Revisiting Embodied Chain-of-Thought for Generalizable Robot Manipulation

Nan Sun[1,2] **Yuan Zhang**[2,†,⋆] **Yongkun Yang**[1] **Wentao Zhao**[1] **Peiyan Li**[2,3]
Jun Guo[1,2] **Wenxuan Song**[2,4] **Pengxiang Ding**[2,5] **Runze Suo**[2,6,8]
Yifei Su[2] **Xin Xiao**[7] **Xinghang Li**[2] **Huaping Liu**[1,⋆]

[1]Department of Computer Science and Technology, BNRist, Institute for Embodied Intelligence and Robotics, Tsinghua University [2]Xiaomi Robotics [3]CASIA
[4]HKUST(GZ) [5]Zhejiang University [6]Fudan University [7]Wuhan University
[8]Shanghai Innovation Institute [†]Project Lead [⋆]Corresponding Author

Abstract

Embodied chain-of-thought (CoT) aims to bridge linguistic reasoning with robotic control, yet its effective form and integration remain underexplored. In this paper, we revisit embodied CoT for robotic control at an unprecedented scale. We curate the largest embodied CoT corpus to date, comprising 978,743 trajectories, 226.3M samples, and 2592.5 hours of data. Through extensive experiments, we show that effective CoT must ground high-level semantic understaning in concrete linguistic action guidance – such as end-effector movement and image-space trajectories – whereas high-level reasoning alone yields marginal gains. More importantly, we identify that explicit CoT does not scale reliably as an autoregressive action prefix, suffering from compounding errors during inference. To address these challenges, we propose ERVLA, a vision-language-action (VLA) model that effectively leverages linguistic reasoning in generalizable robot manipulation. ERVLA is trained using a CoT-dropout strategy, allowing the model to leverage rich reasoning traces during training while predicting actions directly without CoT during inference to bypass autoregressive instability. This approach enables reliable scaling with increasing pre-training data. ERVLA achieves state-of-the-art results on LIBERO-Plus with an 86.9% success rate and reaches 53.2% on VLABench, showcasing superior performance in out-of-distribution settings. Furthermore, ERVLA outperforms competitive state-of-the-art baselines in real-robot experiments, especially in handling semantic ambiguity and long-horizon tasks. Code, data, and model checkpoints are available on our project website.

1 Introduction

Vision-language-action (VLA) models have emerged as a promising paradigm for generalizable robot control by transferring large-scale visual and semantic priors from pre-trained vision-language models (VLMs) into embodied decision making [6, 20, 23, 31, 68]. By grounding language instructions in visual observations and mapping them to robot actions, these models provide a unified interface for open-vocabulary and long-horizon manipulation. However, enhanced perception and broader semantic coverage do not inherently guarantee better action generation [29, 60]. In practice, VLA models often struggles in complex, out-of-distributed scenarios which demands deliberation prior to execution. This bottleneck has motivated the development of reason-before-act policies, which leverages intermediate reasoning to guide action generation. Previous embodied chain-of-thought (ECoT) methods [58] showcase that supervising explicit reasoning traces via next-token prediction can improve robustness and generalization. Since then, the design space has rapidly expanded, encompassing diverse reasoning modalities, architectural designs for reasoning-action coupling, and specialized training recipes [10][47].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

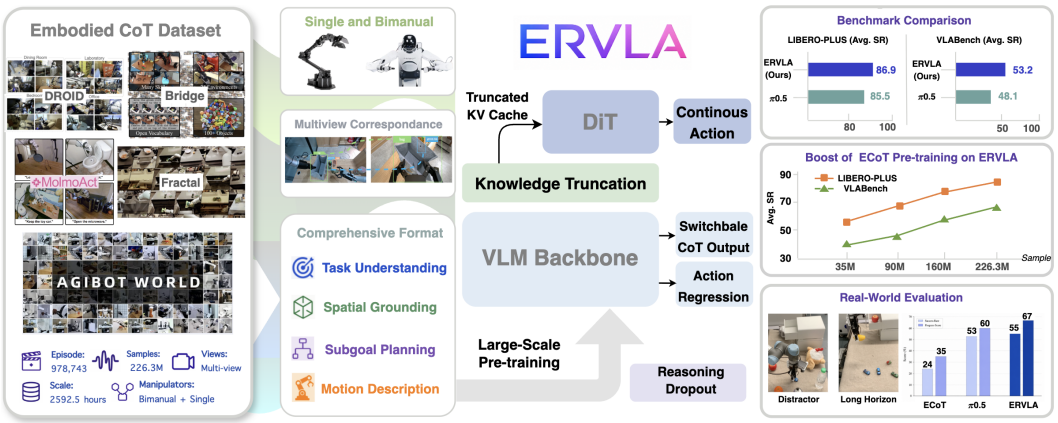

Figure 1: **Overview of ERVLA.** We build the largest and most comprehensive embodied CoT corpus to date, which serves as the reasoning pre-training foundation for ERVLA. ERVLA uses CoT supervision, auxiliary action queries, knowledge-truncated KV conditioning, and reasoning dropout to internalize reasoning into action-aware VLM representations and predict continuous actions without mandatory test-time CoT. It achieves state-of-the-art performance on two simulation benchmarks, benefits from more embodied CoT data, and shows stronger real-world robustness under semantic ambiguity and long-horizon tasks.

Despite the progress, **a systematic understanding of reasoning in VLA remains elusive.** The essence of ECoT is not to make robots "say more," but to translate the semantic abstractions learned by VLMs from internet-scale data into intermediate representations that are genuinely useful for action generation. Existing methods instantiate reasoning through various lenses, including scene understanding, subtask decomposition, spatial grounding, end-effector trajectory prediction, and future-frame prediction [25, 58, 46, 62, 20]. However, these choices are tightly coupled with specific architectures and training objectives, making it difficult to identify what actually improves control. Consequently, the field lacks a principled understanding of which forms of embodied reasoning truly facilitate effective action generation.

A second unresolved question is how should reasoning interact with the policy. Early ECoT-style formulations [58, 46] model reasoning as a prefix of action generation: the model first predicts embodied reasoning traces and then generates discretized action tokens through next-token prediction. This design is simple but slow and brittle, since long reasoning traces increase latency and action generation depends heavily on preceding reasoning, leading to compounding errors during inference. Recent works decouple reasoning from action by distilling reasoning into latent plans [36, 19], conditioning diffusion-based action heads on selected reasoning states [18, 48], or learning from reasoning data during training while predicting actions without CoT at inference time [10]. This raises another central question: should reasoning be taken as a visible trace, a latent variable, or a training signal that reshapes the policy representation?

A third but equally important question is whether embodied CoT can effectively scale. Recent VLA pre-training efforts have begun to include richer semantic or reasoning-related supervision [20, 48]. However, the scaling behavior of CoT data remains unclear. Public reasoning-enhanced robotic datasets are scarce, given that embodied CoT is expensive to annotate and often requires dense labeling of multiple perception tasks within a large amount of images. Consequently, a large-scale embodied CoT dataset and a systematic scaling study is essential to determine whether scaling reasoning supervision yields stronger action generation.

In this paper, we revisit embodied CoT for robot control on an unprecedented scale, with a focus on *linguistic CoT* (see Figure. 1). We first establishes a comprehensive CoT format and an auto-labeling pipeline, extending prior single-view reasoning to multi-view settings. Based on this pipeline, we curate the largest embodied CoT robot dataset to date, consisting of 978,743 trajectories, 226.3M samples, and 2592.5 hours of data (see Figure. 2). We perform extensive experiments on this dataset in a unified setting to compare the effectiveness of various CoT signals.

We identifies a critical yet overlooked failure mode that we term **CoT contamination**. Noisy labels are inevitable in large-scale auto-labeled CoT annotations. These dense but inaccurate supervision, es-

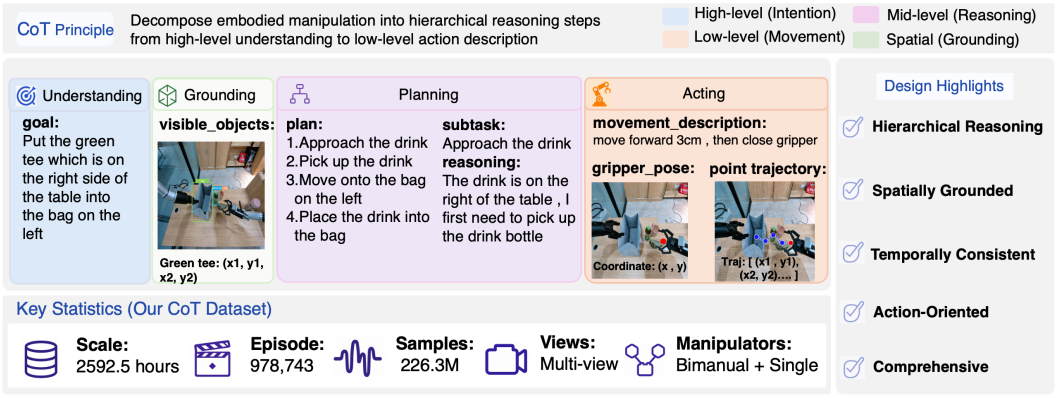

Figure 2: **Embodied CoT data format and statistics.** We introduce a hierarchical embodied CoT schema that decomposes robot manipulation into task understanding, planning, spatial grounding, and action-level descriptions. Each sample connects language reasoning with multi-view observations and executable cues. Using this schema, we construct a large-scale public embodied CoT dataset covering 2,592.5 hours, 978,743 episodes, and 226.3M samples.

pecially highly-jitterred signals such as unstable bounding boxes and drifting end-effector coordinates across consecutive similar frames, can hinder adaptation to embodied control by imposing inconsistent supervision on semantically similar observations. We show that this issue can be mitigated through improved annotation pipelines, sparse supervision for unstable grounding tasks, and properly designed reasoning dropout. Importantly, reasoning dropout is not merely a regularizer, it acts as training scaffolding that reduces noisy CoT contamination and reshapes internal VLA representations. This leads to a central conclusion: **embodied CoT should be treated not merely as a test-time verbalization channel, but as a training signal that reshapes the representation space from perception to action.**

Building on these insights, we present **ERVLA**, an **E**mbodied **R**easoning **V**ision-**L**anguage-**A**ction model trained on a large-scale robot dataset with embodied CoT. ERVLA leverages a mixture-of-transformers architecture [6, 20, 48], where a vision-language model [2] processes visual observations and text instructions, and a diffusion transformer (DiT) [40] generates continuous robot actions via flow matching [32]. The DiT is conditioned on the robot state and the key-value (KV) cache from the VLM, allowing action generation to directly access semantic representations produced by the reasoning model. Drawing inspiration from [8, 20], we train the VLM with an auxiliary action-regression loss through query tokens, so that the backbone is guided not only to produce textual reasoning but also to understand action-relevant semantics. This auxiliary supervision stabilizes training, accelerates convergence, and encourages the VLM representation to align high-level reasoning with low-level action generation. We empirically found that it is important to truncate the KV cache of these tokens in the condition of the flow matching to avoid shortcut copying.

Besides action generation, we supervise the VLM backbone with next-token prediction over embodied CoT traces, bridging semantic reasoning and robotic control. This supervision serves to reshape the representation space of the VLM for action generation. During training, we apply random dropout to the reasoning supervision, utilizing an explicit toggle signal to indicate whether to activate reasoning or not as in large language models [2]. This not only allows the model to generate actions without reasoning at inference time, but also circumvents the CoT contamination problem described above.

We perform extensive experiments on both simulation benchmarks and real-robot platforms. Despite trained on LIBERO data [33], ERVLA achieves a state-of-the-art performance of an 86.9% average success on LIBERO-Plus [15] and reaches 100% success on the background and lighting variations of its Spatial track, demonstrating strong zero-shot generalization capabilities. In VLABench [61], ERVLA achieves an average success rate of 53.2%, showcasing superior performance in challenging out-of-distribution settings demanding semantic understanding and instruction following. In real-robot experiments, ERVLA surpasses competitive state-of-the-art baselines, especially in tackling tasks with semantic ambiguity and long-horizon tasks. The full dataset, model checkpoints, and code are available on our project website.

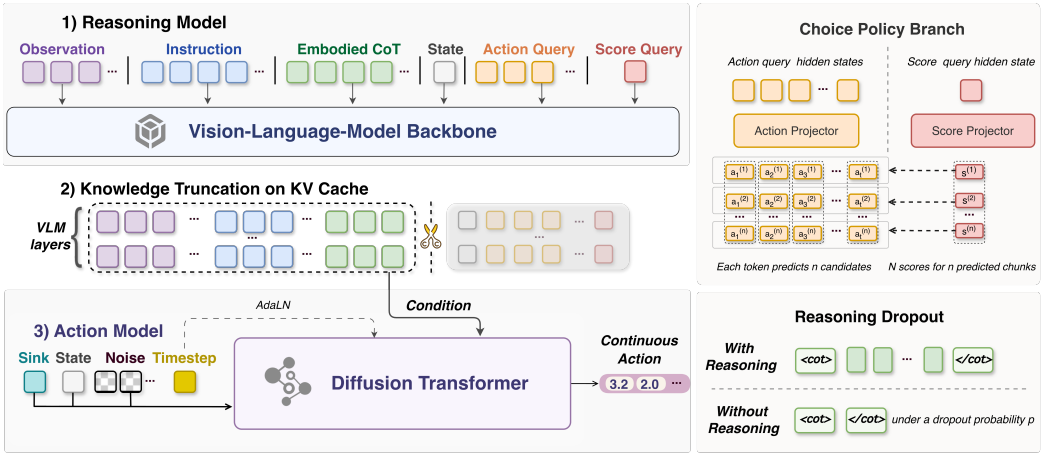

Figure 3: **Demonstration of ERVLA architecture.** ERVLA integrates explicit embodied CoT supervision into the VLM backbone, uses auxiliary action-query regression to align semantic reasoning with action, applies knowledge truncation so that the action model attends only to the semantic-prefix KV cache, and generates continuous actions through a diffusion transformer with flow-matching.

2 Methodology

We start by systematically reviewing and organizing existing linguistic embodied CoT, grouping its major forms and defining the key fields in Sec. 2.1. A detailed review is provided in Appendix A. Second, we introduce ERVLA and study how embodied reasoning should guide action generation in modern VLA architectures built on VLM backbones and diffusion-based action heads in Sec. 2.2. Third, we present the training recipe for building a strong reasoning-aware VLA model in Sec. 2.3.

2.1 Embodied CoT Data Construction

We construct a large-scale embodied CoT corpus as the reasoning pre-training source for ERVLA. Rather than treating CoT as free-form explanations attached to trajectories, our pipeline decomposes embodied reasoning into structured categories that correspond to different roles in control: task understanding, spatial grounding, subgoal planning, and action-oriented motion description. This design is intended to make reasoning supervision both interpretable and learnable: high-level fields provide semantic intent, grounded fields align language with visual entities, planning fields expose task progress, and acting fields connect reasoning to executable motion (see Figure. 2). The corpus is built on open VLA datasets including Bridge [51], Fractal [68], Droid [21], MolmoAct [26], and AgiBot [1], covering both single-arm and dual-arm manipulation, as well as single-view and multi-view observations. In this way, the dataset provides not only language supervision, but a scalable interface for studying how semantic, spatial, planning, and action-level reasoning jointly shape model representations. Detailed annotation procedures, filtering rules, and dataset construction pipelines are provided in Appendix B.

2.2 ERVLA Architecture

With the structured ECoT corpus established, we next present the ERVLA architecture that integrates reasoning pre-training with end-to-end action generation. ERVLA contains a vision-language reasoning module and a diffusion-based action module as depicted in Figure. 3. We instantiate the reasoning backbone with Qwen3-VL-4B [2] and keep explicit embodied CoT supervision in the standard language-modeling form. Given images I, instruction x, optional CoT c, state s, action-query tokens $\{a_i\}$, and a score-query token a_{score}, the VLM outputs hidden states and per-layer key-value caches:

$$\mathbf{H}^{\text{vlm}}, \{(\mathbf{K}_\ell^{\text{vlm}}, \mathbf{V}_\ell^{\text{vlm}})\}_{\ell=1}^{L} = f_{\text{vlm}}(I, x, c, s, \{a_i\}, a_{\text{score}}). \tag{1}$$

The input includes trainable control interfaces: `<state>` token is replaced by projected state embedding, `<a_i>` by action-query embeddings, and `<score>` by a score-query embedding. Unlike

ThinkAct-style designs [18], which mainly use the VLM for planning while modeling actions outside the backbone, ERVLA injects action-sensitive supervision directly into the VLM. We also avoid compressed action tokenization such as FAST [41], which introduces a separate action-token space and may cause discretization error. Instead, ERVLA preserves CoT in the native language space and uses an auxiliary *choice policy* [42] branch to predict N candidate action chunks and their scores:

$$\hat{\mathbf{a}}_t^{(n)} = g_{\text{act}}(\mathbf{H}_a)_{t,n}, \qquad t = 1, \dots, T, \;\; n = 1, \dots, N, \qquad \hat{\mathbf{r}} = g_{\text{score}}(\mathbf{H}_s). \tag{2}$$

Here, $\mathbf{H}_a$ and $\mathbf{H}_s$ denote hidden states at the `<a_i>` and `<score>` positions. Each `<a_i>` corresponds to one temporal slot, where the action head predicts N vectors $\hat{\mathbf{a}}_t^{(n)} \in \mathbb{R}^D$. Grouping vectors with the same candidate index forms N continuous chunks, while $\hat{\mathbf{r}} \in \mathbb{R}^N$ scores them. The final action is generated by a DiT-style diffusion policy rather than next-token prediction:

$$\hat{\mathbf{A}}^{(n)} = [\hat{\mathbf{a}}_1^{(n)}, \dots, \hat{\mathbf{a}}_T^{(n)}] \in \mathbb{R}^{T \times D}, \qquad \hat{\mathbf{v}}_\theta = f_{\text{dit}}(\mathbf{z}_\tau, \tau, s \mid \{(\mathbf{K}_\ell^{\text{vlm}}, \mathbf{V}_\ell^{\text{vlm}})\}). \tag{3}$$

Here, $\mathbf{z}_\tau$ is the noisy action, and the DiT input consists of a sink token, projected state tokens, and projected noisy action tokens. The diffusion timestep τ is encoded by an MLP and injected into each DiT block through AdaLN conditioning [40]. Rather than using pooled VLM features, DiT directly attends to the VLM's per-layer key-value cache.

We further introduce *knowledge truncation*: although the VLM receives both the semantic prefix and appended control-query turns during training, DiT conditions only on the semantic-prefix cache:

$$\text{Attn}(\mathbf{Q}, [\mathbf{K}_\ell^{\text{KT}}; \mathbf{K}_\ell^{\text{dit}}], [\mathbf{V}_\ell^{\text{KT}}; \mathbf{V}_\ell^{\text{dit}}]), \quad \{(\mathbf{K}_\ell^{\text{KT}}, \mathbf{V}_\ell^{\text{KT}})\} = \text{SlicePrefix}\big(\{(\mathbf{K}_\ell^{\text{vlm}}, \mathbf{V}_\ell^{\text{vlm}})\}, m_{\text{cond}}\big). \tag{4}$$

where m_{cond} excludes appended state and action-query turns. In this way, DiT reads only the semantic memory before the control-query tokens, preventing it from exploiting synthetic query shortcuts and reducing interference from training-specific token patterns.

2.3 Training Recipe

Unlike knowledge-insulated VLA training [13], which blocks gradients from the continuous action expert to the VLM backbone, ERVLA enables the flow-matching loss from the DiT action head to back-propagate into the VLM backbone, making the whole reasoning-action system fully end-to-end trainable. Combining explicit CoT verbalization, reasoning dropout, choice policy prediction, and knowledge truncation, ERVLA enables action-sensitive VLM representations, cleaner semantic memory for DiT, and natural support for test-time reasoning dropout, where CoT can be partially used, sparsely refreshed, or omitted.

We optimize ERVLA with the objective $\mathcal{L} = \lambda_{\text{vlm}}\mathcal{L}_{\text{vlm}} + \lambda_{\text{flow}}\mathcal{L}_{\text{flow}} + \lambda_{\text{choice}}\mathcal{L}_{\text{choice}} + \lambda_{\text{score}}\mathcal{L}_{\text{score}}$. Here, $\mathcal{L}_{\text{vlm}}$ is the token-level CoT cross-entropy loss, and $\mathcal{L}_{\text{flow}}$ is the rectified flow loss for continuous actions. For the choice policy branch, the model predicts N candidate chunks and supervises the best one under mean absolute error:

$$\mathcal{L}_{\text{choice}} = \frac{1}{B}\sum_{b=1}^{B} \min_n d_b^{(n)}, \qquad d_b^{(n)} = \frac{1}{T_b D}\left\|\hat{\mathbf{A}}_b^{(n)} - \mathbf{A}_b^*\right\|_1. \tag{5}$$

The score branch predicts these candidate-wise errors:

$$\mathcal{L}_{\text{score}} = \frac{1}{B}\sum_{b=1}^{B}\left\|\hat{\mathbf{r}}_b - \text{sg}\Big([d_b^{(1)}, \dots, d_b^{(N)}]\Big)\right\|_2^2, \tag{6}$$

where $\hat{\mathbf{r}}_b \in \mathbb{R}^N$ is the predicted scores, and $\text{sg}(\cdot)$ denotes stop-gradient through the target errors.

We apply *reasoning dropout* [10] during training: each sample is converted to either `/cot` or `/no_cot` with probability p_{cot}. Thus, explicit CoT is an optional training condition rather than a mandatory inference trace. This reduces reliance on visible reasoning, mitigates noisy CoT supervision, encourages reasoning to be internalized into backbone states and the cache-conditioned action interface, and enables test-time reasoning dropout while preserving the benefits of reasoning-aware pre-training.

3 Experiment

We organize our experiments around three questions:

Table 1: **Ablation of ECoT fields on average success rate in VLABench [61].** The lower block reports performance changes with and without Bridge pre-training [51], relative to the first column.

CoT Field		Ablation Setting														
Textual Description	**Goal**		✓								✓					✓
	Planning			✓							✓					✓
	Subtask				✓						✓		✓		✓	✓
	Movement					✓					✓			✓	✓	✓
	Reasoning						✓				✓		✓	✓		✓
Numerical Coordinate	**Gripper**							✓				✓				✓
	Point traj.								✓			✓			✓	✓
	Bounding box									✓		✓				✓
w/o Pretraining		**19.0**	-1.2	-0.8	-0.6	+4.1	-1.0	-0.7	+4.8	-1.4	+4.9	+2.7	-1.3	+5.2	+7.4	+8.2
w/ Pretraining		**25.2**	-0.8	-0.5	-0.7	+2.0	-0.9	-5.6	+1.4	-6.1	+3.1	-2.4	-1.6	+3.0	+3.7	+2.5
w/ Pretraining + Dropout		**25.2**	-0.6	-0.3	-0.5	+1.9	-0.6	-0.8	+3.0	-1.0	+2.9	+1.5	-0.9	+3.2	+4.4	+4.0

- *What kinds of embodied CoT are more useful for action policy learning?*
- *Can embodied CoT provide an effective transfer interface that turns stronger VLM backbones into stronger VLA action policies?*
- *Can embodied CoT serve as a scalable pre-training signal for generalizable action generation?*

To answer them, we first conduct controlled studies with a unified backbone and autoregressive prediction to identify useful CoT fields, evaluate their pre-training value, and examine VLM-to-VLA transfer with ECoT in Sec. 3.1. We then compare ERVLA with strong baselines and ablate reasoning–action integration and CoT pre-training scale in Sec. 3.2. Finally, we evaluate real-world robot tasks requiring semantic understanding, robust scene grounding, and long-horizon capabilities in Sec. 3.3.

Benchmarks. We evaluate on three complementary benchmarks:

- **LIBERO** [33] is one of the most widely used manipulation benchmarks and provides a strong testbed for studies, but its relatively fixed setting can encourage benchmark-specific overfitting.
- **LIBERO-Plus** [15] extends LIBERO with camera, state, language, background, and layout shifts, providing a stronger test of zero-shot transfer and robustness beyond the original distribution.
- **VLABench** [61] is a harder benchmark with one in-distribution track and multiple generalization tracks, covering cross-category transfer, commonsense reasoning, semantic instruction following, and unseen texture variation. It is suited for testing semantic and distribution shifts.

Baselines. On LIBERO-Plus, we compare with ECoT [58], Emma-X [46], OpenVLA-OFT [24], UniVLA [7], WorldVLA [9], π_0 [6], π_0-FAST [41], Spatial Forcing [27], PokeVLA [65], and $\pi_{0.5}$ [20]. On VLABench, we additionally compare with X-VLA [64] and ACoT [66]. The works cover linguistic, visual, and latent reasoning. We report CoT scaling variants of ERVLA and ablations of the choice policy and knowledge truncation. Full training details are provided in Appendix C.

VLM backbones. To examine whether stronger VLMs transfer better to action through embodied CoT, we evaluate diverse backbones spanning a broad range on the OpenVLM Leaderboard [14]: PaliGemma-2-3B [3], Florence-2-large [55], Cosmos-Reason2-2B [37], Gemma-4-E2B-it, Qwen2.5-VL-3B/7B [44], and Qwen3-VL-2B/4B/8B [2]. All models share the same action-token interface and are evaluated under the StarVLA framework [12], with training details provided in Appendix C.

3.1 Exploring Effective Embodied Chain-of-Thought Signals

To study individual and combined embodied CoT signals, we annotate VLABench [61] and LIBERO [33] using the format in Sec. 2.1. We test on Qwen3-VL-4B [2] with a fixed autoregressive CoT+FAST [41] action interface, so performance differences mainly reflect the CoT signal itself.

Action-related CoT signals support action generation more directly than high-level understanding alone. As shown in Table 1, isolated high-level fields do not help action learning under direct training: *Goal*, *Planning*, *Subtask*, and *Reasoning* all lead to small drops, with *Goal* decreasing performance by -1.2. In contrast, action-related fields are more effective: *Movement* improves performance by +4.1 and *Point trajectory* by +4.8, while coordinate-only cues such as *Bounding box* remain insufficient when used alone. High-level understanding becomes useful only when coupled

Table 2: **Comparison of scaling embodied CoT pre-training data on LIBERO [33] and VLABench [61].** All settings use FAST [41] action representation with embodied CoT supervision and reasoning dropout. Values below the first row report performance changes relative to it.

Pre-Training Data (with dropout)	LIBERO				VLABench				
	Spatial	Object	Goal	Long	In-dist.	Category	Common	Instruction	Texture
–	96.4	96.6	91.8	88.6	38.0	29.0	17.6	25.8	30.0
Bri. [51]	+0.4	+0.8	+0.4	+0.6	+1.0	+0.8	+0.6	-0.4	+1.2
Bri. [51]+Fra. [68]	+0.2	-0.6	+0.2	-0.8	-1.8	+0.2	+0.4	+0.2	-1.0
Bri. [51]+Fra. [68]+Mol. [26]	-0.8	-1.0	-0.6	-1.4	-1.2	-1.0	-0.8	-1.4	-1.2
Bri. [51]+Fra. [68]+Mol. [26]+Dro. [21]	-1.8	-1.6	-2.0	-2.4	-3.6	-3.0	-2.2	-3.2	-3.4

with concrete action guidance, as *Movement+Reasoning* reaches +5.2 and *Subtask+Movement+Point trajectory* reaches +7.4. Full ECoT gives the strongest gain, suggesting that effective embodied CoT should connect semantic understanding to executable motion rather than merely describe task intent.

Pretraining helps only with reliable CoT supervision, while reasoning dropout mitigates CoT contamination. With ECoT pretraining on Bridge [51], unreliable coordinate fields become harmful: *Gripper* and *Bounding box* cause the largest drops, reducing performance by -5.6 and -6.1, respectively. Unlike benchmark annotations obtained by simulator replay and thus closer to ground truth, large-scale detector-generated labels are lossy and can impose inconsistent supervision on semantically similar frames. Reasoning dropout substantially reduces this damage, shrinking the drops of *Gripper* and *Bounding box* to -0.8 and -1.0, while preserving useful action-oriented supervision such as *Point trajectory*, which improves from +1.4 without dropout to +3.0 with dropout. Overall, CoT pretraining works best when reliable acting signals are retained and unstable grounding fields are regularized rather than densely imposed.

Explicit CoT does not automatically scale under autoregressive action modeling. With full CoT supervision, Bridge-only pretraining provides modest gains, but Table 2 shows that simply adding more CoT data does not reliably improve an autoregressive CoT+action-token model. As the pretraining mixture expands to Fractal [68], MolmoAct [26], and Droid [21], performance gradually degrades across VLABench tracks. In the four-source setting, the drops reach -3.6 on In-dist., -3.0 on Cross-category, and -3.4 on Texture, indicating that more explicit CoT can introduce more prefix noise rather than better control. We attribute this to the fragility of autoregressive action decoding under dense and drifting supervision: reasoning traces vary in length and quality, grounding noise perturbs the action prefix, and CoT errors directly affect subsequent action tokens. Thus, explicit CoT is useful as supervision, but autoregressive action decoding is a poor substrate for scaling it.

Embodied CoT makes stronger VLMs more reliably transferable to stronger VLAs. Figure. 5 left shows that under non-CoT recipe, backbone strength and downstream performance remain weakly correlated. With explicit embodied CoT, stronger VLMs become more consistently effective: the best results are dominated by Qwen3-VL series [2], with Qwen3-VL-4B and leading on LIBERO *Spatial* and *Goal* and VLABench *In-dist.*, *Category*, and *Common* as detailed in Table 13 of Appendix D. This suggests that embodied CoT is not merely an auxiliary label space, but a transfer interface that converts stronger VLM semantic priors into action-relevant representations.

3.2 Studies of ERVLA Design Choices and ECoT Scaling

We move beyond the controlled LIBERO setting to LIBERO-Plus. Sec. 3.1 motivates our ERVLA design: naive autoregressive for CoT and action generation does not scale reliably, and explicit CoT is most useful when it reshapes action-relevant representations rather than serving as a fragile token prefix. We instantiate ERVLA with Qwen3-VL-4B [2], add a *choice policy [42]* branch for candidate action chunks and scores, and apply *knowledge truncation* so DiT conditions on semantic CoT memory instead of control-query tokens. With full embodied CoT pre-training and reasoning dropout, we obtain the final ERVLA model and analyze both design choices and pre-training scale.

ERVLA achieves state-of-the-art performance through the choice policy and knowledge truncation. As shown in Tables 3 and 4, ERVLA reaches 86.9 on LIBERO-Plus, surpassing $\pi_{0.5}$ (85.5), and obtains 53.2/70.4/65.9 average SR/IS/PS on VLABench, outperforming strong baselines including atent-interface-based UniVLA [7] and visual-prediction-enhanced WorldVLA [9]. These gains come from the proposed reasoning–action interface rather than simply adding explicit CoT. Removing

Table 3: **Quantitative comparison on the LIBERO-Plus [15] benchmark.** Models are post-trained on LIBERO [33] and evaluated by zero-shot direct transfer. Gray-shaded rows are ERVLA ablations: *No Choice (E2E)* allows DiT loss back-propagated to the VLM; *No CoT* removes explicit embodied chain-of-thought supervision while keeping the same action-learning pipeline; *No Choice + KI* blocks this back-propagation, resembling ThinkAct-style [18] planning and action separation; *Choice w/o KT* lets DiT access the CoT+choice cache, which can introduce shortcut learning.

Method	Task Suite				Perturbation Type							Total
	Spatial	Object	Goal	Long	Camera	Robot	Language	Light	Background	Noise	Layout	
ECoT [58]	31.8	27.9	30.6	8.6	0.3	26.8	40.2	42.6	16.4	10.2	36.9	24.3
Emma-X [46]	33.4	28.4	31.2	8.0	0.2	28.6	42.0	42.8	17.6	10.0	37.4	25.1
OpenVLA-OFT [24]	84.0	66.5	63.0	66.4	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
UniVLA [7]	55.5	36.7	40.7	39.9	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
WorldVLA [9]	32.5	28.6	31.8	8.2	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
π_0 [6]	60.7	61.4	44.9	48.4	13.8	6.0	58.8	85.0	81.4	79.0	68.8	53.6
π_0-FAST [41]	74.4	72.7	57.6	43.4	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
Spatial Forcing [27]	52.9	31.0	28.2	5.4	20.1	13.4	40.9	29.1	33.4	25.7	39.3	29.1
PokeVLA [65]	85.4	81.8	77.6	72.7	84.7	46.1	84.8	94.6	82.6	89.8	77.2	79.3
$\pi_{0.5}$ [20]	90.4	**89.9**	**81.0**	80.8	71.7	75.5	85.9	**96.1**	**95.7**	86.4	**87.5**	85.5
No Choice	70.8	65.4	58.6	55.2	54.6	42.8	64.2	72.4	66.0	68.8	62.0	61.9
No CoT	77.4	71.8	65.2	62.0	68.6	43.2	72.6	82.0	72.6	76.4	68.8	70.8
No Choice + Knowledge Insulation	83.8	78.6	71.4	69.0	80.4	43.8	81.0	91.6	79.2	85.4	74.6	76.5
Choice + No Knowledge Truncation	89.2	88.6	79.4	79.8	70.8	73.4	84.6	94.2	94.4	85.6	86.2	84.7
ERVLA (Ours)	**96.2**	89.6	79.6	**82.1**	**77.2**	75.3	**87.1**	95.1	94.7	**92.3**	86.4	**86.9**

Table 4: **Comparison on the VLABench benchmark.** SR, PS, and IS denote Success Rate, Progress Score, and Intention Score, respectively.

Method	In-dist.			Cross Category			Commonsense			Instruction			Texture			Avg.		
	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑	SR ↑	PS ↑	IS ↑
π_0 [6]	47.0	62.7	67.8	21.2	33.6	44.0	29.1	43.0	54.9	17.3	38.7	58.0	32.2	42.5	50.6	29.4	44.1	55.0
π_0-FAST [41]	56.2	66.8	72.4	31.0	38.2	47.8	38.0	48.6	56.8	35.0	45.0	59.4	39.0	49.0	56.8	39.8	49.5	58.6
X-VLA [64]	–	67.8	–	–	25.1	–	–	48.2	–	–	63.1	–	–	–	–	–	51.1	–
ACoT-VLA [66]	–	66.1	79.8	–	38.9	54.1	–	37.8	52.3	–	39.6	56.8	–	54.6	**74.6**	–	47.4	63.5
$\pi_{0.5}$ [20]	65.4	77.8	80.4	38.2	49.7	52.0	43.9	**57.3**	**60.0**	48.2	64.2	67.0	44.9	**62.3**	65.0	48.1	62.3	64.9
No Choice	50.2	57.8	65.0	33.6	42.0	49.6	36.4	45.4	52.8	42.4	54.6	61.2	33.4	41.2	50.4	39.2	48.2	55.8
No CoT	52.6	59.6	66.0	34.8	44.8	51.6	38.8	47.8	54.0	44.2	56.4	62.4	34.0	43.4	51.6	40.9	50.4	57.1
No Choice + Knowledge Insulation	55.0	61.4	67.2	36.0	47.6	53.8	41.2	50.0	55.0	46.0	58.2	63.4	34.8	45.4	52.6	42.6	52.5	58.4
Choice + No Knowledge Truncation	62.0	73.0	76.2	42.4	55.2	59.4	43.0	52.6	56.4	53.6	66.4	70.2	35.0	51.8	54.6	47.2	59.8	63.4
ERVLA (Ours)	**69.7**	**81.1**	**84.2**	**47.0**	**61.0**	**66.4**	**44.0**	55.0	57.2	**58.0**	**70.2**	**73.8**	**47.4**	**62.3**	70.6	**53.2**	**65.9**	**70.4**

the choice policy branch weakens action-aware representation shaping, while replacing knowledge truncation with naive end-to-end conditioning allows DiT to exploit synthetic control-query tokens and causes large drops on both benchmarks. These results show that ERVLA benefits from both components: the choice policy injects action-level discrimination into the VLM, and knowledge truncation provides DiT with cleaner semantic memory for continuous-action generation.

ECoT scales through synergistic reasoning–action representation learning. The right panel of Figure. 5 shows that ERVLA with the choice policy branch improves steadily as embodied CoT pre-training data increases on both LIBERO-Plus and VLABench. In contrast, autoregressive CoT+action-token designs, as in ECoT [58] and Emma-X [46], become brittle because reasoning noise and length variability directly affect action decoding. An isolated VLM+DiT design, in the spirit of ThinkAct [18], avoids this brittleness but weakly couples reasoning to action. Thus, the bottleneck is not whether embodied CoT is useful, but whether it can guide action learning without becoming a decoding burden. As depicted in Figure. 4, ERVLA uses CoT as high-level semantic supervision, choice policy prediction to internalize prefix reasoning into action-aware VLM represen-

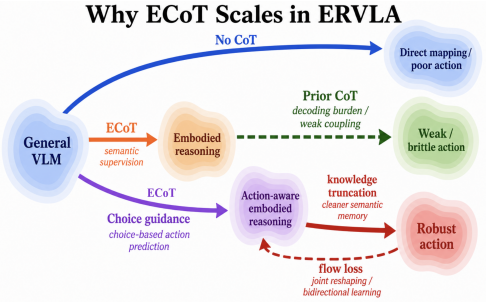

Figure 4: Next-token action decoding after CoT is brittle, while knowledge insulation limits action feedback. Coupling ECoT with choice policy and flow supervision enables reasoning and action generation to co-adapt for robust control.

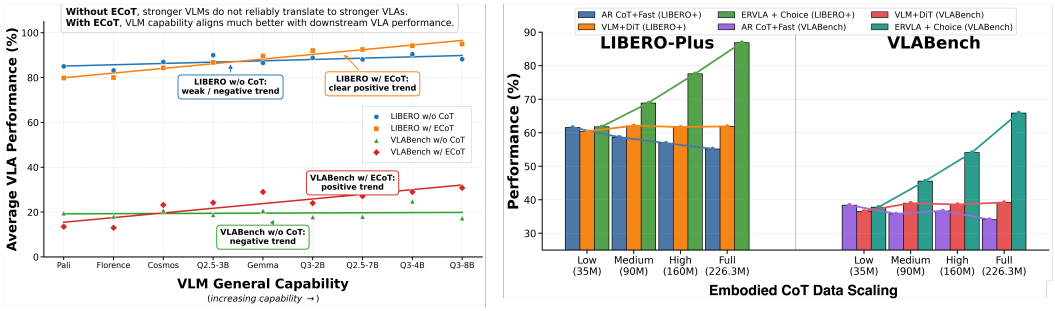

Figure 5: **VLM-to-VLA transfer and embodied CoT scaling. Left:** ECoT better aligns VLM capability with action. **Right:** ERVLA scales steadily with more CoT data on both LIBERO-Plus and VLABench, whereas AR CoT+Fast and isolated VLM+DiT show weaker or saturated scaling.

Table 5: Real-world results across four difficulty tiers. Each tier reports success rate and progress score. Results are averaged over two task families.

Method	Basic		Distractors		Semantic		Long-horizon		Average	
	Succ. rate	Prog. score	Succ. rate	Prog. score	Succ. rate	Prog. score	Succ. rate	Prog. score	Succ. rate	Prog. score
ECoT [58]	60	68	18	30	10	25	6	18	24	35
WorldVLA [9]	78	84	28	42	18	35	12	28	34	47
UniVLA [7]	76	82	31	45	22	38	18	34	37	50
$\pi_{0.5}$ [20]	**97**	**98**	**45**	57	31	45	35	38	53	60
ERVLA (Ours)	96	97	44	**58**	**42**	**58**	**38**	**55**	**55**	**67**

tations, knowledge truncation to give DiT cleaner semantic memory, and flow loss for end-to-end action feedback. Embodied CoT therefore scales when reasoning supervision and action losses are jointly optimized in representation space, rather than treated as literal autoregressive targets.

3.3 Real-world Experiments

To evaluate real-world manipulation beyond simulation, we deploy ERVLA on physical robots with one third-person and one wrist camera. We compare representative reasoning–action paradigms: ECoT [58] for autoregressive linguistic reasoning, WorldVLA [9] for using visual prediction as a reasoning–action interface, UniVLA [7] for latent task-centric reasoning, and $\pi_{0.5}$ [20] as a strong VLM-conditioned DiT baseline with reasoning pre-training for continuous action generation.

Task design. We design two task families for real-world experiments: (i) placing items into drawers and (ii) clearing tabletop waste. The objects include toy cars, fruit models, bottles, and cans, together with a drawer and a basket. We organize the evaluation into four difficulty tiers:

- **Basic.** Clean scenes with explicit object names, mainly testing basic visual grounding and manipulation execution.
- **Distractors.** Irrelevant or visually similar objects are added to the scene, requiring the policy to identify the correct target while ignoring plausible distractors.
- **Semantic.** Instructions refer to objects indirectly through attributes, functions, categories, or spatial relations, requiring semantic interpretation beyond direct object-name matching. Examples include "put away the object that is not a fruit," "place the blue car into the second drawer," and "remove the item that should be thrown away."
- **Long-horizon.** Multiple challenges are combined into multi-object, multi-step tasks, requiring the policy to maintain task progress and execute a sequence of correct subgoals over an extended horizon.

Each tier contains five instructions, yielding 20 tasks in total. We conduct five trials per task and report the mean success rate and progress score.

Results. As depicted in Figure 6, ECoT [58] performs worst due to the latency and brittleness of autoregressive reasoning–action decoding, while WorldVLA [9] and UniVLA [7] remain limited

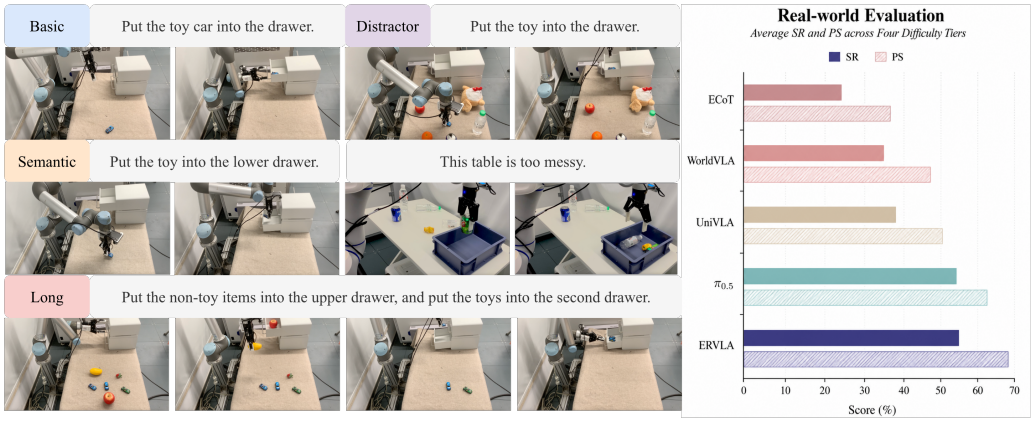

Figure 6: **Real-world evaluation.** Left: representative rollouts across basic, distractor, semantic, and long-horizon settings. Right: average success rate and progress score across four difficulty tiers, where ERVLA shows stronger robustness under semantic and long-horizon real-world generalization.

because visual prediction does not directly resolve semantic intent and latent reasoning can obscure fine-grained grounding. In contrast, ERVLA enables action-sensitive VLM representations and cleaner DiT conditioning. Detailed numbers in Table 5 show that the main gap emerges not in clean cases, where $\pi_{0.5}$ [20]and ERVLA perform similarly, but under semantic ambiguity, distractors, and long-horizon dependencies. ERVLA gains most on the *Semantic* and *Long-horizon* tiers, supporting our claim that embodied CoT is most useful when internalized as action-relevant representation rather than emitted as mandatory explanation. Overall, these results indicate that real-world robustness comes less from stronger low-level action generation alone and more from whether reasoning supervision is converted into grounded, action-aware representations that remain useful under out-of-distribution semantic and temporal shifts.

4 Conclusion

In this work, we argue that embodied CoT should be understood less as an output format and more as an interface between semantic reasoning and action learning. The key lesson is that reasoning does not benefit robot policies simply by being longer, denser, or more explicit; it becomes useful only when its supervision is reliable, action-relevant, and integrated into the policy in a way that shapes representations rather than burdens decoding. From this perspective, both positive and negative of our findings are important: effective reasoning signals can expose the control value of pretrained VLM knowledge, while noisy grounding and naive autoregressive coupling reveal why reasoning-enhanced VLAs can fail to scale. ERVLA instantiates this principle by combining explicit linguistic CoT, reasoning dropout, VLM-level action shaping, and semantically conditioned continuous control into an end-to-end trainable recipe. More broadly, we hope that our dataset, annotation pipeline, model design, and systematic analyses provide a foundation for future work on reasoning-aware robot learning, including more reliable grounding supervision, adaptive test-time reasoning, stronger real-world generalization, and architectures that treat reasoning not as verbal decoration, but as a learnable bridge from perception to action.

Acknowledgments

This work was funded by the National Key Research and Development Program of China under Grant 2024YFB4708000, the National Natural Science Fund under Grant 62120106005, and the Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China under Grant JYB2025XDXM109. This work was conducted at Xiaomi Robotics with research and computing support from Xiaomi Robotics, and benefited from academic guidance and research support from Tsinghua University. The authors declare no competing interests.

References

[1] AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, Shu Jiang, Yuxin Jiang, Cheng Jing, Hongyang Li, Jialu Li, Chiming Liu, Yi Liu, Yuxiang Lu, Jianlan Luo, Ping Luo, Yao Mu, Yuehan Niu, Yixuan Pan, Jiangmiao Pang, Yu Qiao, Guanghui Ren, Cheng Ruan, Jiaqi Shan, Yongjian Shen, Chengshi Shi, Mingkang Shi, Modi Shi, Chonghao Sima, Jianheng Song, Huijie Wang, Wenhao Wang, Dafeng Wei, Chengen Xie, Guo Xu, Junchi Yan, Cunbiao Yang, Lei Yang, Shukai Yang, Maoqing Yao, Jia Zeng, Chi Zhang, Qinglin Zhang, Bin Zhao, Chengyue Zhao, Jiaqi Zhao, and Jianchao Zhu. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems, 2025. URL https://arxiv.org/abs/2503.06669.

[2] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

[3] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bošnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier Henaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. Paligemma: A versatile 3b vlm for transfer, 2024. URL https://arxiv.org/abs/2407.07726.

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

[5] Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies, 2025. URL https://arxiv.org/abs/2506.07339.

[6] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

[7] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions, 2025. URL https://arxiv.org/abs/2505.06111.

[8] Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, et al. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026.

[9] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, Deli Zhao, and Hao Chen. Worldvla: Towards autoregressive action world model, 2025. URL https://arxiv.org/abs/2506.21539.

[10] William Chen, Suneel Belkhale, Suvir Mirchandani, Oier Mees, Danny Driess, Karl Pertsch, and Sergey Levine. Training strategies for efficient embodied reasoning, 2025. URL https://arxiv.org/abs/2505.08243.

[11] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

[12] StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing, 2026. URL https://arxiv.org/abs/2604.05014.

[13] Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better, 2025. URL https://arxiv.org/abs/2505.23705.

[14] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 11198–11201, 2024.

[15] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: In-depth robustness analysis of vision-language-action models, 2025. URL https://arxiv.org/abs/2510.13626.

[16] Shenghao Fu, Qize Yang, Qijie Mo, Junkai Yan, Xihan Wei, Jingke Meng, Xiaohua Xie, and Wei-Shi Zheng. Llmdet: Learning strong open-vocabulary object detectors under the supervision of large language models, 2025. URL https://arxiv.org/abs/2501.18954.

[17] Charles Gaydon, Michel Daab, and Floryne Roche. Fractal: An ultra-large-scale aerial lidar dataset for 3d semantic segmentation of diverse landscapes, 2024. URL https://arxiv.org/abs/2405.04634.

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

[19] Chi-Pin Huang, Yunze Man, Zhiding Yu, Min-Hung Chen, Jan Kautz, Yu-Chiang Frank Wang, and Fu-En Yang. Fast-thinkact: Efficient vision-language-action reasoning via verbalizable latent planning, 2026. URL https://arxiv.org/abs/2601.09708.

[20] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

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

Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset, 2025. URL https://arxiv.org/abs/2403.12945.

[22] Dongyoung Kim, Sumin Park, Huiwon Jang, Jinwoo Shin, Jaehyung Kim, and Younggyo Seo. Robot-r1: Reinforcement learning for enhanced embodied reasoning in robotics. *arXiv preprint arXiv:2506.00070*, 2025.

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

[24] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success, 2025. URL https://arxiv.org/abs/2502.19645.

[25] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

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

[27] Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model, 2025. URL https://arxiv.org/abs/2510.12276.

[28] Peiyan Li, Yixiang Chen, Hongtao Wu, Xiao Ma, Xiangnan Wu, Yan Huang, Liang Wang, Tao Kong, and Tieniu Tan. Bridgevla: Input-output alignment for efficient 3d manipulation learning with vision-language models, 2025. URL https://arxiv.org/abs/2506.07961.

[29] Peiyan Li, Shuyuan Zhang, Wenju Yang, Yitong Hu, Yuhang Wang, Dunzheng Wang, Hang Li, Di Guo, and Huaping Liu. Proactive collaboration via autonomous interaction. *Nature Communications*, 17:6306, 2026.

[30] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, Xiaofan Wang, Bei Liu, Jianlong Fu, Jianmin Bao, Dong Chen, Yuanchun Shi, Jiaolong Yang, and Baining Guo. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation, 2024. URL https://arxiv.org/abs/2411.19650.

[31] Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Xinlong Wang, Di Guo, et al. What matters in building vision–language–action models for generalist robots. *Nature Machine Intelligence*, 8(2):158–172, 2026.

[32] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

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

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

[35] Yicheng Liu, Shiduo Zhang, Zibin Dong, Baijun Ye, Tianyuan Yuan, Xiaopeng Yu, Linqi Yin, Chenhao Lu, Junhao Shi, Luca Jiang-Tao Yu, et al. Faster: Toward efficient autoregressive vision language action modeling via neural action tokenization. *arXiv preprint arXiv:2512.04952*, 2025.

[36] Zhuoyang Liu, Jiaming Liu, Hao Chen, Jiale Yu, Ziyu Guo, Chengkai Hou, Chenyang Gu, Xiangju Mi, Renrui Zhang, Kun Wu, Zhengping Che, Jian Tang, Pheng-Ann Heng, and Shanghang Zhang. Last_0: Latent spatio-temporal chain-of-thought for robotic vision-language-action model, 2026. URL https://arxiv.org/abs/2601.05248.

[37] NVIDIA, :, Alisson Azzolini, Junjie Bai, Hannah Brandon, Jiaxin Cao, Prithvijit Chattopadhyay, Huayu Chen, Jinju Chu, Yin Cui, Jenna Diamond, Yifan Ding, Liang Feng, Francesco Ferroni, Rama Govindaraju, Jinwei Gu, Siddharth Gururani, Imad El Hanafi, Zekun Hao, Jacob Huffman, Jingyi Jin, Brendan Johnson, Rizwan Khan, George Kurian, Elena Lantz, Nayeon Lee, Zhaoshuo Li, Xuan Li, Maosheng Liao, Tsung-Yi Lin, Yen-Chen Lin, Ming-Yu Liu, Xiangyu Lu, Alice Luo, Andrew Mathau, Yun Ni, Lindsey Pavao, Wei Ping, David W. Romero, Misha Smelyanskiy, Shuran Song, Lyne Tchapmi, Andrew Z. Wang, Boxin Wang, Haoxiang Wang, Fangyin Wei, Jiashu Xu, Yao Xu, Dinghao Yang, Xiaodong Yang, Zhuolin Yang, Jingxu Zhang, Xiaohui Zeng, and Zhe Zhang. Cosmos-reason1: From physical common sense to embodied reasoning, 2025. URL https://arxiv.org/abs/2503.15558.

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

[39] Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlas, 2025. URL https://arxiv.org/abs/2512.15692.

[40] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

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

[42] Haozhi Qi, Yen-Jen Wang, Toru Lin, Brent Yi, Yi Ma, Koushil Sreenath, and Jitendra Malik. Coordinated humanoid manipulation with choice policies, 2025. URL https://arxiv.org/abs/2512.25072.

[43] Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Xinyi Ye, Qi Lv, Modi Shi, Guanghui Ren, Cheng Ruan, et al. Eo-1: Interleaved vision-text-action pretraining for general robot control. *arXiv preprint arXiv:2508.21112*, 2025.

[44] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

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

[46] Qi Sun, Pengfei Hong, Tej Deep Pala, Vernon Toh, U-Xuan Tan, Deepanway Ghosal, and Soujanya Poria. Emma-x: An embodied multimodal action model with grounded chain of thought and look-ahead spatial reasoning, 2024. URL https://arxiv.org/abs/2412.11974.

[47] Sinan Tan, Mengmeng Ge, Di Guo, Huaping Liu, and Fuchun Sun. Knowledge-based embodied question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10): 11948–11960, 2023.

[48] Peijun Tang, Shangjin Xie, Binyan Sun, Baifu Huang, Kuncheng Luo, Haotian Yang, Weiqi Jin, and Jianan Wang. Mind to hand: Purposeful robotic control via embodied reasoning, 2025. URL https://arxiv.org/abs/2512.08580.

[49] Gemini Robotics Team, Abbas Abdolmaleki, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Ashwin Balakrishna, Nathan Batchelor, Alex Bewley, Jeff Bingham, et al. Gemini robotics 1.5: Pushing the frontier of generalist robots with advanced embodied reasoning, thinking, and motion transfer. *arXiv preprint arXiv:2510.03342*, 2025.

[50] Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[51] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale, 2024. URL https://arxiv.org/abs/2308.12952.

[52] Yating Wang, Haoyi Zhu, Mingyu Liu, Jiange Yang, Hao-Shu Fang, and Tong He. Vq-vla: Improving vision-language-action models via scaling vector-quantized action tokenizers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11089–11099, 2025.

[53] Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

[54] Yilin Wu, Anqi Li, Tucker Hermans, Fabio Ramos, Andrea Bajcsy, and Claudia PÃŠrez-D'Arpino. Do what you say: Steering vision-language-action models via runtime reasoning-action alignment verification. *arXiv preprint arXiv:2510.16281*, 2025.

[55] Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks, 2023. URL https://arxiv.org/abs/2311.06242.

[56] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

[57] Yifu Yuan, Haiqin Cui, Yaoting Huang, Yibin Chen, Fei Ni, Zibin Dong, Pengyi Li, Yan Zheng, and Jianye Hao. Embodied-r1: Reinforced embodied reasoning for general robotic manipulation. *arXiv preprint arXiv:2508.13998*, 2025.

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

[59] Lihan Zha, Asher J. Hancock, Mingtong Zhang, Tenny Yin, Yixuan Huang, Dhruv Shah, Allen Z. Ren, and Anirudha Majumdar. Lap: Language-action pre-training enables zero-shot cross-embodiment transfer, 2026. URL https://arxiv.org/abs/2602.10556.

[60] Jianke Zhang, Xiaoyu Chen, Qiuyue Wang, Mingsheng Li, Yanjiang Guo, Yucheng Hu, Jiajun Zhang, Shuai Bai, Junyang Lin, and Jianyu Chen. Vlm4vla: Revisiting vision-language-models in vision-language-action models. *arXiv preprint arXiv:2601.03309*, 2026.

[61] Shiduo Zhang, Zhe Xu, Peiju Liu, Xiaopeng Yu, Yuan Li, Qinghui Gao, Zhaoye Fei, Zhangyue Yin, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. Vlabench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks, 2024. URL https://arxiv.org/abs/2412.18194.

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

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

[64] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Ya-Qin Zhang, Jiangmiao Pang, Jingjing Liu, Tai Wang, and Xianyuan Zhan. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model, 2025. URL https://arxiv.org/abs/2510.10274.

[65] Yupeng Zheng, Xiang Li, Songen Gu, Yuhang Zheng, Shuai Tian, Weize Li, Linbo Wang, Senyu Fei, Pengfei Li, Yinfeng Gao, Zebin Xing, Yilun Chen, Qichao Zhang, Haoran Li, and Wenchao Ding. Pokevla: Empowering pocket-sized vision-language-action model with comprehensive world knowledge guidance, 2026. URL https://arxiv.org/abs/2604.20834.

[66] Linqing Zhong, Yi Liu, Yifei Wei, Ziyu Xiong, Maoqing Yao, Si Liu, and Guanghui Ren. Acot-vla: Action chain-of-thought for vision-language-action models, 2026. URL https://arxiv.org/abs/2601.11404.

[67] Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, et al. Beast: Efficient tokenization of b-splines encoded action sequences for imitation learning. *arXiv preprint arXiv:2506.06072*, 2025.

[68] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

A Related Work

A.1 Embodied Reasoning in Robot Manipulation

Vision-language-action (VLA) models seek to turn pretrained visual-language knowledge into executable robot control. Early systems such as RT-2 [68] and OpenVLA [23] demonstrate that web-scale semantic priors can be reused for manipulation, while recent generalist policies further scale this paradigm with larger backbones, broader embodiments, and stronger action heads, including π_0 [6], $\pi_{0.5}$ [20], GR00T [4], Gemini Robotics [49], Eo-1 [43], Hi Robot [45], and CogACT [30]. These models mark a transition from task-specific imitation policies to foundation-style robot policies, where language, vision, and prior world knowledge provide a broad semantic substrate for control. However, this transition also exposes a fundamental mismatch: VLMs are trained to produce symbolic or descriptive outputs, whereas robot policies must produce spatially precise, temporally consistent, and dynamically feasible actions.

The progress of VLAs is closely tied to large-scale robot data, including Open X-Embodiment [38], BridgeData V2 [51], DROID [21], AgiBot World [1], MolmoAct [26], and Fractal [68]. Yet scaling data and backbones alone does not fully resolve this semantic-to-action gap. VLM4VLA [60] shows that stronger VLMs do not necessarily yield stronger VLAs, suggesting that the bottleneck lies less in whether the model "knows" the task and more in whether such knowledge is transformed into action-relevant representations. In manipulation, useful knowledge must be grounded into objects, decomposed into subtasks, aligned with the robot state, and ultimately expressed as short-horizon motion. Embodied reasoning therefore serves as a missing interface: it converts high-level task understanding into structured signals that can guide perception-to-action learning, rather than relying on the policy to discover this bridge implicitly from raw trajectories alone.

A.2 Taxonomy of Embodied Reasoning Signals

To bridge high-level task understanding and motor execution, recent reasoning-aware VLAs introduce intermediate reasoning interfaces in linguistic, spatial, visual, and latent forms. A first line uses explicit linguistic or action-space CoT. ECoT [58] and EMMA-X [46] generate textual plans, subtasks, motion descriptions, and grounded coordinates before action prediction; LAP [59] verbalizes low-level actions in natural language; ACoT-VLA [66] reasons with coarse action intents; and ECoT-Lite studies efficient reasoning strategies such as reasoning dropout [10]. Reinforced and verification-based methods, including Robot-R1 [22], Embodied-R1 [57], Mind-to-Hand [48], and runtime reasoning-action alignment verification [54], further show that reasoning becomes more useful when connected to execution feedback. These works make reasoning interpretable and directly supervisable, but also reveal that not all textual or grounded fields are equally reliable for control, and noisy explicit supervision can directly affect action learning.

A second line represents reasoning visually or through world-model-style prediction. CoT-VLA [62] predicts future visual subgoals, WorldVLA [9] uses visual prediction as a reasoning-action interface, Cosmos Policy [25] adapts video models for visuomotor control, Mimic-Video [39] studies video-action policies beyond standard VLAs, BridgeVLA [28] uses 2D heatmap-style spatial supervision, and world action models investigate whether generative world models can act as zero-shot policies [56]. Related methods further explore spatial alignment and world-knowledge grounding [27, 65]. Visual reasoning naturally captures dynamics and spatial correspondence, but it is more expensive and harder to isolate: future images or heatmaps are slower to generate than text, often require per-step refresh, and may need inverse-dynamics or action-extraction modules to convert visual predictions into executable actions.

A third line compresses reasoning into latent or implicit representations. ThinkAct [18], FastThinkAct [19], LaST$_0$ [36], and UniVLA [7] distill reasoning into visual, verbalizable, spatio-temporal, or task-centric latents, while other systems move reasoning into policy representations rather than explicit traces [20, 30, 45, 13, 53]. These approaches reduce inference overhead, but usually assume that useful reasoning structure has already been learned or distilled. Therefore, our work begins from explicit embodied CoT not as the final deployment format, but as the most controllable interface for identifying which signals help action learning, which contaminate representations, and how reasoning should supervise continuous-action policies. This motivates using embodied CoT as representation-shaping pre-training rather than as a mandatory autoregressive prefix at inference time.

A.3 Action Representations and Policy Integration

A central design choice in VLA systems is how language-conditioned perception is converted into robot actions. Early models such as RT-2 [68] and OpenVLA [23] formulate control as next-token prediction over discretized action tokens, reusing the VLM decoder to unify language and action generation. However, actions are not naturally linguistic objects: tokenization introduces quantization error, autoregressive decoding increases latency, and long reasoning prefixes make action prediction sensitive to exposure bias, reasoning noise, and sequence length. Methods such as OpenVLA-OFT [24], FAST [41], VQ-VLA [52], BEAST [67], and FASTer [35] improve decoding or tokenization efficiency, but still keep reasoning and action tightly coupled through autoregressive token prediction.

Continuous-action policies instead model action chunks directly with imitation, diffusion, or flow objectives. ACT [63], Diffusion Policy [11], RDT [34], DexVLA [53], π_0 [6], and real-time action chunking flow policies [5] better capture smoothness, multimodality, and temporal consistency. Yet this shifts the challenge from action representation to policy integration: the action expert must use VLM semantics without collapsing them into pooled-feature bottlenecks or damaging pretrained knowledge. Knowledge-insulated VLA training protects the backbone by blocking continuous-action gradients [13], but this weakens action feedback and limits the emergence of action-sensitive VLM representations. Other works improve the interface through spatial alignment, cross-embodiment prompting, or world knowledge, such as Spatial Forcing [27], X-VLA [64], and PokeVLA [65].

Our work follows the continuous-action direction but focuses on the missing integration mechanism. Instead of treating CoT as an autoregressive action prefix, ERVLA uses embodied CoT as representation-shaping supervision. The choice branch injects action-level discrimination into the VLM, the DiT policy attends to per-layer VLM cache rather than compressed pooled features, and knowledge truncation restricts DiT conditioning to semantic-prefix memory rather than synthetic control-query tokens. This preserves end-to-end action feedback while avoiding shortcut conditioning. Thus, the key question is not only whether actions should be tokenized or generated continuously, but how reasoning supervision and action losses should jointly shape the representation that connects pretrained semantics to executable control.

B Embodied CoT Dataset Construction Pipeline

Overview. Given a raw robot trajectory, our goal is to convert low-level observations and actions into structured embodied CoT annotations that are useful for action learning. Each trajectory contains language instruction, multi-view RGB observations, proprioceptive states, gripper states, and low-level end-effector actions. We annotate each frame with four groups of reasoning signals: `Understanding`, `Grounding`, `Planning`, and `Acting`. The pipeline is staged rather than fully frame-independent: we first infer episode-level intent and subtask structure, then derive frame-level motion and grounding signals conditioned on this structure. This design reduces temporal inconsistency and avoids asking a language model to rediscover the same task logic at every frame. We demonstrate the entire annotation pipeline in Figure 7.

B.1 Trajectory Segmentation and Episode-Level Planning

For each episode, we first partition the trajectory into coarse semantic segments using end-effector motion and gripper dynamics. Let the robot state at time t be

$$\mathbf{s}_t = [\mathbf{p}_t, \mathbf{R}_t, g_t], \tag{7}$$

where $\mathbf{p}_t \in \mathbb{R}^3$ is the end-effector position, $\mathbf{R}_t \in SO(3)$ is the end-effector orientation, and g_t is the gripper state. We compute translational motion magnitude, rotational change, and gripper transitions:

$$\Delta p_t = \|\mathbf{p}_{t+1} - \mathbf{p}_t\|_2, \qquad \Delta R_t = \|\log(\mathbf{R}_t^\top \mathbf{R}_{t+1})\|_2, \qquad \Delta g_t = |g_{t+1} - g_t|. \tag{8}$$

Candidate boundaries are created when the gripper opens or closes, when motion stops for a sustained window, or when the end-effector changes from approach-like motion to manipulation-like motion. These boundaries are used only as proposals, since low-level motion statistics alone cannot perfectly determine semantic subtask changes.

We then query Qwen3.5-397B [50] once per episode using the task instruction, selected keyframes, and the proposed temporal segments. The VLM generates a globally consistent task goal and

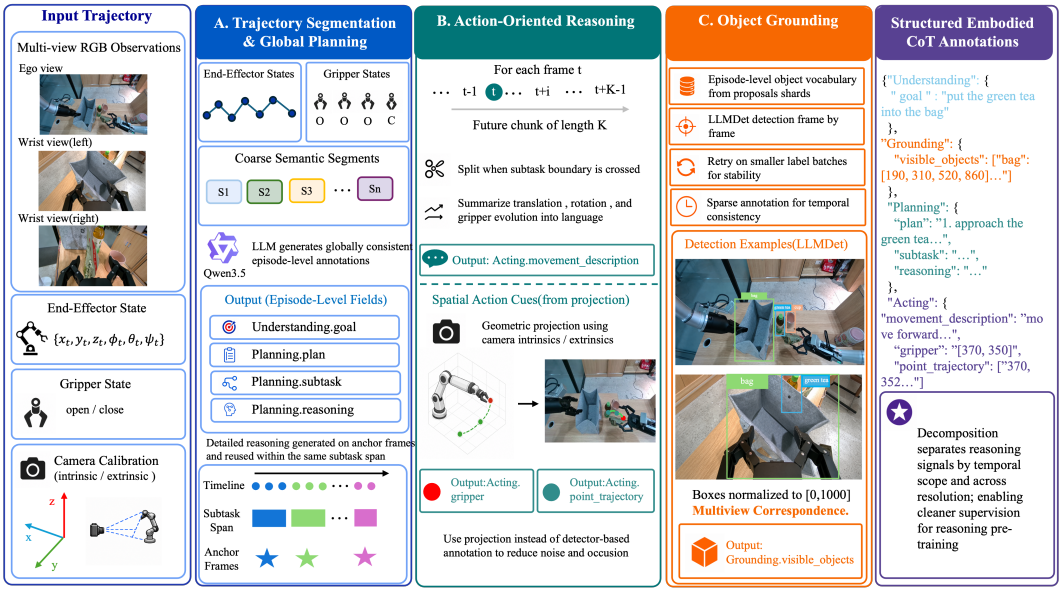

Figure 7: **Embodied CoT dataset construction pipeline.** Raw robot trajectories are converted into structured embodied reasoning supervision through trajectory segmentation, episode-level planning, action-oriented motion annotation, geometric gripper projection, future point-trajectory construction, and sparse object grounding. The resulting annotations provide hierarchical, grounded, and action-oriented CoT signals for scalable VLA pre-training.

subtask sequence, including `Understanding.goal`, `Planning.plan`, `Planning.subtask`, and `Planning.reasoning`. This episode-level generation avoids frame-wise drift: all frames inherit the same global goal and plan, while the current subtask is assigned according to the segment that contains the frame. For long segments, we generate detailed subtask reasoning only on anchor frames and reuse it for neighboring non-anchor frames within the same subtask span. This reduces annotation cost while keeping the reasoning temporally coherent.

B.2 Action-Oriented Reasoning From Future Motion

For each frame, we construct action-oriented CoT from the future motion of the robot. Given a future horizon K, we extract the short-horizon action chunk

$$\mathbf{A}_{t:t+K} = [\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+K-1}], \tag{9}$$

and truncate it if the chunk crosses a subtask boundary. We summarize the chunk into a compact language description stored as `Acting.movement_description`. The description captures the dominant translation direction, rotation trend, and gripper evolution. For example, a chunk can be converted into a phrase such as "move back 3 cm, move left 9 cm, move down 3 cm, keep gripper open." We use thresholded displacement and gripper-state changes to avoid over-describing small jitter:

$$\Delta \mathbf{p}_{t:t+K} = \mathbf{p}_{t+K} - \mathbf{p}_t, \qquad \Delta g_{t:t+K} = g_{t+K} - g_t. \tag{10}$$

Only motion components whose magnitude exceeds a small threshold are verbalized. This makes the movement field stable and action-relevant, rather than a noisy transcription of every small controller fluctuation.

B.3 Geometric Gripper Projection

For datasets with calibrated external cameras and end-effector poses, we annotate `Acting.gripper` through geometric projection rather than detector-based localization. This is important because grippers are often occluded, motion-blurred, or visually ambiguous; directly detecting them from images can introduce large spatial jitter. We apply this projection mainly to static third-person cameras, such as base or front views. For wrist-mounted cameras, the camera moves rigidly with the

end-effector, making the current gripper location nearly fixed in the wrist-camera frame; therefore, we do not use wrist-camera gripper projection as action-oriented supervision.

For a static camera c, let its intrinsic matrix be

$$\mathbf{K}_c = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{11}$$

and let $\mathbf{T}_c^w \in SE(3)$ denote the camera-to-world extrinsic matrix. The corresponding world-to-camera transform is

$$\mathbf{T}_w^c = (\mathbf{T}_c^w)^{-1}. \tag{12}$$

We define a gripper reference point in the end-effector frame, such as the tool center point or a calibrated fingertip midpoint, as

$$\bar{\mathbf{p}}_{\text{grip}}^{ee} = [\mathbf{p}_{\text{grip}}^{ee}, 1]^\top. \tag{13}$$

Its world coordinate at time t is

$$\bar{\mathbf{p}}_{\text{grip}}^{w}(t) = \mathbf{T}_{ee}^{w}(t)\bar{\mathbf{p}}_{\text{grip}}^{ee}, \tag{14}$$

where $\mathbf{T}_{ee}^w(t)$ is the end-effector pose in the world frame. The point is then projected into camera c by

$$\lambda \begin{bmatrix} u_t^c \\ v_t^c \\ 1 \end{bmatrix} = \mathbf{K}_c \begin{bmatrix} \mathbf{I}_{3\times 3} & \mathbf{0} \end{bmatrix} \mathbf{T}_w^c \bar{\mathbf{p}}_{\text{grip}}^{w}(t). \tag{15}$$

Equivalently, if the point in the camera coordinate frame is $\mathbf{p}_{\text{grip}}^c = (X, Y, Z)^\top$, then

$$u_t^c = f_x \frac{X}{Z} + c_x, \qquad v_t^c = f_y \frac{Y}{Z} + c_y. \tag{16}$$

A projection is considered valid only when $Z > 0$ and the pixel lies inside the image. We normalize valid pixel coordinates to a $[0, 1000]$ coordinate system:

$$\tilde{u}_t^c = \left\lfloor 1000 \frac{u_t^c}{W_c} \right\rfloor, \qquad \tilde{v}_t^c = \left\lfloor 1000 \frac{v_t^c}{H_c} \right\rfloor, \tag{17}$$

where W_c and H_c are the image width and height. The normalized coordinate $(\tilde{u}_t^c, \tilde{v}_t^c)$ is stored as `Acting.gripper` for valid external camera views. Invalid projections are omitted rather than replaced by unreliable detector outputs.

B.4 Future Point-Trajectory Annotation

We construct `Acting.point_trajectory` by projecting future gripper positions into the image plane. For each frame t, we select a sequence of future timestamps $\{t + \delta_1, \ldots, t + \delta_M\}$ and project the gripper point at each timestamp using the same camera model:

$$\mathcal{P}_t^c = [(\tilde{u}_{t+\delta_1}^c, \tilde{v}_{t+\delta_1}^c), \ldots, (\tilde{u}_{t+\delta_M}^c, \tilde{v}_{t+\delta_M}^c)]. \tag{18}$$

To make the trajectory stable, we skip consecutive duplicate or near-duplicate projected points, since they often correspond to stationary frames or controller jitter. If fewer than M valid future points remain, we pad the sequence with $[-1, -1]$. This field provides the model with a spatially grounded motion trend, but avoids forcing dense per-frame pixel supervision when the projection is unreliable.

B.5 Object Grounding

For task-relevant object grounding, we use LLMDet [16] to produce bounding boxes. We first build a compact episode-level vocabulary of candidate objects from the instruction, generated plan, subtask descriptions, and detected proposal shards. This step prevents the detector from being queried with an overly large and unstable open vocabulary. For each frame and camera view, we query the detector with small batches of object names and retry failed or low-confidence detections with reduced label sets.

Given a detected box in image coordinates,

$$\mathbf{b} = [x_1, y_1, x_2, y_2], \tag{19}$$

```
"Grounding": {
  "visible_objects": {
    "Base View": {
      "table": [0, 644, 1000, 1000],
      "fridge_open": [0, 0, 771, 696],
      "cola": [146, 321, 296, 602],
      "juice": [402, 208, 502, 604]
    },
    "Front View": {
      "table": [0, 577, 1000, 1000],
      "fridge_open": [258, 144, 888, 654],
      "cola": [625, 504, 673, 596],
      "juice": [692, 479, 727, 608]
    },
    "Wrist View": {
      "table": [0, 0, 1000, 558],
      "fridge_open": [985, 471, 1000, 519]
    }
  }
}
```

```
"Acting": {
  "point_trajectory": {
    "Base View": [
      [459, 23], [458, 52],
      [457, 79], [456, 106],
      [455, 133], [455, 159],
      [454, 182], [453, 209],
      [452, 231], [452, 257]
    ],
    "Front View": [
      [388, 217], [396, 221],
      [407, 225], [418, 229],
      [429, 232], [436, 238],
      [442, 247], [448, 255],
      [454, 263], [460, 270]
    ]
  }
}
```

Figure 8: **Example of multi-view spatial CoT annotation.** The same objects and future gripper motion are represented under different camera views, enabling view-consistent grounding and action-oriented reasoning. Point trajectory is removed from wrist views.

we normalize it to the same $[0, 1000]$ coordinate system:

$$\tilde{\mathbf{b}} = \left[\left\lfloor 1000 \frac{x_1}{W} \right\rceil, \left\lfloor 1000 \frac{y_1}{H} \right\rceil, \left\lfloor 1000 \frac{x_2}{W} \right\rceil, \left\lfloor 1000 \frac{y_2}{H} \right\rceil \right]. \tag{20}$$

The normalized boxes are stored in `Grounding.visible_objects`. Since dense object boxes generated independently for every frame can fluctuate strongly across nearly identical observations, we adopt sparse grounding annotation. Specifically, object boxes are annotated on keyframes or at a fixed sparse interval, while frames without reliable detections omit the field. This choice is deliberate: for reasoning pre-training, missing uncertain grounding is less harmful than imposing inconsistent spatial labels on semantically similar frames.

B.6 Multi-view Correspondence and Bimanual Formatting

A distinctive feature of our dataset is that spatial CoT fields are not stored as a single-view annotation, but as view-indexed multi-view correspondence. To the best of our knowledge, this is the first embodied CoT dataset at this scale to explicitly preserve such multi-view spatial reasoning signals across base, front, and wrist cameras. Specifically, `Grounding.visible_objects`, `Acting.gripper`, and `Acting.waypoint_px` are organized by camera view, so that the same semantic object or future motion trend can be grounded differently under different viewpoints (see Figure 8). This allows the model to learn view-consistent reasoning instead of overfitting to a single camera layout.

For bimanual trajectories, we further preserve arm identity in action-oriented fields. End-effector states, gripper projections, and future waypoint trajectories are computed separately for the left and right arms when both are available. For single-arm data, the same schema is retained with only the active arm populated. This unified format allows single-arm, dual-arm, single-view, and multi-view data to share one language interface for reasoning pre-training, while preserving the spatial structure needed for grounded action learning.

B.7 Quality Control and Filtering

We apply several filtering rules before writing the final annotations. First, trajectories with missing images, invalid states, or inconsistent action lengths are removed. Second, projected gripper points are discarded if they fall outside the image, have negative depth, or jump unrealistically between adjacent frames. Third, object boxes with invalid geometry, extremely small area, or unstable category assignment are removed. Fourth, CoT fields are pruned if they are empty or inconsistent with the available observations. These filters are important because our experiments show that unreliable grounding can cause CoT contamination; therefore, the pipeline favors sparse but trustworthy supervision over dense but noisy labels.

Table 6: **Summary of pre-training robot CoT data.** Statistics are reported after trajectory cleaning, CoT matching, and quality filtering. The total excludes auxiliary VLM-only data, which is mixed only to preserve general multimodal capability.

Dataset	Source	Action Horizon	Trajectories	Samples	Duration (h)	Notes
AgiBot World [1]	Real	30	765,649	204.94M	1879.4	Multi-view, bimanual manipulation
DROID [21]	Real	15	74,682	15.12M	280.0	In-the-wild single-arm manipulation
Fractal [17]	Real	3	86,703	3.70M	342.5	Large-scale tabletop manipulation
BridgeData V2 [51]	Real	5	43,807	1.43M	79.7	Tabletop manipulation and object interaction
MolmoAct [26]	Real	15	7,902	1.10M	10.9	single-arm Franka manipulation
Robot CoT total	–	–	**978,743**	**226.3M**	**2592.5**	Excludes auxiliary VLM-only data
Auxiliary VLM data	Web/VLM	–	–	–	–	Mixed at a small ratio

The final output is a frame-level JSON-style annotation paired with the original trajectory data. A typical sample contains the task instruction, multi-view images, robot state, action chunk, and structured CoT fields:

$$\mathcal{D}_t = \{I_t^{1:C},\ \mathbf{s}_t,\ \mathbf{A}_{t:t+K},\ \texttt{Understanding},\ \texttt{Grounding},\ \texttt{Planning},\ \texttt{Acting}\}. \tag{21}$$

This format supports both explicit CoT supervision and reasoning dropout during training. It also allows different fields to be selectively ablated, enabling controlled analysis of which embodied reasoning signals are helpful, harmful, or scalable.

Overall, the pipeline separates reasoning signals by temporal scope and noise sensitivity. Episode-level fields provide stable semantic intent and task structure; frame-level movement fields connect reasoning to short-horizon control; geometric projection provides cleaner gripper supervision when calibration is available; and sparse object grounding avoids injecting dense detector noise. Based on this pipeline, we construct a large-scale embodied CoT corpus comprising **978,743** trajectories, **226.3M** samples, and **2592.5** hours of data across Bridge [51], Fractal [68], Droid [21], MolmoAct [26], and AgiBot [1] as depicted in Figure. 2. To the best of our knowledge, it is the largest embodied CoT dataset to date and one of the first at this scale to include both multi-view and bimanual settings.

C Training Details

This section provides implementation and training details for all experiments in the paper. We organize the discussion into three groups: (i) controlled embodied CoT studies with autoregressive action-token prediction, (ii) VLM-to-VLA transfer studies, and (iii) ERVLA pre-training and ablations.

C.1 Pre-training Data

Table 6 summarizes the robot CoT data used for ERVLA pre-training. We clean and standardize large-scale robot datasets, then annotate them with structured embodied CoT fields, including task understanding, object grounding, planning, motion descriptions, and action-oriented spatial cues when available. The pre-training mixture contains both single-arm and bimanual manipulation, as well as single-view and multi-view observations. We also mix a small amount of general VLM instruction data to preserve the backbone's multimodal instruction-following ability; these auxiliary VLM samples do not contain robot trajectories and are not counted in the robot CoT totals.

C.2 Post-training Data

In addition to large-scale robot CoT pre-training, we also annotate all downstream post-training data with the same structured ECoT schema. This ensures that benchmark adaptation does not rely on a different prompting or supervision format from pre-training. Specifically, LIBERO and VLABench post-training samples are converted into frame-level annotations containing `Understanding`, `Grounding`, `Planning`, and `Acting` fields. Each sample includes the task goal, visible object boxes, global plan, current subtask, subtask reasoning, movement description, gripper pose, and future waypoint trajectory when available. For multi-view settings, spatial fields are stored as view-indexed dictionaries so that the same object or motion cue can be grounded under different cameras.

Table 7: **Summary of post-training ECoT data.** All datasets are annotated with the same structured ECoT schema, including task understanding, object grounding, planning, subtask reasoning, movement description, gripper pose, and future waypoint trajectory.

Dataset	Trajectories	Samples
VLABench [61]	5,000	575,101
LIBERO-10 [33]	500	138,090
LIBERO-Goal [33]	500	63,728
LIBERO-Object [33]	500	74,507
LIBERO-Spatial [33]	500	62,250
Total	**7,000**	**913,676**

Table 7 summarizes the post-training data used in our experiments. VLABench provides long-horizon and semantically diverse manipulation tasks, while LIBERO provides four post-training suites covering general, goal-conditioned, object-conditioned, and spatial transfer settings. All suites are annotated with the complete ECoT field set and use the same field names as the pre-training corpus. This unified format allows the model to reuse the same reasoning interface during pre-training, post-training, ablation, and evaluation.

Figure. 9 shows one frame-level ECoT annotation from LIBERO-10. The example illustrates three important properties of our post-training format. First, object grounding is multi-view: the same objects are represented separately under `cam0` and `cam1`. Second, action-oriented fields provide both language-level motion description and spatial waypoint supervision. Third, planning fields are episode-consistent: the global plan is shared across the trajectory, while the current subtask and reasoning are attached to each frame through the corresponding subtask span. Non-anchor frames can reuse cached reasoning from an anchor frame, reducing redundant annotation while preserving temporal consistency.

C.3 Controlled CoT Field Studies

The first set of experiments studies which embodied CoT fields are useful for action learning. To isolate the effect of reasoning supervision, we use Qwen3-VL-4B [2] as the unified backbone and adopt the same autoregressive CoT+FAST action-token interface [41] for all variants. Each sample is formatted as a multimodal instruction-following sequence: multi-view observations and the task instruction are provided as input, the model first predicts the selected CoT fields wrapped by `<cot>` and `</cot>`, and then predicts FAST action tokens through next-token prediction. The action horizon, tokenizer, data split, optimizer, and evaluation protocol are kept fixed, so that performance differences mainly reflect the contribution of different CoT fields.

We consider two training regimes. In the direct post-training regime, the model is initialized from Qwen3-VL-4B and trained only on the downstream VLABench data with the selected CoT fields. This setting tests whether a field provides immediate supervision for action learning without additional robot pre-training. In the Bridge pre-training regime, we first pre-train the model on BridgeData V2 [51] annotated with the same structured CoT format, and then post-train it on VLABench. This setting tests whether each CoT field remains beneficial after the model has already learned a basic mapping between embodied reasoning and action tokens. For both regimes, we report changes relative to the corresponding no-CoT baseline.

We also evaluate reasoning dropout. During training, each sample uses explicit CoT with probability $p = 0.5$; otherwise, the CoT span is replaced by an empty `<cot></cot>` block while the model is still supervised to predict the same action tokens. This prevents the policy from relying on explicit reasoning text as a mandatory action prefix and encourages reasoning information to be internalized into action-relevant hidden states. Comparing training with and without dropout allows us to separate useful reasoning supervision from brittle dependence on explicit CoT. These controlled studies show that low-level action-centric fields, such as movement descriptions and end-effector trajectories in 2D image space, are more directly useful for control than high-level task descriptions alone, while dense but noisy grounding fields can cause CoT contamination if used without dropout or field selection.

```
{
  "Understanding": {
    "goal": "turn on the stove and put the moka pot on it"
  },
  "Grounding": {
    "visible_objects": {
      "cam0": {
        "chefmate_8_frypan_1": [457, 480, 1000, 754],
        "moka_pot_1": [395, 527, 688, 789],
        "flat_stove_1": [94, 418, 395, 754]
      },
      "cam1": {
        "chefmate_8_frypan_1": [4, 82, 570, 523],
        "moka_pot_1": [219, 0, 699, 219],
        "flat_stove_1": [672, 102, 1000, 707]
      }
    }
  },
  "Planning": {
    "plan": [
      {"id": 1, "subtask": "approach stove knob"},
      {"id": 2, "subtask": "grasp stove knob"},
      {"id": 3, "subtask": "turn stove knob"},
      {"id": 4, "subtask": "release stove knob and retreat"},
      {"id": 5, "subtask": "grasp moka pot"},
      {"id": 6, "subtask": "place moka pot on stove and retreat"}
    ],
    "subtask_name": "approach stove knob",
    "subtask_reasoning": "The robot is positioned above the counter with the stove knob, moka pot, and
     pan visible below. No actions have been completed yet as the gripper remains open. The robot needs
     to lower its arm to approach the stove knob for interaction."
  },
  "Acting": {
    "movement_description": "move left 4 cm, keep gripper open",
    "gripper_pose": {
      "cam0": [508, 234]
    },
    "waypoint_px": {
      "cam0": [
        [504, 234], [500, 234], [496, 234], [492, 234], [484, 234],
        [480, 234], [473, 234], [469, 234], [461, 234], [453, 234]
      ]
    },
    "subtask_end": 61
  }
}
```


Figure 9: **Example frame-level ECoT annotation from LIBERO-10.** The annotation contains multi-view object grounding, episode-level planning, frame-level subtask reasoning, and action-oriented spatial cues. Coordinates are normalized to the $[0, 1000]$ image coordinate system.

C.4 VLM-to-VLA Transfer Study

The second set of experiments studies whether stronger VLMs become stronger VLAs when embodied CoT is used as a transfer interface. Following the controlled comparison principle of VLM4VLA, we keep the VLA adaptation recipe fixed across all backbones: the same StarVLA [12] training pipeline, FAST [41] action-token interface, action horizon, data preprocessing, optimizer family, augmentation pipeline, and evaluation tasks are used for every model. Each backbone is trained under two settings: without CoT supervision and with full embodied CoT supervision under reasoning dropout. This design isolates whether embodied CoT improves the conversion of pretrained semantic priors into action-relevant representations, rather than confounding the comparison with different policy heads, action tokenizers, or data processing strategies.

For each VLM, we preserve its native multimodal instruction format as much as possible. Images are inserted using the model-specific vision-token convention, while the robot instruction and optional embodied CoT fields are formatted as the text context. For CoT-enabled training, the assistant target first contains the structured embodied CoT block and then the FAST action tokens; for the no-CoT setting, the CoT span is replaced by an empty `<cot></cot>` block while the same action tokens are still supervised. Reasoning dropout is applied with probability $p = 0.5$, so the model sees both explicit-CoT and no-CoT inputs during training. This prevents the policy from treating CoT

Table 8: **Training protocol for the VLM-to-VLA transfer study.** All backbones use the same StarVLA adaptation recipe; only the VLM backbone and CoT setting are changed.

Item	Setting
Framework	StarVLA unified training pipeline
Action interface	FAST action-token prediction
Backbones	PaliGemma-2, Florence-2, Cosmos-Reason2, Gemma, Qwen2.5-VL, Qwen3-VL
CoT settings	w/o CoT; w/ full ECoT + reasoning dropout
Reasoning dropout	$p = 0.5$
Trainable modules	Vision encoder, LLM, token embeddings, and action-token head
Observation	Current visual observation with task instruction
Image preprocessing	Resize with `max_pixels`=90k, random crop ratio 0.95, photometric distortion
Action horizon	10
Action representation	FAST tokens from 7-DoF end-effector action chunks
Action-token loss	Next-token prediction on FAST action tokens
Optimizer	AdamW
Adam betas	$(0.9, 0.95)$
Weight decay	0.1
Peak learning rate	1×10^{-4}
Minimum learning rate	5×10^{-7}
Warmup steps	2,000
LR schedule	Cosine decay
Gradient clipping	1.0
Precision	bfloat16 mixed precision
LIBERO training steps	10,000
VLABench training steps	20,000
Checkpoint interval	2,000 steps

Table 9: **VLM backbone capability order used in the VLM-to-VLA transfer study.** Models are roughly ordered from weaker to stronger general VLM capability based on available public VLM evaluation results. Scores are reported when comparable results are available.

Backbone	Param.	Avg.	MMBench	MMStar	MMMU	MathVista	OCRBench	AI2D	HallusionBench
PaliGemma-2-3B [3]	3B	46.5	65.6	48.3	34.9	28.5	614	68.3	32.2
Florence-2-large [55]	0.77B	52.0	69.5	–	39.0	–	690	–	35.0
Cosmos-Reason2-2B [37]	2B	61.8	74.0	–	48.5	57.0	–	76.0	–
Gemma-4-E2B-it	2B	63.2	75.5	56.8	–	59.5	–	78.0	–
Qwen2.5-VL-3B [44]	3B	64.5	76.8	56.3	51.2	61.2	828	81.4	46.6
Qwen3-VL-2B [2]	2B	66.5	78.4	58.3	53.4	61.3	858	76.9	51.4
Qwen2.5-VL-7B [44]	7B	70.9	82.2	64.1	58.0	68.1	888	84.3	51.9
Qwen3-VL-4B [2]	4B	74.94	83.9	69.8	67.4	73.7	881	84.1	57.6
Qwen3-VL-8B [2]	8B	**76.94**	**84.5**	**70.9**	**69.6**	**77.2**	**896**	**85.7**	**61.1**

as a mandatory autoregressive prefix and instead encourages the VLM hidden states to internalize reasoning information useful for action prediction.

All VLM-to-VLA transfer runs fine-tune the VLM backbone together with the FAST action-token prediction head, rather than freezing the vision encoder or language model. Following the controlled comparison principle of VLM4VLA, we use the same StarVLA adaptation recipe across all backbones. To keep the comparison fair, we fix the action-token interface, data split, optimizer, augmentation, checkpointing, and evaluation protocol (see Table 8). For this transfer study, we use short benchmark-level training schedules: 10,000 steps on LIBERO and 20,000 steps on VLABench.

We evaluate a diverse set of VLM backbones spanning different model families and capability levels, including PaliGemma-2-3B [3], Florence-2-large [55], Cosmos-Reason2-2B [37], Gemma-4-E2B-it, Qwen2.5-VL-3B/7B [44], and Qwen3-VL-2B/4B/8B [2]. Table 9 orders these backbones by general VLM capability. Since all models share the same action-token decoder and training recipe, this study directly tests whether embodied CoT makes pretrained VLM capability more transferable to VLA control.

Table 10: **Large-scale ERVLA pre-training hyperparameters.**

Item	Value
Backbone	Qwen3-VL-4B-Instruct
Action horizon	30
Action / state shape	30×60 / 1×60
Number of choices	5
DiT layers	36
Reasoning dropout	$p_{\text{cot}} = 0.5$
CoT fields	Understanding, Grounding, Planning, Acting
Obs. length / interval	1 / 1
Image augmentation	Resize, random crop, photometric distortion
Max image pixels	90,000
Token packing	Enabled
Max packed length	17,600 tokens
Training steps	120,000
Batch size	64
Precision	bfloat16 mixed precision
Optimizer	DeepSpeed FusedAdam / AdamW-style
Adam betas	$(0.9, 0.95)$
Weight decay	0.1
Learning rate	5×10^{-5}
Warmup steps	2,000
Gradient clipping	1.0
Distributed strategy	DeepSpeed
Checkpoint / validation interval	20,000 / 10,000 steps

C.5 Large-scale ERVLA Pre-training

ERVLA is trained in two stages. First, we perform large-scale embodied CoT pre-training on a mixture of open robot datasets. ERVLA uses Qwen3-VL-4B as the reasoning backbone, a choice branch for candidate continuous action chunks, and a DiT policy for continuous action generation. The final model is trained end-to-end: diffusion loss, choice loss, score loss, and CoT token loss are optimized jointly unless an ablation explicitly blocks gradients.

The model receives semantic inputs consisting of images, instruction, and optional CoT, followed by trainable control-query tokens. The VLM produces hidden states and key-value caches. The choice branch reads action-query hidden states to predict N candidate action chunks and their scores, while DiT generates the final continuous action chunk by attending to the knowledge-truncated semantic-prefix cache. In all ERVLA experiments, we use $N = 5$ choices and a 36-layer DiT; full pre-training hyperparameters are provided in Table 10.

For large-scale reasoning pre-training, we train on a mixture of embodied CoT data constructed from Bridge [51], Fractal [68], Droid [21], MolmoAct [26], and AgiBot [1]. Compared with the base no-CoT pre-training configuration, we enable CoT loading with the same structured fields used in the main paper and apply reasoning dropout with $p_{\text{cot}} = 0.5$.

For single-arm datasets, actions are composed into a padded action vector. For dual-arm datasets such as AgiBot, we compose left-arm and right-arm end-effector position, axis-angle rotation, and gripper signals into a unified action vector. In the full pre-training configuration, actions are padded to 60 dimensions and states are padded to 60 dimensions, allowing a shared action interface across single-arm and dual-arm data. We use streaming MDS datasets and token packing to improve training efficiency. Packing allows multiple shorter multimodal samples to be concatenated into a long sequence up to the maximum training length, reducing padding waste and increasing effective token throughput.

As shown in Table 11, we use a nominal sampling mixture to balance large-scale coverage and dataset diversity. AgiBot receives the largest sampling weight due to its scale, while smaller auxiliary datasets such as MolmoAct are sampled at lower rates to avoid over-representation.

Table 11: **Pre-training data mixture.** We report nominal sampling weights used to mix different embodied CoT sources during pre-training.

Dataset	Traj.	Samples	Weight
AgiBot [1]	765.6K	204.9M	0.518
DROID [21]	74.7K	15.1M	0.180
Fractal [68]	86.7K	3.7M	0.120
Bridge [51]	43.8K	1.4M	0.100
MolmoAct [26]	7.9K	1.1M	0.082
Total	**978.7K**	**226.3M**	**1.00**

C.6 Post-training

For LIBERO post-training, we train on LIBERO-10, LIBERO-Spatial, LIBERO-Object, and LIBERO-Goal. We use delta action targets from the dataset and compose them into the same 60-dimensional padded action vector used in ERVLA pre-training. The action horizon is 10. The robot state contains the end-effector and gripper state, padded to 60 dimensions. We use Gaussian normalization for actions and states. The same CoT fields are enabled with $p_{\text{cot}} = 0.5$.

The LIBERO post-training configuration is used both for the final LIBERO-Plus transfer evaluation and for ablations. All models are post-trained only on LIBERO and evaluated zero-shot on LIBERO-Plus. This setting tests whether reasoning pre-training improves robustness beyond the original LIBERO distribution.

For VLABench post-training, we train on the VLABench CoT dataset. The action horizon is 10. Unlike LIBERO, VLABench uses delta end-effector actions derived from absolute end-effector states and actions. We wrap Euler angles when computing rotational deltas and compose each action into the same 60-dimensional padded action space. We use Gaussian normalization for both actions and proprioceptive states, which are also padded to 60 dimensions. The `use_ideal_length` option is enabled so that training samples satisfy the required action horizon for choice and diffusion supervision.

C.7 ERVLA Ablations

We use the same pre-training and post-training data for ERVLA ablations unless otherwise stated. The ablations are designed to isolate the role of the choice branch, end-to-end gradient flow, and knowledge truncation.

No Choice (End-to-End). This variant removes the auxiliary choice branch and trains the VLM+DiT model end-to-end. Diffusion loss is still allowed to update the VLM backbone. This tests whether end-to-end continuous-action training alone is sufficient without candidate-level action discrimination.

No Choice + Knowledge Insulation. This variant removes the choice branch and blocks gradients from the continuous action expert to the VLM backbone. It resembles planning-action separation in which the VLM representation is preserved but receives weaker feedback from action generation. This tests whether preserving pretrained knowledge by stopping gradients is preferable to end-to-end action shaping.

Choice + No Knowledge Truncation. This variant keeps the choice branch but removes knowledge truncation. DiT is allowed to condition on the full VLM cache, including appended state and action-query turns. This tests whether DiT exploits synthetic control-query tokens as shortcuts when the full cache is exposed.

Full ERVLA. The full model combines explicit CoT supervision, reasoning dropout, the choice policy branch, knowledge truncation, and end-to-end diffusion training. DiT conditions only on the

Table 12: Ablation of ECoT components on VLABench [61]. The upper part lists the CoT fields used, and the lower part reports performance on different VLABench tracks with and without pretraining on the CoT field data of the Bridge dataset [51].

CoT Field		Ablation Setting														
Descriptive	Goal		✓								✓					✓
	Planning			✓							✓					✓
	Subtask				✓						✓		✓		✓	✓
	Movement					✓					✓			✓	✓	✓
	Reasoning						✓				✓		✓	✓		✓
Coordinate	Gripper							✓				✓				✓
	Point traj.								✓			✓			✓	✓
	Bounding box									✓		✓				✓
w/o Pretraining	In-dist.	30.2	29.0	29.4	29.6	34.2	29.2	29.4	35.0	28.8	35.0	33.0	28.8	35.4	37.6	**38.4**
	Category	20.2	19.0	19.4	19.6	24.4	19.2	19.6	25.0	18.8	25.0	22.8	18.8	25.4	27.6	**28.4**
	Commonsense	7.8	6.6	7.0	7.2	12.0	6.8	7.2	12.6	6.4	12.8	10.6	6.6	13.0	15.2	**16.0**
	Instruction	16.4	15.2	15.6	15.8	20.4	15.4	15.6	21.2	15.0	21.4	19.0	15.2	21.6	23.8	**24.6**
	Texture	20.4	19.2	19.6	19.8	24.4	19.4	19.6	25.2	19.0	25.2	23.0	19.0	25.6	27.8	**28.6**
	Avg.	19.0	17.8	18.2	18.4	23.1	18.0	18.3	23.8	17.6	23.9	21.7	17.7	24.2	26.4	**27.2**
w/ Pretraining	In-dist.	35.4	34.6	34.8	34.8	37.4	34.4	29.8	36.8	29.2	38.4	33.0	33.8	38.4	**39.0**	37.8
	Category	25.8	25.0	25.2	25.2	27.8	24.8	20.2	27.2	19.6	28.8	23.4	24.2	28.8	**29.6**	28.4
	Commonsense	15.2	14.4	14.6	14.4	17.2	14.2	9.6	16.6	9.2	18.4	12.8	13.6	18.2	**18.8**	17.6
	Instruction	22.8	22.0	22.4	22.0	24.8	22.0	17.2	24.2	16.8	26.0	20.4	21.2	25.8	**26.6**	25.4
	Texture	26.8	26.0	26.4	26.0	28.8	26.0	21.2	28.2	20.6	29.8	24.4	25.2	29.8	**30.4**	29.2
	Avg.	25.2	24.4	24.7	24.5	27.2	24.3	19.6	26.6	19.1	28.3	22.8	23.6	28.2	**28.9**	27.7
w/ Pretraining + Dropout	In-dist.	35.4	34.8	35.0	34.8	37.2	34.8	34.6	38.4	34.4	38.2	36.8	34.4	38.6	**39.8**	39.4
	Category	25.8	25.2	25.6	25.2	27.6	25.2	25.0	28.8	24.8	28.8	27.2	24.8	29.0	**30.2**	29.8
	Commonsense	15.2	14.6	14.8	14.6	17.2	14.6	14.4	18.2	14.2	18.2	16.6	14.2	18.4	**19.6**	19.2
	Instruction	22.8	22.2	22.6	22.4	24.8	22.2	22.0	25.8	21.8	25.6	24.4	22.0	26.0	**27.2**	26.8
	Texture	26.8	26.2	26.4	26.4	28.6	26.2	26.0	29.8	25.8	29.6	28.4	26.0	30.0	**31.2**	30.8
	Avg.	25.2	24.6	24.9	24.7	27.1	24.6	24.4	28.2	24.2	28.1	26.7	24.3	28.4	**29.6**	29.2

semantic-prefix cache, while gradients from action generation still update the VLM backbone. This allows CoT learning, choice-based action shaping, and continuous action generation to co-adapt.

C.8 Practical Notes

All training runs use bfloat16 mixed precision and DeepSpeed distributed training. We use gradient clipping with a maximum norm of 1.0 and cosine warmup scheduling. For post-training, checkpoints are saved every 10,000 steps. For large-scale pre-training, checkpoints are saved every 20,000 steps. We disable progress bars during distributed runs to reduce logging overhead. For packed training, samples are grouped into long sequences under the maximum token budget; samples that exceed the budget are skipped or deferred by the dataloader, while valid shorter samples are packed together to improve token utilization.

At inference time, ERVLA supports reasoning dropout. The model can use explicit CoT, sparsely refreshed CoT, or `/no_cot` input depending on the evaluation protocol. Unless otherwise specified, benchmark results are reported under the efficient setting without mandatory test-time CoT decoding.

D Detailed Experimental Results

D.1 Simulation Results

This section provides the full simulation results corresponding to the controlled studies and benchmark comparisons in Sec. 3.1 and Sec. 3.2. We report three complementary analyses. First, Table 12 gives the full field-level ECoT ablation on VLABench [61]. Second, Table 13 reports the VLM-to-VLA transfer results across LIBERO [33] and VLABench under the same FAST action interface. Third, Table 5 reports the detailed real-world breakdown used in Sec. 3.3; although it is not a simulation benchmark, we include it here for completeness so that all detailed quantitative results are collected in one appendix section. We visualize representative simulation rollouts in Figures 10 and 11.

Field-level ECoT ablations on VLABench. Table 12 expands the analysis in Sec. 3.1 by reporting every CoT-field configuration across five VLABench tracks. All variants use the same Qwen3-VL-4B

Table 13: Comparison on **LIBERO** and **VLABench**. All values are reported in percentage (%). **F** denotes Fast, and **C+F** denotes CoT+Fast. The best result in each column is highlighted in bold.

Model	LIBERO								VLABench									
	Spatial		Object		Goal		Long		In-dist.		Category		Common		Instruction		Texture	
	F	C+F	F	C+F	F	C+F	F	C+F	F	C+F	F	C+F	F	C+F	F	C+F	F	C+F
PaliGemma-2-3B	90.2	86.4	87.6	86.0	82.4	78.4	75.8	70.8	26.8	15.4	18.6	12.2	13.8	10.2	16.2	17.0	22.2	15.2
Florence-2-large	87.6	86.6	90.2	86.2	78.2	78.6	72.8	70.2	32.8	15.0	13.4	11.8	19.2	10.0	12.4	16.8	15.6	15.0
Cosmos-Reason2-2B	92.0	89.6	89.6	90.6	84.6	84.2	76.0	76.8	21.8	23.8	23.8	20.6	12.6	15.4	28.4	**33.4**	18.8	22.0
Qwen2.5-VL-3B	94.2	93.0	92.0	93.8	86.6	88.2	80.8	81.4	19.6	27.8	16.8	24.2	20.4	15.0	14.2	24.6	24.6	27.6
Gemma-4-E2B-it	90.6	92.6	93.0	93.0	85.8	87.4	78.4	80.6	34.0	37.4	21.6	28.6	16.0	20.4	21.0	30.2	17.4	29.4
Qwen3-VL-2B	95.0	95.6	94.8	**97.4**	89.6	92.0	84.8	87.2	26.8	28.8	14.6	25.2	21.2	13.9	12.6	26.2	18.2	28.8
Qwen2.5-VL-7B	91.0	95.2	94.0	95.4	87.2	90.4	80.6	84.8	28.4	37.8	15.8	28.6	14.8	20.6	18.8	30.0	14.2	30.4
Qwen3-VL-4B	95.8	96.4	96.8	96.6	91.6	91.8	86.0	**88.6**	38.2	40.0	22.8	29.8	18.8	13.6	16.4	26.8	25.8	**31.0**
Qwen3-VL-8B	90.8	**97.2**	97.0	97.2	86.4	**93.2**	83.6	87.0	24.8	**40.8**	16.6	**32.4**	15.2	**22.6**	13.6	25.2	20.0	29.2

backbone and the same autoregressive CoT+FAST action-token interface, so the comparison isolates the effect of the reasoning fields. The upper block specifies which ECoT fields are enabled, while the lower blocks report results under three regimes: direct post-training without Bridge pre-training, Bridge-based ECoT pre-training, and Bridge-based ECoT pre-training with reasoning dropout. The results show a clear pattern: high-level descriptive fields such as goal, planning, subtask, and reasoning are not sufficient when used alone, whereas action-centric fields such as movement description and point trajectory provide stronger gains. However, dense coordinate fields such as gripper position and bounding boxes become harmful under pre-training when noisy labels are repeatedly imposed, revealing CoT contamination. Reasoning dropout mitigates this degradation by preventing the model from over-relying on explicit and potentially noisy CoT fields.

VLM-to-VLA transfer under a fixed action interface. Table 13 provides the detailed results for the VLM-to-VLA transfer study. Each backbone is trained with the same StarVLA protocol and FAST action-token interface, and each method is evaluated with and without full ECoT supervision. The LIBERO results show that many VLMs already achieve strong in-distribution performance, but ECoT improves the stronger Qwen-series backbones more consistently, especially on object, goal, and long-horizon task suites. The VLABench results are more discriminative: without CoT, stronger VLMs do not always transfer reliably to harder semantic and out-of-distribution tracks; with CoT+FAST, the Qwen3-VL series obtains clearer gains, especially on in-distribution, cross-category, commonsense, instruction, and texture generalization tracks. This supports our conclusion that embodied CoT acts as a transfer interface: it helps convert pretrained semantic priors into action-relevant representations, rather than merely adding extra text supervision.

Connection to main benchmark results. The detailed tables also clarify why ERVLA is designed beyond autoregressive CoT+FAST. Table 12 shows that explicit CoT is useful only when the right fields are selected and noisy grounding is regularized. Table 13 shows that ECoT improves VLM-to-VLA transfer, but still remains constrained by autoregressive action decoding. These observations motivate the ERVLA interface evaluated in Tables 3 and 4: CoT is used as representation-shaping supervision, while continuous action generation is handled through the choice branch, knowledge truncation, and DiT-based flow modeling. Together, the detailed simulation results support the main claim that embodied CoT becomes scalable when it is internalized into action-aware representations rather than treated as a mandatory action prefix.

D.2 Real-world Results

We provide the complete real-world task suite and evaluation results in Tables 14 and 5. The robot platform and representative rollouts are described in Sec. 3.3; here we additionally report the full task instructions, evaluation protocol, training setup, and quantitative breakdown across difficulty tiers. As shown in Table 14, each method is evaluated on 20 tasks spanning *Basic*, *Distractors*, *Semantic*, and *Long-horizon* settings, with five trials per task, resulting in 100 real-world rollouts per method. We report two metrics in Table 5: success rate (SR), which measures whether the full instruction is completed, and progress score (PS), which assigns partial credit for correctly completed subtasks



Figure 10: **Visualization of VLABench evaluation results.**

such as opening the drawer, selecting the correct object, placing an object into the target container, or completing one stage of a multi-object instruction. This two-metric protocol is important because many failures in real-world manipulation are not binary: a policy may correctly identify the target object but fail to place it, complete the first object in a multi-object instruction but lose track of the remaining objects, or execute the correct motion primitive but choose an incorrect container.

We collect approximately 10 hours of demonstrations covering object pick-and-place with toy cars, fruit models, cans, bottles, and drawer opening and closing. The real-world demonstrations are annotated with the same embodied CoT schema used in the main simulation experiments, including task understanding, object grounding, planning, subtask reasoning, movement description, gripper pose, and future waypoint trajectory when available. This keeps the real-world adaptation setting consistent with our pre-training and benchmark post-training setup, so the comparison reflects whether the learned reasoning-action interface transfers to physical execution rather than whether a new real-world prompt format is introduced. All models are fine-tuned on the same real-world demonstrations using the same optimization schedule: 8 NVIDIA A100 GPUs, per-GPU batch size 16, and 20,000

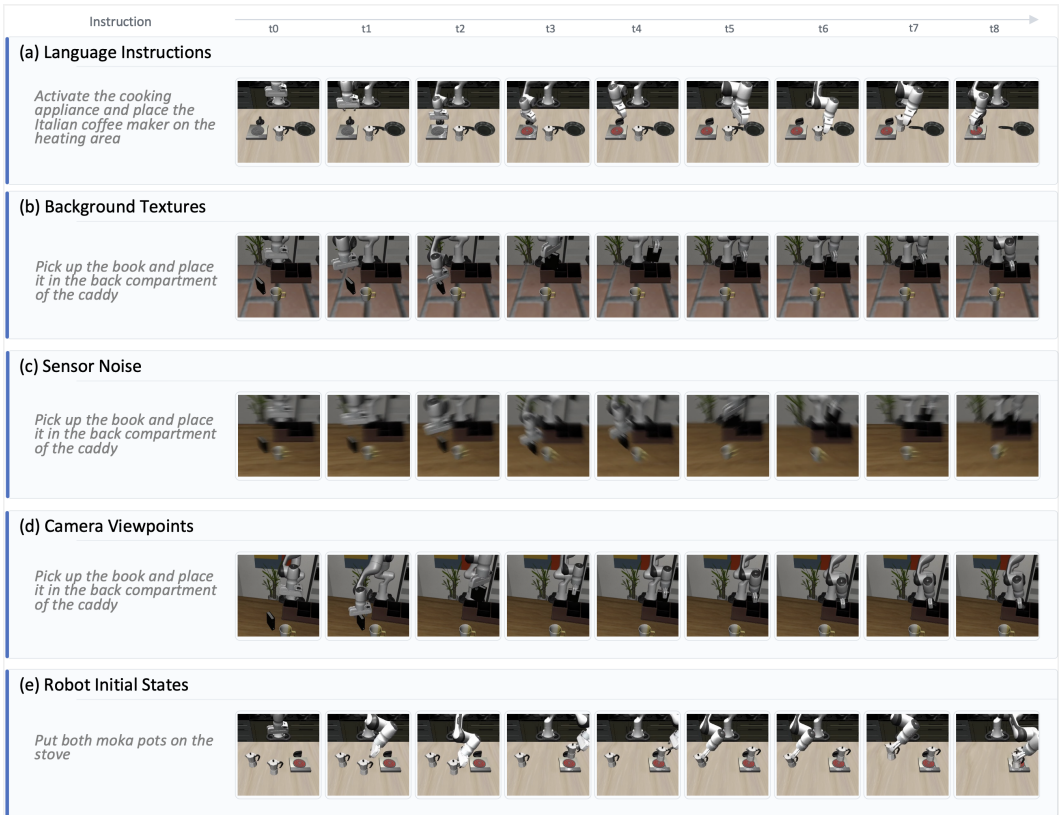

Figure 11: **Visualization of LIBERO-PLUS evaluation results.**

Table 14: **Real-world task suite.** We evaluate 20 tasks across four difficulty tiers. Each task is tested with five trials.

Tier	Task family	Instruction
Basic	Drawer placing	Put the blue toy car into the drawer.
	Drawer placing	Put the red toy car into the drawer.
	Drawer placing	Put the yellow toy car into the drawer.
	Table clearing	Put the empty can into the basket.
	Table clearing	Put the bottle into the basket.
Distractors	Drawer placing	Put the toy car into the drawer. (with fruit models)
	Drawer placing	Put the blue toy car into the drawer. (with other toy cars)
	Drawer placing	Put the toy car into the drawer. (with cans and bottles)
	Table clearing	Put the can into the basket. (with the fruits and other bottles)
	Table clearing	Clear the bottle from the table. (with toy cars)
Semantic	Drawer placing	Put away the object that is not a fruit.
	Drawer placing	Put the toy into the lower drawer.
	Drawer placing	Put the toy into the drawer above the lower one.
	Table clearing	Remove the item that should be thrown away.
	Table clearing	This table is too messy.
Long-horizon	Drawer placing	Put the non-toy items into the upper drawer, and put the toys into the second drawer.
	Drawer placing	Put all toy cars into the upper drawer while leaving the fruits on the table.
	Drawer placing	Put all fruits into the lower drawer.
	Table clearing	Clear all waste items from the table and place them into the basket.
	Clearing and Placing	Put the non-toy items into the upper drawer, and put the toys into the second drawer.

training steps. To reduce execution jitter and make continuous-action rollouts comparable, we apply the same temporal action ensemble strategy [30] to all continuous-action methods.

The four difficulty tiers separate different sources of real-world generalization error. *Basic* evaluates clean pick-and-place and drawer interaction with explicit object names. *Distractors* adds irrelevant

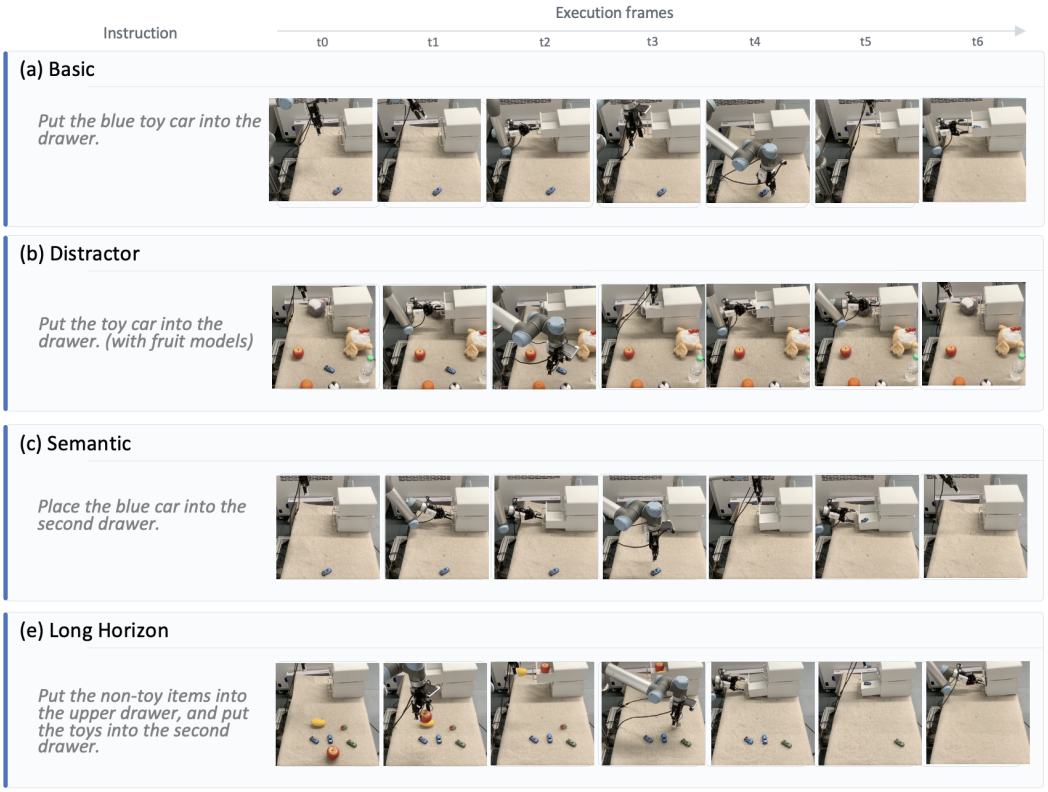

Figure 12: **Visualization of real-world evaluation results.**

or visually similar objects to test target selectivity under clutter. *Semantic* replaces direct object names with category, function, or spatial-reference descriptions, requiring the policy to infer the intended object or container. *Long-horizon* combines semantic grounding with multi-object, multi-step execution, testing whether the policy can preserve progress over an extended task.

Table 5 shows that the main performance gap appears beyond clean manipulation. On the *Basic* tier, $\pi_{0.5}$ and ERVLA perform similarly, reaching 97/98 and 96/97 SR/PS, respectively, suggesting that strong continuous-action policies already handle simple low-level manipulation well. However, under *Distractors*, *Semantic*, and *Long-horizon* settings, the policy must resolve ambiguity, ignore irrelevant objects, and maintain task intent over time, making reasoning-aware representations more important.

ERVLA shows the clearest gains on semantic and long-horizon tasks (see Figure. 12). On the *Semantic* tier, it improves over $\pi_{0.5}$ from 31 to 42 SR and from 45 to 58 PS, indicating better conversion of semantic intent into action-relevant representations. On the *Long-horizon* tier, ERVLA improves PS from 38 to 55, showing that it more often completes meaningful intermediate stages even when the full instruction is not perfectly solved. This supports our design: embodied CoT pre-training provides subtask and movement-level structure, while reasoning dropout encourages these signals to be internalized rather than emitted as fragile test-time text.

Overall, the real-world results show that explicit reasoning alone is insufficient: ECoT suffers from autoregressive latency and error propagation, while visual or latent reasoning in WorldVLA and UniVLA can obscure fine-grained semantic grounding. ERVLA instead uses embodied CoT as training supervision, reasoning dropout to avoid dependence on explicit inference-time traces, and cache-conditioned continuous action generation to preserve both semantic grounding and smooth control.

E Limitations and Future Work

Our work shows that embodied CoT is most useful when it is absorbed into action-relevant representations rather than exposed as a mandatory verbal trace. This also reveals a central limitation: the value of reasoning pre-training is bounded by the quality of the reasoning substrate. In our pipeline, goals, plans, movement descriptions, object boxes, gripper locations, and point trajectories provide a structured bridge from semantics to control, but they are not equally reliable. Linguistic fields are usually robust but sometimes underspecified; dense grounding fields are more directly actionable but more vulnerable to detector errors, calibration bias, and occlusion. Reasoning dropout and knowledge truncation reduce the damage of imperfect supervision, yet they do not remove the need for better annotation quality. A natural next step is to make uncertainty part of the annotation itself, so that the model learns not only what to reason about, but also when each reasoning signal should be trusted.

Our study also treats reasoning primarily as an offline pre-training signal. This makes inference efficient and avoids mandatory test-time CoT decoding, but leaves open how much reasoning should be dynamically refreshed during execution. Long-horizon manipulation may require memory of completed subtasks, recovery from failed attempts, or clarification when the scene does not match the instruction. Our results suggest that the answer is unlikely to be simply "more explicit reasoning." Future systems should decide when to verbalize, when to internalize, and when to act directly. In this sense, our work points toward a practical form of embodied reasoning: not a robot that talks through every motion, but one whose actions carry the structure of reasoning even when no reasoning text is produced.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's main contributions: constructing a large-scale embodied CoT dataset, analyzing which CoT signals are useful or harmful for action learning, identifying CoT contamination, proposing ERVLA, and evaluating it on simulation and real-world robot tasks. The claims are supported by controlled ablations, benchmark comparisons on LIBERO-Plus and VLABench, scaling studies, and real-world experiments. The paper also discusses limitations and does not claim unrestricted generalization beyond the evaluated manipulation settings.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper includes a dedicated Limitations and Future Work section. It discusses the dependence of reasoning pre-training on annotation quality, the unreliability of dense grounding signals such as detector-based boxes or projected gripper locations, the remaining need for uncertainty-aware annotations, and the open question of when reasoning should be dynamically refreshed during long-horizon execution.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [No]

   Justification: The paper does not present formal theoretical results, theorems, or proofs. Its contributions are empirical, architectural, and dataset-oriented, supported by controlled experiments, ablations, and benchmark evaluations.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper and appendix describe the dataset construction pipeline, annotation format, model architecture, training objectives, hyperparameters, pre-training and post-training data, ablation settings, evaluation benchmarks, real-world task suite, and metrics.The code, dataset, annotation pipeline, and checkpoints are available on our project website.

   Guidelines:

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

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The code, dataset, annotation pipeline, and model checkpoints are available on our project website.

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The appendix provides detailed training settings for controlled CoT field studies, VLM-to-VLA transfer studies, ERVLA pre-training, post-training, and ablations. It reports data mixtures, action horizons, model backbones, optimizer settings, learning rates, warmup schedules, precision, checkpoint intervals, reasoning dropout settings, and benchmark-specific training steps. The evaluation section describes LIBERO, LIBERO-Plus, VLABench, and the real-world task suite.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper reports average performance across benchmark suites, tracks, task tiers, and real-world trials, but does not include formal statistical significance tests, confidence intervals, or error bars for all experiments. For real-world experiments, the paper specifies the number of trials per task and reports averaged success rate and progress score, but does not conduct significance testing.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper reports the major compute setup for training and real-world fine-tuning, including GPU type and count for real-world training, distributed training strategy, mixed precision, batch sizes or effective training settings, and training steps. The appendix also provides pre-training and post-training schedules, checkpoint intervals, and training configurations sufficient to estimate compute requirements.

   Guidelines:

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: The work studies robotic manipulation using existing public robot datasets, auto-annotated embodied CoT data, simulation benchmarks, and controlled real-world robot experiments. The paper does not involve deceptive data collection, human-subject experiments, or unsafe deployment claims. The release plan focuses on reproducible research assets.

Guidelines:

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

10. **Broader impacts**

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

Answer: [N/A]

Justification: The work is primarily a foundational study of embodied reasoning, robot learning datasets, and VLA model training. It does not target a specific deployed application or user-facing product. The potential impacts are indirect and mainly concern future research on generalizable robot manipulation.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The released assets are intended for robot manipulation research and are based on robot trajectories, annotations, training code, and model checkpoints. The paper does not introduce high-risk scraped web datasets, image-generation models, or general-purpose language models requiring special misuse safeguards beyond standard research release practices.

Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The paper cites the existing datasets, models, and benchmarks used in the work, including BridgeData V2, DROID, Fractal, MolmoAct, AgiBot, LIBERO, LIBERO-Plus, VLABench, Qwen3-VL, StarVLA, FAST, and other baselines. The release package will include documentation for the derived dataset, code, and model checkpoints, including references to the original assets.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces a new large-scale embodied CoT corpus, an annotation pipeline, training code, and model checkpoints. The dataset construction pipeline, annotation schema, filtering rules, multi-view formatting, bimanual formatting, pre-training data mixture, and post-training data are documented in the paper and appendix. The paper states that the dataset, annotation pipeline, code, and checkpoints are available on our project website..

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.

- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or human-subject studies. The real-world experiments are robot manipulation evaluations conducted by the authors, and no participant study or crowdworker data collection is involved.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve human subjects, participant data, or crowdsourcing experiments. Therefore, IRB approval or equivalent human-subject review is not applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: LLM/VLM usage is central to the method. The paper describes the use of Qwen3.5 for episode-level annotation, VLM backbones such as Qwen3-VL for ERVLA, and structured embodied CoT supervision generated through the annotation pipeline. These uses are part of the scientific contribution rather than only writing or editing assistance.

    Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.