STRUCTURED LATENT REASONING FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models can benefit from intermediate reasoning, but explicit chain-of-thought reasoning incurs substantial autoregressive decoding overhead in closed-loop control. Existing latent reasoning methods reduce this cost through continuous latent representations, yet leave the structured reasoning roles of explicit traces implicit. We propose Structured Latent Reasoning (SLR), which organizes latent representations into semantically distinct reasoning units corresponding to different forms of intermediate reasoning and supervises them through auxiliary text decoding. Because the relevance of these units varies across manipulation transitions, we further introduce transition-adaptive learning, which reweights their supervision according to relevance signals from the action policy. At inference, SLR operates directly on the structured latent units, avoiding autoregressive reasoning generation. SLR achieves 73.7% average success on SIMPLER, improving over LaRA-VLA by 4.9 points, and reaches 79.0% average task completion on real-robot tasks while maintaining gains under distribution shift. SLR runs at 341 ms per control step, combining the structured semantics of explicit reasoning with the inference efficiency of latent reasoning.

1 INTRODUCTION

Vision-language-action (VLA) models (Zitkovich et al., 2023; Kim et al., 2024; Black et al., 2026) have recently emerged as a unified framework for mapping visual observations and language instructions to robot actions. Many manipulation tasks require the policy to reason about task decomposition, object grounding, motion specification, and actuation before executing an action. Explicit chain-of-thought methods expose these intermediate decisions as textual (Zawalski et al., 2025) or visual reasoning traces Zhao et al. (2025), providing structured intermediate targets for the policy. However, generating these explicit reasoning autoregressively introduces substantial computation and latency into every closed-loop control cycle.

Recent latent reasoning approaches avoid this autoregressive cost by representing intermediate computation with continuous latent tokens that are passed directly to the action policy (Bai et al., 2026; Huang et al., 2026a; Liu et al., 2026; Zhong et al., 2026a). This substantially improves inference efficiency, but the latent representations are typically learned without explicitly preserving the structure present in explicit reasoning traces. As illustrated in Fig. 1(a), explicit CoT provides structured intermediate reasoning at the cost of autoregressive generation, whereas latent CoT improves efficiency but leaves this intermediate structure implicit. Preserving such structure is relevant for manipulation, where intermediate reasoning involves functionally distinct roles such as task decomposition, object grounding, motion specification, and actuation. We therefore ask: *Can latent reasoning retain the structured reasoning of explicit CoT while preserving its inference efficiency?*

We address this gap with **Structured Latent Reasoning (SLR)**, a framework that organizes the latent tokens into semantically distinct reasoning units, each corresponding to a particular form of intermediate reasoning. Moreover, these reasoning units are not equally relevant to action generation at every transition. As illustrated in Fig. 1(b), the action policy relies on different reasoning units across a manipulation trajectory. This motivates **transition-adaptive learning**, which adapts the supervision of individual reasoning units according to their relevance to the current transition, rather than treating all units uniformly throughout the trajectory.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

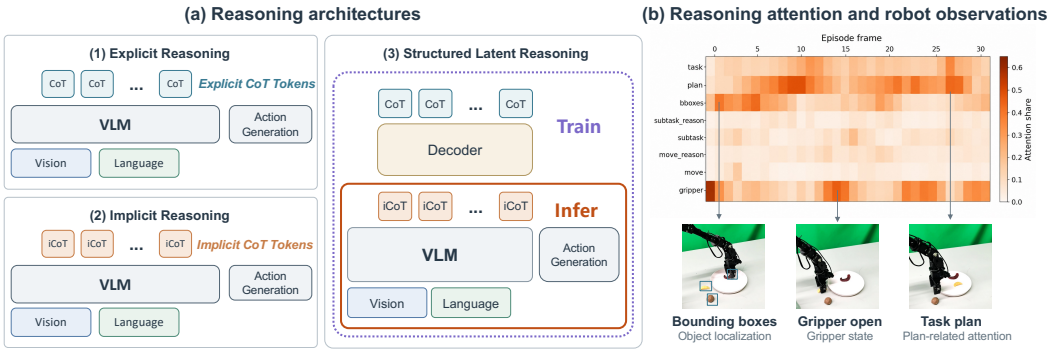


Figure 1: Motivation and overview of Structured Latent Reasoning (SLR). (a) Explicit chain-of-thought reasoning preserves structured intermediate reasoning but incurs autoregressive decoding overhead, while existing latent reasoning methods improve inference efficiency but leave the internal reasoning structure implicit. SLR organizes continuous latent representations into semantically distinct reasoning units while avoiding explicit reasoning generation at inference. (b) The relevance of different reasoning units changes across transitions in a manipulation trajectory, motivating our transition-adaptive learning strategy.

Concretely, SLR uses unit-wise auxiliary supervision to encourage each reasoning unit to encode its designated reasoning content, while transition-adaptive learning reweights this supervision using relevance signals from the action policy. The auxiliary supervision is applied only during training. At inference, the action policy generates actions directly from the latent reasoning units without explicit reasoning generation, retaining the efficiency of latent reasoning.

We evaluate SLR on both simulation and real-world manipulation tasks. On SIMPLER, SLR achieves 73.7% average success, outperforming the strongest latent reasoning baseline, LaRA-VLA, by 4.9 points. On real-robot tasks, SLR reaches 79.0% average task completion, compared with 71.0% for $\pi_{0.5}$, and maintains gains under distribution shift, improving over the strongest baseline by 12.1 points on unseen objects and 7.5 points on unseen instructions. Importantly, these gains do not require explicit reasoning generation at inference: SLR runs at 341,ms per control step, over $2\times$ faster than Fast-ThinkAct and approximately $16\times$ faster than ECoT. Controlled ablations and representation analyses further show that the gains arise from structured reasoning supervision and its transition-adaptive weighting.

Our contributions are three-fold:

1. We introduce Structured Latent Reasoning (SLR) for VLA models by decomposing the continuous latent representation into semantically distinct reasoning components and supervising them with auxiliary text decoding.
2. We propose transition-adaptive learning for SLR. Instead of applying a uniform text-decoding objective throughout a trajectory, our method adjusts the learning density of different reasoning components at each transition according to feedback from the action policy.
3. Experiments in simulation and real-world window manipulation tasks demonstrate consistent improvements over existing latent reasoning baselines under both in-distribution and out-of-distribution settings.

2 Related Work

Vision-language-action (VLA) models unify visual perception, language understanding, and robot control within a common policy framework. Recent systems such as OpenVLA (Kim et al., 2024) and π_0 (Black et al., 2026) demonstrate the effectiveness of large pretrained vision-language models for general-purpose manipulation. However, directly mapping observations and instructions to actions leaves the intermediate reasoning process largely implicit. To make this process more explicit, a line of work introduces chain-of-thought (CoT) reasoning into VLA policies. Language-based

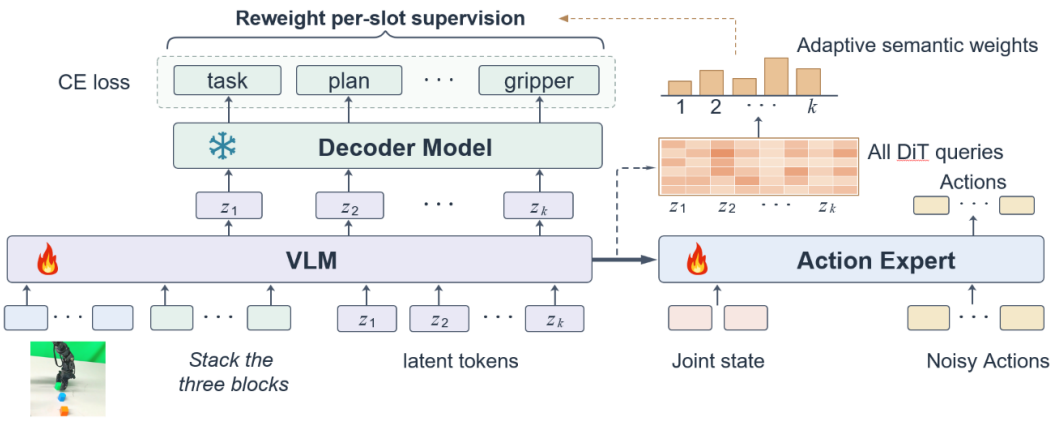


Figure 2: Overview of the proposed Structured Latent Reasoning (SLR) framework. The VLM produces a set of semantically structured latent tokens, each supervised by a training-only decoder to encode a designated reasoning type. The action expert conditions on these latent tokens for control, while its attention to each unit provides transition-adaptive weights that reweight the corresponding semantic supervision during training.

approaches, such as ECoT and ECoT-Lite (Zawalski et al., 2025; Chen et al., 2025), generate textual intermediate reasoning before action prediction, while ThinkAct (Huang et al., 2025) and other visual reasoning approaches use visual plans or predicted goal states to guide action generation. Related methods, including CoT-VLA and FlowVLA (Zhao et al., 2025; Zhong et al., 2026b), further explore visual or action-oriented forms of intermediate reasoning. Although these methods improve the availability of intermediate planning signals, explicit reasoning can introduce additional inference cost, and visual or textual reasoning signals are not always well aligned with the requirements of high-frequency continuous control.

To reduce this overhead, recent works investigate latent reasoning for VLA models. FastThinkAct (Huang et al., 2026a) compresses intermediate planning into a verbalizable latent representation, while LaST0 (Liu et al., 2026) incorporates latent spatio-temporal reasoning for modeling future robotic states. Latent Reasoning VLA (Bai et al., 2026) studies latent thinking and prediction without explicitly generating the full reasoning trace at inference time. DualCoT-VLA (Zhong et al., 2026a) further separates latent reasoning into visual and linguistic streams, but this separation is modality-based rather than organized around functionally distinct reasoning roles for control. More broadly, existing latent reasoning methods primarily optimize latent representations for downstream prediction, without explicitly preserving distinct semantic reasoning roles. SLR instead structures the latent space into semantically typed reasoning units with role-specific supervision, and further adapts this supervision as the relevance of different reasoning roles changes across transitions. In this way, SLR combines the structured semantics of explicit reasoning with the inference efficiency of latent reasoning.

3 Methods

3.1 Task Formulation

We consider language-conditioned robot manipulation in a closed-loop control setting. An episode is a language-conditioned trajectory $\tau = (o_0, a_0, \ldots, o_{T-1}, a_{T-1}, o_T)$, where o_t is the observation available at step t, a_t is the action issued to the robot, and o_{t+1} is the observation received after executing a_t. A natural-language instruction l specifies the task to be completed over the episode. The learning problem is therefore to infer the next action from the current observation and the instruction. We model this conditional mapping with a vision-language-action (VLA) policy, denoted by $\pi(a_t \mid o_t, l)$. For tasks that require multi-step decomposition, object grounding, or precise motion specification, the policy may benefit from intermediate reasoning before selecting a_t. Explicit chain-of-thought methods represent this computation as a textual trace r_t and use $\pi(a_t \mid o_t, l, r_t)$, but autoregressively generating the trace adds decoding cost to every control cycle.

Latent reasoning methods replace the explicit trace r_t with continuous latent states Z_t whose generation and use for action prediction factorize as

$$\pi(a_t \mid o_t, l) = \pi(Z_t \mid o_t, l)\, \pi(a_t \mid o_t, l, Z_t). \tag{1}$$

The model first produces Z_t from the observation and instruction, then uses these states to condition action prediction without decoding a textual reasoning trace. In our formulation, latent generation is deterministic, with $Z_t = F_\theta(o_t, l)$, where F_θ denotes the vision-language backbone.

For training, let $\mathcal{D}$ denote the set of demonstration transitions. The action objective is

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{(o_t,l,a_t)\sim\mathcal{D}}\left[\ell_{\text{act}}(a_t; o_t, l, Z_t)\right], \qquad Z_t = F_\theta(o_t, l), \tag{2}$$

where ℓ_{act} is the training loss associated with the action prediction model, such as a negative log-likelihood or a diffusion denoising loss. We build on this formulation by organizing Z_t into reasoning units with explicit semantic supervision.

3.2 Structured Latent Reasoning

Action supervision encourages latent states to encode information useful for control, but does not specify which type of intermediate reasoning each latent unit should represent. To establish this correspondence, we propose **Structured Latent Reasoning** (SLR), which assigns each unit a reasoning type and supervises it with a corresponding textual target. Unlike action-only latent learning, which supervises Z_t only through an aggregate action loss, SLR supplies a separate text target and loss for each indexed unit.

We organize the latent states as an ordered collection $Z_t = (z_t^{(1)}, \ldots, z_t^{(K)})$. Each index k corresponds to a fixed reasoning type across transitions and trajectories, while the content represented by that unit varies with the observation and instruction. For example, an object-grounding unit describes the relevant object in the current scene, whereas a motion-specification unit describes the motion required at the current stage of execution. We denote the target associated with unit k at transition t by $r_t^{(k)}$.

We supervise these units with an auxiliary text decoder derived from the language backbone. For each unit, the decoder predicts $r_t^{(k)}$ conditioned on $z_t^{(k)}$. Applying the decoder separately to individual units encourages the designated reasoning content to be recoverable from its assigned unit. The backbone still computes the units jointly, allowing them to share information from the observation and instruction.

For a target containing $N_t^{(k)}$ tokens, the unit-level loss is

$$\ell_{t,k} = -\frac{1}{N_t^{(k)}} \sum_{j=1}^{N_t^{(k)}} \log p_\psi\left(r_{t,j}^{(k)} \mid r_{t,<j}^{(k)}, z_t^{(k)}\right), \tag{3}$$

where ψ denotes the decoder parameters. Averaging over reasoning units and training transitions gives

$$\mathcal{L}_{\text{text}} = \mathbb{E}_{t\sim\mathcal{D}}\left[\frac{1}{K}\sum_{k=1}^{K} \ell_{t,k}\right], \tag{4}$$

where the expectation is over transitions sampled from the training demonstrations.

Text supervision establishes the intended association between each unit and its reasoning content, while the action objective trains the policy to use the full collection of units for control. The auxiliary decoder is used only during training. At inference time, the policy predicts actions directly from the observation, instruction, and generated latent states.

3.3 Transition-Adaptive Learning

The uniform objective in Eq. equation 4 assigns equal weight to every reasoning unit at every transition. However, the information needed for action prediction can change during execution. Object grounding may be particularly useful when selecting a target, while motion specification and actuation may become more useful as the robot interacts with it.

To account for this variation, we further propose **Transition-Adaptive Learning** for SLR. This method uses feedback from the action policy to adjust the text-supervision weight of each reasoning unit at the current transition. Its purpose is to place greater emphasis on reasoning content that the policy uses more strongly for the current action, while retaining supervision for all units.

We obtain this feedback from the attention of the action-model queries to the latent units. Averaging the attention assigned to unit k over the cross-attention layers, heads, action-model queries, and repeated diffusion samples yields a non-negative score $s_t^{(k)}$, which we use as a proxy for the policy's use of that unit. We convert these scores into normalized supervision weights:

$$w_t^{(k)} = \frac{s_t^{(k)}}{\sum_{i=1}^{K} s_t^{(i)}}, \tag{5}$$

where the normalization is performed over the valid reasoning units of each transition.

We replace the uniform text objective with

$$\mathcal{L}_{\text{text}}^{\text{TA}} = \mathbb{E}_{t\sim\mathcal{D}}\left[\sum_{k=1}^{K} \text{sg}\left(w_t^{(k)}\right) \ell_{t,k}\right], \tag{6}$$

where sg denotes stop-gradient. This prevents the text loss from directly optimizing the weighting signal. When $\lambda = 0$, Eq. equation 6 reduces to the uniform objective $\mathcal{L}_{\text{text}}$.

The complete training objective combines action prediction with adaptive reasoning supervision:

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \beta\mathcal{L}_{\text{text}}^{\text{TA}}, \tag{7}$$

where β controls the strength of text supervision.

4 Experiments

Our experiments examine whether Structured Latent Reasoning improves manipulation performance, which aspects of its supervision contribute to these improvements, and how the resulting latent units are used during control. We evaluate SLR in simulation and on a real robot, followed by controlled ablations, analyses of the latent representation, and inference-time measurements.

4.1 Experimental Setup

Simulation benchmark. We conduct simulation experiments on SimplerEnv (Li et al., 2024b), which evaluates the real-to-sim generalization of robot manipulation policies trained on real-world data. We use the WidowX setup and evaluate each task on 24 predefined object configurations with 4 policy seeds, resulting in 96 rollouts per task. We report binary full-task success rates and the unweighted average across tasks. Complete environment and evaluation details are provided in Appendix B.

Baselines. We compare SLR with representative VLA methods spanning different forms of chain-of-thought reasoning. The no-CoT baselines include OpenVLA (Kim et al., 2024), Octo (Team et al., 2024), OpenVLA-OFT (Kim et al., 2025), π_0 (Black et al., 2026), π_0-FAST (Pertsch et al., 2025), $\pi_{0.5}$ (Intelligence et al., 2025), CogACT (Li et al., 2024a), and StarVLA-GR00T (Community, 2026). For explicit reasoning baselines, we include textual-CoT methods ThinkAct (Huang et al., 2026b) and ECoT (Zawalski et al., 2025), as well as visual-CoT methods F1 (Lv et al., 2025) and UD-VLA (Chen et al., 2026). We further compare against LaRA-VLA (Bai et al., 2026), a latent-CoT method most closely related to our reasoning formulation.

Implementation details. SLR uses Qwen3-VL-4B-Instruct (Bai et al., 2025) as its vision–language backbone and a GR00T-style DiT-B flow-matching action head. Eight typed latent units are supervised with frame-level Embodied CoT annotations (Zawalski et al., 2025). A Qwen-initialized auxiliary decoder provides training-only supervision and is omitted entirely at inference. We train on Bridge (Walke et al., 2024) using a 30k-step uniform-supervision stage followed by an attention-weighted continuation of up to 70k steps. Training uses AdamW on 8 GPUs with a global batch size of 128; the backbone and action-head learning rates are 10^{-5} and 10^{-4}, respectively. Full architecture, loss, optimization, and checkpoint details are reported in Appendix E.

Table 1: Full-task success rates (%) on four SIMPLER WidowX Bridge tasks.

CoT Type	Method	Put spoon	Put carrot	Stack block	Put eggplant	Avg.
No cot	OpenVLA (Kim et al., 2024)	0.0	0.0	0.0	4.1	1.0
	Octo (Team et al., 2024)	47.2	9.7	4.2	56.9	29.5
	π_0 (Black et al., 2026)	29.1	0.0	16.7	62.5	27.1
	π_0-FAST (Pertsch et al., 2025)	29.1	21.9	10.8	66.6	32.1
	OpenVLA-OFT (Kim et al., 2025)	34.2	30.0	30.0	72.5	41.7
	$\pi_{0.5}$ (Intelligence et al., 2025)	49.3	64.7	44.7	69.7	57.1
	CogACT (Li et al., 2024a)	71.7	50.8	15	67.5	51.3
	StarVLA-GR00T (Community, 2026)	83.0	59.4	18.8	100.0	65.3
Textual cot	ECoT (Zawalski et al., 2025)	40.2	11.7	0.0	28.4	20.1
	ThinkAct (Huang et al., 2026b)	58.3	37.5	8.7	70.8	43.8
Visual cot	F1 (Lv et al., 2025)	50	70.8	50	66.7	59.4
	UD-VLA (Chen et al., 2026)	58.3	62.5	54.1	75	62.5
Latent cot	lara-vla (Bai et al., 2026)	95.8	62.5	25	91.7	68.8
	SLR (Ours)	83.3	58.3	54.2	99.0	**73.7**

4.2 Simulation Experiment

Table 1 reports full task success for each task and the average across the four tasks in SIMPLER. SLR achieves the best average success rate (73.7%), outperforming the strongest latent reasoning baseline LaRA-VLA (68.8%) and all explicit reasoning methods. We organize the analysis around three comparisons.

(1) **SLR outperforms ECOT despite using the same reasoning traces**: ECOT generates textual traces autoregressively at every control cycle, while SLR encodes the same reasoning into continuous latent tokens and removes the decoder at inference, improving over ECOT by 53.6 points on average with the largest gap on stacking (54.2% vs. 0.0%), which shows that the gain comes from the latent interface rather than the reasoning content.

(2) **SLR outperforms visual CoT methods without using visual CoT at inference**: UD-VLA and F1 reach 62.5% and 59.4% average success, while SLR reaches 73.7% without generating any visual trace, indicating that the same reasoning content can be preserved in latent units and passed directly to the action policy.

(3) **SLR outperforms latent reasoning baselines because of its structured reasoning**: SLR improves over LaRA-VLA from 68.8% to 73.7%, consistent with our hypothesis that unstructured latent tokens become a bottleneck when the task requires order-constrained composition.

4.3 Real-World Evaluation

Tasks and setup. We evaluate the methods on a single-arm WidowX with an external third-person RGB camera and a wrist-mounted RGB camera. The task suite contains four tasks covering (1) long-horizon execution (*put all the things on the plate*), (2) tool use and sequential execution (*wipe the plate with a towel and put the carrot on the plate*), (3) order constraints (*stack the three blocks in order from left to right*), and (4) spatial relations (*put the orange duck on the left plate, and put the green duck on the right plate*). These axes cover the reasoning types that SLR is designed to preserve in the latent representation.

We collect 100 training demonstrations for each task. All methods are trained using the same task-specific demonstrations, with reasoning annotations provided to methods that require them. We evaluate each model on 10 trials per task and assign each trial a graded task-completion score, with partial credit for partially completed tasks. We report the mean score across trials. Further hardware and data-collection details are provided in Appendix C.

Results. Figure 3 reports the mean real-world task-completion score of SLR and three baselines: $\pi_{0.5}$, ECOT, and StarVLA-GR00T. SLR achieves the highest mean score on all four tasks, reaching 79.0% on average compared with 71.0% for $\pi_{0.5}$, 63.0% for STARVLA, and 22.3% for ECOT. Its task scores are 85.0% for long-horizon execution, 61.0% for tool use and sequential execution,

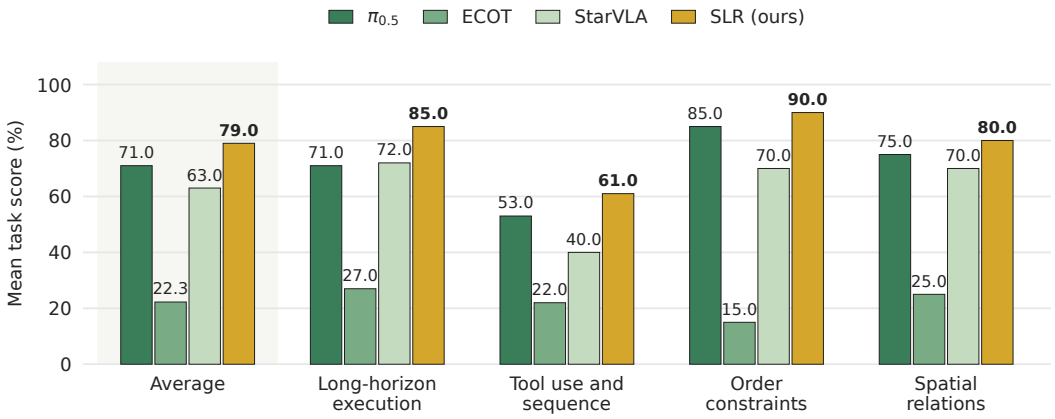


Figure 3: Mean task-completion scores (%) on four real-world manipulation tasks. Each trial can receive partial credit. The shaded group shows the unweighted average across tasks.

90.0% for order constraints, and 80.0% for spatial relations, indicating that the benefit of structured latent reasoning is not tied to a single task type. Example real-robot trajectories are shown in Appendix D.

4.4 Generalization Experiment

We evaluate generalization along two axes: OOD objects and OOD instructions. For OOD objects, we use three held-out object configurations per task; depending on the task, each configuration replaces one or two objects. For OOD instructions, we use three alternative prompts per task. Each object configuration and each instruction variant is evaluated for five trials, giving 15 trials per task and 60 trials per method for each axis. We report mean graded task-completion scores. The scoring rubrics follow the in-distribution evaluation, except that object generalization for plate placement scores the two replacement objects, awarding 0.5 for one and 1.0 for both. The task-specific substitutions, exact prompts, and scoring rubrics are provided in Appendix C.

Figure 4 reports the results. SLR achieves a mean score of 73.8% on OOD objects and 72.0% on OOD instructions, outperforming all baselines in both settings. We organize the analysis around two comparisons.

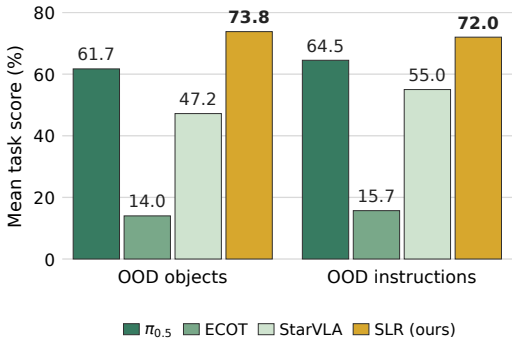


Figure 4: Real-robot generalization scores (%) for unseen objects and instructions.

(1) OOD objects. SLR reaches a mean score of 73.8%, compared with 61.7% for $\pi_{0.5}$, 47.2% for StarVLA, and 14.0% for ECOT. The gap between SLR and the strongest baseline is 12.1 points. This result suggests that structured latent units preserve object grounding under object appearance changes. Because the reasoning units are supervised with object-level targets, the action policy can reuse the grounded object representation even when the object appearance differs from training.

(2) OOD instructions. SLR reaches a mean score of 72.0%, compared with 64.5% for $\pi_{0.5}$, 55.0% for StarVLA, and 15.7% for ECOT. Instruction generalization requires mapping paraphrased or recomposed language to the same underlying task structure. Because SLR supervises reasoning units rather than a single entangled latent vector, the action policy can reuse grounded object representations while adapting to instruction variation. The gain over the strongest baseline is 7.5 points. ECOT again performs poorly, indicating that explicit textual reasoning is particularly fragile when the instruction distribution shifts.

5 ANALYSIS

5.1 ABLATION STUDIES

We conduct controlled ablations to distinguish the effects of latent tokens, text supervision, and adaptive weighting. As shown in Table 2, we observe three main findings. (1) **Latent tokens alone are insufficient.** Training latent tokens only with the action objective achieves 65.6% average success, slightly below the 66.7% action-only baseline. This suggests that latent tokens do not automatically acquire useful reasoning structure without explicit supervision. (2) **Reasoning supervision improves latent representations.** Adding auxiliary text decoding raises performance from 65.6% to 67.7%, indicating that supervising intermediate reasoning content makes the latent representation more useful for action prediction. (3) **Adaptive supervision provides the largest gain.** SLR further improves performance to 73.7%, outperforming uniform text decoding by 6.0 points. This result shows that adapting supervision to the relevance of different reasoning units is important for effectively exploiting structured latent reasoning.

Table 2: Component ablation (average success, %).

Variant	Avg.
Action only	66.67
+ latent tokens	65.63
+ text decoding	67.71
+ adaptive weighting (SLR)	**73.70**

5.2 WHERE DOES THE GAIN FROM ADAPTIVE WEIGHTING COME FROM?

The component ablation shows that transition-adaptive weighting substantially improves performance. We further investigate where this gain comes from by comparing it with three alternatives under the same training budget and total supervision weight: flat weighting assigns equal weights to all reasoning units, random weighting samples independent weights at each transition, and shuffled weighting uses the adaptive weight vector from another transition while preserving the reasoning-unit order. These variants isolate whether the gain comes from non-uniform weighting itself or from aligning the weighting signal with the current transition.

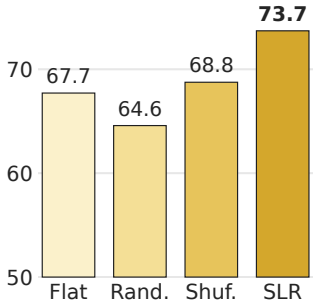


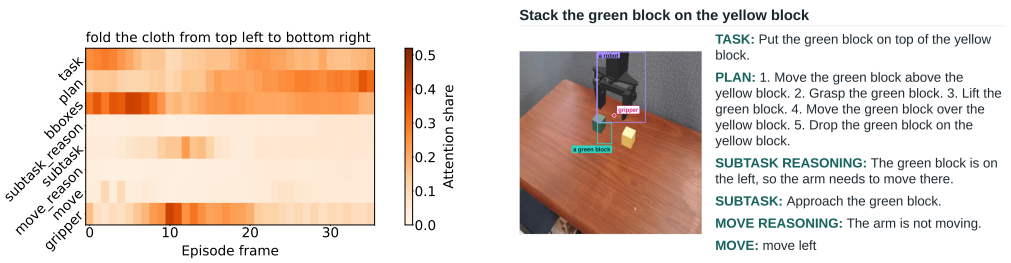
Figure 5: SIMPLER weighting ablation (average success, %; y-axis starts at 50%).

As shown in Figure 5, flat weighting achieves 67.7% average success, while random weighting drops to 64.6%, showing that arbitrary non-uniform supervision does not improve learning. Shuffled weighting reaches 68.8%, but remains 4.9 points below SLR at 73.7%. Since shuffled weighting retains weight vectors produced by the same mechanism but breaks their transition-wise correspondence, this gap indicates that the gain primarily comes from assigning reasoning-unit supervision according to its relevance to the current transition.

5.3 DO STRUCTURED LATENT UNITS LEARN DISTINCT REASONING ROLES?

We examine the learned latent units from two complementary perspectives: how they are used by the action policy and what reasoning content they encode.

Dynamic usage across transitions. Figure 6(a) visualizes the attention assigned to different reasoning units over a complete manipulation episode. The attention pattern changes systematically with the stage of execution rather than remaining fixed throughout the trajectory. At the beginning of the episode, the policy places relatively high attention on object-grounding information such as bounding boxes, consistent with the need to localize the target object before interaction. Around the grasping stage, attention shifts toward gripper-related reasoning as the policy must coordinate contact and gripper closure. Toward the end of the episode, higher-level planning information becomes more prominent, consistent with checking whether the planned manipulation has been completed and determining the remaining action. These phase-dependent shifts suggest that the action policy selectively recruits different reasoning units as the control state evolves, rather than using all latent units uniformly. This behavior also supports the transition-dependent supervision used in SLR, which allows the relative importance of different reasoning roles to vary across the trajectory.



(a) Attention to reasoning units across an episode.

(b) Reasoning content decoded from latent units.

Figure 6: Analysis of structured latent reasoning. (a) Attention assigned by the action policy to different reasoning units varies across transitions. (b) Offline decoding reveals role-specific reasoning content from individual latent units.

Distinct reasoning content. We further use the training-time text decoder offline to decode individual latent units on held-out observations without providing reference reasoning text. As shown in Figure 6(b), the decoded outputs recover role-specific information at the task, plan, subtask, and motion levels while remaining grounded in the current observation. Together, these results provide qualitative evidence that SLR learns semantically differentiated reasoning units that are dynamically used by the action policy.

5.4 Inference Efficiency

We measure end-to-end inference latency on a single RTX 4090, from an observation–instruction pair to the predicted action chunk, including visual processing, reasoning, and action generation. As shown in Figure 7, SLR requires 341.30 ms per inference, compared with 805.00 ms for Fast-ThinkAct, 5460.32 ms for ECOT, and 7513.00 ms for ThinkAct. Thus, SLR is over $2\times$ faster than Fast-ThinkAct and approximately $16\times$ and $22\times$ faster than ECOT and ThinkAct, respectively, by avoiding autoregressive reasoning generation. LaRA-VLA and StarVLA are faster at 253.12 ms and 130.20 ms, respectively. SLR introduces only 88.18 ms additional latency over LaRA-VLA while achieving stronger manipulation performance, and its auxiliary text decoder incurs no inference-time cost since it is removed after training.

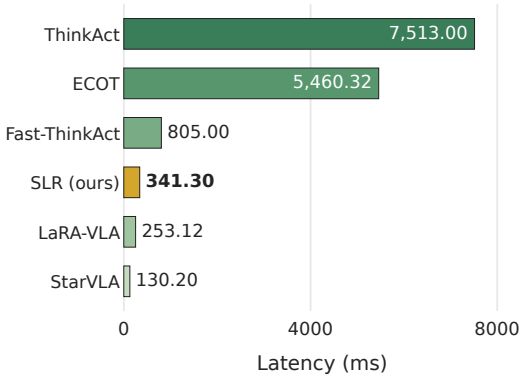


Figure 7: Inference latency (ms) on an RTX 4090.

6 Conclusion

In this paper, we introduced Structured Latent Reasoning (SLR), a framework for incorporating structured intermediate reasoning into continuous latent representations for vision-language-action models. SLR organizes latent tokens into semantically distinct reasoning units and supervises each unit with a corresponding reasoning target through a training-only auxiliary text decoder. We further introduced transition-adaptive learning, which uses feedback from the action policy to redistribute reasoning supervision according to the relevance of each unit at different stages of execution. Across simulation and real-world manipulation tasks, SLR consistently improves over existing latent reasoning baselines and generalizes better to unseen objects and instructions. Our analyses further show that the learned units encode differentiated reasoning content and are used dynamically throughout manipulation trajectories. By removing explicit reasoning generation at inference, SLR retains the efficiency of latent reasoning while providing a more structured interface between reasoning and action.

AI USE STATEMENT

We used generative AI tools to assist with language editing and the preparation of statistical visualizations. All AI-assisted text was reviewed and revised by the authors, and all generated or AI-assisted figures were checked against the underlying experimental data for correctness. Generative AI tools were not treated as authoritative sources for scientific claims or experimental conclusions. The authors take full responsibility for the final content of the paper, including all text, claims, analyses, and visualizations produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the model architecture, training procedure, optimization settings, reasoning supervision, and evaluation protocols in the appendix. The appendix also specifies the simulation and real-robot experimental setups, task definitions, evaluation criteria, and ablation configurations used in our experiments. To further support reproducibility, we plan to publicly release the training and evaluation code, model checkpoints, and trained weights upon publication.

REFERENCES

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.

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiao-Guang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, Badong Chen, and Shanghang Zhang. Latent reasoning vla: Latent thinking and prediction for vision-language-action models. *ArXiv*, abs/2602.01166, 2026. URL https://api.semanticscholar.org/CorpusID:285271357.

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.

Jiayi Chen, Wenxuan Song, Pengxiang Ding, Ziyang Zhou, Han Zhao, Barrett Tang, Donglin Wang, and Haoang Li. Unified diffusion vla: Vision-language-action model via joint discrete denosing diffusion process. In *International Conference on Learning Representations*, volume 2026, pp. 139291–139311, 2026.

William Chen, Suneel Belkhale, Suvir Mirchandani, Oier Mees, Danny Driess, Karl Pertsch, and Sergey Levine. Training strategies for efficient embodied reasoning. *arXiv preprint arXiv:2505.08243*, 2025.

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

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. *arXiv preprint arXiv:2601.09708*, 2026a.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Frank Wang, and Fu-En Yang. Thinkact: Vision-language-action reasoning via reinforced visual latent planning. *Advances in Neural Information Processing Systems*, 38:82782–82802, 2026b.

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.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *ArXiv*, abs/2406.09246, 2024. URL https://api.semanticscholar.org/CorpusID:270440391.

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

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, 2024a. URL https://arxiv.org/abs/2411.19650.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation, 2024b. URL https://arxiv.org/abs/2405.05941.

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.

Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions, 2025. URL https://arxiv.org/abs/2509.06951.

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

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

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.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In *Conference on Robot Learning*, pp. 3157–3181. PMLR, 2025.

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.

Zhide Zhong, Junfeng Li, Junjie He, Haodong Yan, Xin Gong, Guanyi Zhao, Yingjie Cai, Jiantao Gao, Xu Yan, Bingbing Liu, Yingcong Chen, Liuqing Yang, and Haoang Li. Dualcot-vla: Visual-linguistic chain of thought via parallel reasoning for vision-language-action models, 2026a. URL https://arxiv.org/abs/2603.22280.

Zhide Zhong, Haodong Yan, Junfeng Li, Xiangchen Liu, Xin Gong, Tianran Zhang, Wenxuan Song, Jiayi Chen, Xinhu Zheng, Hesheng Wang, et al. Flowvla: Visual chain of thought-based motion reasoning for vision-language-action models. *IEEE Transactions on Robotics*, 2026b.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

A Training Details Ablation

Training schedule. SLR uses a two-stage training schedule: an initial stage with uniform token-level supervision followed by transition-adaptive learning. We compare this schedule with training with uniform supervision throughout. As shown in Table 3, uniform supervision throughout reaches 67.7% average success, while the two-stage schedule reaches 73.7%. This comparison evaluates the two training schemes as a whole; it does not isolate the effect of the initial stage or establish whether adaptive weighting from the beginning would be less stable. We use the two-stage schedule in all main experiments.

Decoding weight. We additionally examine the text-decoding loss weight during the 30k-step uniform warm-up stage. As shown in Figure 8, weights of 0.001, 0.01, 0.1, and 0.5 yield average success rates of 57.29%, 57.29%, 59.38%, and 56.25%, respectively. We therefore use 0.1 for the warm-up stage in all main experiments. The degradation at 0.5 suggests that overly strong auxiliary supervision can interfere with action learning.

Table 3: Effect of the training schedule on average success rate (%).

Schedule	Avg.
Uniform	67.71
Warm-up + adaptive (SLR)	**73.70**

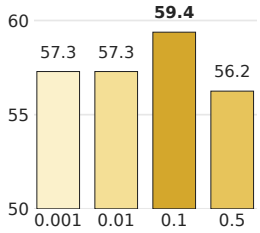


Figure 8: Average SIMPLER success across text-decoding loss weights during uniform warm-up. The vertical axis begins at 50%.

B Details of the SIMPLER Experimental Setup

Tasks and environment. We evaluate the four standard WidowX Bridge tasks in SIMPLER: *put a spoon on a tablecloth*, *put a carrot on a plate*, *stack a green cube on a yellow cube*, and *put an eggplant in a basket*. The simulator runs at 500 Hz and the policy is queried at 5 Hz, with a maximum episode length of 120 control steps. We use the benchmark's Bridge RGB overlays and fixed robot initialization. Policy observations are resized to 224×224.

Action prediction. The policy predicts a 16-step, 7-D delta end-effector action chunk. At each control transition, the flow-matching action head starts from a sampled Gaussian action trajectory and applies four Euler denoising updates. We temporally ensemble successive predictions with an ensemble horizon of seven before executing the current action.

Evaluation protocol. Each run evaluates the same 24 predefined object episodes for every task. For the main SLR result, we repeat this evaluation with policy sampling seeds 0–3, producing 96 rollouts per task. The simulator's binary full-task success signal is averaged first over object episodes and policy seeds and then equally over the four tasks. All of our model variants use the same object episodes and robot initialization. External baseline entries in Table 1 are compiled from published evaluations on the four-task SIMPLER Bridge suite.

C Details of the Real-Robot Experimental Setup

Platform and observations. Experiments use a single-arm WidowX manipulator with a parallel gripper. The perception system consists of one external third-person RGB camera and one wrist-

mounted RGB camera. Demonstrations are recorded at 640×480 and 5 Hz; both views are resized to 224×224 for policy input.

Tasks and demonstrations. We collect 100 training demonstrations for each of four tasks: placing all objects on a plate, wiping the plate and then placing a carrot on it, stacking three blocks in the instructed order, and placing two colored ducks on their corresponding plates. The associated frame-level reasoning annotations cover all recorded frames in each dataset. We train a separate policy for each task. Each model is evaluated on 10 trials per task. Trials receive graded task-completion scores, with partial credit for partially completed tasks; we report the mean score over trials.

Task-completion scoring. We assign each trial a score between 0 and 1 according to the following task-specific rubric. Scores for sequential subtasks are cumulative.

- *Put all the things on the plate.* Three objects are initially on the table. Placing one, two, or all three objects on the plate earns 0.3, 0.6, or 1.0, respectively.
- *Wipe the plate with a towel and put the carrot on the plate.* Picking up the towel earns 0.2. Wiping the plate and returning the towel to the table adds 0.3. Placing the carrot on the plate adds the remaining 0.5.
- *Stack the three blocks in order from left to right.* Completing one block-stacking step earns 0.5; completing both steps to form the ordered three-block stack earns 1.0.
- *Put the orange duck on the left plate, and put the green duck on the right plate.* Placing either duck on its designated plate earns 0.5; placing both correctly earns 1.0.

Generalization conditions. We evaluate instruction and object generalization separately for each of the four tasks. Each condition has three variants per task: three alternative prompts for instruction generalization and three held-out object configurations for object generalization. Each variant is evaluated for five trials, giving 15 trials per task and 60 trials per method for each generalization condition (120 trials per method across both conditions). The alternative prompts are listed below. These prompts retain the original task objective, and each trial uses the same scoring rubric as its corresponding in-distribution task. For the object condition, we use three held-out object configurations per task. In the plate-placement task, each configuration replaces two of the three objects with objects absent from the training data (six held-out objects across the three configurations); placing one or both replacement objects on the plate earns 0.5 or 1.0. In the wiping task, each configuration replaces the carrot with one unseen object (three held-out objects total), and the original 0.2/0.3/0.5 rubric applies with the replacement object as the final placement target. In the stacking task, each configuration replaces one of the three blocks with an unseen block (three held-out blocks total); the original stacking rubric applies. In the duck-placement task, each configuration replaces both ducks with unseen ducks (six held-out ducks total); the original left/right placement rubric applies.

Generalization instruction prompts. For reproducibility, the exact alternative prompts are:

Plate placement. (1) Gather every object and place them together on top of the plate. (2) Make sure nothing is left outside the plate; everything should end up on it. (3) Move each of the items from where they are now onto the plate.

Wiping and placement. (1) Use the towel to clean the plate first, then place the carrot on it. (2) Before the carrot goes onto the plate, give the plate a wipe with the towel. (3) Clean the surface of the plate using the towel, and afterwards set the carrot there.

Ordered stacking. (1) Build a tower using the blocks according to their current left-to-right arrangement. (2) Pile the three blocks together while preserving the order in which they appear from left to right. (3) Form a vertical stack whose block ordering follows their horizontal arrangement from leftmost to rightmost.

Duck placement. (1) Place each duck on its corresponding side: the orange one goes to the left plate and the green one to the right. (2) Move the green duck onto the plate on the right, while the orange duck should end up on the plate on the left. (3) Assign the left plate to the orange duck and the right plate to the green duck.

Fine-tuning and action representation. Each task-specific SLR policy is initialized from the 30k Bridge SLR checkpoint and fine-tuned for 20k updates with transition-adaptive decoder supervision. The policy uses both camera views and a 10-D state comprising the end-effector position, a 6-D rotation representation, and the signed gripper state. It predicts an eight-step, 7-D body-frame delta end-effector chunk at 5 Hz: three translation components, three rotation- vector components, and one gripper command. The deployment server converts these relative predictions into absolute end-effector targets in the robot base frame for execution. The auxiliary text decoder is not loaded during deployment.

Real-robot reasoning annotations. We adapt the synthetic annotation approach of Embodied CoT (Zawalski et al., 2025) and its public implementation[1] to our real-robot demonstrations. The pipeline combines task specifications, visual detections, recorded robot signals, and language-model-generated explanations. Each task specification supplies a canonical task description, a scene description, object and target identities, and an ordered plan. Task and plan targets are shared across the frames of an episode; the remaining fields are annotated at each step of the 5 Hz demonstration.

Visual and motion features. Object boxes are obtained from the external camera with `IDEA-Research/grounding-dino-base`, using the task specification's object queries and default box/text thresholds of 0.3/0.2. Boxes are clipped to the image boundaries and stored as pixel coordinates $(x_{\min}, y_{\min}, x_{\max}, y_{\max})$ in the source image. Task-specific checks constrain object identities and spatial assignments; for example, the two duck-placement targets are identified by their left-to-right image positions. For gripper localization, external-camera frames are resized to 256×256. OWL-ViT (`google/owlvit-base-patch16`) detects the gripper, SAM (`facebook/sam-vit-base`) segments it, and a RANSAC linear fit relates recorded 3-D end-effector positions to detected 2-D positions within each episode. The gripper target concatenates the current position and the next four positions in this annotation coordinate system, repeating the final position at the episode boundary. Movement labels are generated deterministically from the recorded action sequence, and gripper open/close events provide additional phase cues.

Text generation and checks. The annotation client is configured with the API model identifier `Qwen3.8-27B`, with thinking disabled and a maximum of 512 output tokens per request. For each frame, its text prompt contains the task and scene descriptions, ordered plan options, the episode's movement sequence, the current movement, previously accepted subtasks, and gripper-state cues. The language model produces the current subtask and concise explanations of the subtask and movement; the movement label must match the deterministic input. Subtasks must follow the configured plan without skipping or moving backward. For duck placement, the configured gripper-event state machine fixes the subtask, and the language model supplies its explanation. Outputs are checked for the required field tags, valid plan entries, movement-label agreement, and explicit contradictions with gripper state. The implementation allows up to five generation attempts per frame. Remaining invalid explanations are repaired from a nearby compatible frame or, when no suitable frame is available, a deterministic template, with repair provenance retained in the annotation records.

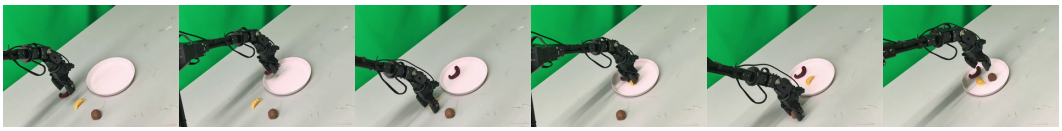
Alignment and supervision fields. The pipeline merges text and visual annotations by dataset identity, episode index, and frame index, checks episode lengths and frame coverage, and exports eight tagged fields: task, plan, object boxes, subtask reasoning, current subtask, motion reasoning, motion command, and gripper positions. These fields supervise the corresponding latent units. Annotation is performed offline and can use the complete demonstration, including future movement information and gripper positions. These are training targets; the deployed SLR policy receives only its current observations, robot state, and task instruction.

D Qualitative Real-Robot Trajectories

Figure 9 presents seven recorded real-robot executions. Each row is a single image assembled from six frames in chronological order from left to right.

[1] `https://github.com/MichalZawalski/embodied-CoT`

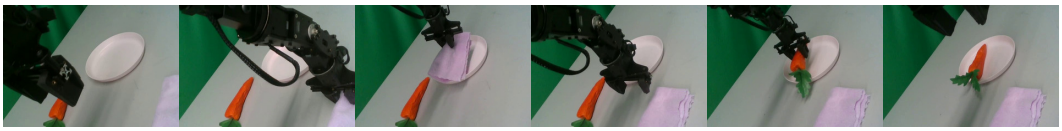
(a) Place all objects on the plate (in distribution)



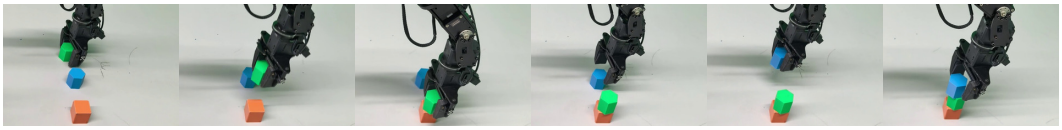
(b) Place all objects on the plate (unseen objects)



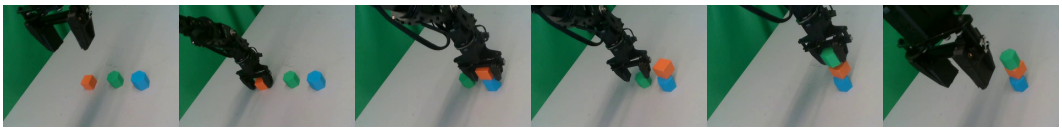
(c) Wipe the plate, then place the carrot



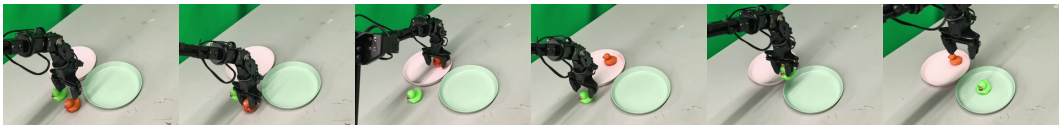
(d) Stack three blocks in order (clip 1)



(e) Stack three blocks in order (clip 2)



(f) Place the colored ducks on the instructed plates (clip 1)



(g) Place the colored ducks on the instructed plates (clip 2)



Figure 9: Ordered frame sequences from seven task-labeled real-robot videos. Six frames are sampled at evenly spaced points within each recording, and each row uses a consistent aspect ratio for display.

E Implementation and Training Details

Architecture. SLR uses the Qwen3-VL-4B-Instruct-Action vision–language backbone and a GR00T-style DiT-B flow-matching action head. We insert eight sequential latent units in the following order: task, plan, visible-object bounding boxes, subtask reasoning, current subtask, motion reasoning, motion command, and gripper position. A learned two-layer latent projection propagates the hidden state into the next latent position. For SIMPLER, the model receives one RGB image resized to 224×224 and predicts a 16-step chunk of 7-D delta end-effector actions. The flow-matching head generates each chunk using four denoising iterations.

Data and reasoning supervision. The Bridge training corpus contains 53,192 demonstrations and 1,893,026 frames recorded at 5 Hz. We align the frame-level annotations released with Embodied

CoT (Zawalski et al., 2025) to the LeRobot trajectories, obtaining reasoning targets for 1,496,265 frames. Unmatched frames retain action supervision but do not contribute to the decoder objective. Each latent unit is independently supervised by its corresponding tagged field.

The auxiliary decoder consists of a learned linear projection followed by the language-model stack and LM head initialized from the same Qwen3-VL-4B checkpoint. The language-model stack and LM head are frozen; the decoder projection, VLM, latent projection, and action head remain trainable. The total objective is the flow-matching action loss plus the unit-wise teacher-forced decoding loss with coefficient 0.1. Teacher action, teacher language-modeling, slot-distillation, and pooled distillation losses are disabled. During action inference, the decoder modules are not constructed or loaded.

Transition-adaptive training. The first 30k updates use an unweighted decoder objective. We then load the 30k model weights into a new run and train for up to 70k additional updates. In this stage, each unit's decoding loss is weighted by the action DiT's detached cross-attention to that unit. We clamp the weights to be nonnegative and normalize them to sum to one over the valid units of each transition, falling back to uniform weights if their sum is zero. The continuation restores model weights but starts a fresh optimizer and learning-rate schedule. The SLR result in Table 1 uses the 45k checkpoint of this adaptive run, corresponding to 75k effective updates; training continues to a maximum of 100k effective updates.

Optimization. The default run uses eight processes with 16 samples per device, for a global batch size of 128. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 10^{-8}, and gradient clipping at 1.0. The learning rates are 10^{-5} for the backbone and other base parameters and 10^{-4} for the action head. Each run uses 10k warm-up updates followed by cosine decay to 5×10^{-7}. Training uses bfloat16, gradient checkpointing, DeepSpeed ZeRO-2, and seed 42. The loss-weighting variants use the same global batch size of 128; the action-only ablation uses eight samples per device, giving a global batch size of 64.

F Additional Qualitative Latent-Unit Examples

Figure 10 shows six training episodes from the 45k checkpoint with visible changes in the action policy's relative attention to the eight latent reasoning units. For example, in episode 00116 the task unit is prominent early and the current-subtask unit becomes more prominent later; in episode 00119, attention shifts from object bounding boxes toward the plan unit. The plots show attention share among latent units, rather than absolute attention magnitude or a causal measure of their contribution.

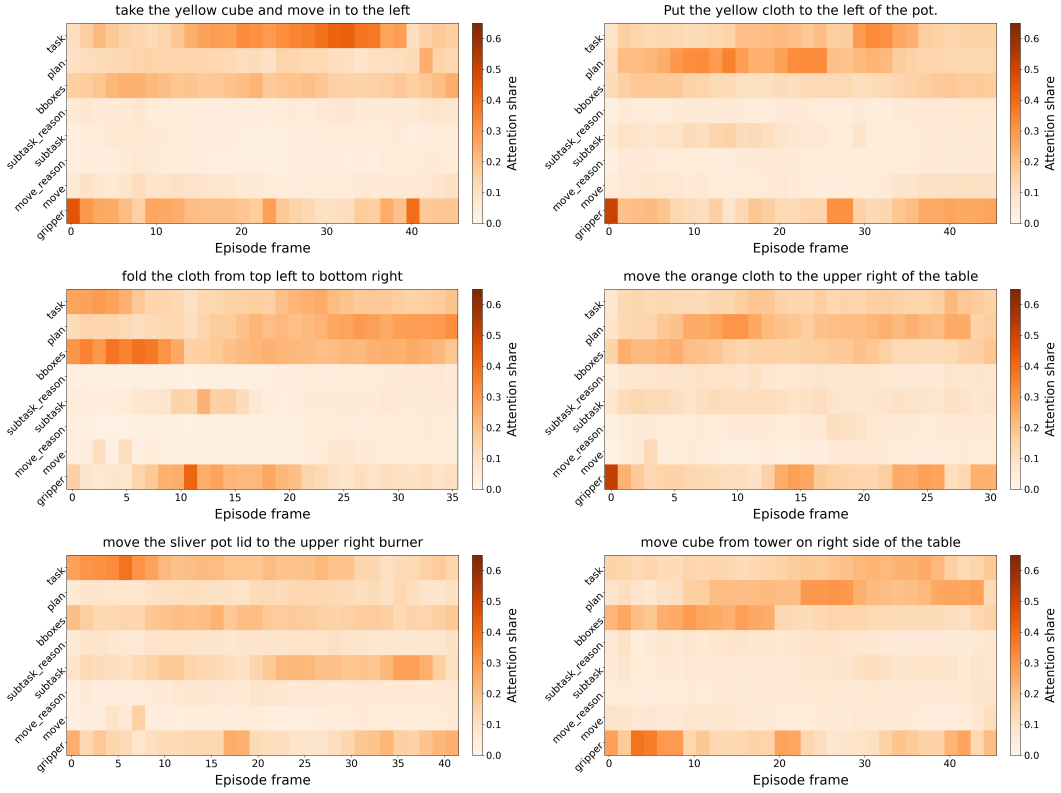
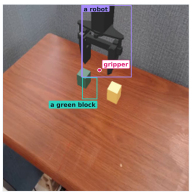


Figure 10: Temporal attention in six training episodes from the 45k checkpoint, arranged left to right and top to bottom as episodes 00026, 00039, 00079, 00089, 00116, and 00119. Rows within each heatmap are latent reasoning units and columns are episode frames. All panels use the same linear attention-share scale from 0 to 0.65.

Figure 11 presents six offline decodings from SIMPLER evaluation rollouts: two frames each for stacking, carrot placement, and spoon placement. The decoded task and plan fields recover the intended goals, while the subtask and motion fields illustrate different execution stages. The images retain the decoder's raw outputs, including occasional imperfect object labels and boxes; no reference text or manual corrections are supplied. The decoder is absent during policy execution.

Stack the green block on the yellow block



TASK: Put the green block on top of the yellow block.

PLAN: 1. Move the green block above the yellow block. 2. Grasp the green block. 3. Lift the green block. 4. Move the green block over the yellow block. 5. Drop the green block on the yellow block.

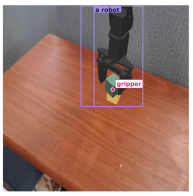
SUBTASK REASONING: The green block is on the left, so the arm needs to move there.

SUBTASK: Approach the green block.

MOVE REASONING: The arm is not moving.

MOVE: move left

Stack the green block on the yellow block



TASK: Place the green block on top of the yellow block.

PLAN: Move to the green block, grasp the green block, lift the green block, move to the yellow block, release the green block.

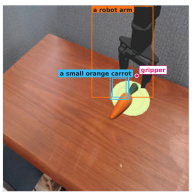
SUBTASK REASONING: The green block needs to be lifted up.

SUBTASK: Move the arm up.

MOVE REASONING: The arm needs to move up to lift the green block.

MOVE: move up

Put carrot on plate



TASK: put carrot on plate

PLAN: 1. Move to the carrot. 2. Grasp the carrot. 3. Move the carrot to the plate. 4. Release the carrot.

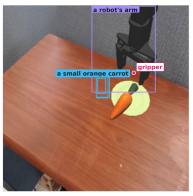
SUBTASK REASONING: The carrot has been moved above the plate and the robot needs to now move the carrot to the plate.

SUBTASK: Move the carrot above the plate.

MOVE REASONING: The arm is above the plate and needs to move up to release the carrot.

MOVE: move up

Put carrot on plate



TASK: put carrot on plate

PLAN: 1. Move to the carrot. 2. Grasp the carrot. 3. Move the carrot to the plate. 4. Release the carrot.

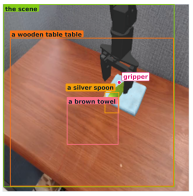
SUBTASK REASONING: The carrot has been moved above the plate and now needs to be released.

SUBTASK: Move back to the starting position.

MOVE REASONING: The arm is above the plate and needs to move up to continue moving away from the plate.

MOVE: move backward up

Put the spoon on the towel



TASK: The spoon is on the towel.

PLAN: 1. Move to the spoon 2. Grasp the spoon 3. Move the spoon to the towel 4. Put the spoon down

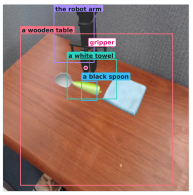
SUBTASK REASONING: The spoon can now be placed on the towel.

SUBTASK: 1. Move the spoon to the towel

MOVE REASONING: The spoon is above the towel, so the arm needs to move up to reach it.

MOVE: move up

Put the spoon on the towel



TASK: The spoon needs to be placed on the towel.

PLAN: 1. Move to the spoon and grasp it 2. Move to the towel and put the spoon on it

SUBTASK REASONING: The spoon needs to be moved to the towel now.

SUBTASK: 2. Move the spoon to the towel.

MOVE REASONING: The spoon is already above the towel, so it can be stopped

MOVE: open gripper

Figure 11: Selected offline latent-unit decodings from the 45k checkpoint. Top row: stacking at environment steps 0 and 35. Middle row: carrot placement at steps 45 and 55. Bottom row: spoon placement at steps 35 and 64. Each panel pairs an observation with the decoded reasoning fields.