RECURRENT-DEPTH VLA: IMPLICIT TEST-TIME COMPUTE SCALING OF VISION–LANGUAGE–ACTION MODELS VIA LATENT ITERATIVE REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Current Vision-Language-Action (VLA) models utilize fixed computational depth, processing simple adjustments and complex multi-step manipulations with same amount of compute. While Chain-of-Thought (CoT) prompting enables variable compute, it scales memory linearly and struggles with continuous action spaces. We introduce Recurrent-Depth VLA (RD-VLA), an architecture that employs a recurrent action head with weight-tied layers, enabling arbitrary depth with a constant memory footprint. We train the model using truncated backpropagation through time (TBPTT), allowing for efficient supervision of the refinement process. At inference, an adaptive stopping criterion based on latent convergence enables the model to dynamically allocate compute per sample. Our experiments on complex manipulation tasks demonstrate that recurrent depth is critical for success: tasks failing (0%) with single-iteration inference achieve +90% success with four iterations, while simpler tasks saturate quickly. RD-VLA provides a scalable path for test-time compute in robotics, bypassing the data and memory overhead of CoT while replacing discrete, token-based reasoning with latent reasoning, which maintains a constant memory footprint regardless of depth, and does not require any special data collection.

1 INTRODUCTION

Humans do not operate with a fixed computational budget. When performing trivial maneuvers, such as adjusting a grip or nudging an object, we rely on near-reflexive, low-effort responses. However, when environments become cluttered or actions require long-horizon foresight, human cognition adaptively reallocates resources—slowing down to process sensory evidence and refine internal models before acting (Verguts et al., 2015). This ability to scale "compute" to task complexity is a hallmark of efficient reasoning that remains a significant challenge for modern robotics. Currently, most Vision-Language-Action (VLA) models (Kim et al., 2024; Black et al., 2024; Lee et al., 2025) are limited to a fixed computational depth, processing every control step with the same number of parameters regardless of difficulty; a simple gripper adjustment receives the same compute as high-precision navigation in a cluttered space.

While generalist VLAs built upon Multimodal Large Language Models (MLLMs) exhibit impressive visual understanding, they often remain brittle and fail to fully leverage the latent reasoning capabilities of their backbone models. To bridge this gap, reasoning-centric VLAs (Huang et al., 2025; Lee et al., 2025; Zhao et al., 2025a; Zawalski et al., 2024) have emerged. These models incorporate intermediate thinking processes—typically via supervised Chain-of-Thought (CoT) which categorized into textual reasoning (generating pseudo-CoT labels) or visual reasoning (generating sub-goal images or 2D traces) (Zheng et al., 2024; Qu et al., 2025; Lee et al., 2025). However, a fundamental limitation persists: these models perform reasoning at the token level. By tethering the "thinking" process to specific task settings and explicit sequences, these methods are often inefficient or misaligned with the requirements of continuous robotic control. But unlike language modeling, reasoning for physical manipulation is inherently subtler and less conducive to tokenization; consequently, attempting to verbalize these dynamics carries significant overhead, requiring the curation of custom reasoning datasets and scaling memory linearly with the length of the reasoning chain.

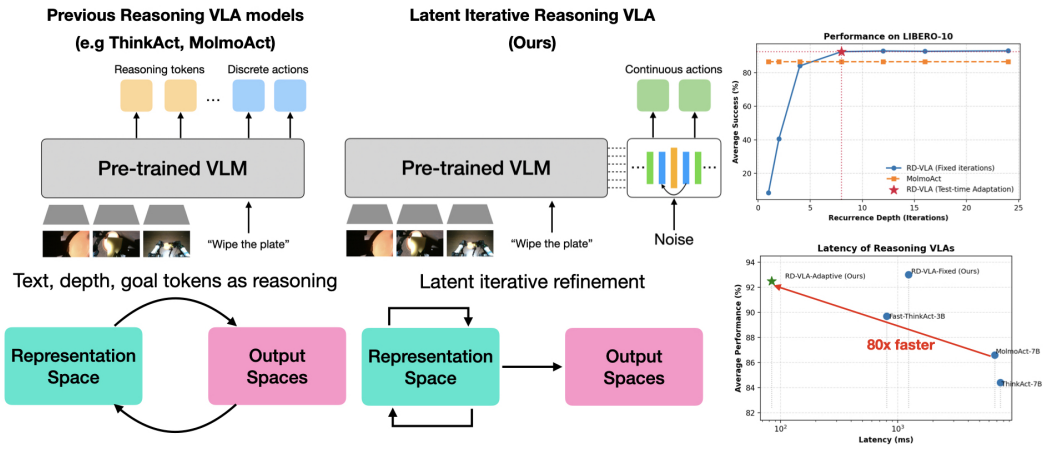


Figure 1: **Recurrent-Depth VLA.** (Left) Previous reasoning VLAs (e.g., ThinkAct, MolmoAct) generate explicit reasoning tokens in output space, requiring expensive autoregressive decoding. (Center) Our approach reasons in latent space bypassing token generation overhead. (Right) RD-VLA achieves comparable performance to autoregressive reasoning baselines on LIBERO-10 while being substantially faster due to the efficiency of latent reasoning with adaptive compute.

Beyond these constraints, a fundamental architectural limitation persists: these models perform reasoning in the output space. This requires iterative transitions between the model's high-dimensional latent space and a discretized, low-bandwidth output space (e.g., text, depth bins, or coordinates). This cyclical re-projection is fundamentally non-ideal, as it forces the model to collapse continuous internal states into lossy, discretized representations only to re-encode them for subsequent reasoning steps. The resulting information bottlenecks and quantization noise inherently limit reasoning fidelity.

To address this, we look toward the iterative nature of biological systems. Human cognition is fundamentally characterized by recurrent neural dynamics, where the repeated recruitment of the same neural circuitry transforms initial sensory inputs into progressively refined, deliberative representations (Lamme & Roelfsema, 2000). An architectural analogy in neural networks is the recursive reuse of layers, in which a model iteratively processes information until a refined decision is reached. This iterative process differs from Diffusion Policies (Chi et al., 2025), which generate actions through denoising. While effective for modeling multi-modal distributions, this process is fundamentally a generative sampling technique rather than a deliberative one: it refines the action signal but does not scale or enrich the underlying representation of the scene.

We introduce Recurrent-Depth VLA (RD-VLA), a new class of VLA model that enables adaptive test-time compute for visuomotor control. Unlike existing reasoning-centric methods, our approach enables adaptive behavior emerges naturally from its recurrent architecture, eliminating the need for curated reasoning traces or explicit supervision of iteration counts. Inspired by recent advances in latent reasoning and looped transformer architectures (Zhu et al., 2025; Geiping et al., 2025), we extend the principle of latent iterative refinement to the VLA domain. Our core insight is that robotic reasoning can occur entirely within the representation space, bypassing the need to decode intermediate tokens or perform expensive denoising iterations in action space. RD-VLA achieves this by decomposing the action head into three stages: an initial encoding stack, a weight-tied recurrent block for iterative refinement, and a final projection layer. By recursively updating hidden states through the shared recurrent block, the model progressively refines its internal representations within a fixed-dimensional latent space. This design enables arbitrarily deep computation at test time while maintaining a constant memory footprint, with minimal parameter overhead due to weight reuse.

When coupled with an adaptive stopping criterion (Figure 3), RD-VLA dynamically allocates computation on a per-sample basis, yielding a scalable and resource-efficient framework for high-precision robotic control at inference. To our knowledge, RD-VLA is the first Vision-Language-Action (VLA) model to support scaling test-time computation through implicit latent-space reason-

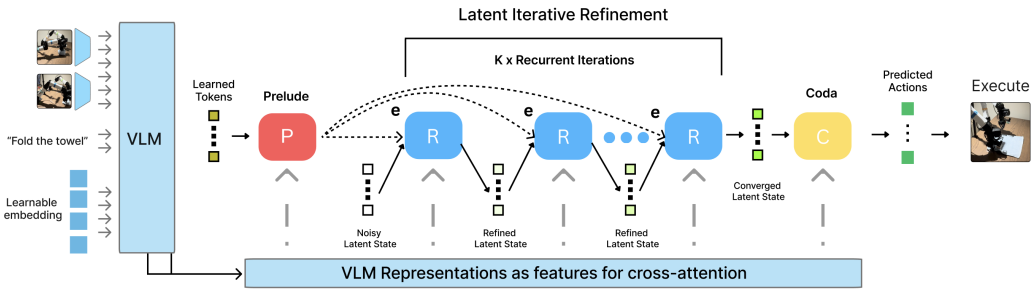


Figure 2: **Recurrent-Depth VLA Architecture.** The Prelude (P) grounds learned queries via cross-attention to mid-layer VLM features. The weight-tied Recurrent Core (R) iteratively refines a noisy latent scratchpad over K iterations, cross-attending to final-layer VLM representations and proprioception. The Coda (C) decodes the converged state into actions. Recurrence depth K adapts dynamically at inference based on task complexity.

ing via a weight-tied recurrent core. We further observe task-dependent convergence behavior: the required unrolled depth emerges naturally from the execution context, reflecting the underlying complexity of the action being performed. This property enables adaptive inference-time stopping and execution schedules that modulate the action horizon based on the model's internal reasoning trajectory, resulting in inference speeds up to an order of magnitude faster than prior action-reasoning approaches. Empirically, RD-VLA outperforms strong end-to-end VLAs and all reasoning-centric VLAs on the LIBERO benchmark (Liu et al., 2023a), achieving 93.0% success with fixed iterations and 92.5% with uncertainty-based adaptive computation. On the CALVIN benchmark (Mees et al., 2022), RD-VLA achieves an average task length of 3.39 and a task-5 success rate of 45.3%, demonstrating strong long-horizon generalization. Finally, RD-VLA transfers effectively to real-world robotic settings, achieving robust performance on challenging long-horizon tasks such as bread toasting and towel folding.

2 Related Work

2.1 Vision-Language-Action Models

Large language models (LLMs) (Achiam et al., 2023; Dubey et al., 2024; Jiang et al., 2024) and vision language models (VLMs) (Liu et al., 2023b; Wang et al., 2024; Karamcheti et al., 2024; Zhu et al., 2023) have shown strong problem-solving capabilities and deep semantic understanding of visual and textual data. Leveraging large-scale Internet data, they generalize to unseen tasks, though acquiring comparable real-world robotic data remains challenging. Despite advances in assembling large robotic datasets (Ebert et al., 2021; Mendonca et al., 2023), the Open X-Embodiment dataset (O'Neill et al., 2024) is the largest, featuring 22 robots, 527 skills, and 160,266 tasks from 21 institutions.

Using this dataset, Transformer-based general robot policies like Octo (Team et al., 2024) and RT-1 (Brohan et al., 2022) were trained. However, models trained from scratch for robotic tasks struggled to generalize to new environments, tasks, and objects. RT-2 (Zitkovich et al., 2023) addressed these issues by using a large (55B-parameter) pretrained vision-language model to generate actions, while OpenVLA (Kim et al., 2024) emerged as a leading open-source alternative. Building on these, π_0 (Black et al., 2024) introduced a generalist policy using flow-matching for multi-modal actions, with $\pi_{0.5}$(Intelligence et al., 2025) providing a more efficient, high-performance iteration for real-time deployment.

2.2 Reasoning and Efficient-Compute VLA Models

As Vision-Language-Action (VLA) models scale in parameter count, balancing the tradeoff between computational demand and the real-time requirements of robotic control has become increasingly

critical. Recent research has split into two primary directions: optimizing the efficiency of existing backbones and introducing structured reasoning stages to move beyond reactive policies.

To mitigate the high latency of large Transformer backbones, several works explore dynamic resource allocation. TinyVLA (Wen et al., 2025) focuses on data-efficient distillation to create high-performance, small-scale models. To reduce redundant computation in invariant visual scenes, VLA-Cache (Xu et al., 2025) introduces an adaptive token caching mechanism that reuses textual and visual features across control steps.

A second direction aims to improve performance by grounding actions in explicit reasoning via mid-level representations. Mobility-VLA (Chiang et al., 2024) employs long-context VLMs to reason over topological graphs for navigation, while TraceVLA (Zheng et al., 2024) utilizes visual trace prompting to enhance spatial-temporal awareness. Several recent works have integrated explicit reasoning directly into the control loop. Embodied Chain of Thought (ECoT) (Zawalski et al., 2024) leverages embodied chain-of-thought to generate textual justifications before action emission. ThinkAct (Huang et al., 2025) utilizes reinforced visual latent planning, and MolmoAct (Lee et al., 2025) introduces Action Reasoning Models (ARMs) that generate depth-aware perception tokens and editable trajectories.

2.3 Recurrent Transformers

Recurrent Transformers is an architecture where all or some portion of layers in transformers are recurred, which means that representation of the later layer gets reinjected back into the earlier layers. The initial work focused on recurring only one transformer layer (Dehghani et al., 2019), while later works explored various architectures and methods to train recurrent transformers (Geiping et al., 2025; Gatmiry et al., 2024; Hao et al., 2025; McLeish et al., 2025). Although various research mostly focused on inspecting inductive biases of recurrent architectures on toy scales, and in algorithmic setting (Wang et al., 2025; Jolicoeur-Martineau, 2025; Saunshi et al., 2024; McLeish et al., 2024), recent results had shown that these model could scale to Foundation Models sizes (Geiping et al., 2025; McLeish et al., 2025; Zhu et al., 2025). Recurrent transformers allow models to scale computation by repeating layers, enabling several useful properties: test-time scaling, where the number of computation steps can be increased at inference; uncertainty quantification for test-time scaling, since operating at the representation level makes it possible to define metrics that reflect model confidence; adaptive compute (Geiping et al., 2025).

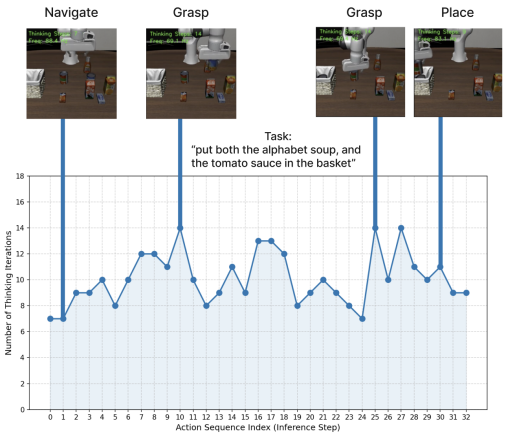


Figure 3: **Case study for adaptive computation.** In a LIBERO rollout, the model dynamically selects different numbers of iterations before terminating, depending on the execution state. It uses fewer iterations (7–9) at steps 1 and 30, which correspond to simpler motions like navigation and placing, and more iterations (about 14) at steps 10 and 25, where the actions are more complex, such as grasping.

3 METHOD

We introduce **Recurrent-Depth VLA (RD-VLA)**, a framework designed to decouple computational depth from the fixed architectural constraints of pretrained vision-language backbones. While standard VLA architectures typically utilize fixed-depth MLP heads or compute-intensive iterative processes in the output space such as diffusion or flow-matching action heads (Black et al., 2024), RD-VLA shifts the computational burden to a weight-tied recurrent transformer core operating entirely within a continuous latent manifold. This design allows the model to scale its test-time compute by unrolling the recurrent block to an arbitrary depth r, enabling dynamic allocation of compute based on task complexity (Figure 3). The RD-VLA action head is a modular framework designed to be backbone-agnostic. We instantiate our framework using a pre-trained VLM backbone (see A.1.1Appendix A.1 for details on the specific Qwen2.5-0.5B and vision encoder setup).

3.1 RECURRENT-DEPTH ARCHITECTURE

Following the Huggin approach (Geiping et al., 2025), we partition the architecture into a functional triplet: the **Prelude**, the **Recurrent Core**, and the **Coda** (Fig. 2). The Prelude and Coda serve as non-recurrent interface layers that map representations into and out of a dedicated **latent manifold** optimized for iterative reasoning.

Prelude (P_ϕ) The process begins with a non-recurrent interface that consumes $K = 8$ learned **queries**. First K queries self attend to each other bidirectionally. Then by performing cross-attention over the VLM's **middle-layer** visual features $h^{(12)}_{vis+lat}$, the Prelude transforms these queries into a grounded latent foundation $S_{pre} = P_\phi(\text{Queries}, h^{(12)}_{vis+lat}) \in \mathbb{R}^{K\times D}$.

Parallel to this, we initialize a **latent scratchpad** $S \in \mathbb{R}^{K\times D}$ from a high-entropy truncated normal distribution to serve as the evolving state for the reasoning process:

$$S_0 \sim \text{TruncNormal}(0, \gamma_{init} \cdot \sigma_{init}) \tag{1}$$

This noisy initialization transforms S into a blank workspace that the model must iteratively "clean" and refine, ensuring the model learns a stable refinement operator.

Recurrent Core (R_θ) The core iterative refinement occurs within a weight-tied transformer block. To maintain representational stability and prevent the model from losing its grasp on the physical observations over long unrolls (representational collapse), we utilize a persistent **Input Injection** strategy. Specifically, for each iteration $k = 1 \dots r$, the previous scratchpad state S_{k-1} is concatenated with the fixed S_{pre} along the feature dimension to form a $2D$-dimensional context. This is mapped back to the manifold dimension via a learned adapter and normalized:

$$x_k = \text{RMSNorm}\left(\gamma_{adapt} \cdot W_{adapt}[S_{k-1}; S_{pre}]\right) \tag{2}$$

where $W_{adapt} \in \mathbb{R}^{D\times 2D}$. The scratchpad state is then updated through the weight-tied recurrent block $S_k = R_\theta(x_k, [h^{(24)}_{vis+lat}; p])$. Similar to Prelude, R_θ performs bidirectional self-attention across K, followed by gated cross-attention to a manifold consisting of the 64 task-aligned latent tokens, 512 vision tokens from the VLM's final layer, and the robot's current proprioception p.

Coda (C_ψ) and Action Projection Once the recurrence reaches the desired depth r, the converged scratchpad S_r is processed by the non-recurrent Coda. The Coda moves the representation out of the latent manifold, attending to the self and high-level VLM features ($h^{(24)}_{vis+lat}$). Finally, an **output projection** layer maps these refined features to the robot's action space:

$$\mathbf{a} = W_{out} \cdot \text{RMSNorm}(C_\psi(S_r, [h^{(24)}_{vis}; h^{(24)}_{lat}; p])) \tag{3}$$

where W_{out} is the final linear layer producing the control commands $\mathbf{a}$. This ensures that any intermediate state S_k is a valid representation, while S_{k+1} is a strictly more refined iteration.

3.2 ADAPTIVE COMPUTATION

Leveraging the properties of the model, we implement an adaptive computation mechanism at inference by utilizing the model's own internal convergence as a proxy for reasoning certainty. We define

Table 1: Comparison on the LIBERO (Liu et al., 2023a) benchmark. **Bold*** indicates the best performance, **Bold** the second best, and *Italics* the third best. "Params" denotes backbone scale in **B**illions. Here we compare three types of VLAs. **End-to-end (E2E) VLAs** directly predict robot actions without intermediate reasoning steps. **Token reasoning** VLAs first perform explicit reasoning by generating tokens before producing action outputs. Finally, **latent reasoning**, our approach, performs iterative reasoning in a latent space before emitting actions.

Method		Params	Spat.	Obj.	Goal	Long	Avg.
E2E VLAs	SmolVLA (Shukor et al., 2025)	2.2	93.0	94.0	91.0	77.0	88.8
	OpenVLA (Kim et al., 2024)	7	84.7	88.4	79.2	53.7	76.5
	WorldVLA (Cen et al., 2025)	7	87.6	96.2	83.4	60.0	81.8
	π_0-FAST (Black et al., 2024)	3	96.4	96.8	88.6	60.2	85.5
Token Reasoning	CoT-VLA (Zhao et al., 2025a)	7	87.5	91.6	87.6	69.0	81.1
	FlowVLA (Zhong et al., 2025)	8.5	93.2	95.0	91.6	72.6	88.1
	SpatialVLA (Qu et al., 2025)	4	88.2	89.9	78.6	55.5	78.1
	ThinkAct (Huang et al., 2025)	7	88.3	91.4	87.1	70.9	84.4
	Fast-ThinkAct (Huang et al., 2026)	3	92.0	97.2	90.2	79.4	89.7
	TraceVLA (Zheng et al., 2024)	7	84.6	85.2	75.1	54.1	74.8
	MolmoAct (Lee et al., 2025)	7	87.0	95.4	87.6	77.2	86.6
Latent Reasoning	**RD-VLA (Fixed)**	**0.5**	**92.0**	**99.0**	**96.0**	**84.8**	**93.0**
	RD-VLA (Adaptive)	**0.5**	**88.6**	**98.8**	**96.8**	**85.8**	**92.5**

a stopping criterion based on the Kullback-Leibler (KL) divergence between the action distributions of consecutive iterations. Approximating KL via Mean Squared Error (MSE) in the action space, the inference loop terminates at step k^* when:

$$||\mathbf{a}_k - \mathbf{a}_{k-1}||_2^2 < \delta \tag{4}$$

where $\mathbf{a}_k$ is the predicted action chunk at step k and δ is a convergence threshold (e.g., $1e^{-3}$). This allows the model to self-regulate: terminating instantly for trivial movements while allocating extended compute for complex situations.

3.3 Adaptive Execution

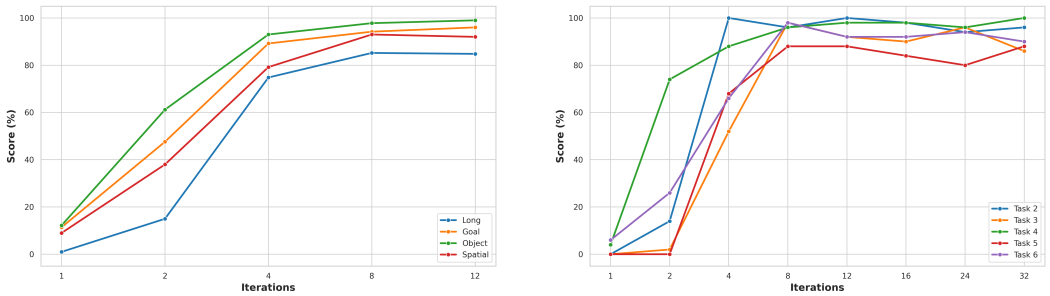
Adaptive computation determines how long to recur. At the same time adaptive execution determines how many actions to execute. We observe that instances requiring deep recurrence ($k^* > 8$) often correspond to states of high uncertainty. In these regimes, executing a long horizon of actions is dangerous, as small errors in the initial plan compound over time. We discuss two strategies to couple the depth of reasoning with the execution of action in Appendix C.4.

4 Experiments

We evaluate our approach in both simulation and real-world settings. Simulation experiments are conducted on two widely used manipulation benchmarks, LIBERO (Liu et al., 2023a) and CALVIN (Mees et al., 2022), while real-world experiments are performed on a bimanual YAM manipulator. Our evaluation is designed to answer the following questions:

- 4.1 How does our approach scale with recurrent computation?
- 4.2 Is adaptive computation necessary for robotic manipulation tasks?
- 4.3 What is the most effective strategy for adaptive computation at inference time?
- 4.4 Is representation-level reasoning for action prediction more effective than token-level reasoning?
- 4.5 How well does our approach perform in real-world deployment?

In this section, we evaluate Recurrent-Depth VLA (RD-VLA) across three dimensions: (1) the scaling behavior of latent recurrence on standard benchmarks, (2) the efficacy of adaptive test-time compute, and (3) performance comparisons against state-of-the-art baselines. We utilize the LIBERO



(a) Aggregate Performance

(b) Task-Dependent Convergence

Figure 4: **Performance scaling with recurrent depth.** (a) Aggregate performance across all LIBERO task suites shows consistent log-linear improvement, saturating between 8–12 iterations. (b) Individual task analysis reveals distinct convergence profiles: simpler tasks converge quickly (e.g., Task 4 at iter 2), while complex tasks require deeper recurrence (e.g., Task 5 at iter 3), demonstrating the emergent adaptive behavior of the model.

(Liu et al., 2023a) and CALVIN (Mees et al., 2022) benchmarks for quantitative analysis and conduct real-world evaluations to assess physical robustness.

4.1 Performance Scaling via Recurrent Computation

We first establish the relationship between computational depth and task success rates by evaluating RD-VLA with a fixed number of recurrent iterations $N_{inf} \in \{1, \ldots, 32\}$ on all task suites of LIBERO. As shown in Figure 4a, performance exhibits a clear log-linear improvement with increased recurrence. Starting from near-random performance at $N_{inf} = 1$ (8.4% average), the model achieves substantial gains at each doubling of compute: 40.5% at $N_{inf} = 2$ (382% increase), 84.1% at $N_{inf} = 4$ (108% increase), and 92.6% at $N_{inf} = 8$ (10% increase). Performance saturates between 8 and 12 iterations, with the peak of 93.1% achieved at $N_{inf} = 24$. Notably, scaling beyond $N_{inf} = 12$ yields diminishing returns, suggesting that for the given task distribution, the model shows an increasing trend of its performance on different tasks by scaling up the compute budget at test-time.

4.2 Necessity of Task-dependent Computation

Beyond aggregate performance, we observe that individual tasks exhibit distinct convergence profiles, reflecting that different tasks have different requirements for computation complexity. Figure 4b illustrates this task-dependent behavior on selected Long-horizon tasks from LIBERO.

Critically, the number of required iterations emerges naturally from the task context rather than being prescribed. Some tasks (e.g., Task 4) achieve near-perfect performance with just two iterations, while others (e.g., Task 5) require three or more iterations before any meaningful success. This observation reveals the phenomenon that different tasks might have different optimal iteration counts in the latent reasoning process. This further motivates our adaptive computation strategy: rather than using a fixed compute budget, the model should dynamically allocate iterations based on the difficulty of the current state.

4.3 Adaptive Computation Strategies

We evaluate whether the model can dynamically calibrate its own compute budget using the KL-divergence stopping criterion described in Section 3.2. Table 3 presents comprehensive ablation studies comparing fixed computational depths against three adaptive strategies: Binary Adaptation, Linear Decay, and Pure KL thresholding.

Results demonstrate that adaptive computation maintains parity with the best fixed-depth models while reducing average inference cost. With $\tau = 5 \times 10^{-4}$, the Binary Adaptation strategy achieves 92.5% success rate (comparable to the 93.0% peak of fixed recurrence at $N = 12$) using an aver-

Table 2: Comparison on the CALVIN ABC→D benchmark. We report tasks completed in a row (↑) and average episode length (↑).

CALVIN ABC→D	**Params**	**Tasks completed in a row ↑**					**Avg. len ↑**
		1	2	3	4	5	
OpenVLA(Kim et al., 2024)	7	91.3	77.8	62.0	52.1	43.5	3.27
VLAS (Zhao et al., 2025b)	7	87.2	64.2	40.9	28.1	19.6	2.40
LCB (Shentu et al., 2025)	7	73.6	50.2	28.5	16.0	9.9	1.78
DeeR (Yue et al., 2025)	3	86.2	70.1	51.8	41.5	30.4	2.82
RoboFlamingo (Li et al., 2024)	3	82.4	61.9	46.6	33.1	23.5	2.48
GR-1 (Wu et al., 2023)	3.1	85.4	71.2	59.6	49.7	40.1	3.06
SuSIE (Black et al., 2023)	1.3	87.0	69.0	49.0	38.0	26.0	2.69
RT-1 (Brohan et al., 2023)	–	53.3	22.2	9.4	3.8	1.3	0.90
RD-VLA (Ours)	**0.5**	**91.4**	**79.5**	**67.9**	**54.9**	**45.3**	**3.39**

age of only $\bar{k} = 7.93$ iterations, which is a **34% reduction** in compute. Figure **??** illustrates the distribution of adaptive exits across task categories, showing that the model dynamically adjusts the number of recurrent steps based on the difficulty of the current state.

4.4 Performance against Other Baselines

We compare RD-VLA against state-of-the-art VLA methods on the LIBERO and CALVIN benchmarks. Table 1 presents results on LIBERO, where methods are grouped into three categories: end-to-end VLAs, token-level reasoning methods, and our latent reasoning approach.

RD-VLA achieves state-of-the-art performance of **93.0%** with fixed recurrence, significantly outperforming all prior methods, including the strong Fast-ThinkAct baseline (89.7%). Remarkably, our model achieves this with only **0.5B parameters** which is 14× smaller than 7B token-reasoning methods. The adaptive variant maintains competitive performance (92.5%) while providing the efficiency benefits of dynamic compute allocation.

Table 2 presents results on the CALVIN ABC→D benchmark, which evaluates long-horizon task chaining. RD-VLA achieves the highest average chain length of **3.39**, outperforming OpenVLA (Kim et al., 2024) (3.27) and all other baselines. This validates that latent refinement effectively extends the model's sequential planning capabilities.

4.5 Real-world Experiments

To validate the capability of our method in the real-world setting, we deployed RD-VLA on a bimanual robot arm (bimanual YAM) across four household tasks: placing a cube in a bowl, wiping a dish, folding a towel, and toasting bread. We train and compare our model in each task, both using fixed 8 iterations and Pure KL with threshold $\tau = 10^{-4}$. Figure 5 presents task progression scores compared to Diffusion Policy and $\pi_{0.5}$ baselines. RD-VLA with fixed 8 iterations demonstrates superior robustness across all tested scenarios, reaching nearly 100% completion on dish wiping and significantly outperforming baselines on the remaining tasks. The adaptive variant (RD-VLA-adaptive, Pure KL with threshold $\tau = 10^{-4}$) maintains competitive performance, matching or closely trailing the fixed-strategy model while achieving the highest score on cube placement. This demonstrates the viability of dynamic computation in physical settings, even when slightly trading off performance on complex manipulation tasks like towel folding.

5 Conclusion and Future Work

We introduced **Recurrent-Depth VLA (RD-VLA)**, a novel architecture that shifts robotic reasoning from the discrete output space to the continuous latent space. This work serves as the first implementation of *latent iterative reasoning* for visuomotor policies, demonstrating that effective

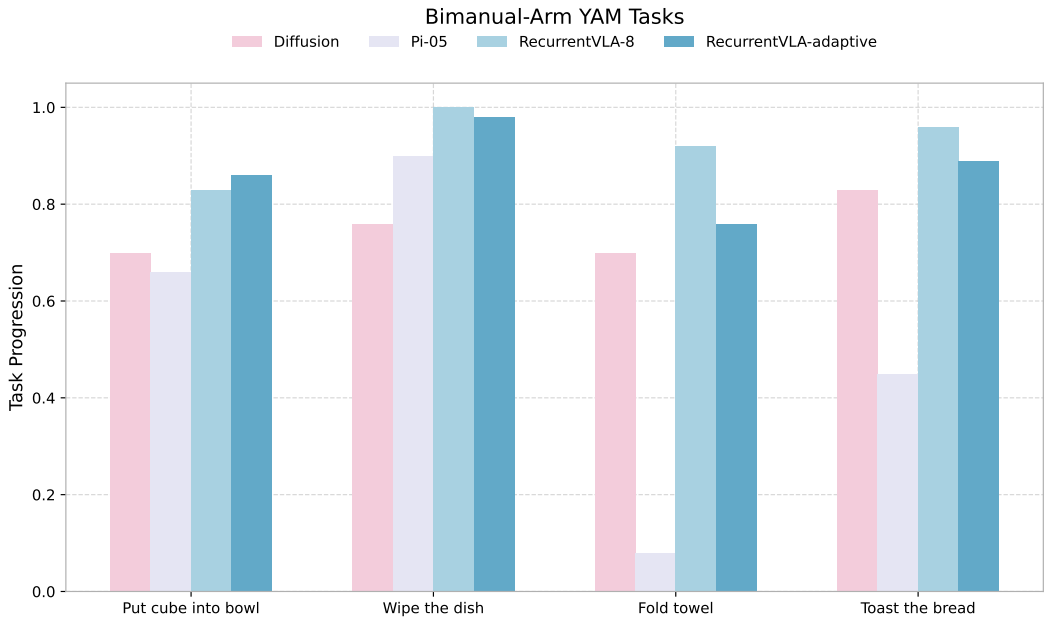


Figure 5: Performance on real-world tasks compared to $\pi_{0.5}$ and Diffusion Policy baselines. RD-VLA variants consistently outperform baselines across all tasks, with the fixed 8-iteration model achieving near-perfect performance on dish wiping.

test-time compute scaling can be achieved without the memory and latency overhead of autoregressive Chain-of-Thought generation. Our experiments provide substantial evidence that the model successfully utilizes recurrent iterations to refine its internal state, with performance scaling log-linearly with compute depth. Crucially, this architecture unlocks new capabilities for embodied agents: the ability to think longer for harder tasks and the capacity to measure its own uncertainty through latent convergence. We aimed to open a new design space for efficient, reasoning-capable robotic policies. We believe that optimizing the regimes for adaptive compute and exploring the scaling laws of latent recurrence represent promising avenues for future research. By validating the efficacy of iterative latent refinement, RD-VLA provides a foundation for the next generation of resource-efficient and robust foundation models in robotics.

REFERENCES

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

Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pretrained image-editing diffusion models, 2023. URL `https://arxiv.org/abs/2310.10639`.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian

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

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

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.

Hao-Tien Lewis Chiang, Zhuo Xu, Zipeng Fu, Mithun George Jacob, Tingnan Zhang, Tsang-Wei Edward Lee, Wenhao Yu, Connor Schenck, David Rendleman, Dhruv Shah, et al. Mobility vla: Multimodal instruction navigation with long-context vlms and topological graphs. *arXiv preprint arXiv:2407.07775*, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2019. URL https://arxiv.org/abs/1807.03819.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

Khashayar Gatmiry, Nikunj Saunshi, Sashank J. Reddi, Stefanie Jegelka, and Sanjiv Kumar. On the role of depth and looping for in-context learning with task diversity, 2024. URL https://arxiv.org/abs/2410.21698.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025. URL https://arxiv.org/abs/2502.05171.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

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. *arXiv preprint arXiv:2507.16815*, 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*, 2026.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025. URL https://arxiv.org/abs/2510.04871.

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

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.

V A Lamme and P R Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends Neurosci*, 23(11):571–579, November 2000.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, et al. Molmoact: Action reasoning models that can reason in space. *arXiv preprint arXiv:2508.07917*, 2025.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, Hang Li, and Tao Kong. Vision-language foundation models as effective robot imitators, 2024. URL https://arxiv.org/abs/2311.01378.

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

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023b.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. Transformers can do arithmetic with the right embeddings, 2024. URL https://arxiv.org/abs/2405.17399.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence, 2025. URL https://arxiv.org/abs/2511.07384.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022. URL https://arxiv.org/abs/2112.03227.

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

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

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)*, pp. 6892–6903. IEEE, 2024.

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

Nikunj Saunshi, Stefani Karp, Shankar Krishnan, Sobhan Miryoosefi, Sashank J. Reddi, and Sanjiv Kumar. On the inductive bias of stacking towards improving reasoning, 2024. URL https://arxiv.org/abs/2409.19044.

Yide Shentu, Philipp Wu, Aravind Rajeswaran, and Pieter Abbeel. From llms to actions: Latent codes as bridges in hierarchical robot control, 2025. URL https://arxiv.org/abs/2405.04798.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

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

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

Tom Verguts, Eliana Vassena, and Massimo Silvetti. Adaptive effort investment in cognitive and physical tasks: A neurocomputational model. *Frontiers in Behavioral Neuroscience*, 9:57, 2015.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

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

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation, 2023. URL https://arxiv.org/abs/2312.13139.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation. *arXiv preprint arXiv:2502.02175*, 2025.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2025.

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

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

Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 1702–1713, 2025a.

Wei Zhao, Pengxiang Ding, Min Zhang, Zhefei Gong, Shuanghao Bai, Han Zhao, and Donglin Wang. Vlas: Vision-language-action model with speech instructions for customized robot manipulation, 2025b. URL https://arxiv.org/abs/2502.13508.

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

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. *arXiv preprint arXiv:2508.18269*, 2025.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models, 2025. URL https://arxiv.org/abs/2510.25741.

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*, pp. 2165–2183. PMLR, 2023.

A APPENDIX

A MODEL AND TRAINING DETAILS

A.1 MODEL ARCHITECTURE

The **RD-VLA** architecture used for experiments employs a specialized depth-distributed design. The motivation behind this design is to ground the initial planning phase in mid-level semantic features (VLM's middle layer representations) while performing iterative reasoning and final action decoding using the most abstract, high-level representations from the VLM backbone.

A.1.1 VLM BACKBONE

We utilize the **Qwen2.5-0.5B** language model (Team, 2024) as the backbone. Visual perception is handled by a fused encoder combining **SigLIP** (Zhai et al., 2023) and **DINOv2** (Oquab et al., 2023). The vision encoder processes 224×224 input images, generating 256 vision tokens per image (512 total for a dual-camera setup), which are projected into the LLM's embedding space.

A.1.2 RECURRENT-DEPTH ACTION HEAD

The action head operates on a latent dimension of $D = 896$ and is structured into three functional stages. Crucially, each stage utilizes cross-attention to specific intermediate layers of the frozen VLM backbone:

1. **Prelude (P_ϕ):** A single-layer non-recurrent interface.
   - **Function:** Initializes the latent reasoning scratchpad.
   - **Grounding:** It cross-attends to **VLM Layer 12**. This connects the initial state to mid-level visual-semantic features, providing a grounded starting point for the reasoning process.
2. **Recurrent Core (R_θ):** The central iterative reasoning engine.
   - **Structure:** Consists of **2 transformer layers**.
   - **Grounding:** At every iteration step, both layers cross-attend to **VLM Layer 24** (index 23), the final layer of the backbone. This ensures that the iterative refinement process is conditioned on the model's most abstract and fully contextualized representations.
   - **Depth:** The core is unrolled for a mean of $\mu = 32$ iterations during training.
3. **Coda (C_ψ):** The final decoding stage.
   - **Structure:** A single-layer transformer block.
   - **Function:** Projects the converged latent state into the 7-dimensional action space (6-DoF pose + gripper).
   - **Grounding:** Like the core, it conditions on **VLM Layer 24** to ensure the final output policy is derived from high-level semantics.

A.2 TRAINING DETAILS

We train RD-VLA using an implicit differentiation objective that encourages the model to refine its internal state over variable compute paths.

A.2.1 OPTIMIZATION

The model is trained with the following hyperparameters:

- **Optimizer:** AdamW with a learning rate of 2×10^{-5}.
- **Schedule:** A linear warmup is applied for the first 500 steps, followed by a cosine decay schedule.
- **Batch Size:** We use a batch size of 64 with 1 gradient accumulation step, resulting in an effective batch size of 64.
- **Steps:** The model is trained for a total of 120,000 steps, with checkpoints saved every 5,000 steps. We then ran eval on each version to choose the best one for each benchmark.

A.2.2 Recurrent Training via TBPTT

To enable test-time compute scaling, we train the model with randomized recurrence depths using Truncated Backpropagation Through Time (TBPTT):

- **Depth Sampling:** The number of recurrent iterations is sampled from a Poisson distribution with a mean of $\mu_{\text{rec}} = 32$. This forces the model to learn a stable update operator that remains valid across a wide range of depths.
- **Truncated Gradients:** Gradients are propagated only through the final $d = 8$ iterations of the unrolled sequence to manage memory usage while capturing sufficient reasoning dependencies.
- **Initialization:** The latent scratchpad is initialized from a truncated normal distribution with $\sigma = 0.632$ (derived from $\sqrt{2/5}$), simulating a noisy starting state that requires refinement.

A.3 Parameter-Efficient Fine-Tuning (LoRA)

To adapt the pre-trained Qwen2.5-0.5B backbone for robotic control without destroying its generalist capabilities, we employ Low-Rank Adaptation (LoRA).

A.3.1 Configuration

- **Rank:** We use a rank of $r = 64$, applied to the attention modules of the LLM.
- **Dropout:** LoRA dropout is set to 0.0, relying on the frozen backbone and limited rank for regularization.
- **Frozen Components:** The entire vision encoder (SigLIP + DINOv2) and the base weights of the LLM are kept frozen. Only the LoRA adapters, the Projector, and the Recurrent-Depth Action Head are trainable.

A.3.2 Deployment

Upon completion of training, we perform a merge operation where the learned LoRA weights are algebraically folded back into the base model parameters:

$$W_{\text{final}} = W_{\text{base}} + BA \tag{5}$$

This ensures that the deployed model does not incur additional inference latency compared to the standard backbone, maintaining the efficiency required for real-time robotic control.

B Extended Convergence and Scaling Analysis

This section provides a comprehensive empirical analysis of the recurrent core's convergence properties and test-time scaling behavior. We organize the results into two parts: global convergence diagnostics (§B.5), and adaptive computation trade-off (§B.6).

B.1 Detailed Ablation Studies

Table 3 presents the detailed performance breakdown across all adaptive thresholds and fixed iteration counts. These results support the main text findings that adaptive computation ($\tau = 5e^{-4}$) achieves parity with the best fixed models while reducing computational cost.

Table 3: Comprehensive Ablation of Recurrent-Depth VLA. We compare fixed computational depths against adaptive strategies (Binary, Linear Decay, and Pure KL). For adaptive runs, we report Mean Iterations ($\bar{k}$) and Standard Deviation (σ) to illustrate reasoning convergence.

Strategy & Threshold τ		$\bar{k}$	σ	Spatial	Object	Goal	Long	Avg.
Fixed Recurrence	Rec=1	1.0	0.00	9.0	12.2	11.4	1.0	8.4
	Rec=2	2.0	0.00	38.0	61.2	47.6	15.0	40.5
	Rec=4	4.0	0.00	79.2	93.0	89.2	74.8	84.1
	Rec=8	8.0	0.00	93.0	97.8	94.2	85.2	92.6
	Rec=12	12.0	0.00	92.0	99.0	96.0	84.8	93.0
	Rec=16	16.0	0.00	91.6	99.0	95.2	85.2	92.8
	Rec=24	24.0	0.00	92.4	99.2	94.2	86.6	**93.1**
	Rec=32	32.0	0.00	91.4	98.8	93.8	84.4	92.1
Binary Adaptation	$\tau = 1e^{-4}$	11.04	1.20	91.2	98.2	96.0	79.8	91.3
	$\tau = 2e^{-4}$	9.71	1.11	90.6	97.2	96.0	80.8	91.2
	$\tau = 5e^{-4}$	7.93	1.03	88.6	98.8	96.8	85.8	**92.5**
	$\tau = 1e^{-3}$	6.61	0.89	88.6	97.8	94.6	84.8	91.5
	$\tau = 5e^{-3}$	4.27	0.41	83.2	93.6	87.4	61.8	81.5
	$\tau = 1e^{-2}$	3.36	0.19	74.6	88.6	81.6	43.6	72.1
Linear Decay	$\tau = 1e^{-4}$	11.55	1.03	91.2	97.2	96.2	80.0	91.2
	$\tau = 2e^{-4}$	9.86	1.10	91.2	98.4	96.6	82.0	**92.1**
	$\tau = 5e^{-4}$	7.90	1.05	88.8	97.6	94.4	82.0	90.7
	$\tau = 1e^{-3}$	6.62	0.90	89.0	98.4	94.6	83.8	91.5
	$\tau = 5e^{-3}$	4.26	0.42	84.2	94.4	90.6	62.2	82.9
	$\tau = 1e^{-2}$	3.36	0.19	76.2	87.2	81.6	42.6	71.9
Pure KL (Threshold)	$\tau = 1e^{-4}$	10.58	1.40	91.2	98.2	93.8	81.8	91.3
	$\tau = 2e^{-4}$	9.25	1.12	92.0	98.6	95.0	82.0	**91.9**
	$\tau = 5e^{-4}$	7.66	0.95	90.8	99.4	93.2	82.0	91.4
	$\tau = 1e^{-3}$	6.57	0.79	89.8	98.2	93.8	79.6	90.4
	$\tau = 5e^{-3}$	4.30	0.40	86.2	91.4	87.2	50.8	78.9
	$\tau = 1e^{-2}$	3.38	0.18	75.8	82.2	81.8	33.8	68.4

B.2 Real-World Rollouts

To validate that the learned policy transfers beyond simulation, we deploy RD-VLA on a real robotic manipulator across four household tasks. Each strip shows temporally sampled keyframes from a single episode using adaptive computation with $\tau = 1 \times 10^{-4}$.

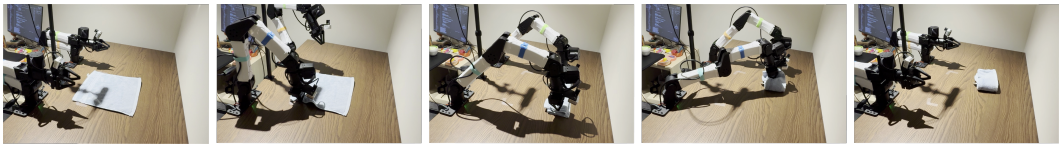


Figure 6: **Real-world:** *"Fold the towel."* A deformable-object manipulation task requiring the model to reason about cloth geometry. The policy successfully grasps the towel edge, lifts, and executes a folding motion.

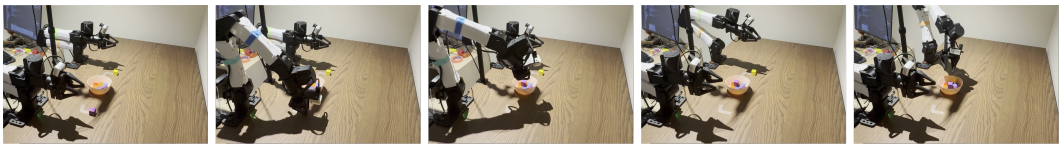


Figure 7: **Real-world:** *"Put cube into bowl."* A precision placement task requiring accurate spatial reasoning to locate small objects and place them within a constrained target region.

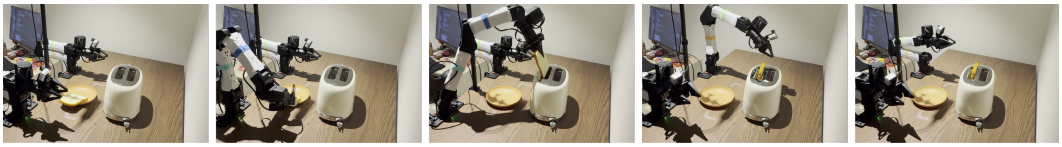


Figure 8: **Real-world:** *"Toast the bread."* The model must reason about the narrow insertion geometry of the toaster slot, requiring fine-grained position and orientation control during the final placement phase.

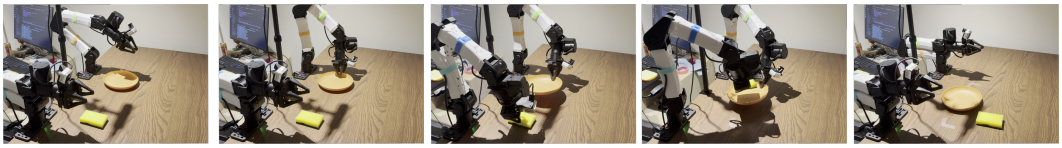
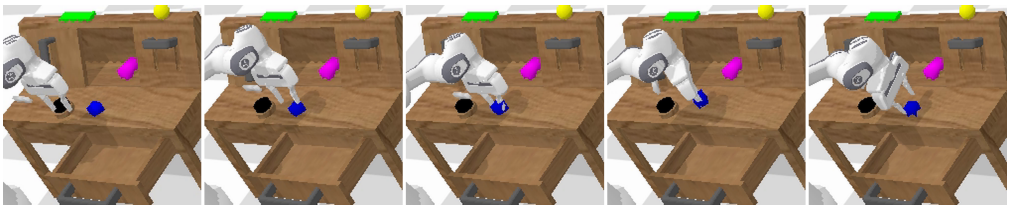


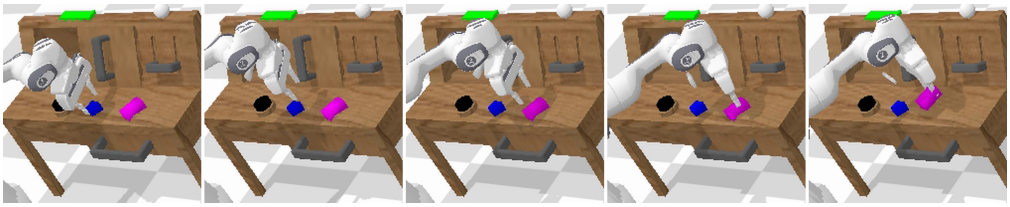
Figure 9: **Real-world:** *"Wash the dish."* A contact-rich task involving picking up a dish and performing a washing motion. The policy demonstrates robustness to the dynamic forces encountered during the scrubbing interaction.

B.3 CALVIN BENCHMARK ROLLOUTS



(a) *"Rotate the blue block right."*

(b) *"Lift the blue block from the drawer."*



(c) *"Lift the pink block from the table."*

Figure 10: **CALVIN benchmark rollouts.** Successful executions of three language-conditioned manipulation tasks in the CALVIN environment.

B.4 QUALITATIVE ROLLOUT VISUALIZATIONS WITH ADAPTIVE COMPUTATION

We present representative rollouts from the LIBERO benchmark executed with adaptive computation using a KL-divergence threshold of $\tau = 1 \times 10^{-4}$. Each strip shows temporally sampled keyframes from a single episode, with the number of recurrent thinking steps and the resulting control frequency overlaid on each frame. These visualizations illustrate how the model dynamically allocates more computation to challenging moments (e.g., grasping, placement) while maintaining higher frequency during simpler motions (e.g., free-space transit).

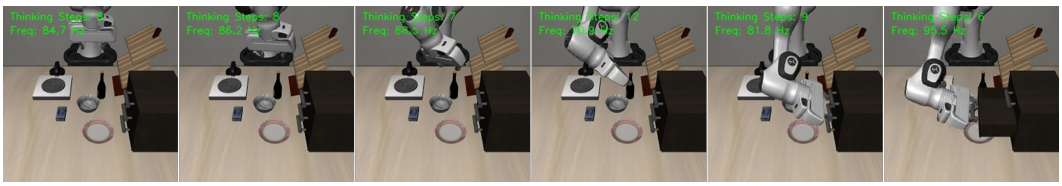


Figure 11: **LIBERO-Goal:** *"Open the middle drawer of the cabinet."* The model allocates additional thinking steps during the grasp-and-pull phase, then reduces computation as the drawer slides open.

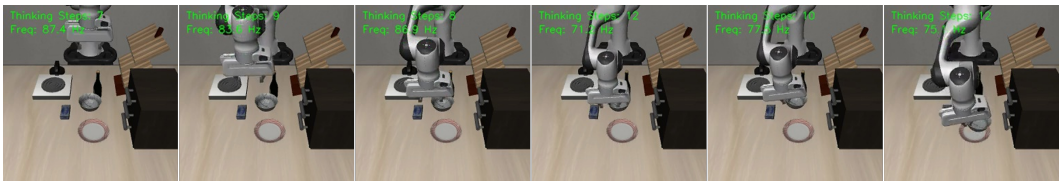


Figure 12: **LIBERO-Goal:** *"Put the bowl on the plate."* Thinking steps peak during the precision placement phase, reflecting the spatial accuracy required for stable placement.

Figure 13: **LIBERO-Long:** *"Put both the cream cheese box and the butter in the basket."* A multi-step task requiring two sequential pick-and-place operations. The model increases computation at each grasp and release transition.

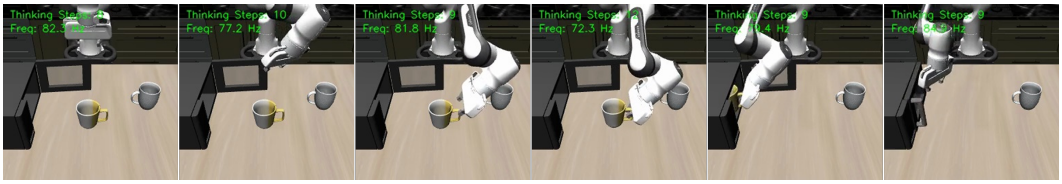


Figure 14: **LIBERO-Long:** *"Put the yellow and white mug in the microwave and close it."* The model successfully chains grasping, placing inside the microwave, and closing the door, with elevated thinking steps during each sub-goal transition.

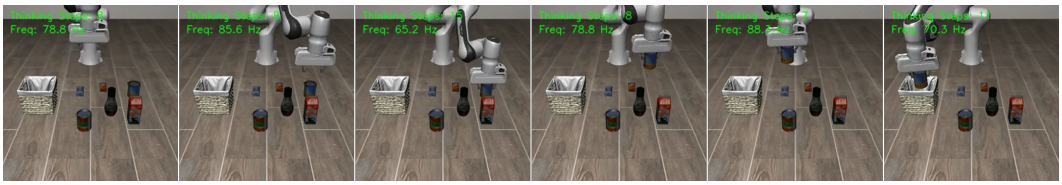


Figure 15: **LIBERO-Object:** *"Pick up the alphabet soup and place it in the basket."* Computation increases during object identification and grasp planning in a cluttered scene.

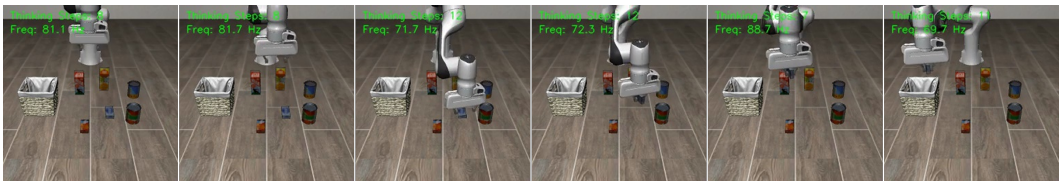


Figure 16: **LIBERO-Object:** *"Pick up the cream cheese and place it in the basket."* Similar adaptive pattern with peak computation at the grasp point.

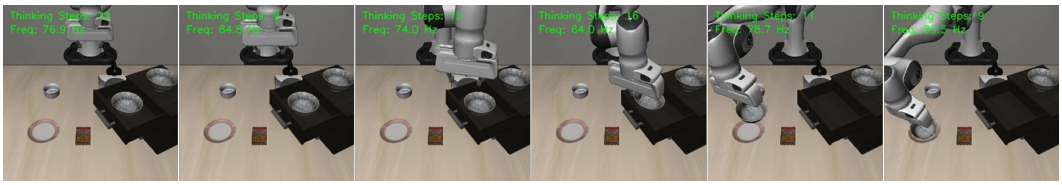


Figure 17: **LIBERO-Spatial:** *"Pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate."* A spatially complex task requiring the model to reason about the drawer location, reach inside, and perform a precise placement.

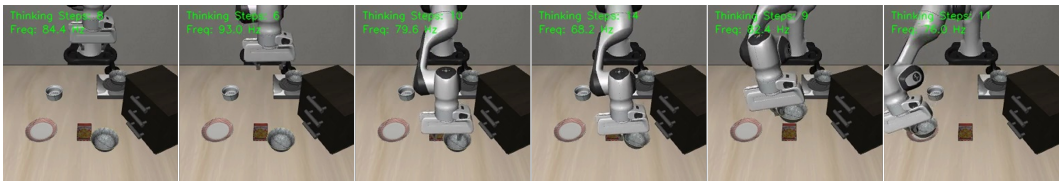


Figure 18: **LIBERO-Spatial:** *"Pick up the black bowl next to the cookie box and place it on the plate."* The spatial reference ("next to the cookie box") requires disambiguation among similar objects, reflected in elevated thinking steps during the approach phase.

Figure 19: **LIBERO-Goal (failure):** *"Put the wine bottle on top of the cabinet."* The model fails to achieve stable placement on the cabinet top. Despite increased thinking steps during the lift phase, the required vertical reach and precision placement exceed the policy's capability.

Figure 20: **LIBERO-Long (failure):** *"Put both the alphabet soup and the tomato sauce in the basket."* The model succeeds on the first object but fails to complete the second pick-and-place, illustrating the compounding difficulty of long-horizon multi-step tasks.

Figure 21: **LIBERO-Object (failure):** *"Pick up the orange juice and place it in the basket."* The model fails to drop it on time (under max number of steps allowed).

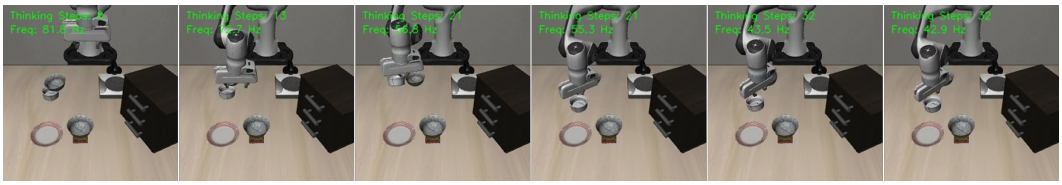


Figure 22: **LIBERO-Spatial (failure):** *"Pick up the black bowl on the ramekin and place it on the plate."* The stacked spatial relationship ("on the ramekin") presents a challenging disambiguation problem that the model fails to resolve, resulting in an incorrect grasp target. One can observe how failure cases spike *Thinking Steps* to high values like *32* and *21*.

Figure 23: **LIBERO-Long task observations.** Dual-camera inputs for 10 LIBERO-Long tasks. Each cell shows the main (third-person) and wrist (eye-in-hand) camera views that constitute the model's visual observation at a given timestep.

B.5 GLOBAL CONVERGENCE DIAGNOSTICS

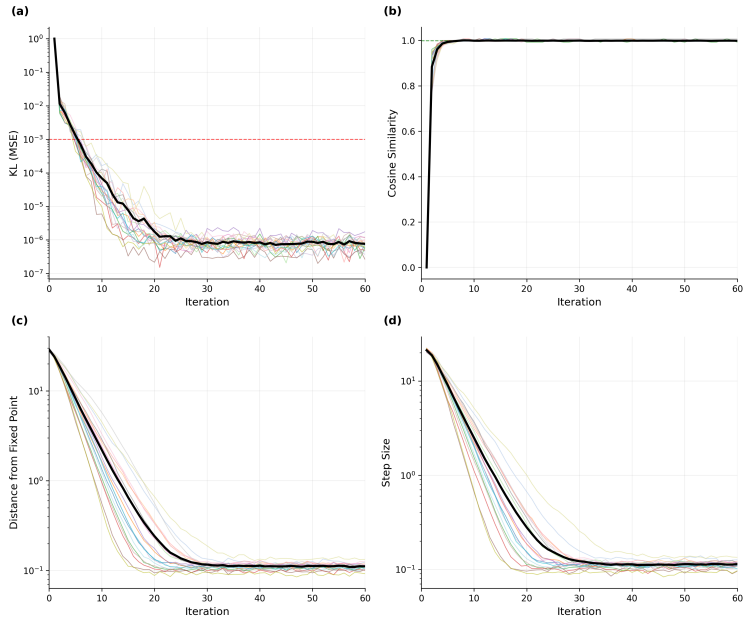
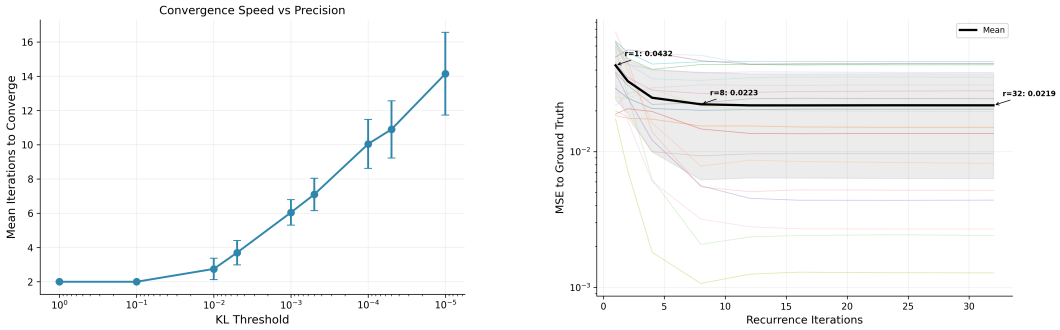


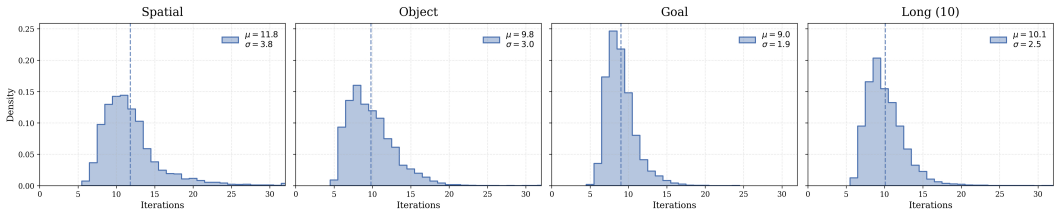
Figure 24: **Convergence diagnostics.** (a) Output change (KL divergence / MSE) between consecutive iterations drops below 10^{-5} by iteration $\sim$15 and plateaus near 10^{-6}. (b) Cosine similarity between consecutive outputs rapidly approaches 1.0. (c) Distance from the final fixed point decays exponentially with iteration count. (d) Step size (state change per iteration) decreases log-linearly, confirming deceleration toward the attractor. Together, these diagnostics confirm that the recurrent core behaves as a contractive dynamical system that reliably converges to a stable fixed point.



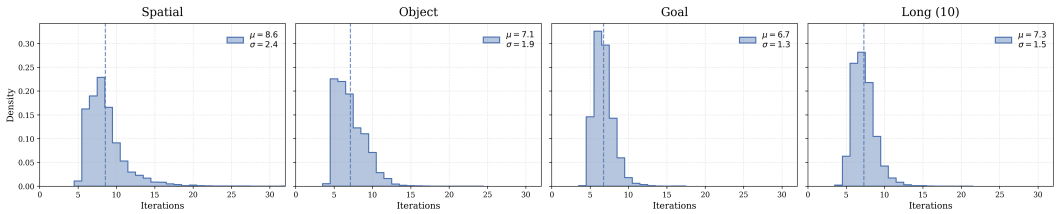
(a) (b)

Figure 25: **Test-time compute scaling.** (a) Compute-precision trade-off: reaching a tighter convergence threshold requires more iterations: $\sim$3 iterations suffice for KL < 0.01, $\sim$6 for KL $< 10^{-3}$, and $\sim$14 for KL $< 10^{-5}$. This curve provides practical guidance for choosing the iteration budget based on the desired convergence threshold and latency trade-off during deployment. (b) Mean MSE to ground-truth actions drops from 0.0432 at r=1 to 0.0219 at r=32, a 1.9$\times$ improvement. Individual samples (colored lines) show consistent gains, with the majority of improvement realized by r=8 iterations, indicating a log-linear scaling relationship between compute depth and action prediction quality.

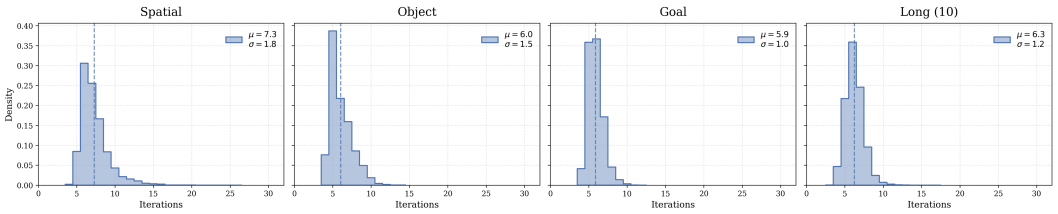
B.6 ADAPTIVE COMPUTATION TRADE-OFF



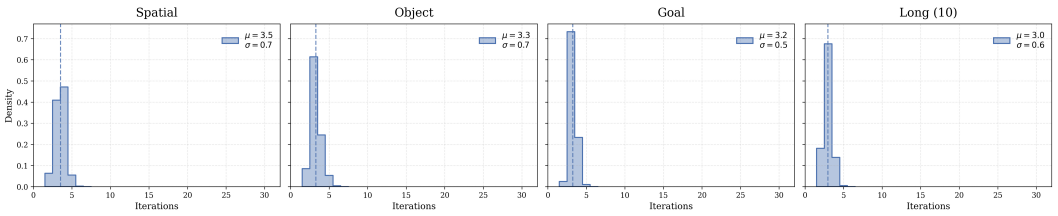
(a) Iteration distribution for $\tau = 1 \times 10^{-4}$



(b) Iteration distribution for $\tau = 5 \times 10^{-4}$



(c) Iteration distribution for $\tau = 1 \times 10^{-3}$



(d) Iteration distribution for $\tau = 1 \times 10^{-2}$

Figure 26: **Iteration distributions under varying KL-divergence thresholds.** Full suite of empirical convergence distributions across the LIBERO benchmark for $\tau \in \{10^{-4},\ 5{\times}10^{-4},\ 10^{-3},\ 10^{-2}\}$. As the threshold τ increases, the mean number of iterations k decreases, illustrating the trade-off between computational cost and reasoning depth. These distributions complement the Pareto curve in Figure 27 by showing the full shape of the iteration budget at each precision level.

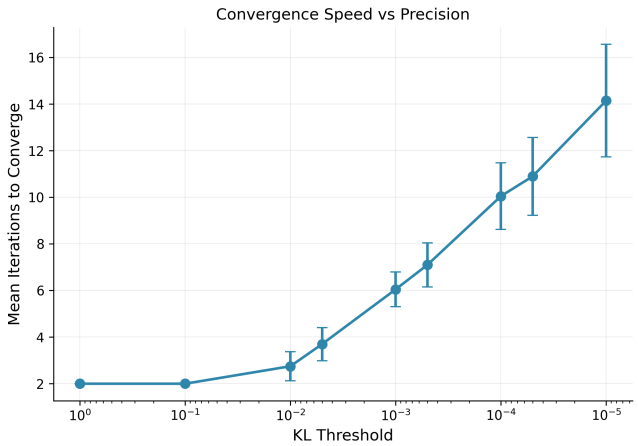


Figure 27: **Compute–precision trade-off.** Reaching a tighter convergence threshold requires more iterations: $\sim$3 iterations suffice for KL < 0.01, $\sim$6 for KL $< 10^{-3}$, and $\sim$14 for KL $< 10^{-5}$. This Pareto curve provides practical guidance for choosing the iteration budget based on the desired precision–latency trade-off during deployment.

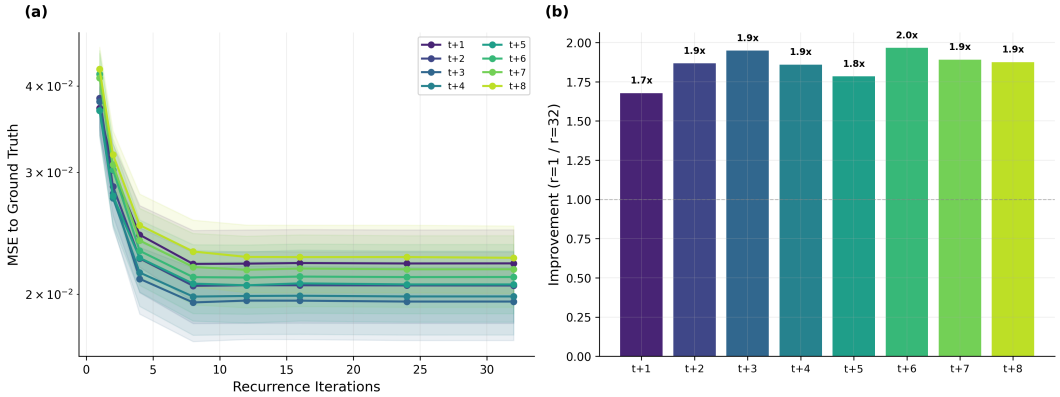


Figure 28: **Test-time scaling across the action chunk.** (a) MSE to ground truth decreases with additional iterations for all 8 positions in the predicted action chunk. (b) The improvement ratio (r=1 vs. r=32) is uniform across chunk positions ($\sim$1.7–2.0$\times$), indicating that the recurrence benefits near-term and far-term predictions equally rather than favoring any particular temporal horizon.

C Latent Dynamics Visualization

C.1 Latent Trajectory Analysis

To build qualitative intuition for the recurrent core's behavior, we visualize both the trajectories of latent states during iterative refinement (§C.1) and the spectral properties of the learned dynamical system (§**??**).

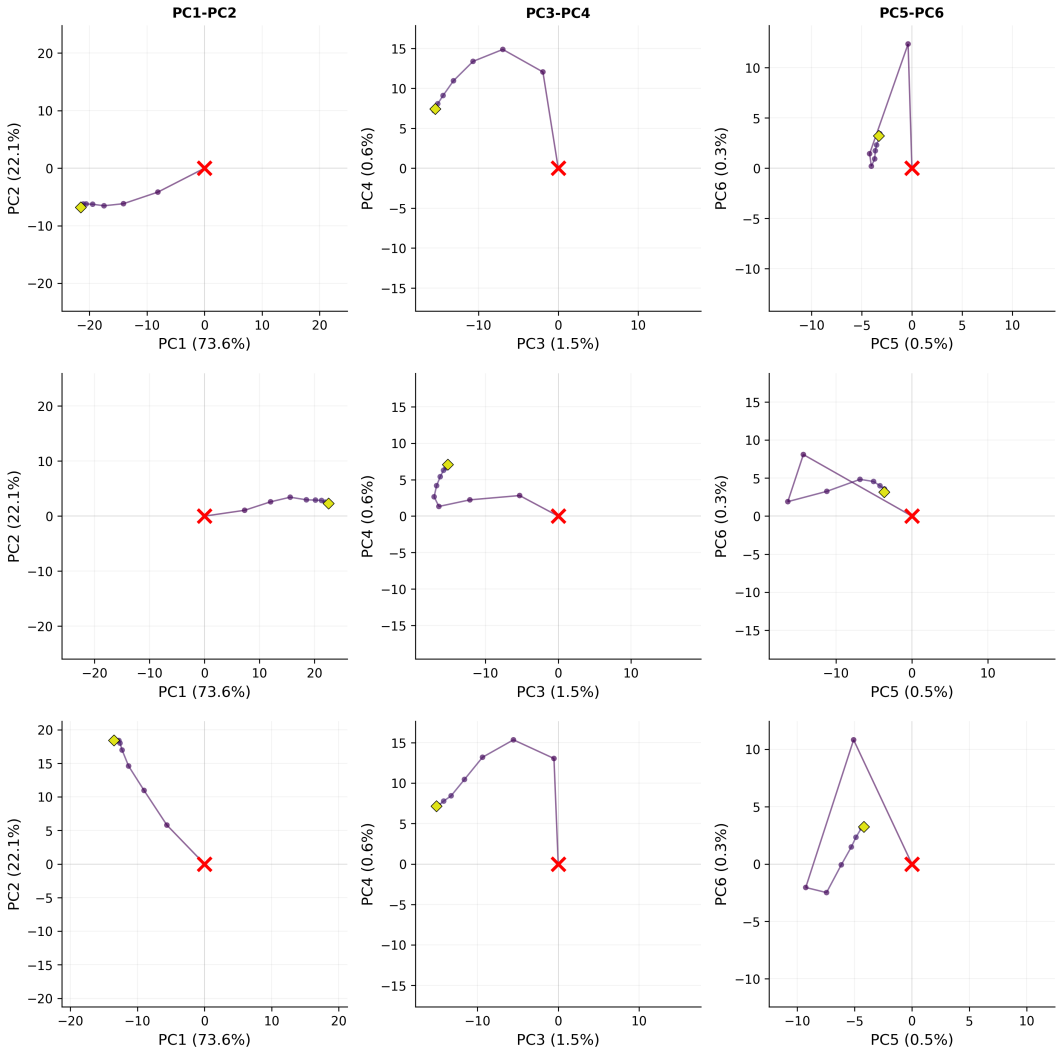


Figure 29: **Latent trajectory diversity across inputs (Sample Set 1).** Each row shows one sample's iterative refinement path projected onto principal component pairs (PC1–2, PC3–4, PC5–6). The red × marks the starting point (centered at the origin); the yellow ◇ indicates the converged fixed point. Despite diverse starting directions and arc lengths, all trajectories smoothly converge, demonstrating that the model adapts its "thinking path" to each input while reliably reaching a stable solution.

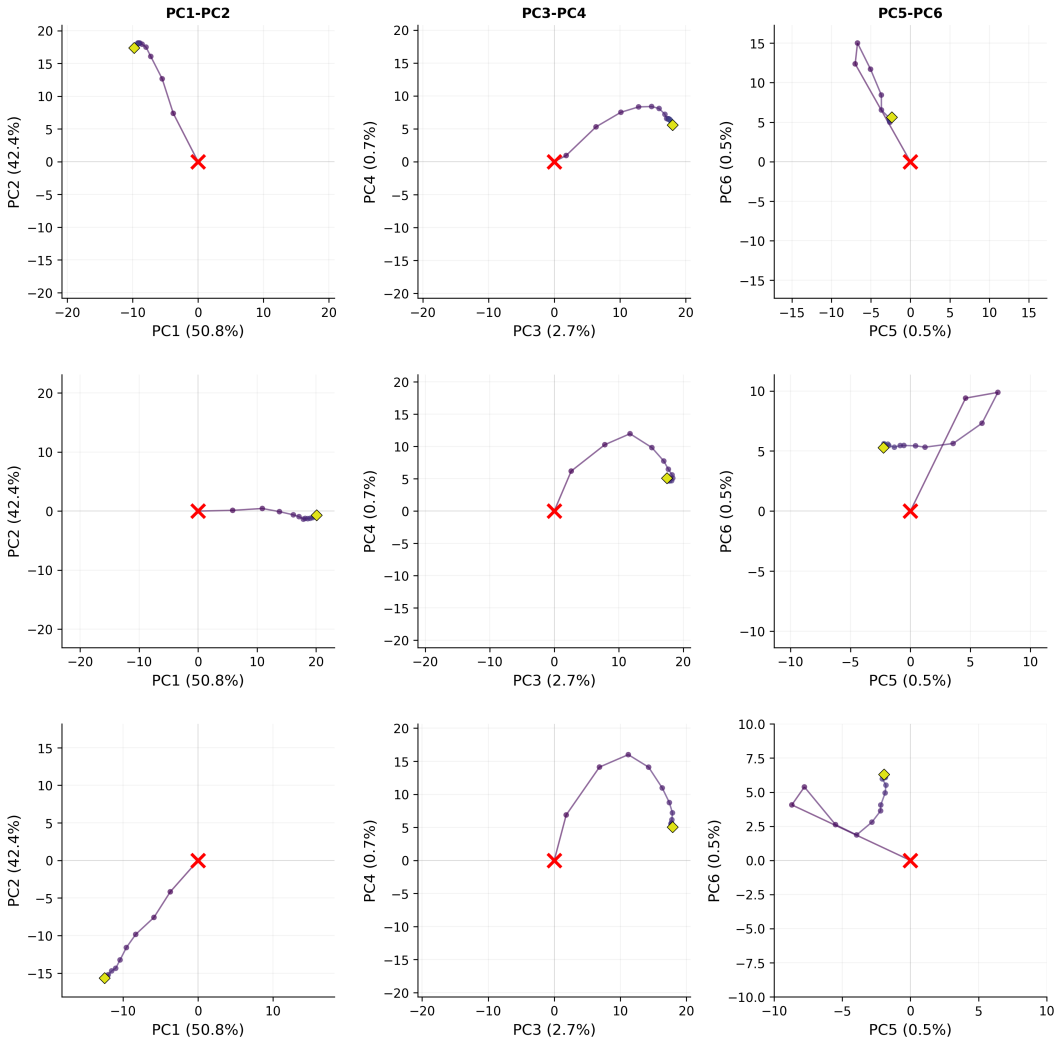


Figure 30: **Latent trajectory diversity across inputs (Sample Set 2).** Same visualization as Figure 29 for a second set of samples. The variation in trajectory curvature and length across PC subspaces further illustrates input-dependent computation: the recurrent core allocates different amounts of latent "travel" depending on the complexity of the observation.

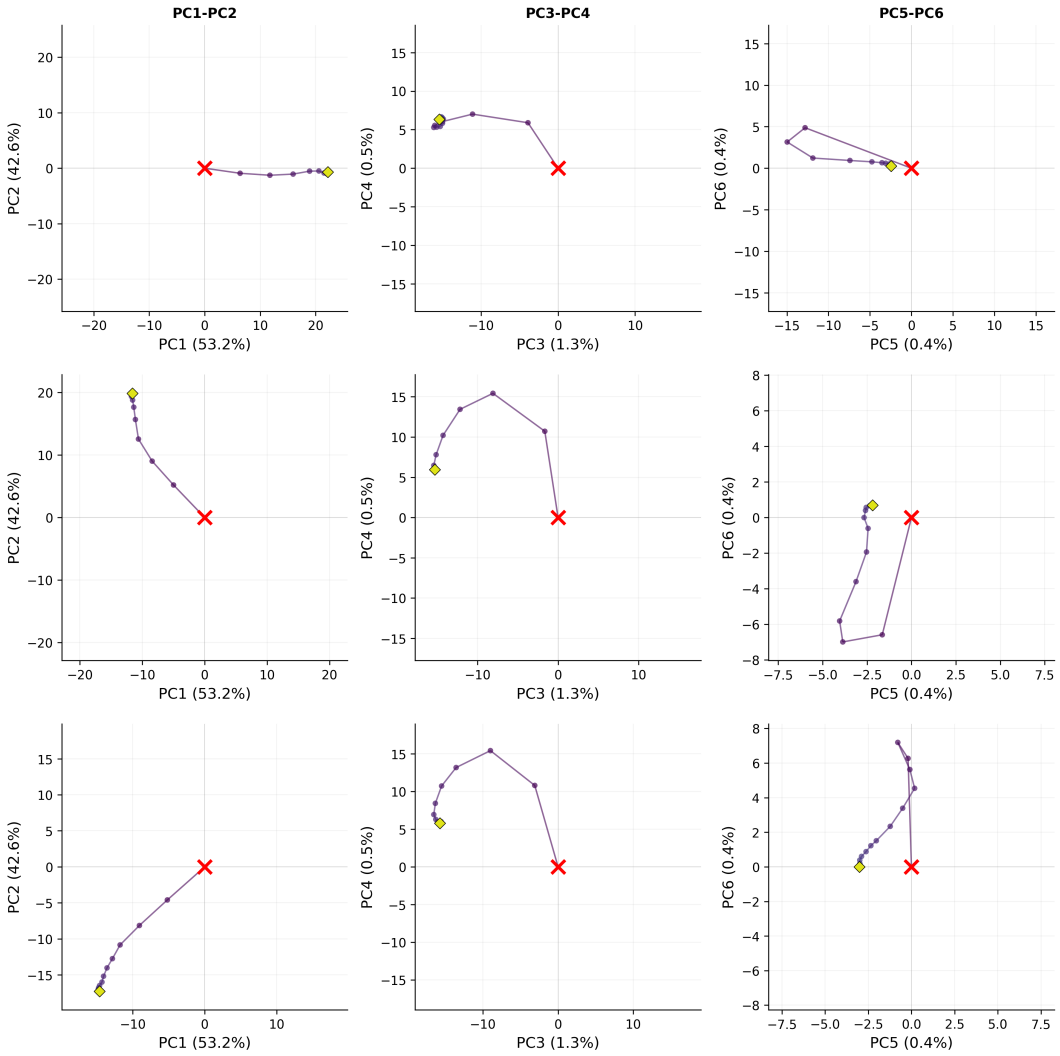


Figure 31: **Latent trajectory diversity across inputs (Sample Set 3).** Same visualization as Figures 29–30 for a third set of samples, confirming the generality of the convergence behavior across diverse observations from the LIBERO benchmark.

C.2 ATTENTION VISUALIZATIONS

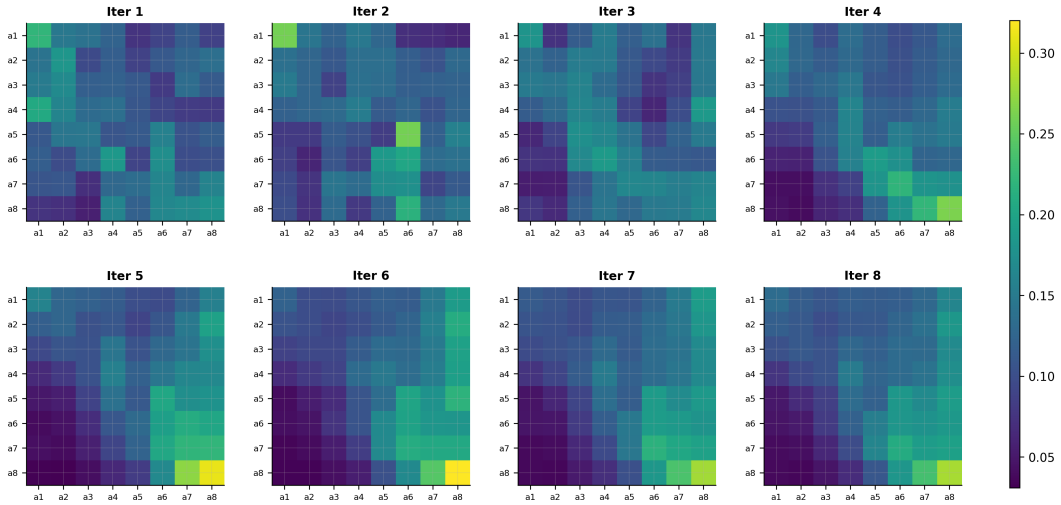


Figure 32: **Evolution of self-attention patterns across iterations (Recurrent Layer 1 - Head 5).** Attention weights of a single head visualized at successive recurrent iterations for a random sample. Early iterations exhibit diffused attention across the latent tokens; as the state converges, attention sharpens to a sparse, structured pattern suggesting that the model attends to specific latent tokens.

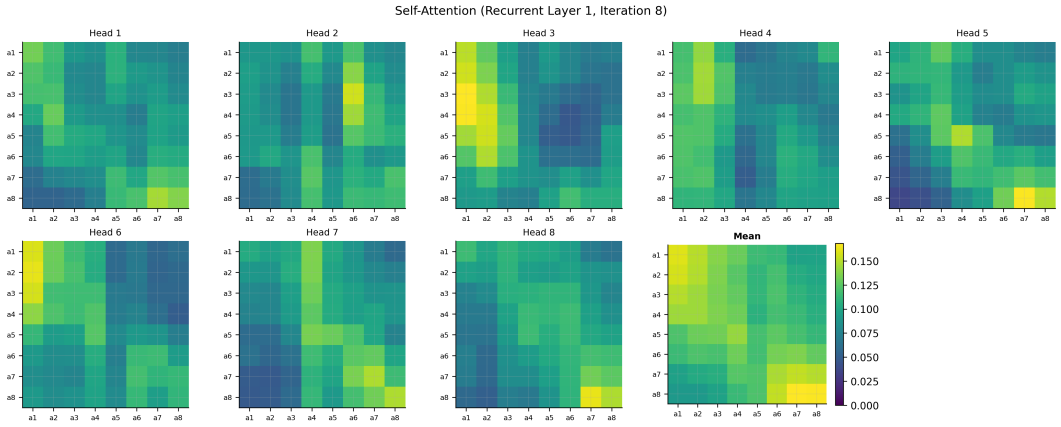


Figure 33: **Multi-head attention patterns at iteration** r=8**.** All self-attention heads visualized at a single mid-convergence iteration for a random sample. Different heads attend to distinct subsets of latent tokens, indicating functional specialization within the recurrent core: some heads attend broadly (contextual integration), while others focus on narrow token subsets (local refinement). The mean attention map exhibits a block-diagonal structure, indicating that the model learns to segment the action sequence into distinct temporal clusters. This suggests a strong locality bias, where actions primarily attend to neighboring tokens within the same sub-phase of the task, rather than attending globally across the entire sequence.

C.3 DISCUSSION AND LIMITATIONS

Our primary objective was to investigate latent iterative reasoning in robotic control, rather than to hyper-optimize a specific model for state-of-the-art dominance. While RD-VLA achieves competitive or superior performance on standard benchmarks such as LIBERO (Liu et al., 2023a), we emphasize that these results were obtained with minimal hyperparameter tuning and a relatively small backbone (0.5B parameters). We expect that scaling this architecture to larger backbones and training on more diverse datasets will yield significant performance gains.

A key limitation observed in our experiments is the boundary of depth generalization. While performance scales predictably with the number of recurrent steps up to some optimal number of iterations, extending recurrence beyond this number of iterations may lead to state saturation or performance degradation rather than continued refinement. Addressing this problem—perhaps through architectural innovations or specific training protocols remains an open challenge for scaling latent reasoning in robotics.

Recurrent architectures inherently expose internal state dynamics that can serve as proxies for model uncertainty. This offers a suite of test-time intervention capabilities, such as adaptive computation or uncertainty-aware execution. For instance, the system could autonomously halt execution or request operator assistance if the variance between recurrent states exceeds a safety threshold. While we demonstrated the viability of these mechanisms, the design space for such interventions is vast. We leave the specific implementation of such mechanisms to future work, focusing here on the architectural foundation that makes them possible.

While this work highlights the latency and memory limitations of token-based Chain-of-Thought (CoT) reasoning for high-frequency control, our Recurrent-Depth approach offers a complementary pathway. Future research could investigate hybrid approaches where recurrent depth is modulated per-token to enhance CoT reasoning capabilities in embodied agents.

C.4 Adaptive strategies

C.4.1 Threshold-Based Adaptive Execution

This method modulates the execution horizon using a binary reasoning threshold τ. We hypothesize that high iteration counts imply higher epistemic uncertainty. Consequently, if convergence requires $k^* > \tau$ steps, we truncate the horizon to a shorter duration H_{short} to mitigate compounding errors, while retaining H_{long} for confident predictions ($k^* \leq \tau$):

$$H_{exec} = \begin{cases} H_{long} & \text{if } k^* \leq \tau \\ H_{short} & \text{if } k^* > \tau \end{cases} \tag{6}$$

C.4.2 Linear Decay Execution

To provide a continuous scaling mechanism, we implement a linear decay schedule that reduces the execution horizon inversely to the reasoning depth. Given a base iteration budget τ_{base}, the horizon H_{exec} decreases by one step for every additional iteration required to converge:

$$H_{exec}(k^*) = \max\left(H_{min}, H_{max} - \max(0, k^* - \tau_{base})\right) \tag{7}$$

This approach forces the agent to replan more frequently as computational demand increases, favoring safety over efficiency in complex states.