Training Strategies for Efficient Embodied Reasoning

William Chen[1], **Suneel Belkhale**[2], **Suvir Mirchandani**[2]
Oier Mees[1], **Danny Driess**[3], **Karl Pertsch**[1], **Sergey Levine**[1]
[1]UC Berkeley, [2]Stanford University, [3]Physical Intelligence
ecot-lite.github.io

Abstract: Robot chain-of-thought reasoning (CoT) – wherein a model predicts helpful intermediate representations before choosing actions – provides an effective method for improving the generalization and performance of robot policies, especially vision-language-action models (VLAs). While such approaches have been shown to improve performance and generalization, they suffer from core limitations, like needing specialized robot reasoning data and slow inference speeds. We thus introduce two simple and lightweight approaches that enjoy the performance benefits of learning to generate robot reasonings without the test-time compute expenses. To design these methods, we give several hypotheses for why robot reasoning improves policies – (1) better representation learning, (2) improved learning curricularization, and (3) increased expressivity – then devise simple variants of robot CoT reasoning to isolate and test each one. Of these mechanisms, we find that the improved representations from learning to reason are the most important factor in increasing performance and generalization. Our results provide a more complete characterization of why reasoning helps policy performance, allowing us to design novel recipes for robot reasoning that achieve significant performance gains over non-reasoning policies, state-of-the-art results on the LIBERO-90 benchmark, and a 3x inference speedup compared to standard robot reasoning.

Keywords: Robot Reasoning, Vision-Language-Action Models

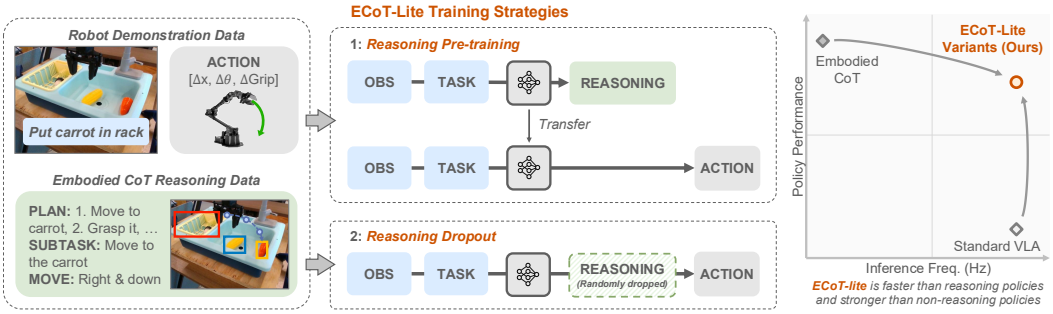

Figure 1: Illustration of our ECoT-Lite approaches, which enjoy the benefits of learning to reason without the test-time compute. Past robot reasoning policies are performant but slow. By testing hypotheses on why robot reasoning improves policy performance, we find two simple lightweight alternatives for training policies with embodied reasoning data *without* producing reasonings at test time, boosting performance over non-reasoning VLAs while maintaining fast inference speeds.

1 Introduction

Useful robots must generalize to a broad range of real-world scenarios. As such, policy *generalization* has been a long-standing focus for the robotics community. Recent works have shown that training policies on large and diverse datasets of robot experience can significantly improve their generalization ability [1–4]. Vision-language-action models (VLAs [5, 6]) have been proposed as a particularly promising approach for generalizable policy training: by fine-tuning a pre-trained vision-language model (VLM) with diverse robot control data, these models combine the scalability of large transformer architectures with rich semantic knowledge from the VLM's internet pre-training.

Correspondence to: verityw@berkeley.edu

While the standard approach to improving policy generalization in such a data-driven regime is to collect increasingly large robot datasets, often through tedious human teleoperation [1, 7–9], the flexibility of the VLA architecture [10–13] has recently given rise to an alternative, *orthogonal* paradigm for improving generalization: embodied chain-of-thought reasoning (ECoT [14]). Inspired by work in reasoning for large language models [15, 16], these approaches train VLAs to split the robot action prediction problem into a sequence of reasoning steps, like identifying the locations of objects and the robot's end-effector [17], planning subtasks [5], or predicting object affordances [18, 19]. Importantly, there is ample evidence that training robot policies to perform such reasoning steps can significantly improve their ability to generalize to new scenes, objects, and task instructions [14], *without* the need to collect additional robot demonstrations.

However, while embodied chain-of-thought reasoning is a promising approach for improving policy generalization, existing robot reasoning approaches come with significant costs: training data needs to be annotated with detailed reasoning instructions, and performing extended reasoning steps during inference can significantly slow down policy rollouts, to the point where a single action prediction may take multiple seconds [14]. The latter downside has made it particularly challenging to use sophisticated robot reasoning approaches in practice.

In this work, we develop more practical strategies for training policies with embodied reasoning data, which we call ECoT-Lite. These recipes **learn to reasoning, but do not generate them at test time,** thereby retaining most of the generalization benefits of regular CoT policies while retaining high throughput. We develop ECoT-Lite by identifying distinct hypotheses for *why* embodied chain-of-thought reasoning may help generalization, e.g., it may improve the policy's representations or increase the effective model capacity via increased inference compute. After instantiating novel reasoning recipes which isolate and test our hypotheses, we find that full test-time reasoning has the best performance, as expected. However, perhaps surprisingly, we confirm that simply *learning* to reason confers significant performance benefits, even if reasonings are absent at test time. This suggests that reasoning helps policies via better representation learning, while also offering readily-applicable recipes for leveraging embodied reasoning without hindering policy speeds.

Concretely, ECoT-Lite achieves state-of-the-art performance on the widely used LIBERO simulation benchmark [20], and outperforms standard VLAs on BridgeData V2 evaluations by 10-19%, while speeding up inference from 1-1.2 Hz for conventional embodied chain-of-thought reasoning approaches to the 3.5+ Hz of standard VLAs. In doing so, we also develop a deeper understanding of why embodied chain-of-thought works, which we hope will inspire future robot reasoning research.

In summary, our core contributions are: **(1)** a detailed empirical analysis of the mechanisms by which embodied chain-of-thought reasoning improves policies, from which we develop **(2)** a set of novel, simplified strategies for training robot reasoning policies, which we validate with **(3)** comprehensive simulated and real-robot evaluations that show that our new recipes retain substantially improved generalization performance while making robot reasoning policies significantly more practical.

2 Related Work

Vision-language-action models. Vision language action models (VLAs) are a way to train robot policies by adapting vision-language models (VLMs) [21–24] to output robot actions [5, 6, 10–13, 25–38]. Several works have shown that by framing action prediction as a vision-language problem, the model benefits from the internet-scale pre-training on general language and vision-language tasks, leading to better robustness and generalization capabilities [5]. VLAs are trained via behavioral cloning (BC) on large datasets [1, 7, 8, 39], typically representing actions as discrete tokens.

Chain-of-thought in LLMs and VLMs. Many studies have shown that chain-of-thought reasoning (CoT) – wherein a model generates intermediate reasoning steps before producing an answer to a query [15, 16] – can significantly improve the performance of LLMs and VLMs. The intuition is that each intermediate step may be easier for a model to predict compared to directly generating the correct answer. CoT has been particularly successful for tasks involving mathematics [40, 41], where this intuition is theoretically justified: the marginal of a fix-sized transformer architecture is not expressive enough to solve problems like integer arithmetic or dynamic programming, but

employing suitably-long CoT allows LLMs to solve these problems [42]. Other analyses of why reasoning helps LLMs make similar arguments, showing that intermediate reasoning steps increase the expressivity of a transformer [43, 44]. While these theoretical analyses give insights on the complexity of logical problems that can be solved with CoT, this view is incomplete, since it does not address why reasoning empirically helps with *common-sense* problem solving (like VQA or robotics) [15], which differs from math problems in that it requires connecting fragmented or implicit knowledge about the real world instead of logical deduction over well-defined operations on abstract symbols. In this work, we aim to fill this gap by investigating several hypotheses about why CoT benefits real-world robotics problems. We focus specifically on tasks that rely on semantic reasoning, similar to common-sense question answering, as distinct from reasoning over logical symbols.

Robot reasoning. Inspired by the success of CoT for LLMs, there are several works that try to leverage CoT reasoning for robotics [5, 14, 46, 47]. Instead of mapping observations directly to robot actions as in standard VLAs, CoT in robotics first generates robot "reasonings" before predicting the final actions [14]. Typical reasoning steps include breaking down goals into subtasks [5, 48–51]; grounded features including representations of robot goals or motions [46, 52–55]; and task-relevant visuo-semantic features [14, 56] such as object bounding boxes or semantic keypoints. Prior works show that predicting these intermediate reasonings improves policy generalization compared to training only with action prediction, but fail to dissect *why* this is the case. Compared to LLMs, robot reasoning in VLAs cannot usually be elicited by prompting alone. Instead, the model is explicitly trained to produce reasonings [14], which raises the question to what extent the CoT itself or the additional training signal contributes to the increased performance. Our work investigates this by dissecting the different aspects of CoT both at training and inference time.

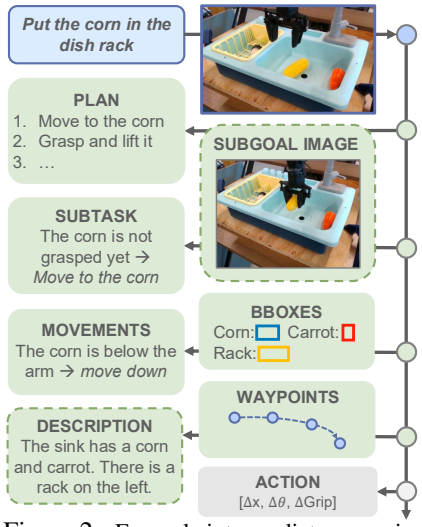

Figure 2: Example intermediate reasoning steps. We use Embodied Chain-of-Thought Reasoning (ECoT [14]) as a representative robot reasoning approach for this work, and thus indicate which steps it does *not* use with dashed borders (but they are used in other similar works [45, 46]).

Robot representation learning. Training a VLA to reason can be interpreted as a form of representation learning, which has a long history in robot learning. Numerous authors have suggested pre-training base models on egocentric/embodied data [57, 58] or grounded spatial reasoning tasks [59, 60], as the resulting representations might be more conducive to learning control. Burns et al. [61] provide experimental support that representations conducive to spatial-semantic understanding tasks (e.g., segmentation) are indicative of their visual robustness in robot policies. Further, a core motivation behind VLAs is that VLMs learn generalizable visuo-linguistic representations during pre-training [5]. Even a frozen VLM's representations can be used to learn policies, with performance increasing further if representations of reasoning are included [56]. However, when analyzing a wider set of generalization axes, Gao et al. [62] found that co-training with VLM training data was not uniformly beneficial for all types of generalization. Finally, Gemini Robotics Team et al. [33] connects embodied representation learning and reasoning by both training their base VLM on embodied data and then fine-tuning it into a reasoning robot policy.

In summary, previous work either focuses on representation learning, theoretical analysis of CoT for math problems, or simply shows that CoT improves performance for robotics. To our knowledge, our work is the first to dissect why training on and using CoT at inference time is beneficial in embodied robotics settings, investigating both the representation learning and test-time compute aspects.

3 Preliminaries

Vision-language-action models. Given a task expressed in natural language ℓ, obtaining a robot policy can be seen as learning a function π from which we can sample actions $a \sim \pi(\cdot|\ell, o)$ conditioned on the robot observation o (e.g. image observations or the robot's proprioceptive state).

The idea of vision-language-action models is to represent robot actions as a sequence of text tokens, for example through simple discretization and binning strategies, such that a policy can be trained by finetuning a vision-language model into outputting robot actions autoregressively.

Embodied chain-of-thought reasoning. We consider *Embodied Chain-of-Thought Reasoning* (ECoT) as a representative robot reasoning approach for this work [14]. ECoT makes a simple change to the VLA formulation: the policy first generates reasoning text before predicting the action tokens. Possible reasoning steps include both high-level planning (such as subtask prediction) and low-level grounded features such as motions, gripper positions, and object bounding boxes (see Fig. 2). A typical approach to obtain the reasoning data for training is to annotate robot action trajectories with various foundation models. At training time, these reasoning texts are tokenized and prepended to the action tokens, which the model learns to generate via next-token prediction (same as for the base VLM's pre-training). Each inference step involves decoding the reasoning text followed by actions, then executing the latter, making ECoT policies' inference speeds low.

4 Why Does Embodied Chain-of-Thought Reasoning Improve Performance?

We aim to develop more practical recipes for training embodied reasoning policies that avoid its conventional drawbacks. We start by formulating hypotheses for *why* reasoning improves policy performance. We will then use these hypotheses to develop multiple novel, lightweight recipes for reasoning policy training in Section 5, and test them empirically in Section 6.

Hypothesis 1: Embodied reasoning improves representation learning. One possible explanation for the benefit of reasoning is that the additional knowledge supplied to the model via the reasoning steps influences the model's representations. For example, the reasoning trace could indicate that a particular object's location is relevant. If improved representations were the primary explanation for the increase in policy performance, we would expect that actually generating reasonings at test-time is less important compared to supervising the model with the information contained in the reasoning steps, such that its internal representations are attentive to the features that the reasoning steps require it to predict. If this holds true, we can design approaches that use reasoning during training for improved representation learning, but forego slow reasoning at test time to retain low policy latency.

Hypothesis 2: Embodied reasoning provides learning curriculum. An alternative hypothesis is that embodied reasoning features provide the policy with an implicit learning curriculum, where it can first learn the comparatively straight-forward reasoning tasks, like mapping from low-level movement primitives to robot actions, and then "work its way up" to the full observation-to-action prediction task. Thus, the model may learn a more generalizable mapping from images to actions, instead of struggling from the start with the end-to-end prediction task and resorting to memorizing solutions. Similar effects have been observed in LLM training [63]. If this holds true, we can design training recipes where we simply use embodied reasoning as a "scaffolding" for training, but remove it entirely during inference to simplify and speed-up policy rollouts.

Hypothesis 3: Embodied reasoning increases effective model expressivity. Finally, embodied reasoning extends the *length* of the sequence of tokens that the VLA is operating over. As such, the model leverages more compute during training and inference than comparably-sized, regular VLA policies, and thus effectively increases its expressiveness. To this end, multiple works in language and vision-language modeling have observed that merely increasing the *number* of tokens, even without adding new information, can improve model performance [22, 23, 64, 65]. Similarly, we may design a simplified training recipe that introduces additional tokens *without* the need to perform extensive reasoning annotation on the data ahead of time.

5 ECoT-Lite: Practical Training Recipes for Embodied Reasoning Policies

In this section, we use the hypotheses from the previous section to design a number of simplified embodied chain-of-thought training recipes. Our goal is to retain most of the generalization abilities of the original ECoT recipe introduced in Section 3, but mitigate the need for either extensive data annotation or high-latency inference. We provide an overview of our proposed approaches in Fig. 3.

Reasoning pre- or co-training. Following **Hyp. 1**, we can use the reasoning data to shape the policy's *representations* by pre- or co-training the VLA on it – both commonly used in other domains

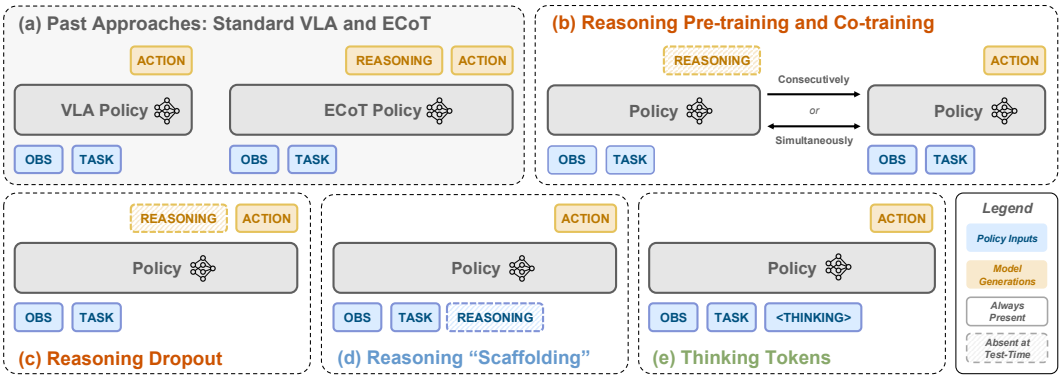

Figure 3: ECoT-Lite training recipes. Blue indicates inputs, orange indicates outputs/generations, dashed border represents absence during test-time (and random drop-out during training). **(a)**: Standard VLA [5, 6] and embodied CoT [14] training. **(b)** Pre-train or co-train VLA models with embodied reasoning data. **(c)**: Provide reasoning data as a "scaffolding" in context during training. **(d)**: Train with reasoning dropout, remove reasoning during inference. **(e)**: Introduce non-semantic "thinking tokens" to increase effective model expressivity.

[66–68]. For pre-training, the base VLM is trained to just produce reasonings, then trained on just actions. For co-training, the VLM is trained on both objectives simultaneously – half of the training batch maps observations to reasonings only, the other half to actions only. This treats reasoning as an auxiliary loss. At inference time, both pre-training and co-training policies operate as standard VLAs, without explicit reasoning and thus with low latency.

Test-time reasoning dropout. Similar to the above approaches, the main goal here is to use embodied reasoning data to shape the policy's representation (**Hyp. 1**). However, instead of treating reasoning as a fully separate auxiliary task, this approach trains the model to explicitly use the reasoning information *for action prediction*, by *sequencing* reasoning and action tokens. Importantly, it performs *dropout* on the reasoning steps such that part of the training examples contain *no* reasoning. This allows us to again use the policy without reasoning prediction at inference time, while explicitly using reasoning for action prediction during training may encourage better representational transfer than simple pre- or co-training. Thus, this approach allows us to isolate the impact of generating test-time reasonings by turning it on or off and measuring the resulting performance.

Reasoning "scaffolding". As outlined in **Hyp. 2**, we may want to use reasoning only as a scaffold for the learning process, *without* requiring the policy to expend capacity for learning to actually predict the reasoning itself. We thus can provide reasoning examples *in context* during training, but with no loss on them. We can again apply dropout on the reasoning steps such that a fraction of the training examples contain *no* reasoning scaffold, i.e., train the policy to perform direct observation-to-action mapping. At inference time, we can then use this policy like a standard VLA, with low latency.

Thinking tokens. We can increase the effective capacity of the policy, *without* semantic reasoning, by introducing empty "thinking tokens" in the context of the model during both, training and inference (**Hyp. 3**). In this way, we artificially increase the token sequence length and thus the computational resources at the VLA's disposal. Similar approaches have been explored in the context of LLM training [64, 65]. We set the number of thinking tokens to a random value within the range of reasoning token lengths in our embodied reasoning data, to enable a fair, compute-matched comparison.

6 Experiments

Our goal is to test the performance of our lightweight embodied reasoning recipes from Section 5 in comparison to standard VLA and ECoT inference. Specifically, we aim to answer the following:

(1) How does the performance of the ECoT-Lite policies compare to regular VLAs and ECoT?

(2) Which of the hypotheses in Section 4 is best supported by empirical results and explains why embodied reasoning improves policy generalization?

(3) How can we decide *which* embodied reasoning strategy to use in a given scenario?

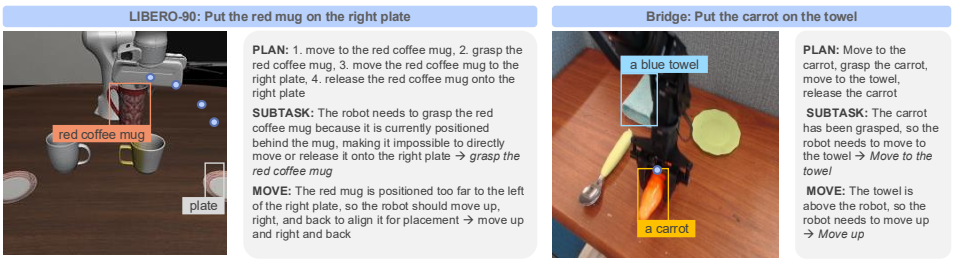

Figure 4: Example ECoT reasonings for LIBERO and Bridge. See Fig. 6 for more examples.

6.1 Experimental Setup

Environments & Tasks. We evaluate our approach in two environments: the widely-used, simulated LIBERO robot manipulation benchmark [20], and a similar distribution of evaluation tasks from the BridgeData V2 [8] environment that was used in prior works on VLAs and robot reasoning [6, 14]. In both environments, we test on a diverse set of tasks that requires policies to generalize beyond their training data: in LIBERO we introduce multiple levels of randomization to initial object positions and distractor objects; in the BridgeData V2 evaluations we follow Zawalski et al. [14] and Kim et al. [6] and evaluate on in-distribution, motion generalization, spatial relations, and unseen objects. For both environments we train policies on the publicly available LIBERO and Bridge V2 training datasets. For more details on robot setups, datasets, and tasks, see Appendix B or Fig. 4 for reasoning examples. We refer to Appendix C for details of how the embodied reasoning annotations are created.

Policy training. For each environment, we use standard, auto-regressive VLA architectures (Section 3) from prior work (see Appendix D for details). We emphasize that our policy architecture choices are merely trying to mirror prior state-of-the-art policies on the respective environments, but the embodied reasoning training recipes we develop are *agnostic* to the choice of the underlying VLA and can readily be applied to other VLA architectures.

Comparisons. We compare our ECoT-Lite training recipes to standard (non-reasoning) VLA training [6, 31], and prior embodied chain-of-thought policy training approaches [14]. We ensure that all approaches are trained on the same amount of robot demonstration and reasoning data, using the same amount of computational resources and hyperparameters, and evaluated under comparable initial states, lighting conditions, and camera angles. All these factors are controlled to best ensure that performance differences come from how reasoning data is used, as discussed in our three hypotheses. In total, our results contain 121,500 simulated and 444 real-robot trials.

6.2 ECoT-Lite Enables Generalizable Policies with Fast Inference

We present our LIBERO policies' performance in Fig. 5 (top). ECoT and reasoning dropout both significantly increases performance in all three LIBERO-90 splits over the equivalent standard MiniVLA policy. Both achieve around 90% on standard LIBERO-90, **exceeding past LIBERO-90 state-of-the-art results by Mete et al. [69] (88.6%)**. Critically, reasoning dropout does not generate test-time reasonings, so it is much faster than full ECoT. The reasoning pre-training policy yields the second biggest increase in performance over the standard VLA (+5.4%), while co-training and scaffolding yield smaller improvements. No thinking token condition improves over the baseline.

As reasoning dropout and pre-training were the most effective in LIBERO, we validate their real-world applicability in Bridge, presented in Fig. 5 (bottom). Both improve significantly over the standard VLA baseline, and while ECoT remains the most performant, our novel approaches are around 3× faster. Unlike with LIBERO, we find that reasoning dropout is *less* effective than reasoning pre-training for Bridge. Being the most performant approaches, reasoning pre-training and dropout are the main algorithmic contributions for ECoT-Lite. We now discuss these results in more detail.

6.3 Representation Learning Results

All our representation learning policies improve on the non-reasoning VLA, in support of **Hyp. 1**. We find that reasoning co-training only makes a marginal improvement over the baseline (+1.9%), but pre-training makes a more significant one (+5.4%). Gao et al. [70] similarly notes a mixed impact of co-training on actions and VQA data. Our result is perhaps more surprising, as we are co-training

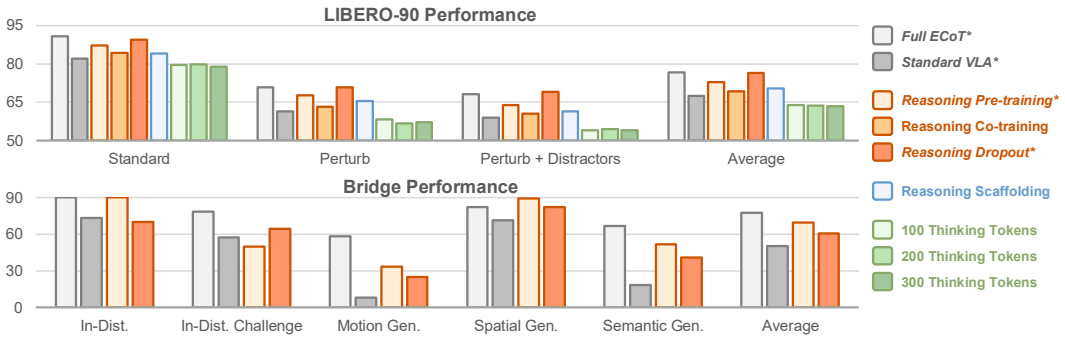

Figure 5: **Top:** Performance of all methods on LIBERO-90 benchmarks. The most performant approaches are ECoT and the ECoT-Lite reasoning dropout policy, both of which beat past state-of-the-art on the standard LIBERO-90 evaluations (90.8% and 89.4% vs. 88.6% by Mete et al. [69]). Reasoning pre-training also improves performance significantly. See Table 1 for numerical values and standard errors. **Bottom:** We replicate the reasoning dropout and pre-training policies in Bridge to validate their real-world effectiveness. Both ECoT-Lite approaches improve on the standard VLA's performance. While full ECoT is the most performant, the ECoT-Lite policies do not generate test-time reasonings, making their inference speeds much faster. See Table 2 for per-task numerical values and standard errors. Asterisks in legend indicate method appears in top and bottom.

on data from the same domain as our action prediction task, while VQA data is generally more distinct from robot data. Indeed, the fact that reasoning pre-training, dropout, or standard ECoT all significantly improve performance shows that learning from the reasoning data *should* be helpful for improving action prediction; co-training just seems to be an ineffective way to do so.

Why is pre-training better than the baseline? Reasoning pre-training causes the model's internal representations to capture the reasoning features that are helpful for robust action prediction. In contrast, tuning a base VLM to predict actions in the standard VLA recipe may cause the resulting policy to learn a poor mapping, as it has to leverage representations learned from *general* vision-language data (which may be less suited to action prediction, due to low exposure to robot data). This in turn leads to lower performance. Such an argument is commonly applied to adapting LLMs [66].

Why is co-training worse than pre-training? When co-training, the policy does still learn to generate reasonings, so its representations should also be improved. However, it critically learns these *at the same time* that it fits a mapping from observations to actions. We suspect that, once learning to reason induces good representations, the VLA has already learned a poor mapping from images to actions, i.e., the model learns to predict actions and reasonings with separate parameters (with little overlap). At this point, it may be hard for action prediction to "switch" to being based on the good reasoning representations. In contrast, by having reasoning as the sole pre-training objective, the model devotes all parameters to modeling the features needed to reason; when action fine-tuning starts, the model *must* tune parameters that were used for reasoning. Similar arguments have been presented in past works [71], which we discuss in more detail in Appendix A.1.

When does generating test-time reasonings matter? We find that our reasoning dropout policy performs best on LIBERO-90 *regardless* of if it generates test-time reasonings. We suspect this may be a byproduct of LIBERO-90 being narrow (containing 90 tasks vs. the thousands in real robot datasets like Bridge or DROID [7, 8, 39]). As a result, the LIBERO reasoning steps are not very diverse; features like the plan, subtasks, and object labels have little variation for any given task. Thus, it is likely much easier for the model to fully memorize and "internalize" the reasonings, making generating them at test time superfluous. This is corroborated by our more diverse Bridge evaluation, wherein enabling test-time reasoning *does* improve performance compared to disabling it. Qualitatively, this performance gap mainly comes from the reasoning dropout policy picking up incorrect objects or colliding with obstacles – failure modes that would intuitively be alleviated by generating bounding boxes or motion rationales in test-time reasonings (see Fig. 8 for examples).

6.4 Learning Curriculum Results

Our reasoning scaffolding approach yields slightly better performance than the baseline (+2.9%), comparable to reasoning co-training. This weakly supports **Hyp. 2**, that having reasonings in-context

during training *does* improve the learned observation to action mapping, even when reasonings are absent at test time. Notably, this approach only differs from reasoning dropout in that the latter also assigns losses to reasoning tokens; it is therefore the combination of scaffolding *and* learning to reason that leads to the most significant improvement for LIBERO.

6.5 Improved Expressivity Results

We find that adding thinking tokens degrades performance relative to the baseline (-3.8% on average). This runs counter to **Hyp. 3** and empirical results in language modeling, wherein thinking tokens improve performance in various domains [64, 65], suggesting that the main benefit of robot reasoning is *not* increased policy expressivity, but in actually learning to produce semantically-meaningful reasoning steps. We note that these experiments use MiniVLA [31], which is much smaller than other popular VLAs (1B vs. 55B for RT-2, 7B for OpenVLA, and 3B for π_0 [5, 6, 10]). Our results nonetheless indicate that expressivity is not a bottleneck for it.

7 Which Robot Reasoning Approach is Best for My Problem?

As shown in Fig. 5, we find that when trained on the same demonstration data, standard VLAs are universally outperformed by ECoT and our two ECoT-Lite variants, indicating that all three successfully use the same embodied reasoning datasets to improve policy performance. To determine which one is appropriate, we thus now consider their compute, inference, and performance tradeoffs, then make prescriptions for when to use each.

First, we find that **embodied chain-of-thought reasoning** is the most performant approach. However, it is also the slowest: with TensorRT-LLM FP8 compilation on an H100 GPU [72, 73], the policy only achieves around 1-1.2 Hz control frequency (or 0.3-0.5 Hz on 4090 without compilation), compared to VLAs of the same architecture, which achieve around 3-4 Hz (4090, no compilation[1]). As both ECoT-Lite variants also map observations directly to actions, they share that faster inference speed.

Of these variants, **reasoning dropout** matches the state-of-the-art performance of full ECoT on all variants of LIBERO (Fig. 5, top). As it is trained the same way as ECoT (just with the reasoning occasionally dropped out), dropout has the same resource demands as ECoT, and even affords users the flexibility of "turning on" the reasonings at test time.

Finally, **reasoning pre-training** does better than dropout on Bridge in all but one split (Fig. 5, bottom). It has the downside of needing more gradient steps in order to learn reasoning and action prediction consecutively (whereas ECoT and reasoning dropout learns them simultaneously). However, this also means that it does not need *paired* reasoning steps and action data[2]. Last, as it forgoes training on reasonings and actions in the same context window, each training datapoint requires less memory, which is particularly salient when actions are expressed as many tokens.

Our prescription is: use full ECoT to maximize performance, at the cost of slower inference. Use reasoning dropout in narrower task domains or if the option to "turn on" test-time robot reasonings is needed. Use reasoning pre-training in more diverse task domains, or if unpaired embodied reasoning data is available (and training for more steps is acceptable).

8 Discussion

We investigated three hypotheses for why reasoning improves learned robot policies, namely that they aid representation learning, provide learning curricula, or increased policy expressivity. By isolating each of these improvement mechanisms in extensive simulated experiments, we find that reasoning pre-training and test-time reasoning dropout are effective ways for maintaining the improved performance conferred by robot reasoning while avoiding its slower inference speeds. We validate these approaches' effectiveness in real-world manipulation experiments, then finally give explicit prescriptions for when each approach is appropriate.

[1] Most of the compilation speed-up comes from FP8, which must be run on Hopper GPUs (it is unsupported on 4090) [73]. We also find that TensorRT-LLM compilation does not speed up non-reasoning policies much.

[2] In principle, it can learn from arbitrary embodied robot reasoning data, including from other embodiments. However, we leave investigations into this possibility to future works.

9 Limitations

While we have shown the effectiveness of our proposed approaches, they do not address all of robot reasonings' limitations. Our results show that policy expressivity is likely *not* a bottleneck for VLAs, with our thinking token approach failing to improve performance. While unsurprising, this means that to enjoy the benefits of robot reasoning, users need robot reasoning training data, which can be difficult or expensive to extract.

Additionally, while we have conducted extensive empirical analysis on our proposed approaches' *performance* and have provided intuitive speculation on why they are or are not effective, we leave investigation of reasoning's impact on actual model learning dynamics to future works. Such fine-grained analysis may prove useful for understanding *why* certain training schemes yield better representational transfer or grounding than others.

Lastly, as part of our analysis, we hold many design choices (e.g., policy architecture, training hyperparameters, and reasoning corpora) constant to make our comparisons more controlled and fair. While we have presented empirical evidence of ECoT-Lite's performance benefits, further optimizations are possible – for example, for reasoning pre-training, one could investigate if said approach enables better cross embodiment *reasoning* transfer, taking advantage of how said approach does not need paired reasoning-action data.

Acknowledgments

We thank Aviral Kumar, Evan Hernandez, Kyle Stachowicz, Seohong Park, and Vivek Myers for insightful discussions. This research was partly supported by ONR N00014-25-1-2060, NSF IIS-2150826, and ARL DCIST CRA W911NF-17-2-0181, with additional support from Volkswagen and NVIDIA. We also thank the NVIDIA Academic Grant Program for providing the compute resources used in this work.

References

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

S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2024.

[2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023.

[3] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[4] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *Conference on Robot Learning*, 2024.

[5] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, P. Florence, C. Fu, M. G. Arenas, K. Gopalakrishnan, K. Han, K. Hausman, A. Herzog, J. Hsu, B. Ichter, A. Irpan, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, L. Lee, T.-W. E. Lee, S. Levine, Y. Lu, H. Michalewski, I. Mordatch, K. Pertsch, K. Rao, K. Reymann, M. Ryoo, G. Salazar, P. Sanketi, P. Sermanet, J. Singh, A. Singh, R. Soricut, H. Tran, V. Vanhoucke, Q. Vuong, A. Wahid, S. Welker, P. Wohlhart, J. Wu, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[6] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. 2024.

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

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

[9] E. Rosete-Beas, O. Mees, G. Kalweit, J. Boedecker, and W. Burgard. Latent plans for task agnostic offline reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, Auckland, New Zealand, 2022.

[10] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[11] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[12] J. Jones, O. Mees, C. Sferrazza, K. Stachowicz, P. Abbeel, and S. Levine. Beyond sight: Finetuning generalist robot policies with heterogeneous sensors via language grounding. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Atlanta, USA, 2025.

[13] H. Zhen, X. Qiu, P. Chen, J. Yang, X. Yan, Y. Du, Y. Hong, and C. Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

[14] M. Zawalski, W. Chen, K. Pertsch, O. Mees, C. Finn, and S. Levine. Robotic control via embodied chain-of-thought reasoning. In *Conference on Robot Learning*, 2024.

[15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[16] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners, 2023.

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

[18] R. Xu, J. Zhang, M. Guo, Y. Wen, H. Yang, M. Lin, J. Huang, Z. Li, K. Zhang, L. Wang, et al. A0: An affordance-aware hierarchical model for general robotic manipulation. *arXiv preprint arXiv:2504.12636*, 2025.

[19] J. Borja-Diaz, O. Mees, G. Kalweit, L. Hermann, J. Boedecker, and W. Burgard. Affordance learning from play for sample-efficient policy learning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Philadelphia, USA, 2022.

[20] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023. URL https://arxiv.org/abs/2306.03310.

[21] S. Karamcheti, S. Nair, A. Balakrishna, P. Liang, T. Kollar, and D. Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models, 2024.

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

[23] A. Steiner, A. S. Pinto, M. Tschannen, D. Keysers, X. Wang, Y. Bitton, A. Gritsenko, M. Minderer, A. Sherbondy, S. Long, S. Qin, R. Ingle, E. Bugliarello, S. Kazemzadeh, T. Mesnard, I. Alabdulmohsin, L. Beyer, and X. Zhai. Paligemma 2: A family of versatile vlms for transfer, 2024. URL https://arxiv.org/abs/2412.03555.

[24] W. Dai, J. Li, D. Li, A. M. H. Tiong, J. Zhao, W. Wang, B. Li, P. Fung, and S. Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023.

[25] A. Szot, B. Mazoure, O. Attia, A. Timofeev, H. Agrawal, D. Hjelm, Z. Gan, Z. Kira, and A. Toshev. From multimodal llms to generalist embodied agents: Methods and lessons. *arXiv preprint arXiv:2412.08442*, 2024.

[26] H. Huang, F. Liu, L. Fu, T. Wu, M. Mukadam, J. Malik, K. Goldberg, and P. Abbeel. Otter: A vision-language-action model with text-aware visual feature extraction. *arXiv preprint arXiv:2503.03734*, 2025.

[27] J. Wen, Y. Zhu, J. Li, M. Zhu, K. Wu, Z. Xu, N. Liu, R. Cheng, C. Shen, Y. Peng, F. Feng, and J. Tang. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2409.12514*, 2024.

[28] J. Liu, H. Chen, P. An, Z. Liu, R. Zhang, C. Gu, X. Li, Z. Guo, S. Chen, M. Liu, et al. Hybridvla: Collaborative diffusion and autoregression in a unified vision-language-action model. *arXiv preprint arXiv:2503.10631*, 2025.

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

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

[31] S. Belkhale and D. Sadigh. Minivla: A better vla with a smaller footprint, 2024. URL https://ai.stanford.edu/blog/minivla/.

[32] K. Pertsch, K. Stachowicz, B. Ichter, D. Driess, S. Nair, Q. Vuong, O. Mees, C. Finn, and S. Levine. Fast: Efficient action tokenization for vision-language-action models, 2025. URL https://arxiv.org/abs/2501.09747.

[33] Gemini Robotics Team, S. Abeyruwan, J. Ainslie, J.-B. Alayrac, M. G. Arenas, T. Armstrong, A. Balakrishna, R. Baruch, M. Bauza, M. Blokzijl, S. Bohez, K. Bousmalis, A. Brohan, T. Buschmann, A. Byravan, S. Cabi, K. Caluwaerts, F. Casarini, O. Chang, J. E. Chen, X. Chen, H.-T. L. Chiang, K. Choromanski, D. D'Ambrosio, S. Dasari, T. Davchev, C. Devin, N. D. Palo, T. Ding, A. Dostmohamed, D. Driess, Y. Du, D. Dwibedi, M. Elabd, C. Fantacci, C. Fong, E. Frey, C. Fu, M. Giustina, K. Gopalakrishnan, L. Graesser, L. Hasenclever, N. Heess, B. Hernaez, A. Herzog, R. A. Hofer, J. Humplik, A. Iscen, M. G. Jacob, D. Jain, R. Julian, D. Kalashnikov, M. E. Karagozler, S. Karp, C. Kew, J. Kirkland, S. Kirmani, Y. Kuang, T. Lampe, A. Laurens, I. Leal, A. X. Lee, T.-W. E. Lee, J. Liang, Y. Lin, S. Maddineni, A. Majumdar, A. H. Michaely, R. Moreno, M. Neunert, F. Nori, C. Parada, E. Parisotto, P. Pastor, A. Pooley, K. Rao, K. Reymann, D. Sadigh, S. Saliceti, P. Sanketi, P. Sermanet, D. Shah, M. Sharma, K. Shea, C. Shu, V. Sindhwani, S. Singh, R. Soricut, J. T. Springenberg, R. Sterneck, R. Surdulescu, J. Tan, J. Tompson, V. Vanhoucke, J. Varley, G. Vesom, G. Vezzani, O. Vinyals, A. Wahid, S. Welker, P. Wohlhart, F. Xia, T. Xiao, A. Xie, J. Xie, P. Xu, S. Xu, Y. Xu, Z. Xu, Y. Yang, R. Yao, S. Yaroshenko, W. Yu, W. Yuan, J. Zhang, T. Zhang, A. Zhou, and Y. Zhou. Gemini robotics: Bringing ai into the physical world, 2025. URL https://arxiv.org/abs/2503.20020.

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

[35] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[36] H. Etukuru, N. Naka, Z. Hu, S. Lee, J. Mehu, A. Edsinger, C. Paxton, S. Chintala, L. Pinto, and N. M. M. Shafiullah. Robot utility models: General policies for zero-shot deployment in new environments. *arXiv preprint arXiv:2409.05865*, 2024.

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

[38] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024.

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

[40] Z. Sprague, F. Yin, J. D. Rodriguez, D. Jiang, M. Wadhwa, P. Singhal, X. Zhao, X. Ye, K. Mahowald, and G. Durrett. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*, 2024.

[41] C. Snell, J. Lee, K. Xu, and A. Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

[42] G. Feng, B. Zhang, Y. Gu, H. Ye, D. He, and L. Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective, 2023. URL https://arxiv.org/abs/2305.15408.

[43] W. Merrill and A. Sabharwal. The expressive power of transformers with chain of thought, 2024. URL https://arxiv.org/abs/2310.07923.

[44] Z. Li, H. Liu, D. Zhou, and T. Ma. Chain of thought empowers transformers to solve inherently serial problems, 2024. URL https://arxiv.org/abs/2402.12875.

[45] J.-J. Hwang, R. Xu, H. Lin, W.-C. Hung, J. Ji, K. Choi, D. Huang, T. He, P. Covington, B. Sapp, Y. Zhou, J. Guo, D. Anguelov, and M. Tan. Emma: End-to-end multimodal model for autonomous driving, 2024. URL https://arxiv.org/abs/2410.23262.

[46] Q. Zhao, Y. Lu, M. J. Kim, Z. Fu, Z. Zhang, Y. Wu, Z. Li, Q. Ma, S. Han, C. Finn, A. Handa, M.-Y. Liu, D. Xiang, G. Wetzstein, and T.-Y. Lin. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models, 2025. URL https://arxiv.org/abs/2503.22020.

[47] J. Clark, S. Mirchandani, D. Sadigh, and S. Belkhale. Action-free reasoning for policy generalization, 2025. URL https://arxiv.org/abs/2502.03729.

[48] P. Sharma, A. Torralba, and J. Andreas. Skill induction and planning with latent language, 2022.

[49] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter. Inner monologue: Embodied reasoning through planning with language models, 2022.

[50] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. 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.

[51] O. Mees, J. Borja-Diaz, and W. Burgard. Grounding language with visual affordances over unstructured data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, 2023.

[52] S. Belkhale, T. Ding, T. Xiao, P. Sermanet, Q. Vuong, J. Tompson, Y. Chebotar, D. Dwibedi, and D. Sadigh. Rt-h: Action hierarchies using language, 2024.

[53] J. Gu, S. Kirmani, P. Wohlhart, Y. Lu, M. G. Arenas, K. Rao, W. Yu, C. Fu, K. Gopalakrishnan, Z. Xu, P. Sundaresan, P. Xu, H. Su, K. Hausman, C. Finn, Q. Vuong, and T. Xiao. Rt-trajectory: Robotic task generalization via hindsight trajectory sketches, 2023. URL https://arxiv.org/abs/2311.01977.

[54] D. Niu, Y. Sharma, G. Biamby, J. Quenum, Y. Bai, B. Shi, T. Darrell, and R. Herzig. Llarva: Vision-action instruction tuning enhances robot learning, 2024.

[55] H. Zhou, X. Yao, O. Mees, Y. Meng, T. Xiao, Y. Bisk, J. Oh, E. Johns, M. Shridhar, D. Shah, J. Thomason, K. Huang, J. Chai, Z. Bing, and A. Knoll. Bridging language and action: A survey of language-based robot manipulation. *arXiv preprint arXiv:2312.10807*, 2024.

[56] W. Chen, O. Mees, A. Kumar, and S. Levine. Vision-language models provide promptable representations for reinforcement learning. *Transactions on Machine Learning Research*, 2025.

[57] A. Majumdar, K. Yadav, S. Arnaud, Y. J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, P. Abbeel, J. Malik, D. Batra, Y. Lin, O. Maksymets, A. Rajeswaran, and F. Meier. Where are we in the search for an artificial visual cortex for embodied intelligence?, 2024. URL https://arxiv.org/abs/2303.18240.

[58] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang. Language-driven representation learning for robotics, 2023.

[59] B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Driess, P. Florence, D. Sadigh, L. Guibas, and F. Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities, 2024. URL https://arxiv.org/abs/2401.12168.

[60] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence. Palm-e: An embodied multimodal language model, 2023.

[61] K. Burns, Z. Witzel, J. I. Hamid, T. Yu, C. Finn, and K. Hausman. What makes pre-trained visual representations successful for robust manipulation?, 2023. URL https://arxiv.org/abs/2312.12444.

[62] J. Gao, S. Belkhale, S. Dasari, A. Balakrishna, D. Shah, and D. Sadigh. A taxonomy for evaluating generalist robot policies. *arXiv preprint arXiv:2503.01238*, 2025.

[63] K. Kang, A. Setlur, D. Ghosh, J. Steinhardt, C. Tomlin, S. Levine, and A. Kumar. What do learning dynamics reveal about generalization in llm reasoning?, 2024. URL https://arxiv.org/abs/2411.07681.

[64] J. Pfau, W. Merrill, and S. R. Bowman. Let's think dot by dot: Hidden computation in transformer language models, 2024. URL https://arxiv.org/abs/2404.15758.

[65] S. Goyal, Z. Ji, A. S. Rawat, A. K. Menon, S. Kumar, and V. Nagarajan. Think before you speak: Training language models with pause tokens, 2024. URL https://arxiv.org/abs/2310.02226.

[66] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks, 2020. URL https://arxiv.org/abs/2004.10964.

[67] H. Lang, M. Agrawal, Y. Kim, and D. Sontag. Co-training improves prompt-based learning for large language models, 2022. URL https://arxiv.org/abs/2202.00828.

[68] M. Brief, O. Ovadia, G. Shenderovitz, N. B. Yoash, R. Lemberg, and E. Sheetrit. Mixing it up: The cocktail effect of multi-task fine-tuning on llm performance – a case study in finance, 2024. URL https://arxiv.org/abs/2410.01109.

[69] A. Mete, H. Xue, A. Wilcox, Y. Chen, and A. Garg. Quest: Self-supervised skill abstractions for learning continuous control, 2024. URL https://arxiv.org/abs/2407.15840.

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

[71] Y. Ren, S. Guo, W. Bae, and D. J. Sutherland. How to prepare your task head for finetuning, 2023. URL https://arxiv.org/abs/2302.05779.

[72] NVIDIA. Tensorrt-llm. https://github.com/NVIDIA/TensorRT-LLM?tab=readme-ov-file, 2024.

[73] W. Chen, M. Zawalski, K. Pertsch, O. Mees, C. Finn, and S. Levine. Tensorrt-openvla, 2025. URL https://github.com/rail-berkeley/tensorrt-openvla.

[74] A. Kumar, A. Raghunathan, R. Jones, T. Ma, and P. Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution, 2022. URL https://arxiv.org/abs/2202.10054.

[75] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

[76] M. Deitke, C. Clark, S. Lee, R. Tripathi, Y. Yang, J. S. Park, M. Salehi, N. Muennighoff, K. Lo, L. Soldaini, J. Lu, T. Anderson, E. Bransom, K. Ehsani, H. Ngo, Y. Chen, A. Patel, M. Yatskar, C. Callison-Burch, A. Head, R. Hendrix, F. Bastani, E. VanderBilt, N. Lambert, Y. Chou, A. Chheda, J. Sparks, S. Skjonsberg, M. Schmitz, A. Sarnat, B. Bischoff, P. Walsh, C. Newell, P. Wolters, T. Gupta, K.-H. Zeng, J. Borchardt, D. Groeneveld, C. Nam, S. Lebrecht, C. Wittlif, C. Schoenick, O. Michel, R. Krishna, L. Weihs, N. A. Smith, H. Hajishirzi, R. Girshick, A. Farhadi, and A. Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models, 2024. URL https://arxiv.org/abs/2409.17146.

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

[78] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[79] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer. Sigmoid loss for language image pre-training, 2023.

[80] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.

[81] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning, 2017.

[82] Qwen, :, A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei, H. Lin, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Lin, K. Dang, K. Lu, K. Bao, K. Yang, L. Yu, M. Li, M. Xue, P. Zhang, Q. Zhu, R. Men, R. Lin, T. Li, T. Tang, T. Xia, X. Ren, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Wan, Y. Liu, Z. Cui, Z. Zhang, and Z. Qiu. Qwen2.5 technical report, 2024.

[83] W. Merrill and A. Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers, 2023. URL https://arxiv.org/abs/2207.00729.

[84] A. Yao. Circuits and local computation.

Table 1: Performance of all proposed approaches and baselines on all Libero-90 benchmark variants (Mean ± StdErr). Success rates are computed for 50 trials on all 90 tasks (4500 episodes per variant, or 13500 total episodes per policy).

Libero-90 Variant	Baselines		Representation Learning			Learning Curriculum	Improved Expressivity		
	Full ECoT	Standard VLA	Reasoning Pre-training	Reasoning Co-training	Reasoning Dropout	Reasoning Scaffolding	100 Thinking Tokens	200 Thinking Tokens	300 Thinking Tokens
Standard	**90.8%**	82.0%	*87.1%*	84.2%	**89.4%**	84.1%	79.5%	79.8%	78.9%
Perturb	**70.8%**	61.4%	*67.6%*	63.1%	**70.8%**	65.4%	58.3%	56.8%	57.2%
Perturb + Distractors	**68.2%**	58.9%	*63.9%*	60.5%	**69.1%**	61.3%	54.0%	54.4%	54.0%
Average	**76.6% ± 0.36%**	67.4% ± 0.40%	*72.8% ± 0.38%*	69.3% ± 0.40%	**76.4% ± 0.37%**	70.3% ± 0.39%	63.9% ± 0.41%	63.7% ± 0.41%	63.4% ± 0.41%

A Full Results and Further Discussion

Table 2: Per-task performance of the best ECoT-Lite variants, replicated in Bridge and compared against ECoT and standard VLA baselines.

Split	Task	Past Methods		ECoT-Lite Variants	
		Full ECoT	Standard VLA	Reasoning Pre-training	Reasoning Dropout
In Dist.	Put the [mushroom / corn / eggplant] in the [pot / bowl]	**88.9%**	72.2%	**88.9%**	72.2%
	Place the [spoon / carrot] in on the [towel / plate]	**91.7%**	75.0%	**91.7%**	66.7%
In Dist. Challenge	Put the [broccoli / spoon / cube] on the towel (all green objects)	**83.3%**	16.7%	66.7%	66.7%
	Put the [green / pink] spoon on the [plate / towel]	75.0%	**87.5%**	37.5%	62.5%
Motion Gen.	Put the [carrot / mushroom] on the [plate / pot] (target location is high up)	**58.3%**	8.3%	33.3%	25.0%
Spatial Gen.	Put the [banana / tomato] in the [left / right] bowl	**91.7%**	**91.7%**	83.3%	75.0%
	Put the [green / orange] toy in the [left / right] pot	75.0%	50.0%	**87.5%**	75.0%
	Move the [mushroom / carrot] to the [left / right] of the [carrot / mushroom]	75.0%	62.5%	**100%**	**100%**
Semantic Gen.	Put the watermelon on the towel	66.7%	0%	**83.3%**	83.3%
	Put the toothbrush on the plate	66.7%	33.3%	**83.3%**	50.0%
	Put the screw in the bowel	**66.7%**	33.3%	**66.7%**	16.7%
	Reach for the [ketchup / wrench / mallet]	**66.7%**	11.1%	0%	22.2%
	Aggregate	**77.5% ± 4.0%**	50.5% ± 4.7%	69.4% ± 4.4%	60.4% ± 4.6%

We provide numerical performance values for all our simulated LIBERO and real-world Bridge evaluations in Table 1 and Table 2 respectively. These are identical to the values shown in Fig. 5, just written as explicit success rate values (and divided per-task for the Bridge evaluations).

A.1 Further Discussion: Pre-training vs. Co-training

We provide a (heavily abstracted) visualization of our argument as to why reasoning co-training seems to work poorly compared to pre-training in Fig. 7.

Looking at past LLM literature, both pre-training [66] and co-training [67, 68] on task domain data have been shown to empirically boost model performance. We note that, in these co-training works, the co-training dataset is typically quite large, whereas our robot embodied reasoning datasets are not. If our hypothesis that co-training causes the model to learn action prediction and reasoning separately for LIBERO, it is possible that scaling up reasoning data (e.g., introducing more embodied tasks *or* more robot embodiments) would force more parameter sharing between the two tasks, leading to better transfer.

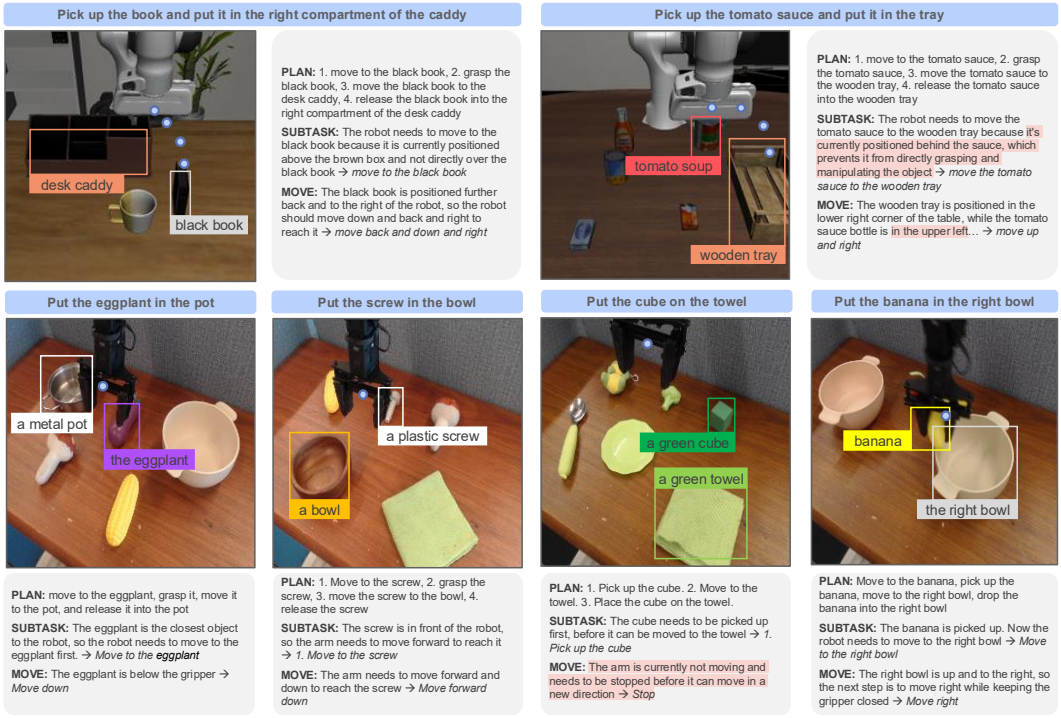

Figure 6: More examples of ECoT reasoning steps generated by our policies (LIBERO on top, Bridge on bottom). Red highlights indicate incorrect or hallucinated features (though they still lead to task successes). Note the slight stylistic differences in the LIBERO reasonings, as the training data generation pipeline for that domain uses a different set of foundation models.

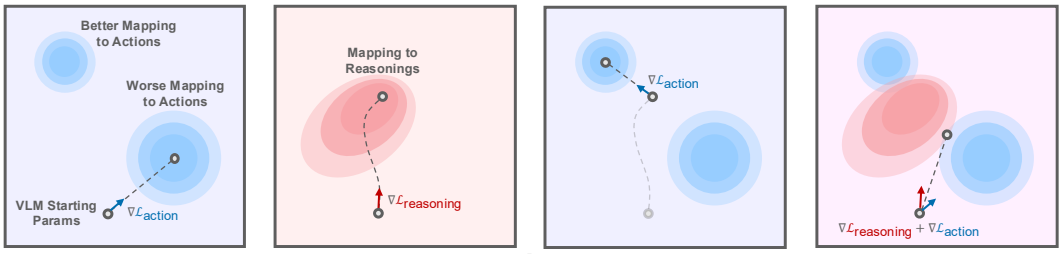

Figure 7: Very abstract illustration of our argument as to why reasoning pre-training seems more effective than co-training, despite using the same data. Blue indicates the loss landscape of the action prediction task, red corresponds to that of reasoning, and darker is lower loss in both cases. Co-training linearly mixes these two loss landscapes and aims to optimize both simultaneously, while pre-training optimizes reasoning and actions disjointly and consecutively. The latter seems to find a better mapping from observations to actions than the former, as shown by our LIBERO results. We suspect this is from the model dedicating all of its parameters and representational capacity to learning reasoning when pre-training, which leads to a part of parameter space that makes learning good actions easier. Note that we do not illustrate the loss landscapes of any approach wherein actions can attend to reasonings (dropout, scaffolding, or ECoT). In that case, since the representations of actions depend on the representations of reasonings, the overall loss landscape is not merely a linear combination of the two tasks' separate landscapes, meaning it is not easily related to the above abstraction.

While not in the context of co- vs. pre-training, several past works have investigated when fine-tuning is effective. For instance, while Ren et al. [71] investigate the setting of how to best initialize a model head atop a pre-trained backbone, they give a somewhat similar argument as what we give in Section 6.3. Specifically, they posit if a model head is over-tuned (i.e., to near minimal loss) while the pre-trained backbone is frozen, it could degrade downstream fine-tuning and performance – paralleling our argument that learning action prediction in tandem with reasoning leads to the former getting stuck in a poor-performance solution.

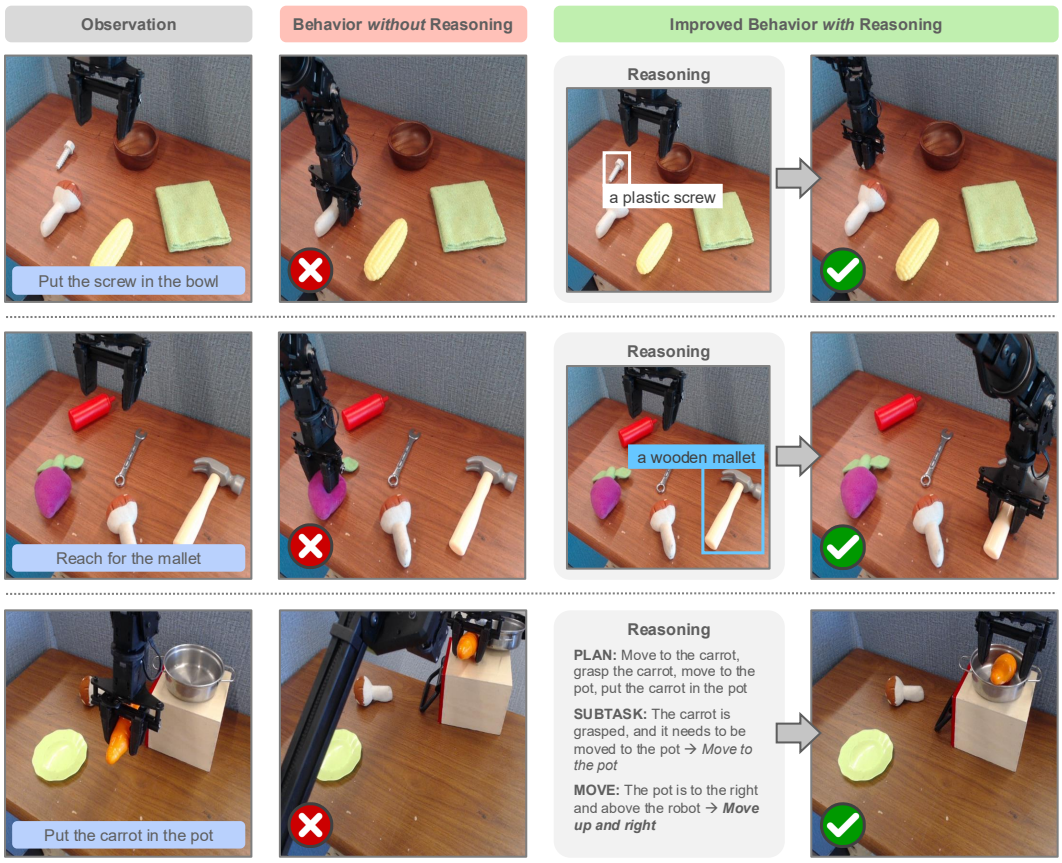

Figure 8: Qualitative examples of the importance of test-time robot reasonings. We show policy behaviors on three different Bridge tasks with the reasonings disabled (reasoning dropout) or enabled (full ECoT), as well as the parts of the reasoning that intuitively lead to correct behaviors. The former leads to failure, while the latter leads to success. In the top and middle tasks, the target grasp object is out-of-distribution. However, the reasoning policy succeeds in labeling it with a bounding box, leading to correct grasping. At the bottom, disabling reasoning causes the robot to collide with the platform and pot, while enabling it causes the arm to move sufficiently high.

Our argument also connects to the work of Kumar et al. [74] in a nuanced way. The authors show how, if a model's pre-trained representations are "good" for a downstream task, excessive fine-tuning can destroy those representations, lowering its performance on out-of-distribution inputs. They find that an effective alternative is to only fine-tune a task-specific linear head on the task of interest, thereby both (1) enforcing a simplicity bias when adapting and (2) better preserving the base model's representations (both of which lead to better performance). Connecting this to our results, co-training serves to "preserve" the reasoning representations throughout training, while reasoning pre-training followed by action fine-tuning does not. Given that the reasoning pre-training policy matches or outperforms even the full ECoT policy in all but one of the semantic generalization tasks, it seems that any distortion of pre-trained features induced by action fine-tuning is not very impactful. Rather, what is important is that the action predictions are grounded in the reasoning representations – preserving them does not matter if the actions do not depend strongly on them in the first place.

A.2 Further Discussion: Dropout vs. Full ECoT

We provide examples of when enabling reasoning results in better performance in Fig. 8. These examples are drawn from the motion generalization and semantic generalization splits of our Bridge evaluations (Table 2). The reasonings contain steps that intuitively help the robot choose good actions. When the reasonings are disabled (in reasoning dropout) or completely absent (in the standard VLA recipe), the policy tends to fail at these tasks.

```
close the top drawer of the cabinet
close the top drawer of the cabinet and put the black bowl on top of it
put the black bowl in the top drawer of the cabinet
put the butter at the back in the top drawer of the cabinet and close it
put the butter at the front in the top drawer of the cabinet and close it
put the chocolate pudding in the top drawer of the cabinet and close it
open the bottom drawer of the cabinet
open the top drawer of the cabinet
open the top drawer of the cabinet and put the bowl in it
put the black bowl on the plate
put the black bowl on top of the cabinet
open the top drawer of the cabinet
put the black bowl at the back on the plate
put the black bowl at the front on the plate
put the middle black bowl on the plate
put the middle black bowl on top of the cabinet
stack the black bowl at the front on the black bowl in the middle
stack the middle black bowl on the back black bowl
put the frying pan on the stove
put the moka pot on the stove
turn on the stove
turn on the stove and put the frying pan on it
close the bottom drawer of the cabinet
close the bottom drawer of the cabinet and open the top drawer
put the black bowl in the bottom drawer of the cabinet
put the black bowl on top of the cabinet
put the wine bottle in the bottom drawer of the cabinet
put the wine bottle on the wine rack
close the top drawer of the cabinet
put the black bowl in the top drawer of the cabinet
put the black bowl on the plate
put the black bowl on top of the cabinet
put the ketchup in the top drawer of the cabinet
close the microwave
put the yellow and white mug to the front of the white mug
open the microwave
put the white bowl on the plate
put the white bowl to the right of the plate
put the right moka pot on the stove
turn off the stove
put the frying pan on the cabinet shelf
put the frying pan on top of the cabinet
put the frying pan under the cabinet shelf
put the white bowl on top of the cabinet

turn on the stove
```

Figure 9: First half of the 90 task prompts for LIBERO-90. Note some are repeated in different scenes (and thus they appear multiple times in this list)

Disabling reasonings does not see to affect the spatial generalization tasks (which test the policies' ability to understand spatial relations, listed as the fourth split in Table 2). We suspect this comes from how Bridge containing numerous examples of left/right directional references in its task labels, allowing non-reasoning policies to "internalize" those concepts.

Curiously, we find that reasoning dropout does very well in the semantic generalization tasks (except the reaching ones).

B Environment Details

B.1 LIBERO-90 Description

LIBERO-90 provides a suite of 90 simulated manipulation benchmarking tasks. The platform has both standardized evaluation features (episode spawn ranges, objects, scenes, success detection, controllable random seeds, etc) and a demonstration dataset of 50 rollouts per episode [20]. We use the latter to train our LIBERO policies via behavioral cloning, as many past works have done [10, 31, 69, 75]. We note that we filter out any trajectories that, when rolled out, do not lead to successes, leaving us with 3,917 of the total 4,500 trajectories. Additionally, we emphasize that this is the only demonstration data we train upon. As it is drawn from the default episode distribution, that makes our challenge splits (described in more detail below) a strict distribution shift from the training data, thereby testing our policies' generalization.

In evaluating different policies, we use the same episode random seeds for each one, thereby ensuring that they are tested on the same set of initial conditions. Our evaluation code builds upon that of Belkhale and Sadigh [31], which starts episodes with 10 no-op steps (to allow spawned objects to

```
turn on the stove and put the frying pan on it
pick up the alphabet soup and put it in the basket
pick up the cream cheese box and put it in the basket
pick up the ketchup and put it in the basket
pick up the tomato sauce and put it in the basket
pick up the alphabet soup and put it in the basket
pick up the butter and put it in the basket
pick up the milk and put it in the basket
pick up the orange juice and put it in the basket
pick up the tomato sauce and put it in the basket
pick up the alphabet soup and put it in the tray
pick up the butter and put it in the tray
pick up the cream cheese and put it in the tray
pick up the ketchup and put it in the tray
pick up the tomato sauce and put it in the tray
pick up the black bowl on the left and put it in the tray
pick up the chocolate pudding and put it in the tray
pick up the salad dressing and put it in the tray
stack the left bowl on the right bowl and place them in the tray
stack the right bowl on the left bowl and place them in the tray
put the red mug on the left plate
put the red mug on the right plate
put the white mug on the left plate
put the yellow and white mug on the right plate
put the chocolate pudding to the left of the plate
put the chocolate pudding to the right of the plate
put the red mug on the plate
put the white mug on the plate
pick up the book and place it in the front compartment of the caddy
pick up the book and place it in the left compartment of the caddy
pick up the book and place it in the right compartment of the caddy
pick up the yellow and white mug and place it to the right of the caddy
pick up the book and place it in the back compartment of the caddy
pick up the book and place it in the front compartment of the caddy
pick up the book and place it in the left compartment of the caddy
pick up the book and place it in the right compartment of the caddy
pick up the book and place it in the front compartment of the caddy
pick up the book and place it in the left compartment of the caddy
pick up the book and place it in the right compartment of the caddy
pick up the red mug and place it to the right of the caddy
pick up the white mug and place it to the right of the caddy
pick up the book in the middle and place it on the cabinet shelf
pick up the book on the left and place it on top of the shelf
pick up the book on the right and place it on the cabinet shelf
pick up the book on the right and place it under the cabinet shelf
```

Figure 10: Latter half of the 90 task prompts for LIBERO-90. Note some are repeated in different scenes (and thus they appear multiple times in this list)

settle) and ends them in failure after 400 more steps have passed without succeeding. The observation and action spaces for this environment are the standard ones: the former just contains RGB images from a fixed, third-person camera, while the latter is a 7D vector (6 joint angles and one for gripper open/close). The environment also provides proprioception, depth images, and wrist camera views as part of the observation space. While these have been shown to improve performance (including for VLAs [31]), we do not use them, as our goal is to determine the impacts of robot reasoning on VLA performance, *not* to optimize our systems for LIBERO in particular. We emphasize that this is a design choice, and not a core limitation of any of our approaches. We provide the full list of 90 tasks in Fig. 9 and Fig. 10.

B.2 Custom Challenge Split Details

We develop a custom challenge split for the existing Libero-90 environments in order to test policy generalization. We introduce two types of randomization: *Perturbation* and *Distractors*. Perturbations involve expanding the initial state distribution in which task-relevant objects are randomized at the beginning of each episode. Task specifications for the standard Libero-90 tasks specify a rectangular initial distribution for each object. We generate the regions for the perturbed environments by expanding the width and height of these regions by a factor of $1.2\times$ for each task-relevant object. Additionally, we mandate that at least one task-relevant object (e.g., the black bowl in the *put the black bowl on top of the cabinet*) exists in the expanded portion of its initial region in order to test spatial generalization of the policy. As a further test of generalization, we introduce distractor objects into tasks. Specifically, we sample 1–2 random objects from the Libero object suite (excluding objects that are already present in the scene as well as large objects such as *microwave*) and randomly place

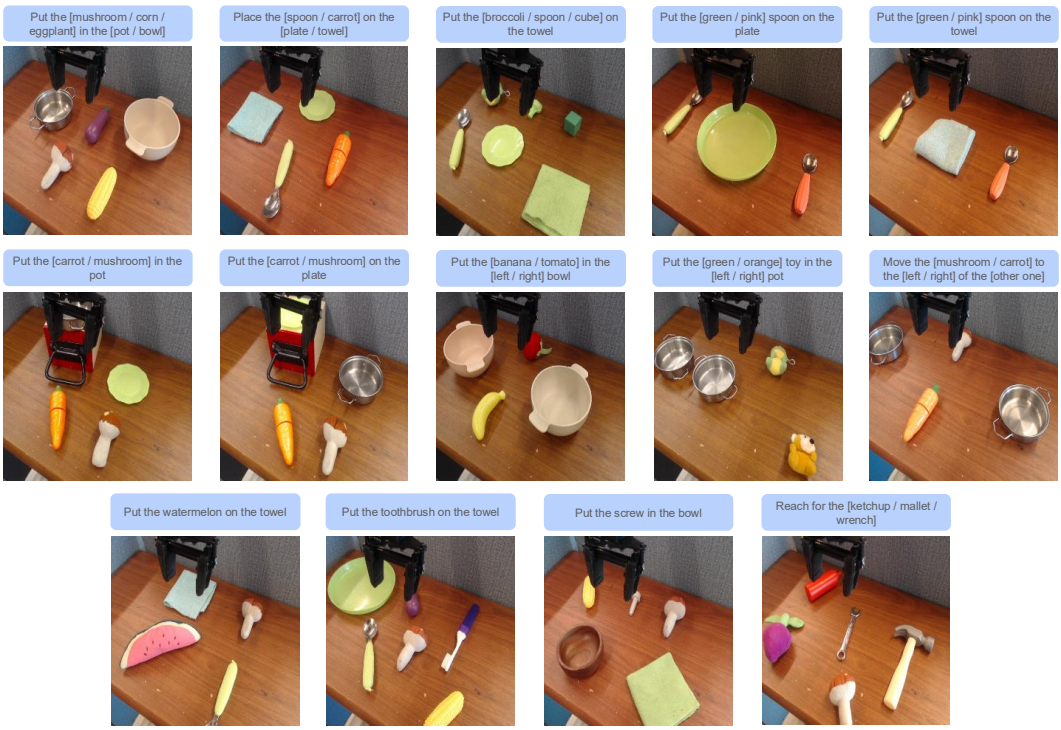

Figure 11: Full list of task prompts used for our Bridge evaluation, as well as example starting states. We test all combinations of the words in brackets for each scene.

these in the scene in regions that do not collide with existing objects. We conduct evaluations under *Perturbation* as well as the combination of *Perturbation + Distractors*.

B.3 Bridge Description

Our real-world robot evaluations are done on a BridgeData WidowX station [1, 7, 8], which is a popular testbed for language-conditioned policies [3, 6], including the original ECoT [14]. The observation space contains RGB images from a fixed, third-person camera, while the action space is a 7D vector (six elements containing delta Cartesian position and rotation, one for gripper open/close). We list all tasks in our Bridge evaluations in Fig. 11. All permutations of the bracketed words are tested for a particular scene (and, indeed, all scenes have more than one possible task in them, making them better tests of language conditioning [70]).

C Embodied Reasoning Datasets

We train all embodied reasoning policies using the same set of reasoning annotations: following prior work [14], we train policies to predict natural language subtasks and explanations, object bounding boxes, the end-effector location, and primitive movements ("move left", "move up") and explanations. While we stick to this particular choice of reasoning steps in this work, since it has been shown to work well in prior work [14], we emphasize that the choice of reasoning steps is *not* a contribution of our work, and we expect the recipes we develop to transfer to other reasoning choices that have been explored in the literature [45, 46, 54]. For BridgeData V2 we directly use the embodied reasoning annotations released by Zawalski et al. [14]. For Libero, we generate reasoning annotations for the aforementioned steps using the simulator state (for bounding boxes, gripper locations, and movements), and a LLM-based synthetic data generation pipeline akin to that of Zawalski et al. [14] (for plans, subtasks, and subtask or motion reasonings).

C.1 LIBERO-90 Embodied Chain-of-Thought Pipeline and Dataset Details

We now describe how we reproduced the ECoT data and policy pipeline for LIBERO-90.

As with the original Bridge pipeline, we extract the following reasoning features: a list of high-level subtasks for accomplishing the overall task (**PLAN**), a natural language rationale for which subtask it should currently follow (**SUBTASK REASON, SUBTASK**, an equivalent rationale for what low-level language motion (move left, close gripper, etc) to execute (**MOVE REASON, MOVE**), object bounding boxes (**OBJECTS**), and gripper pixel coordinates (**GRIPPER**).

These are roughly what were used in the original ECoT work, with some small changes to how they were extracted to better suite LIBERO. First, we exclude the TASK step used by Zawalski et al. [14]; as there are significantly fewer tasks in LIBERO-90 than Bridge and the evaluation instructions are always ones that are seen during training, we expect that this step is superfluous. Next, to extract object bounding boxes and gripper positions, we use the ground truth segmentation masks and object names provided by the simulator (bounding boxes are simply the maximum and minimum height and width coordinates of each mask, while the gripper position is the center of its bounding box). Finally, we use Molmo and Llama2 to come up with and produce subtasks, subtask reasons, and move reasons [76, 77]. We note these pipeline differences from the original one by Zawalski et al. [14] result in the stylistic disparities between LIBERO and Bridge reasonings (e.g., Molmo's synthetic reasonings are much ore verbose). See Fig. 12, Fig. 13, Fig. 14, and Fig. 15 for the prompts used in our LIBERO reasoning data generation pipeline. We run this pipeline on all 3,917 trajectories in our demonstration dataset, generating labels for roughly 90% of them.

D Policy Architectures and Training Hyperparameters

All our policy-training experiments use OpenVLA and MiniVLA [6, 31]. The former, used for our real-robot experiments, is trained to produce discretized 7D robot manipulator actions (representing changes in Cartesian position, rotation, and gripper open/close state), expressed by binning each dimension uniformly into 256 bins, as also done by Brohan et al. [5]. The base VLM is a 7B parameter PrismaticVLM [21], which uses DINOv2 and SigLIP [78, 79] as visual encoders and Llama2 [77] as its language backbone. The latter, used in our large-scale simulation experiments, instead discretizes an action chunk [80] of horizon length 10 into seven tokens by passing it through a learned VQ-VAE [81] with codebook size of 256 (see Appendix D.1 for details). At inference time, the VLA likewise autoregressively generates seven action tokens, but the VQ-VAE then decodes this into a full action chunk of some specified size. MiniVLA uses a smaller base VLM that swaps out the Llama2 backbone with Qwen2.5, making the full VLA 1B parameters [82].

Other variants of the core VLA recipe exist, such as generating actions with a diffusion head or with parallelized action embedding tokens [10, 75]. In principle, these can also be used in reasoning VLAs, but we opt to use the discretization schemes used by Kim et al. [6], Brohan et al. [5], and Belkhale and Sadigh [31] for simplicity.

D.1 VQ-VAE Action Chunking

The MiniVLA codebase provides support for training policies with VQ-VAE [81] tokenization of actions. We use that code to train the tokenizer to convert ten 7D actions (so 70 elements total) into seven discrete tokens, drawn from a codebook of size 256. These tokens are then decoded into a reconstruction of the original action chunk, so the VQ-VAE effectively provides a compressed, discretized representation of the action sequence.

These seven tokens are what our MiniVLA policies predict. At inference time, after generating these tokens, they are passed through the decoder of the VQ-VAE to convert them back to a 10-step action chunk. We note that such action chunking was shown by Belkhale and Sadigh [31] in the original MiniVLA LIBERO comparisons to be more effective than standard one-step action generation. However, they only consider it as a representation learning approach, as they run the action chunk

"closed loop" (generate multiple actions, execute the first one, then generate again). However, we find that executing all ten such actions before re-querying the VLA to work better in all cases (while reducing the number of VLA queries by ten times!), so all LIBERO experiments are run as such.

D.2 Training Hyperparameters

Following Zawalski et al. [14], we mainly use the default hyperparameters for MiniVLA and OpenVLA when training reasoning policies using their respective codebases [6, 31]. The LIBERO MiniVLA policies are trained for 200k gradient steps with batch size 128, which is approximately how long it takes for the VQ-VAE action chunking policy to reach 85% teacher-forced action token accuracy. Some policies we tested were trained with 100k steps and achieved near-identical performance (there was no observed degradation from overtraining), but for fairness, all reported LIBERO policies use 200k steps. The Bridge OpenVLA policies are trained for 80k gradient steps with batch size 256[3], approximately to 95% action token accuracy[4]. All policies were trained on 8x A100s, except the co-trained LIBERO policy, which was trained on 8x H200s (the higher GPU memory is needed to fit the doubled batch size, as described in Appendix E.2). We used the default hyperparameters provided by the MiniVLA codebase for training the horizon length 10 VQ-VAE action chunk tokenizer, using the final checkpoint (after 200 epochs).

E Experiment and ECoT-Lite Policy Training Details

E.1 Reasoning Pre-training

In reasoning pre-training, the policy takes in the observation and task language and maps it to the corresponding embodied reasoning, trained using the standard VLM NLL objective. Implementation-wise, this simply involves removing the action tokens from the end of each datapoint in the ECoT dataloader. See left part of (b) in Fig. 3. In action fine-tuning, the checkpoint from the end of pre-training is loaded and tuned with the standard VLA objective. See right part of (b) in Fig. 3. Note that, by doing these two learning objectives in sequence, we cannot train for the same number of total gradient steps as other approaches while also keeping the number of steps for reasonings and actions individually identical. In LIBERO, we do 100k steps of reasoning pre-training, then 200k steps of action-tuning. In Bridge, we do 20k steps of reasoning and 80k of actions.

E.2 Reasoning Co-training

In reasoning co-training, the policy learns to map images and task language to reasonings and actions simultaneously (albeit in different training data points, i.e., actions cannot attend to reasonings, nor vice-versa). These are the same two objectives as used in reasoning pre-training and action fine-tuning, just at the same time – each batch sample has an equal probability of being for action and reasoning prediction. In this setting, we train for 200k steps (as done for all other LIBERO policies, minus reasoning pre-training), but double the batch size to 256. That way, the policy is trained on the same amount of reasoning and action data as our ECoT policy (wherein each batch sample contains both reasonings and actions, rather than one or the other).

E.3 Reasoning Dropout

In this case, for each batch sample with a reasoning annotation, we randomly and uniformly choose a number from zero to the total number of reasoning steps and drop out that many steps. The remaining reasoning steps (if any) are then put together into the reasoning, and then prepended to the action. Losses are assigned to both the action tokens and reasonings (if present). Thus, some fraction of

[3]Note the higher batch size. We suspect that MiniVLA's Qwen backbone may have an inefficient sharding implementation, as we were unable to fit 256 samples per batch, despite said LLM being smaller.

[4]This is higher than the MiniVLA threshold since it does NOT do action chunking, so it is easier to achieve high accuracy.

training points lack reasonings, meaning that the policy is trained to sometimes map observations and tasks directly to actions (without reasonings), allowing it to behave like a standard VLA, simply by starting its generations at the start of action indicator (which, in our case, is just the string "ACTION:", expressed as tokens). Note that for identical batch size and gradient step counts, policies trained with reasoning dropout inherently are exposed to less reasoning data than if all samples had "full" reasonings. To control for this, *all our policies trained on reasoning data use reasoning dropout during training*, meaning we are controlling for the amount of reasoning data they are trained on – the only caveat being that, for reasoning pre-training and co-training, we do not allow it to drop out the full reasoning during training (as these approaches involve learning to generate the reasonings with no actions later in the sequence, that would leave some batch samples with no loss assigned to any tokens).

Conversely, this also means that it can act as an ECoT reasoning policy as well by starting its generations from the start of reasoning indicator – the full ECoT and reasoning dropout policies *are the same policy*. Other than adding test-time dropout of the reasonings, the model's inputs, outputs, and hyperparameters are identical to ECoT – note that the illustrations in Fig. 3 for ECoT in (a) and reasoning dropout in (c) are identical, with the exception that the reasonings are disabled during testing (whereas full ECoT is forced to generate *all reasoning steps* during inference).

Notably, we find that a significant portion of Bridge transitions lack reasonings in the annotation dataset released by Zawalski et al. [14]. Thus, we find that their released policy can actually be run with reasonings dropped out at test-time. This is therefore the policy we use in the Bridge comparison.

E.4 Reasoning Scaffolding

This policy is trained the same way as reasoning dropout, except that no loss is assigned to reasoning tokens (that is, it does not learn to generate the reasonings, only accept them as part of the context). Critically, it also uses the random reasoning dropout described above so the policy is able to map images (without in-context reasonings) to actions. All hyperparameters are thus the same as dropout. Note how (c) and (d) in Fig. 3 are identical, except the reasonings are listed as an input in the latter.

E.5 Thinking Token Policies

We now provide details for our "thinking token" approach, inspired by Pfau et al. [64] and Goyal et al. [65].

E.5.1 Theory of Thinking Tokens

This approach processes some number of filler thinking tokens before producing an answer, thereby increasing the expressivity by the Transformer (as all those tokens now can support additional computations). Critically though, this does *not* improve the complexity class of the Transformer, which is known to be TC^0; it only increases the expressivity of the fixed-size network within that class [64, 83]. This is in contrast to true chain-of-thought reasoning, wherein a Transformer autoregressively generates different tokens. As discussed by Feng et al. [42], in that case, by generating a suitably long and appropriate sequence of tokens, Transformers can express computations in class NC^1, whereas such a computation cannot be solved by querying the Transformer to generate the answer directly (without intermediate computations)[5].

Intuitively, being able to autoregressively generate a reasoning consisting of appropriate different tokens allows Transformers to "lock in" intermediate computations for all subsequent ones to attend to. Feng et al. [42] discusses the case of integer arithmetic, wherein the correct intermediate steps are to encode long addition (adding corresponding digits of the numbers from right to left, carrying overflows to the next digit) within the CoT.

In contrast, uniform (or non-uniform but low-complexity) sequences of thinking tokens do NOT enjoy this benefit. Since each thinking token is uniform, they do not "lock in" the computations of

[5] Assuming TC^0 cannot express NC^1 [84].

```
I am trying to give a robotic arm a short list of high-level steps to complete the task: '{instruction}.' The robotic arm
is in a scene containing the following {number of objects} objects:
{list of object names}

Break down this instruction into a Python list of high-level steps. These steps should describe the broad physical actions
the robot should take to accomplish the task, not any perceptual steps. For example, for the task 'Place the mushroom in
the pot,' a possible list of tasks could be: ['move to the mushroom', 'grasp the mushroom', 'move the mushroom to the pot',
'release the mushroom into the pot']
```

Figure 12: The Llama2 prompt for generating a plan (list of subtasks) from an instruction.

```
I want to analyze each step my robot took to accomplish the task: '{instruction}.' To accomplish this task, it takes these
subtasks in this order:
{numbered list of subtasks}
Here is a mapping from timestep to the motion that my robot took:
{list of time step indices to language motions, deduplicated}

Please provide me with a Python dictionary mapping from each of the {number of subtasks} subtasks (as strings) to the
integer timestep when that subtask begins. Remember that subtasks are in order, so later subtasks must map to later
timesteps. The first subtask should always start at step 0.
```

Figure 13: The Llama2 prompt for aligning the generated subtasks to specific time steps.

previous tokens; their initial embeddings do not depend on any past computations, whereas in CoT, since that token was generated and fed back into the Transformer autoregressively, this is not the case.

Notably, it is unclear to what extent this argument holds for the case of robot reasoning, or other problems wherein training the CoT to exactly represent the sequential computations needed to solve a problem is intractable. Thus, while we control for number of tokens to induce similar expressivity in our thinking token and ECoT policies, we note that they are not perfectly identical.

E.5.2 Practical Implementation Details of Thinking Token Policies

To train a thinking token policy, we replace the reasonings in ECoT with a similar number of thinking tokens. Following the practice of using rarely-used tokens to symbolize model actions in VLAs [6], we choose the " ." (space followed by a period) token as our dedicated thinking token, as it never appears in our training data and does not intuitively have any semantic meaning related to our robot tasks. This is a design choice – we can alternatively add a brand-new extra thinking token, as done by Goyal et al. [65].

Implementation wise, for each datapoint, we uniformly sample a number from 50-350, then include that many of the thinking tokens as part of the prompt (i.e., no loss is assigned to them). This is therefore functionally identical to the standard VLA training process, just with the input prompts altered with appended thinking tokens – note the similarity between the standard VLA in (a) and the thinking token policy in (e) in Fig. 3. Note that the VLAs we use have decoder-only LLM backbones, meaning prompt and generated tokens are treated identically (only differing in which ones have losses assigned to them during training). This is not the case for some VLA architectures, like those based on PaliGemma [22, 23], which applies non-causal attention to prompt tokens only.

At test time, we append varying number of thinking tokens to the prompt, followed by the start of action prefix. The policy therefore only generates the seven action tokens, but gets to attend to the representations of both the policy inputs and all intermediate thinking tokens (thereby expanding its expressivity).

```
The robot's task is: '{instruction}.' The robot's plan for completing this instruction is:
{numbered list of subtasks}

Based on the position of objects, the robot's pose relative to those objects, the layout of the scene, explain why the

robot's current subtask is: {current subtask}. Answer concisely in a single sentence.
```

Figure 14: The Molmo prompt for generating subtask reasonings for each subtask generated via Fig. 12. The VLM also accepts the current observation image.

```
I want you to explain a robot's actions to me. The robot's task is: '{instruction}.' The robot's plan for completing this
instruction is:
{numbered list of subtasks}
The robot's current subtask in achieving that task is: {subtask}.
The robot just took the following {number of past deduped language motion actions} actions:
{numbered list of language motion actions}
Based on the position of objects, the robot's pose relative to those objects, the layout of the scene, why should the robot
'{current language motion}' in order to complete that subtask in this situation?

Answer with something like: '<some reason>, thus the robot should <some motion>'. Answer specifically, but concisely in

one sentence.
```

Figure 15: The Molmo prompt for generating movement reasonings. The VLM also accepts the current observation image.