ElegantVLA: Learning When to Think for Efficient Vision-Language-Action Models

Ye Li[1], **Huanan Liu**[1], **Kangye Ji**[1], **Yuan Meng**[1,*], **Jiajun Fan**[2], **Yuansong Wang**[1], **Shiyu Qin**[1], **Chenglei Wu**[1], **Shu-Tao Xia**[1], **Zhi Wang**[1*],
[1] Tsinghua University [2] University of Illinois at Urbana-Champaign

Abstract

Vision-Language-Action (VLA) models have emerged as a powerful paradigm for generalist robotic control. However, their high computational cost and limited control frequency hinder real-time robotic manipulation, especially when large vision-language backbones and iterative action heads are executed at every control step. Existing VLA acceleration methods often optimize individual components or rely on fixed acceleration rules, treating different control steps with largely fixed computation and overlooking the non-uniform reasoning demands of sequential embodied control. Inspired by human motor control, where cognitive and feedback resources concentrate on goal-sensitive stages, we argue that VLA models should learn when to invest full computation and when to reuse prior computation. To this end, we propose **ElegantVLA**, a plug-in phase-adaptive inference framework that accelerates VLA models through intra-model dynamic compute scheduling. ElegantVLA introduces a lightweight scheduler that observes temporal representation similarity, robot-motion cues, and episode progress to jointly allocate computation across the vision encoder, LLM, and action head. For perception-language reasoning, the scheduler selects a five-level Vision–LLM compute mode, from full recomputation to multi-step temporal reuse, based on visual-language representation stability. For action generation, it selects a three-level denoising mode, reusing intermediate denoising states during stable motion while preserving full refinement for goal-sensitive stages. By coordinating these decisions, ElegantVLA provides a general acceleration framework for modern VLA pipelines with explicit action-generation modules, without modifying or retraining the base model. Extensive experiments on GR00T, CogACT, and real-world tasks show that ElegantVLA preserves or improves task success while substantially accelerating inference. On GR00T, it achieves up to 2.55× average speedup. On CogACT, it delivers a 3.77× average speedup. In GR00T-based real-world experiments across six tasks, it reduces computation by 2.18× and increases control frequency from 13.8 Hz to 26.3 Hz. The project website is available at Project Page.

1 Introduction

Vision-Language-Action (VLA) models have emerged as a powerful paradigm for generalist robotic control by connecting visual perception, language understanding, and action generation in a unified policy [1–6]. Recent systems have shown impressive generalization across manipulation tasks and environments, but this capability comes with a substantial inference cost. Modern VLA pipelines typically combine a vision encoder, a large language model (LLM), and an action-generation module such as an autoregressive decoder, flow model, or diffusion head. Executing all these components

*Corresponding Authors: yuanmeng@tsinghua.edu.cn, wangzhi@sz.tsinghua.edu.cn.

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

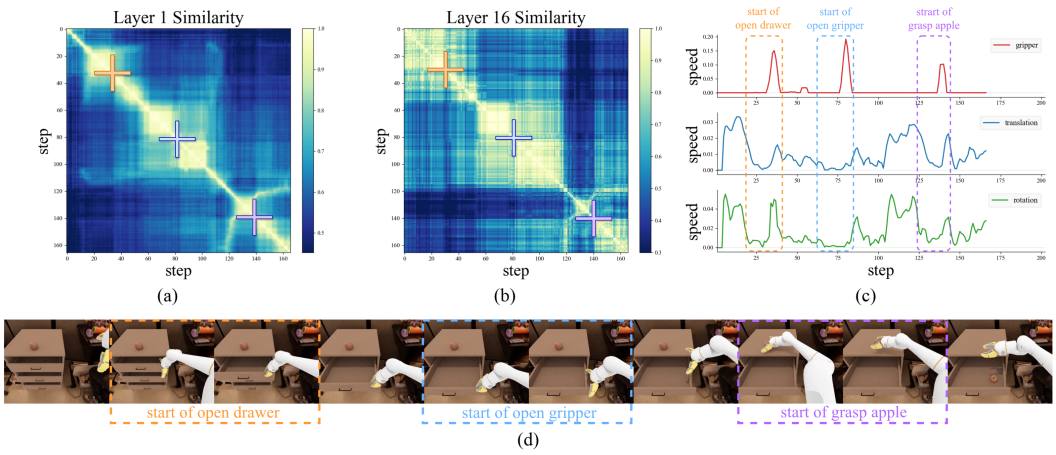

Figure 1: **Observation motivating phase-adaptive cache scheduling.** VLA execution exhibits temporally varying inference demand rather than uniform step-wise complexity. In this drawer-opening and apple-grasping rollout, (a,b) show pairwise CKA similarities of the first and final LLM-layer representations, (c) shows robot motion speeds, and (d) shows the rollout. Similar changes appear in both final-layer and first-layer similarities, suggesting that a lightweight first-layer probe can inform cache/recompute scheduling.

at every control step limits the control frequency of VLA policies, making real-time deployment difficult in latency-sensitive robotic manipulation.

Existing acceleration methods have made progress across model design, perception, action generation, and inference optimization [7, 8]. Some works reduce VLA computation through early exit, quantization, layer skipping, or adaptive computation routing [9–15]. Others exploit visual redundancy through token pruning, token caching, token fusion, or action-aware visual selection [16–22]. Recent work also accelerates action generation through diffusion-policy caching, parallel decoding, speculative decoding, and flow-specific inference design [23–26]. However, most existing methods still optimize a specific component, or rely on a fixed acceleration rule, while the whole VLA pipeline is repeatedly invoked under largely uniform computation across control steps. This overlooks a key property of robotic control: the computational demands of perception, reasoning, and action generation vary significantly across different task phases.

This non-uniformity is also natural from the perspective of human motor control. Human movements are not controlled with the same cognitive and feedback effort throughout an entire task. Stable transport phases can rely more on prediction and low-level feedback, while goal-sensitive phases such as contact, alignment, grasping, insertion, and placement require more careful correction and control [27–29]. VLA policies exhibit an analogous structure during sequential interaction. Adjacent control steps often share similar visual observations, semantic context, and action intent, whereas goal-sensitive stages may induce abrupt representation changes and require full effort.

Therefore, in this work, we propose a phase-adaptive VLA acceleration method to further reduce computational redundancy during inference. This design raises two core challenges. First, how can a VLA system estimate the per-step per-module computational demand without explicit phase labels? The computational demand of each control step is not explicitly given and changes continuously as the task evolves, making it difficult to decide when full inference is necessary. Second, how can computation be coordinated across the full VLA pipeline without degrading task success? Many existing methods make acceleration decisions at the level of individual modules, or combine several component-wise rules, without explicitly modeling how these decisions interact across the full perception-reasoning-action pipeline over time. As a result, locally aggressive reuse may save computation immediately, but can introduce temporal errors that propagate and accumulate during task execution. An effective method must therefore coordinate reuse decisions across modules while accounting for their long-term effects on control stability.

To address these challenges, we propose ElegantVLA, a plug-in phase-adaptive inference framework for efficient VLA models. First, we design multi-level computation strategies for the perception-

reasoning and action generation modules respectively. A lightweight scheduler then determines the computation strategy for each control step based on task-relevant temporal representation similarity and robot-motion cues. For perception-language reasoning, the scheduler selects a five-level Vision–LLM compute mode, from full recomputation to multi-step temporal reuse, based on visual-language representation stability. For action generation, ElegantVLA selects a three-level denoising mode, reusing intermediate denoising states during stable motion while preserving full refinement for precision-sensitive stages. In this way, ElegantVLA turns VLA acceleration into an intra-model dynamic compute scheduling problem, rather than a static compression or single-module pruning problem. Second, we introduce reinforcement learning (RL) to train the scheduler for globally optimal computation allocation. We formulate scheduling as a sequential decision problem, where each decision trades off task performance, computation cost, and reuse stability. The RL policy is conditioned on interpretable scheduling signals, including CKA-based temporal representation similarity, robot motion speeds, and episode progress, while its training objective encourages efficient reuse without allowing errors to accumulate across control steps. This enables ElegantVLA to coordinate recomputation and reuse across perception, reasoning, and action generation under one policy.

We validate ElegantVLA in simulation and real-world manipulation. On GR00T in SimplerEnv, ElegantVLA improves success rate from 64.00% to 65.88% while achieving up to 2.55× average FLOPs speedup. On CogACT, it reaches 3.72× and 3.77× average FLOPs speedups on Visual Matching and Variant Aggregation, respectively. In GR00T-based real-world experiments across six tasks, ElegantVLA reduces computation by 2.18× and increases control frequency from 13.8 Hz to 26.3 Hz, while improving average success from 61.67% to 65.00%. The improvement is visible on conveyor-belt pickup tasks, where faster control helps the policy react to moving targets and improves success on pineapple bun and toast. The main contributions are as follows:

(1) We formulate efficient VLA inference as a phase-adaptive intra-model compute allocation problem. Inspired by human motor control, we argue that computational effort should vary with task progress, semantic change, and robot state rather than remaining fixed across control steps.

(2) We propose ElegantVLA, a unified temporal reuse framework for VLA acceleration. ElegantVLA dynamically coordinates computation across the vision encoder, LLM, and action head through full computation, partial recomputation, and temporal reuse.

(3) We develop a scheduler learning framework tailored to phase-adaptive VLA inference. The scheduler learns stable recomputation and reuse decisions by balancing task performance, compute efficiency, and reuse stability.

2 Related Work

Vision Language Action Models. Representative systems such as OpenVLA, GROOT, π_0, and RDT have demonstrated strong generalization across manipulation settings by combining semantic perception, language-conditioned reasoning, and action generation [1, 2, 30, 4–6]. Recent VLA architectures increasingly separate perception-language reasoning from action generation: a vision encoder and LLM produce task-conditioned representations, while an action decoder, diffusion policy, or flow-based action head generates continuous control. This modular design improves the expressiveness and flexibility of VLA policies, but it also distributes inference cost across the full perception-reasoning-action pipeline. Thus, efficient VLA deployment requires understanding computation across perception, reasoning, and action generation, rather than accelerating a single backbone.

Efficient VLA Inference. Recent work has explored VLA acceleration through model-level compression and routing, perception-side token compression, and faster action generation [7–14, 16–19, 31, 32, 20–22, 33]. Another line targets diffusion-based action heads with block-wise caching, sparse denoising, parallel decoding, or flow-specific inference design [23, 24, 34, 25, 26]. These methods reveal substantial redundancy in VLA inference, but they often optimize a single module or rely on component-wise rules. More broadly, existing methods rarely model how computation demand evolves during execution or how cross-module reuse interacts over long horizons. Inspired by human motor control, where prediction, feedback correction, and precision-dependent control are allocated non-uniformly over time [27–29], ElegantVLA learns a phase-adaptive scheduler that coordinates recomputation and reuse across the vision encoder, LLM, and action head.

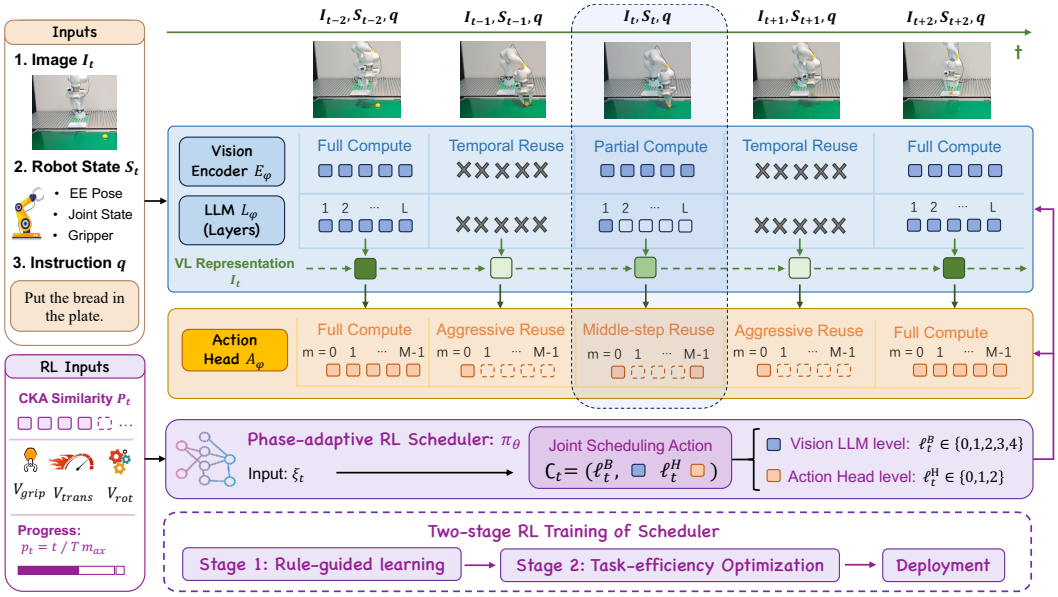

Figure 2: **Overview of ElegantVLA.** ElegantVLA learns when to think during VLA control by attaching a phase-adaptive scheduler to a frozen base policy. At each step, the scheduler uses temporal representation similarity, robot-motion signals, and episode progress to select a joint compute action for the Vision-LLM backbone and the action head. It allocates full recomputation to phase-sensitive moments and reuses cached computation during stable phases, enabling efficient inference without modifying or retraining the base VLA model.

3 Method

3.1 Phase-Adaptive Scheduling Framework

ElegantVLA formulates efficient VLA inference as a temporal compute allocation problem for embodied control. This view is motivated by human motor control, where prediction and feedback are allocated unevenly across movement phases rather than applied uniformly at every instant [27–29]. Analogously, a VLA policy need not spend the same amount of perception-language reasoning and action refinement at every control step. We instantiate this view as a scheduler-based inference wrapper for frozen VLA policies, operating on two branches: the *Vision-LLM backbone*, which contains the vision encoder and LLM, and the *action head*, which converts visual-language representations and robot states into executable actions. Given an image observation $\mathbf{I}_t$, robot state $\mathbf{s}_t$, and language instruction q, a standard VLA policy performs

$$\mathbf{z}_t = \mathcal{L}_\phi(\mathcal{E}_\phi(\mathbf{I}_t), q), \qquad \hat{\mathbf{a}}_{t:t+K-1} = \mathcal{H}_\phi(\mathbf{z}_t, \mathbf{s}_t), \tag{1}$$

where $\mathcal{E}_\phi$, $\mathcal{L}_\phi$, and $\mathcal{H}_\phi$ are kept fixed. ElegantVLA adds a lightweight scheduler π_θ that selects how to execute the Vision-LLM backbone and action head at each control step:

$$\mathbf{c}_t = (\ell_t^{\mathrm{B}}, \ell_t^{\mathrm{H}}) \sim \pi_\theta(\cdot \mid \boldsymbol{\xi}_t). \tag{2}$$

Here ℓ_t^{B} controls the computation level of the Vision-LLM backbone, ℓ_t^{H} controls how much action-head refinement is recomputed, and $\boldsymbol{\xi}_t$ is a compact observation built from temporal representation similarity and robot motion. The Vision-LLM decision is organized as a five-level compute ladder, ranging from full recomputation to multi-step temporal reuse. The action-head decision follows a three-level refinement-reuse ladder over iterative action generation: full refinement, middle-step reuse, and reuse after the first refinement step. Optimizing the action head separately enlarges the scheduling space, allowing ElegantVLA to preserve action precision when needed while still exploiting reuse for faster generation. Together, these decisions make VLA inference better aligned with movement regularities, enabling task-complexity-aware temporal variable-frequency acceleration under a target accuracy constraint. Figure 2 provides an overview of how ElegantVLA couples backbone-level recomputation and action-head reuse through a learned phase-adaptive scheduler.

3.2 Semantic-Stability-Guided Vision–LLM Scheduling

The Vision-LLM backbone dominates per-step inference cost because it repeatedly encodes visual tokens and executes all LLM layers at the control frequency. Following the motor-control view that prediction and feedback demands vary with movement phase [27–29], stable motion require less frequent VLM reasoning than precision-sensitive phases. ElegantVLA therefore allocates backbone computation in a phase-adaptive manner by exposing five ordered compute levels to the scheduler.

As suggested by Figure 1, temporal representation changes that appear in the final LLM layer are already visible after the first layer. This motivates using first-layer CKA as a lightweight probe for backbone reuse decisions. To estimate this stability, we use centered kernel alignment (CKA), which compares the similarity structure of hidden representations and is less tied to low-level pixel changes than raw input differences. We compute CKA after the first LLM layer: at this point, visual tokens have interacted with the language instruction and the representation has been shaped by the LLM prior, making it a lightweight but semantically meaningful probe. Specifically, ElegantVLA maintains an anchor hidden-state matrix $\mathbf{H}_\tau$ from a recent full computation and compares it with the current first-layer hidden-state matrix $\mathbf{H}_t$:

$$\rho_t = \mathrm{CKA}(\mathbf{H}_t, \mathbf{H}_\tau), \tag{3}$$

where a larger ρ_t indicates stronger temporal semantic stability. The Vision-LLM action space contains five computation levels, $\ell_t^{\mathrm{B}} \in \{0, 1, 2, 3, 4\}$, ordered by increasing reuse aggressiveness. When $\ell_t^{\mathrm{B}} = 0$, ElegantVLA computes both the vision encoder and the complete LLM, refreshing the anchor and output representation. When $\ell_t^{\mathrm{B}} = 1$, ElegantVLA refreshes the vision encoder but only recomputes the first and last LLM layers, while reusing intermediate LLM representations from the most recent full computation. This preserves input sensitivity and output alignment while avoiding the dominant cost of repeatedly executing all intermediate Transformer blocks. When $\ell_t^{\mathrm{B}} = j$ for $j \in \{2, 3, 4\}$, ElegantVLA skips both the vision encoder and the LLM for $j - 1$ consecutive control steps and directly reuses the most recent visual-language representation. ElegantVLA feeds ρ_t to the RL scheduler as one decision signal for selecting among these reuse levels.

Let $\bar{\mathbf{z}}$ denote the cached visual-language output representation from the most recent full or partial execution, and let $\bar{\mathbf{u}}$ denote the cached intermediate LLM states used by partial recomputation. At a high level, the representation sent to the action head is

$$\mathbf{z}_t = \begin{cases} \mathcal{L}_\phi(\mathcal{E}_\phi(\mathbf{I}_t), q), & \ell_t^{\mathrm{B}} = 0, \\ \mathcal{L}_\phi^{\mathrm{last}}(\bar{\mathbf{u}}, \mathcal{L}_\phi^{\mathrm{first}}(\mathcal{E}_\phi(\mathbf{I}_t), q)), & \ell_t^{\mathrm{B}} = 1, \\ \bar{\mathbf{z}}, & \ell_t^{\mathrm{B}} \in \{2, 3, 4\}, \end{cases} \tag{4}$$

This design aligns VLA inference with movement regularities and enables task-complexity-aware temporal variable-frequency acceleration while preserving optimal action accuracy.

3.3 Motion-Aware Action-Head Scheduling

Temporal redundancy in VLA inference appears at different levels. The Vision-LLM backbone mainly captures high-level semantic redundancy in visual-language contexts, while the action head exhibits low-level redundancy in the iterative generation of continuous actions. These two forms of redundancy are related but not identical: a stable semantic context may still require fresh action refinement due to changes in robot state or contact, whereas a refreshed visual-language representation may still lead to a smooth action-generation trajectory. ElegantVLA therefore schedules the action head as an independent branch, decoupling semantic reuse from action-generation precision. Since action-refinement redundancy is tied to low-level motion continuity, ElegantVLA exposes robot motion signals, including gripper speed, end-effector translation speed, and end-effector rotation speed, to the scheduler. These signals help distinguish smooth transport phases, where refinement updates are more reusable, from precision-sensitive phases, where fresh action refinement is needed. For an iterative action head with M refinement steps,

$$\mathbf{x}_t^{m+1} = \mathcal{F}_\phi^m(\mathbf{x}_t^m, \mathbf{z}_t, \mathbf{s}_t), \qquad m = 0, \ldots, M-1, \tag{5}$$

where $\mathcal{F}_\phi^m$ denotes the m-th refinement step inside the action head $\mathcal{H}_\phi$, and $\mathbf{x}_t^m$ is the intermediate action state.

ElegantVLA reuses cached refinement updates only on selected steps, allowing the scheduler to reduce action-generation cost without forcing the entire action head to be skipped. The action-head

Table 1: **Simulation results on CogACT in SimplerEnv.** Each cell reports success rate (%, ↑) / speedup (↑). ElegantVLA achieves 3.72× and 3.77× average speedup on Visual Matching and Variant Aggregation, respectively, while also obtaining the best average task success.

SIMPLER	Method	Success Rate (%, ↑) / Speedup (↑)				AVG
		PickCan	MoveNear	Drawer	PutDrawer	
Visual Matching	CogACT [5]	91.30 / 1.00	85.00 / 1.00	71.80 / 1.00	50.90 / 1.00	74.80 / 1.00
	Random Dropping	9.70 / 1.20	20.40 / 1.20	53.50 / 1.20	0.00 / 1.20	20.90 / 1.20
	FastV [35]	82.67 / 1.31	72.92 / 1.31	66.67 / 1.28	46.30 / 1.30	67.14 / 1.30
	VLA-Cache [16]	87.33 / 1.47	77.92 / 1.46	69.44 / 1.44	47.22 / 1.47	70.48 / 1.46
	MoLe-VLA [36]	86.40 / 1.53	80.20 / 1.53	70.60 / 1.53	50.40 / 1.53	71.90 / 1.53
	ElegantVLA	**89.67 / 3.32**	**80.42 / 3.05**	**89.35 / 4.10**	**50.93 / 4.41**	**77.59 / 3.72**
Variant Aggregation	CogACT [5]	89.60 / 1.00	80.80 / 1.00	28.30 / 1.00	46.60 / 1.00	61.30 / 1.00
	Random Dropping	4.00 / 1.20	16.10 / 1.20	15.60 / 1.20	0.00 / 1.20	8.90 / 1.20
	FastV [35]	76.00 / 1.29	71.67 / 1.29	24.34 / 1.26	47.62 / 1.26	54.91 / 1.28
	VLA-Cache [16]	80.00 / 1.47	75.00 / 1.45	30.69 / 1.43	43.39 / 1.43	57.27 / 1.45
	MoLe-VLA [36]	**89.20** / 1.49	**79.50** / 1.49	29.90 / 1.49	46.20 / 1.49	61.20 / 1.49
	ElegantVLA	82.30 / **3.32**	77.17 / **3.26**	**73.54 / 4.34**	**57.14 / 4.15**	**72.54 / 3.77**

Table 2: **Simulation results on GR00T in SimplerEnv.** Each cell reports success rate (%, ↑) / speedup (↑). ElegantVLA improves the overall success rate from 64.00% to 65.88%, with 2.35× and 2.55× average speedup on Google Robot and WidowX, respectively.

SIMPLER	Method	Success Rate (%, ↑) / Speedup (↑)						AVG
		CloseD	MoveNear	OpenD	PickCan	PickObj	PlaceD	
Google Robot	GR00T [30]	82.00 / 1.00	92.00 / 1.00	49.50 / 1.00	99.50 / 1.00	92.50 / 1.00	11.00 / 1.00	71.08 / 1.00
	FastV [35]	80.00 / 1.13	88.00 / 1.13	64.00 / 1.13	94.00 / 1.13	88.00 / 1.13	12.00 / 1.13	71.00 / 1.13
	VLA-Cache [16]	72.00 / 1.13	90.00 / 1.13	62.00 / 1.13	94.00 / 1.13	84.00 / 1.13	10.00 / 1.13	68.67 / 1.13
	ElegantVLA	**81.00 / 2.39**	**95.50 / 1.90**	**74.50 / 2.31**	**98.00 / 2.35**	**88.50 / 2.59**	**12.50 / 2.39**	**75.00 / 2.35**

SIMPLER	Method	Success Rate (%, ↑) / Speedup (↑)							AVG
		Carrot	CloseD	OpenD	Basket	Sink	Spoon	Stack	
WidowX	GR00T [30]	60.50 / 1.00	98.50 / 1.00	93.00 / 1.00	37.50 / 1.00	37.50 / 1.00	71.50 / 1.00	7.00 / 1.00	57.93 / 1.00
	FastV [35]	58.00 / 1.13	98.00 / 1.13	84.00 / 1.13	22.00 / 1.13	30.00 / 1.13	**74.00** / 1.13	12.00 / 1.13	54.00 / 1.13
	VLA-Cache [16]	**72.00** / 1.13	98.00 / 1.13	84.00 / 1.13	24.00 / 1.13	24.00 / 1.13	72.00 / 1.13	12.00 / 1.13	53.14 / 1.13
	ElegantVLA	63.50 / **2.58**	**99.50 / 2.54**	**85.00 / 2.56**	**31.50 / 2.52**	**41.00 / 2.57**	**72.50 / 2.55**	**13.50 / 2.55**	**58.07 / 2.55**

action space contains three refinement levels, $\ell_t^{\mathrm{H}} \in \{0, 1, 2\}$. For each level, we define $\mathcal{R}(\ell_t^{\mathrm{H}})$ as the set of refinement steps whose updates are reused from the cache:

$$\mathcal{R}(\ell_t^{\mathrm{H}}) = \begin{cases} \varnothing, & \ell_t^{\mathrm{H}} = 0, \\ \{1, \ldots, M-2\}, & \ell_t^{\mathrm{H}} = 1, \\ \{1, \ldots, M-1\}, & \ell_t^{\mathrm{H}} = 2, \end{cases} \tag{6}$$

Thus, $\ell_t^{\mathrm{H}} = 0$ recomputes all steps, $\ell_t^{\mathrm{H}} = 1$ reuses only middle steps, and $\ell_t^{\mathrm{H}} = 2$ reuses all steps after the first refinement. The scheduled action-head update is

$$\mathbf{x}_t^{m+1} = \begin{cases} \mathcal{F}_\phi^m(\mathbf{x}_t^m, \mathbf{z}_t, \mathbf{s}_t), & m \notin \mathcal{R}(\ell_t^{\mathrm{H}}), \\ \mathbf{x}_t^m + \bar{\boldsymbol{\Delta}}^m, & m \in \mathcal{R}(\ell_t^{\mathrm{H}}), \end{cases} \tag{7}$$

where $\bar{\boldsymbol{\Delta}}^m$ is the cached update from a recent execution.

When $\ell_t^{\mathrm{H}} = 0$, ElegantVLA recomputes all refinement steps and updates the cache. When $\ell_t^{\mathrm{H}} = 1$, ElegantVLA recomputes the boundary refinement steps and reuses the middle updates, preserving input and output adaptation while reducing intermediate computation. When $\ell_t^{\mathrm{H}} = 2$, ElegantVLA recomputes only the first refinement step and reuses the remaining updates, providing the most aggressive acceleration for smooth action trajectories. This factorized design enlarges the scheduling

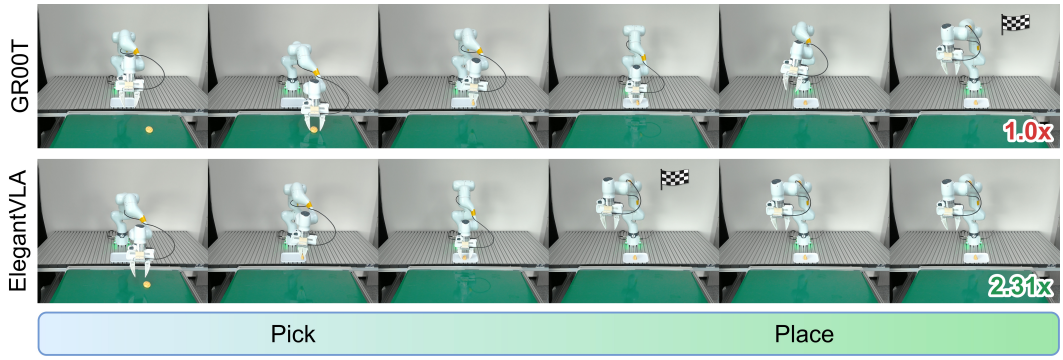

Figure 3: **Qualitative visualization of faster real-world execution.** On real-world pineapple-bun pickup, ElegantVLA preserves the pick-and-place phases while completing the task 2.31× faster than full-computation GR00T.

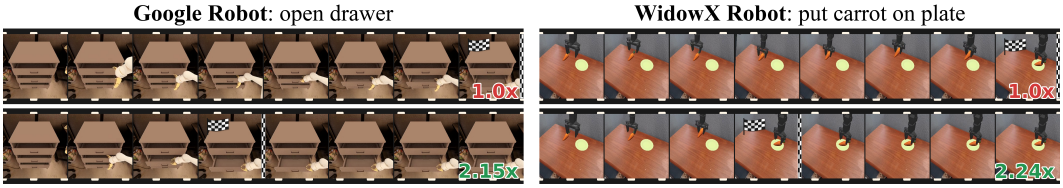

Figure 4: **Qualitative visualization of faster simulated execution.** On Google Robot drawer opening and WidowX carrot placement, ElegantVLA completes the rollouts faster while preserving the key manipulation phases.

space: ElegantVLA can refresh high-level visual-language reasoning while reusing low-level action refinement, or reuse the Vision-LLM backbone while recomputing the action head when precise control is needed.

3.4 Two-Stage Joint RL Scheduler Design

The scheduler serves as the perception-aware compute allocator in ElegantVLA. Given the current semantic stability and robot motion state, it learns to choose how much to think at each control step, mirroring the human-inspired principle of allocating effort according to movement phase. This is a sequential decision problem: each action $\mathbf{c}_t = (\ell_t^{\mathrm{B}}, \ell_t^{\mathrm{H}})$ jointly selects a Vision-LLM compute level and an action-head refinement level, while aggressive reuse may affect future control accuracy through error accumulation. The scheduler observation is

$$\boldsymbol{\xi}_t = [\rho_t,\ v_t^{\mathrm{grip}},\ v_t^{\mathrm{trans}},\ v_t^{\mathrm{rot}},\ p_t]^\top, \tag{8}$$

where ρ_t is the temporal representation similarity, the next three terms are gripper speed, end-effector translation speed, and end-effector rotation speed, and $p_t = t/T_{\max}$ is the normalized episode progress. These signals jointly describe high-level semantic stability, low-level motion continuity, and coarse task progress.

We train π_θ with Maskable PPO while keeping the base VLA frozen. Action masking is important because valid Vision-LLM choices are state-dependent: after a multi-step reuse decision, the backbone level is fixed within the induced reuse window, while the action head still requires a decision at every control step.

Directly optimizing the scheduler from terminal accuracy and FLOPs is difficult because an episode contains hundreds of scheduling decisions but only sparse end-of-episode feedback. We therefore use a two-stage training procedure. The first stage uses a rule-guided teacher to provide dense shaping rewards:

$$r_t^{(1)} = r_t^{\mathrm{succ}} - \lambda_C r_t^{\mathrm{FLOPs}} - \lambda_{\mathrm{B}} d(\ell_t^{\mathrm{B}}, \tilde{\ell}_t^{\mathrm{B}}) - \lambda_{\mathrm{H}} d(\ell_t^{\mathrm{H}}, \tilde{\ell}_t^{\mathrm{H}}) - \lambda_R k(\ell_t^{\mathrm{B}}), \tag{9}$$

where r_t^{FLOPs} measures the computation cost of the selected execution path, $\tilde{\ell}_t^{\mathrm{B}}$ and $\tilde{\ell}_t^{\mathrm{H}}$ are rule-guided computation levels, $d(\cdot,\cdot)$ penalizes disagreement with the teacher when multiple valid

Table 3: **Real-world results on Franka Research 3.** Each cell reports success rate (%, ↑) / speedup (↑). Results are reported over 10 trials per task with the full-computation GR00T policy as the baseline.

Setting	Method	Success Rate (%, ↑) / Speedup (↑)						AVG
		Phone Stand	Pen Holder	Stack Bowls	Pineapple Bun	Toast	Chocolate	
Real Robot	GR00T [30]	60.00 / 1.00	70.00 / 1.00	40.00 / 1.00	70.00 / 1.00	70.00 / 1.00	60.00 / 1.00	61.67 / 1.00
	ElegantVLA	60.00 / **2.18**	70.00 / **2.24**	40.00 / **2.01**	**80.00 / 2.31**	**80.00 / 2.21**	60.00 / **2.11**	**65.00 / 2.18**

Table 4: **Latency and frequency on GR00T in SimplerEnv.** Each cell reports frequency (Hz, ↑) / latency (ms, ↓), measured on an RTX 4090. ElegantVLA more than doubles the average control frequency while reducing per-step latency by over half on both robot suites.

SIMPLER	Method	Frequency (Hz, ↑) / Latency (ms, ↓)						AVG
		CloseD	MoveNear	OpenD	PickCan	PickObj	PlaceD	
Google Robot	GR00T [30]	15.99 / 62.56	16.17 / 61.83	17.62 / 56.75	15.88 / 62.96	16.90 / 59.18	17.46 / 57.28	16.64 / 60.09
	ElegantVLA	**36.36 / 27.50**	**27.19 / 36.78**	**35.86 / 27.88**	**36.20 / 27.63**	**38.20 / 26.18**	**36.36 / 27.50**	**35.03 / 28.55**

SIMPLER	Method	Frequency (Hz, ↑) / Latency (ms, ↓)							AVG
		Carrot	CloseD	OpenD	Basket	Sink	Spoon	Stack	
WidowX	GR00T [30]	17.26 / 57.94	17.25 / 57.98	17.18 / 58.21	15.72 / 63.60	17.07 / 58.57	17.48 / 57.21	15.16 / 65.96	16.69 / 59.92
	ElegantVLA	**37.36 / 26.76**	**37.86 / 26.41**	**37.20 / 26.88**	**37.70 / 26.53**	**38.20 / 26.18**	**37.03 / 27.00**	**38.03 / 26.29**	**37.70 / 26.53**

choices exist, and $k(\ell_t^{\mathrm{B}})$ denotes the temporal reuse horizon introduced by the chosen Vision-LLM level. This stage gives the scheduler a stable initialization and prevents destructive exploration. Stage two removes teacher shaping terms and optimizes task-efficiency feedback, allowing the policy to move beyond hand-designed rules while retaining stable reuse. At deployment, the teacher and rewards are removed; only the learned scheduler is attached to the frozen VLA policy.

4 Experiments

4.1 Main Results

Simulation results. Table 1 shows that ElegantVLA achieves the best average success-speedup trade-off on CogACT. On Visual Matching and Variant Aggregation, ElegantVLA reaches 77.59% and 72.54% average success with 3.72× and 3.77× speedup, respectively, outperforming existing acceleration baselines that provide only 1.20×–1.53× speedup. The gains are especially clear on drawer-style tasks, where adaptive recomputation helps preserve control accuracy during contact-sensitive phases. Table 2 further verifies the same trend on GR00T, where ElegantVLA improves the overall success rate from 64.00% to 65.88% while achieving up to 2.55× average FLOPs speedup; Appendix B gives the simulation task protocol.

Real-world results. We evaluate ElegantVLA on six real-robot tasks: phone placement, pen insertion, bowl stacking, and conveyor-belt pickup of pineapple buns, toast, and chocolate buns. Our real-world platform uses an NVIDIA RTX 4090 GPU and Huawei Ascend hardware. Table 3 shows that ElegantVLA improves average success from 61.67% to 65.00% with a 2.18× speedup, preserving its simulation efficiency gains. The gains are most apparent on conveyor-belt pickup tasks, suggesting faster responses to moving targets; Appendix A provides task descriptions and detailed results.

4.2 Analysis

Frequency and Latency. Table 4 shows that the computation reduction of ElegantVLA translates into practical wall-clock gains. On Google Robot, ElegantVLA increases the average control frequency from 16.64 Hz to 35.03 Hz and reduces latency from 60.09 ms to 28.55 ms. On WidowX, it improves the average frequency from 16.69 Hz to 37.70 Hz and reduces latency from 59.92 ms to

Table 5: **RL training ablation on CogACT in SimplerEnv.** Each cell reports success rate (%, ↑) / speedup (↑). Stage2 improves the average success rate from 70.45% to 77.59% on Visual Matching and from 69.03% to 72.54% on Visual Aggregation while also increasing the average speedup.

SIMPLER	Method	Success Rate (%, ↑) / Speedup (↑)				AVG
		PickCan	MoveNear	Drawer	PutDrawer	
Visual Matching	CogACT [5]	94.33 / 1.00	83.75 / 1.00	74.04 / 1.00	40.74 / 1.00	73.22 / 1.00
	Stage1	88.00 / 3.07	72.50 / **3.12**	88.89 / 3.10	32.41 / 4.40	70.45 / 3.42
	Stage2	**89.67** / **3.32**	**80.42** / 3.05	**89.35** / **4.10**	**50.93** / **4.41**	**77.59** / **3.72**
Visual Aggregation	CogACT [5]	88.97 / 1.00	77.00 / 1.00	27.51 / 1.00	55.03 / 1.00	62.13 / 1.00
	Stage1	81.09 / 3.08	70.67 / 3.12	70.90 / 3.12	53.44 / 3.90	69.03 / 3.31
	Stage2	**82.30** / **3.32**	**77.17** / **3.26**	**73.54** / **4.34**	**57.14** / **4.15**	**72.54** / **3.77**

Table 6: **Module scheduling ablation on GR00T Google Robot tasks.** Each cell reports success rate (%, ↑) / speedup (↑). Jointly scheduling both branches achieves the best average success and substantially larger speedup than keeping either branch at full computation.

SIMPLER	Method	Success Rate (%, ↑) / Speedup (↑)						AVG
		CloseD	MoveNear	OpenD	PickCan	PickObj	PlaceD	
Google Robot	GR00T [30]	82.00 / 1.00	92.00 / 1.00	49.50 / 1.00	99.50 / 1.00	92.50 / 1.00	11.00 / 1.00	71.08 / 1.00
	Force LLM Full	73.50 / 1.47	90.00 / 1.47	47.50 / 1.47	96.00 / 1.46	88.00 / 1.47	**14.00** / 1.45	68.17 / 1.46
	Force AH Full	**86.00** / 1.56	87.00 / 1.56	70.50 / 1.56	**98.00** / 1.55	87.00 / 1.56	8.00 / 1.56	72.75 / 1.56
	ElegantVLA	81.00 / **2.39**	**95.50** / **1.90**	**74.50** / **2.31**	**98.00** / **2.35**	**88.50** / **2.59**	12.50 / **2.39**	**75.00** / **2.35**

26.53 ms. The gains are consistent across tasks, while the lower frequency on MoveNear suggests that the scheduler remains conservative when continuous spatial reasoning is needed.

RL training. Table 5 validates the role of the two-stage scheduler training. Compared with Stage1, Stage2 improves the average success rate from 70.45% to 77.59% on Visual Matching and from 69.03% to 72.54% on Visual Aggregation, while also increasing the average speedup. This indicates that RL learns a better task-efficiency trade-off after rule-guided initialization, rather than simply applying more aggressive reuse; Appendix C.1 ablates threshold-only scheduling, Stage1 training, and two-stage training, further highlighting the effectiveness of ElegantVLA.

Module scheduling. Table 6 studies whether the Vision-LLM backbone and action head should be scheduled jointly. Keeping either branch at full computation limits the benefit: Force LLM Full obtains 68.17% success with 1.46× speedup, and Force AH Full obtains 72.75% success with 1.56× speedup. The full ElegantVLA reaches 75.00% success with 2.35× speedup, showing that semantic reasoning and action refinement contain complementary temporal redundancy; Appendix C.2 ablates CKA- and speed-based scheduler inputs, confirming their complementarity.

Qualitative execution behavior. Figures 3 and 4 show that ElegantVLA finishes representative rollouts faster while preserving key manipulation phases. The visualized speedups, 2.31× on real pineapple-bun pickup and about 2.2× in simulation, suggest that the scheduler removes redundant inference during stable motion without skipping precision-sensitive transitions such as approach, grasp/contact, and placement. Additional rollouts are provided in Appendices D.1 and D.2.

5 Conclusion

We presented ElegantVLA, a phase-adaptive inference framework that learns when to recompute and when to reuse computation in frozen VLA policies. By jointly scheduling the Vision-LLM backbone and the action head, ElegantVLA allocates more computation to precision-sensitive phases and avoids redundant inference during stable motion. Across CogACT, GR00T, and real-world manipulation, ElegantVLA achieves substantial speedups, up to 3.77× in simulation and 2.18× on the real robot, while preserving competitive task success. These results suggest that efficient embodied policies should not only reduce per-step cost, but also adapt computation to the temporal demands of interaction.

Acknowledgments and Disclosure of Funding

This work is supported in part by the National Natural Science Foundation of China under Grants 92467204, 62472249, 62571298, and 62402264, the Shenzhen Science and Technology Program under Grant KJZD20240903102300001, and the Tsinghua SIGS KA Cooperation Fund.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

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

[7] Weifan Guan, Qinghao Hu, Aosheng Li, and Jian Cheng. Efficient vision-language-action models for embodied manipulation: A systematic survey. *arXiv preprint arXiv:2510.17111*, 2025.

[8] Zhaoshu Yu, Bo Wang, Pengpeng Zeng, Haonan Zhang, Ji Zhang, Zheng Wang, Lianli Gao, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on efficient vision-language-action models. *arXiv preprint arXiv:2510.24795*, 2025.

[9] 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, 2024.

[10] Seongmin Park, Hyungmin Kim, Wonseok Jeon, Juyoung Yang, Byeongwook Jeon, Yoonseon Oh, and Jungwook Choi. Quantization-aware imitation-learning for resource-efficient robotic control. *arXiv preprint arXiv:2412.01034*, 2024.

[11] Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. *arXiv preprint arXiv:2506.10100*, 2025.

[12] Zebin Yang, Yijiahao Qi, Tong Xie, Bo Yu, Shaoshan Liu, and Meng Li. Dysl-vla: Efficient vision-language-action model inference via dynamic-static layer-skipping for robot manipulation. *arXiv preprint arXiv:2602.22896*, 2026.

[13] Zihao Zheng, Hangyu Cao, Sicheng Tian, Jiayu Chen, Maoliang Li, Xinhao Sun, Hailong Zou, Zhaobo Zhang, Xuanzhe Liu, Donggang Cao, et al. Dyq-vla: Temporal-dynamic-aware quantization for embodied vision-language-action models. *arXiv preprint arXiv:2603.07904*, 2026.

[14] Wenda Yu, Tianshi Wang, Fengling Li, Jingjing Li, and Lei Zhu. Acˆ 2-vla: Action-context-aware adaptive computation in vision-language-action models for efficient robotic manipulation. *arXiv preprint arXiv:2601.19634*, 2026.

[15] Ye Li, Chen Tang, Yuan Meng, Jiajun Fan, Zenghao Chai, Xinzhu Ma, Zhi Wang, and Wenwu Zhu. Prance: Joint token-optimization and structural channel-pruning for adaptive vit inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[16] 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 e-prints*, pages arXiv–2502, 2025.

[17] Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shutao Xia, Zhi Wang, and Wenwu Zhu. Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration. *arXiv preprint arXiv:2506.12723*, 2025.

[18] Xudong Tan, Yaoxin Yang, Peng Ye, Jialin Zheng, Bizhe Bai, Xinyi Wang, Jia Hao, and Tao Chen. Think twice, act once: Token-aware compression and action reuse for efficient inference in vision-language-action models. *arXiv preprint arXiv:2505.21200*, 2025.

[19] Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv preprint arXiv:2509.05614*, 2025.

[20] Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. *arXiv preprint arXiv:2509.22093*, 2025.

[21] Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Vla-pruner: Temporal-aware dual-level visual token pruning for efficient vision-language-action inference. *arXiv preprint arXiv:2511.16449*, 2025.

[22] Titong Jiang, Xuefeng Jiang, Yuan Ma, Xin Wen, Bailin Li, Kun Zhan, Peng Jia, Yahui Liu, Sheng Sun, and Xianpeng Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*, 2025.

[23] Ye Li, Jiahe Feng, Yuan Meng, Kangye Ji, Chen Tang, Xinwan Wen, Shutao Xia, Zhi Wang, and Wenwu Zhu. Ts-dp: Reinforcement speculative decoding for temporal adaptive diffusion policy acceleration. *arXiv preprint arXiv:2512.15773*, 2025.

[24] Kangye Ji, Yuan Meng, Hanyun Cui, Ye Li, Shengjia Hua, Lei Chen, and Zhi Wang. Block-wise adaptive caching for accelerating diffusion policy. *arXiv preprint arXiv:2506.13456*, 2025.

[25] Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Jun Ma, and Haoang Li. Accelerating vision-language-action model integrated with action chunking via parallel decoding. *arXiv preprint arXiv:2503.02310*, 2025.

[26] Yuxiang Lu, Zhe Liu, Xianzhe Fan, Zhenya Yang, Jinghua Hou, Junyi Li, Kaixin Ding, and Hengshuang Zhao. Faster: Rethinking real-time flow vlas. *arXiv preprint arXiv:2603.19199*, 2026.

[27] Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

[28] Christopher M Harris and Daniel M Wolpert. Signal-dependent noise determines motor planning. *Nature*, 394(6695):780–784, 1998.

[29] Emanuel Todorov and Michael I Jordan. Optimal feedback control as a theory of motor coordination. *Nature neuroscience*, 5(11):1226–1235, 2002.

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

[31] Ye Li, ZhongXin Liu, Ge Lan, Malika Sader, and ZengQiang Chen. A ddpg-based solution for optimal consensus of continuous-time linear multi-agent systems. *Science China Technological Sciences*, 66(8):2441–2453, 2023.

[32] Zhongxin Liu, Ye Li, Ge Lan, and Zengqiang Chen. A novel data-driven model-free synchronization protocol for discrete-time multi-agent systems via td3 based algorithm. *Knowledge-Based Systems*, 287:111430, 2024.

[33] Chenghao Liu, Jiachen Zhang, Chengxuan Li, Zhimu Zhou, Shixin Wu, Songfang Huang, and Huiling Duan. Ttf-vla: Temporal token fusion via pixel-attention integration for vision-language-action models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 18452–18459, 2026.

[34] Kangye Ji, Yuan Meng, Zhou Jianbo, Ye Li, Hanyun Cui, and Zhi Wang. Sparse actiongen: Accelerating diffusion policy with real-time pruning. *arXiv preprint arXiv:2601.12894*, 2026.

[35] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024.

[36] Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. *arXiv preprint arXiv:2503.20384*, 2025.

Limitations and Responsible Use

This work is a technical study of inference-time acceleration for already trained vision-language-action robot policies. It does not involve new human-subject data, personal data, or deployment decisions, and we do not identify direct social or ethical concerns specific to the proposed scheduling method. As an acceleration method, ElegantVLA aims to reduce redundant computation while preserving the behavior of the underlying policy; real-world use should still follow the same task validation, robot safety checks, and fallback execution required for the base policy.

A Real-World Protocol and Additional Results

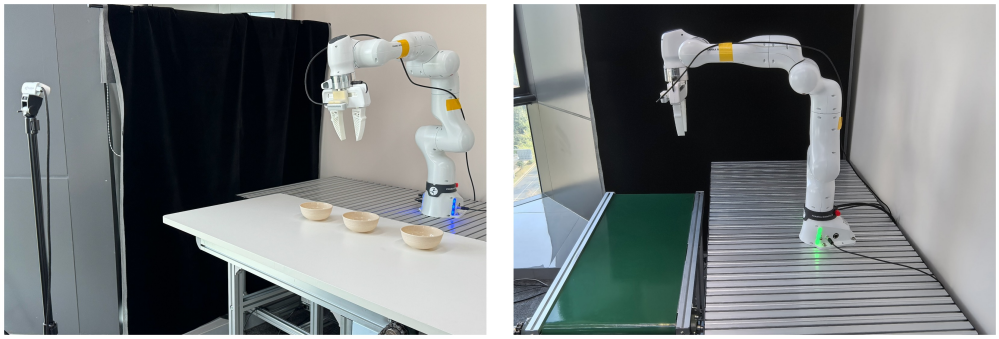

Figure 5: **Real-world experimental platform.** The physical experiments use a Franka Research 3 setup with stationary tabletop manipulation on the left and conveyor-belt pickup on the right.

Table 7: **Real-world task prompts and success criteria.** Natural-language prompts and success criteria used for the six tasks in Table 3.

Task	Instruction prompt	Success criterion
Phone Stand	Place the phone on a stand	The phone-like object is placed on the stand and remains stable.
Pen Holder	Insert the pen into the pen holder	The pen-like object is inserted into the holder without displacing it into a failed configuration.
Stack Bowls	Stack the three bowls on the table together	The three bowls are stacked together and remain stable after release.
Pineapple Bun	move the pineapple bun from the conveyor belt to the center of the plate	The pineapple bun is moved from the conveyor belt to the center of the plate.
Toast	move the toast from the conveyor belt to the center of the plate	The toast is moved from the conveyor belt to the center of the plate.
Chocolate	move the chocolate bun from the conveyor belt to the center of the plate	The chocolate bun is moved from the conveyor belt to the center of the plate.

Platform and protocol. Fig. 5 shows the physical setup. All real-world experiments are conducted on a Franka Research 3 robot equipped with Intel RealSense D435i cameras. The robot is controlled through Polymetis, while the GR00T-based policy server runs high-level VLA inference on an NVIDIA GeForce RTX 4090 GPU. Our real-world platform also uses Huawei Ascend hardware. We use both a front-mounted eye-to-hand camera and a wrist-mounted eye-in-hand camera. For each task and each method, we run 10 trials under the same initialization protocol. The full-computation GR00T policy is used as the real-world baseline. A trial is counted as successful only when the final object state satisfies the task goal and remains stable after execution.

Task design and analysis. Table 7 lists the instruction prompts and success criteria for the six real-world tasks. These tasks separate the evaluation into two regimes aligned with our phase-adaptive view. Phone Stand, Pen Holder, and Stack Bowls are stationary precision tasks, testing whether ElegantVLA preserves accurate contact, placement, and release while removing redundant computation. Pineapple Bun, Toast, and Chocolate Bun are conveyor-belt pickup tasks, where moving targets

Table 8: **Trial-level breakdown for the main real-world results.** Each task is evaluated over 10 real-robot trials, and each cell reports successful trials. The improvement row reports the change relative to full-computation GR00T, showing that ElegantVLA preserves success on stationary precision tasks while improving conveyor-belt pickup.

Method	Phone	Pen	Bowls	Pine apple	Toast	Choco late	Total	Rate
Full computation	6/10	7/10	4/10	7/10	7/10	6/10	37/60	61.67%
ElegantVLA	6/10	7/10	4/10	8/10	8/10	6/10	39/60	**65.00%**
Improvement	+0	+0	+0	+1	+1	+0	+2	+3.33 pp

Table 9: **Latency breakdown for the main real-world results.** We report per-step wall-clock latency for full-computation GR00T and ElegantVLA, the corresponding latency reduction, wall-clock speedup, and FLOPs speedup. The results show that phase-adaptive reuse consistently reduces real-robot inference latency, with the largest gains aligning with tasks that benefit from faster closed-loop correction.

Task	Success rate	Full GR00T ms/step	ElegantVLA ms/step	Reduction	Latency speedup	FLOPs speedup
Phone-on-Stand	60.0%	72.31	37.70	47.9%	1.92×	2.18×
Pen-to-Holder	70.0%	71.74	36.20	49.5%	1.98×	2.24×
Stack-Bowls	40.0%	71.92	41.30	42.6%	1.74×	2.01×
Pineapple-Bun	80.0%	72.74	35.70	50.9%	2.04×	2.31×
Toast	80.0%	72.88	37.40	48.7%	1.95×	2.21×
Chocolate-Bun	60.0%	73.05	39.70	45.7%	1.84×	2.11×
Average	65.0%	72.44	38.00	47.5%	1.91×	2.18×

make delayed actions costly and faster inference should improve closed-loop correction. Together, they test whether stable phases can tolerate temporal reuse while precision- or correction-sensitive phases still receive fresh reasoning and action refinement.

Additional real-world results. Tables 8 and 9 decompose the real-world result in Table 3 into trial counts and latency measurements. Table 8 shows that ElegantVLA matches full-computation GR00T on the three stationary precision tasks, indicating that adaptive reuse does not noticeably degrade contact-sensitive manipulation in this setting. The only success-count improvements appear on Pineapple Bun and Toast, where the moving target makes lower-latency feedback more valuable. Table 9 confirms that these gains are backed by practical runtime reductions: average per-step latency drops from 72.44 ms to 38.00 ms, corresponding to a 1.91× wall-clock speedup and 2.18× FLOPs speedup. This pattern is consistent with the main analysis: ElegantVLA converts redundant computation into faster control while keeping enough fresh reasoning and action refinement for precision-sensitive phases.

Table 10: **CogACT simulation task groups.** We list the task headers used in the CogACT VM and VA result tables. The VM episode counts for PickCan, MoveNear, Drawer, and PutDrawer are 300/240/216/108, and the corresponding VA episode counts are 825/600/189/189.

Header	Expanded task group	Meaning in evaluation
PickCan	Pick Coke Can	The Google Robot grasps an opened Coke can under object-pose, distractor, background, lighting, and camera variants.
MoveNear	Move Near	The Google Robot moves a target object near a reference target, testing spatial relational reasoning and continuous closed-loop correction.
Drawer	Drawer Manipulation	Aggregates drawer opening and closing tasks over top, middle, and bottom drawer variants.
PutDrawer	Put in Drawer	The Google Robot places an object, such as an apple, into a closed drawer; this task combines grasping, drawer geometry, and final placement.

Table 11: **GR00T simulation task abbreviations by robot suite.** Headers are used in the Google Robot and WidowX Robot result tables, with 200 simulation episodes evaluated for each task.

Robot	Header	Expanded task name	Task meaning
Google Robot	CloseD	Close Drawer	Close the drawer to the target final state.
Google Robot	MoveNear	Move Near	Move an object near a specified target or reference region.
Google Robot	OpenD	Open Drawer	Open the drawer to the required extent.
Google Robot	PickCan	Pick Coke Can	Pick or grasp a Coke can from the tabletop.
Google Robot	PickObj	Pick Object	Pick a generic target object rather than a fixed Coke-can object.
Google Robot	PlaceD	Place in Closed Drawer	Place the target object into a closed drawer.
WidowX Robot	Carrot	Carrot on Plate	Place the carrot on the plate.
WidowX Robot	CloseD	Close Drawer	Close the drawer in the WidowX setup.
WidowX Robot	OpenD	Open Drawer	Open the drawer in the WidowX setup.
WidowX Robot	Basket	Eggplant in Basket	Put the eggplant into the basket.
WidowX Robot	Sink	Eggplant in Sink	Put the eggplant into the sink.
WidowX Robot	Spoon	Spoon on Towel	Place the spoon on the towel.
WidowX Robot	Stack	Stack Cube	Stack one cube on another cube.

B Simulation Environment and Task Protocol

Benchmark protocol. All simulation experiments follow the SimplerEnv grouping used in the main tables. We evaluate frozen CogACT and GR00T backbones against their full-computation baselines, changing only the inference execution path. Profiling is performed on an NVIDIA RTX 4090 GPU.

Task groups. GR00T covers six Google Robot tasks and seven WidowX Robot tasks, with 200 episodes per task. CogACT uses the VM and VA task groups: PickCan, MoveNear, Drawer, and PutDrawer, with episode counts 300/240/216/108 for VM and 825/600/189/189 for VA. Tables 10 and 11 expand the compact task headers used in the main and appendix simulation tables. Together, these tasks cover contact-sensitive manipulation and smoother transport phases, matching the phase-adaptive evaluation goal of ElegantVLA.

C Additional Scheduler Ablations

This section supplements the main ablation study with two scheduler-specific questions. First, we examine whether a hand-designed threshold rule or only the first RL training stage is sufficient. Second, we ablate the scheduler observations to test why CKA, robot speed, and task progress are used together. All GR00T results are evaluated in SimplerEnv, and each cell reports success rate and FLOPs speedup.

C.1 Effect of Rule-Based and Two-Stage RL Scheduling

Table 12 compares the full-computation baseline, a rule-based threshold scheduler, the Stage1 RL scheduler, and the final two-stage ElegantVLA scheduler. The threshold scheduler tests whether a manually specified representation-similarity trigger is sufficient. Stage1 RL tests whether the first training stage gives a usable initialization, while ElegantVLA further applies the task-feedback correction from Stage2.

Table 12: **Effect of rule-based and two-stage RL scheduling on GR00T.** Each cell reports success rate (%, ↑) / FLOPs speedup (↑). Threshold is a hand-designed scheduler, Stage1 RL is the first scheduler-training stage, and ElegantVLA is the complete two-stage scheduler.

SIMPLER Method		Success Rate (%, ↑) / FLOPs Speedup (↑)						AVG
		CloseD	MoveNear	OpenD	PickCan	PickObj	PlaceD	
Google Robot	Baseline	82.00 / 1.00	92.00 / 1.00	49.50 / 1.00	99.50 / 1.00	92.50 / 1.00	11.00 / 1.00	71.08 / 1.00
	Threshold	80.50 / 2.08	91.50 / 1.97	67.50 / 2.12	97.00 / 1.91	87.00 / 2.03	12.00 / 2.14	72.58 / 2.05
	Stage1 RL	80.50 / 1.98	80.00 / **2.43**	72.50 / 2.03	**98.50** / 1.85	**90.00** / 1.99	**15.00** / 2.12	72.75 / 2.09
	ElegantVLA	**81.00 / 2.39**	**95.50** / 1.90	**74.50 / 2.31**	98.00 / **2.35**	88.50 / **2.59**	12.50 / **2.39**	**75.00 / 2.35**

SIMPLER Method		Success Rate (%, ↑) / FLOPs Speedup (↑)							AVG
		Carrot	CloseD	OpenD	Basket	Sink	Spoon	Stack	
WidowX	Baseline	60.50 / 1.00	98.50 / 1.00	93.00 / 1.00	37.50 / 1.00	37.50 / 1.00	71.50 / 1.00	7.00 / 1.00	57.93 / 1.00
	Threshold	62.00 / 2.13	99.00 / 2.13	84.50 / 2.20	27.50 / 1.97	25.00 / 2.24	72.00 / 2.18	13.00 / 2.23	54.71 / 2.12
	Stage1 RL	63.00 / **2.59**	99.00 / 2.52	83.50 / **2.57**	15.00 / **2.53**	40.00 / **2.57**	67.00 / **2.55**	11.50 / 2.54	54.14 / **2.55**
	ElegantVLA	**63.50** / 2.58	**99.50 / 2.54**	**85.00** / 2.56	**31.50** / 2.52	**41.00 / 2.57**	**72.50 / 2.55**	**13.50 / 2.55**	**58.07 / 2.55**

Analysis. The threshold scheduler confirms that a representation-similarity trigger is useful, but it is not sufficient as a final policy. It slightly improves the Google Robot average from 71.08% to 72.58%, yet drops WidowX from 57.93% to 54.71%, showing that a fixed hand rule does not transfer cleanly across manipulation regimes. Stage1 RL gives the scheduler a usable initialization and often increases acceleration; on WidowX, it reaches a 2.55× average speedup. However, Stage1 can still be too aggressive: on Google MoveNear, it increases speedup to 2.43× but reduces success to 80.00%. The complete two-stage scheduler recovers the trade-off by using Stage2 task feedback to calibrate reuse, improving Google Robot to 75.00% and WidowX to 58.07%. Thus, the final gain is not simply from reusing more computation, but from learning when reuse should be relaxed to preserve task success.

C.2 Effect of Scheduler Observations

Table 13 ablates the scheduler input signals. Episode progress is included in all reduced-observation variants as a common temporal cue. The ablation therefore focuses on whether the scheduler observes representation similarity through CKA, robot motion through speed, or both.

Analysis. CKA+progress gives the scheduler a semantic stability cue, so it can detect when the Vision–LLM representation changes across time. However, it cannot observe whether the robot is entering contact, moving quickly, or requiring low-level action refinement. Speed+progress gives the opposite view: it captures motion continuity and often keeps FLOPs speedup high, but it cannot directly detect visual-language changes such as object displacement, drawer-state changes, or instruction-relevant scene transitions. The full CKA+speed+progress observation combines these complementary cues. It raises the average success rate to 75.00% on Google Robot and 58.07% on

Table 13: **Effect of scheduler observations on GR00T.** Each cell reports success rate (%, ↑) / FLOPs speedup (↑). Progress is included as a shared temporal cue; the comparison isolates CKA, speed, and their combination.

SIMPLER	Stage	Obs.	Success Rate (%, ↑) / FLOPs Speedup (↑)						AVG
			CloseD	MoveNear	OpenD	PickCan	PickObj	PlaceD	
Google Robot	Baseline	–	82.00 / 1.00	92.00 / 1.00	49.50 / 1.00	99.50 / 1.00	92.50 / 1.00	11.00 / 1.00	71.08 / 1.00
	Stage1	CKA+progress	78.50 / 2.49	79.50 / 2.50	59.00 / 2.46	90.50 / 2.53	82.00 / 2.50	7.00 / 2.45	66.08 / 2.47
	Stage1	speed+progress	79.50 / 2.49	78.00 / 2.46	67.50 / 2.47	90.00 / 2.47	84.00 / 2.47	8.50 / 2.42	67.92 / 2.45
	Stage2	CKA+progress	80.50 / 2.43	79.00 / 2.39	60.50 / 2.43	91.50 / 2.30	81.50 / 2.38	7.00 / 2.44	66.67 / 2.42
	Stage2	speed+progress	78.50 / **2.59**	79.00 / **2.61**	63.00 / **2.58**	92.00 / **2.64**	80.50 / **2.59**	7.50 / **2.55**	66.75 / **2.58**
	Stage2	CKA+speed +progress	**81.00** / 2.39	**95.50** / 1.90	**74.50** / 2.31	**98.00** / 2.35	**88.50** / 2.59	**12.50** / 2.39	**75.00** / 2.35

SIMPLER	Stage	Obs.	Success Rate (%, ↑) / FLOPs Speedup (↑)							AVG
			Carrot	CloseD	OpenD	Basket	Sink	Spoon	Stack	
WidowX	Baseline	–	60.50 / 1.00	98.50 / 1.00	93.00 / 1.00	37.50 / 1.00	37.50 / 1.00	71.50 / 1.00	7.00 / 1.00	57.93 / 1.00
	Stage1	CKA+progress	55.50 / 2.45	94.50 / 2.43	86.50 / 2.45	19.50 / 2.45	34.50 / 2.44	69.00 / 2.44	8.50 / 2.44	52.57 / 2.44
	Stage1	speed+progress	53.50 / 2.53	94.00 / 2.50	79.50 / 2.53	18.50 / 2.50	29.50 / 2.54	62.00 / 2.51	8.50 / 2.51	49.36 / 2.52
	Stage2	CKA+progress	53.00 / 2.43	93.50 / 2.41	83.00 / 2.42	19.50 / 2.44	33.50 / 2.43	58.50 / 2.43	**14.00** / 2.43	50.71 / 2.43
	Stage2	speed+progress	59.00 / 2.52	94.00 / 2.50	**87.00** / 2.54	19.00 / 2.51	37.50 / 2.54	68.00 / 2.51	8.50 / 2.52	53.29 / 2.52
	Stage2	CKA+speed +progress	**63.50 / 2.58**	**99.50 / 2.54**	85.00 / **2.56**	**31.50 / 2.52**	**41.00 / 2.57**	**72.50 / 2.55**	13.50 / **2.55**	**58.07 / 2.55**

WidowX; across all 13 tasks, it improves the average from the reduced-observation range of 57.92–59.50% to 65.88% while preserving a 2.47× FLOPs speedup. This supports the scheduler design: CKA helps decide when backbone reuse is safe, speed helps decide when action-head reuse is safe, and progress provides a shared temporal prior rather than the main ablated signal.

D Qualitative Visualizations

D.1 Additional Real-World Rollouts

To complement the pineapple-bun rollout shown in Fig. 3, we show the remaining five real-world tasks individually in Figs. 6–10. These rollouts evaluate two different effects of acceleration in the physical setup. For stationary manipulation tasks, the goal is to verify that temporal reuse does not sacrifice execution precision. For conveyor-belt pickup tasks, the goal is to verify that lower inference latency gives the policy more responsive closed-loop behavior while the target is moving.

Phone Stand, Pen Holder, and Stack Bowls are stationary precision tasks. They require contact-sensitive behaviors such as approach, alignment, insertion or placement, and release. The qualitative question is therefore not whether the robot simply moves faster, but whether acceleration preserves the fine-grained phases needed for accurate manipulation. The visual sequences show that ElegantVLA retains these phases rather than collapsing the execution into a coarse approach-and-release pattern. This supports the trial-level results in Appendix A, where ElegantVLA preserves success on stationary real-world tasks while reducing redundant computation.

Toast and Chocolate Bun, together with the pineapple-bun example in the main paper, stress a different regime. Because the object pose changes continuously on the conveyor belt, delayed actions make the end-effector chase a stale target state. Here, acceleration is expected to improve responsiveness: the policy can observe the moving object and issue corrective actions with less observation-action delay. The rollouts show more target-centered corrections during approach and grasping, indicating that the compute savings translate into a more sensitive closed-loop response rather than only a lower FLOPs count. Across both task families, the real-world visualizations support the same conclusion as the quantitative breakdown: ElegantVLA preserves precision when the scene is stable and improves reaction to target motion when the scene changes during execution.

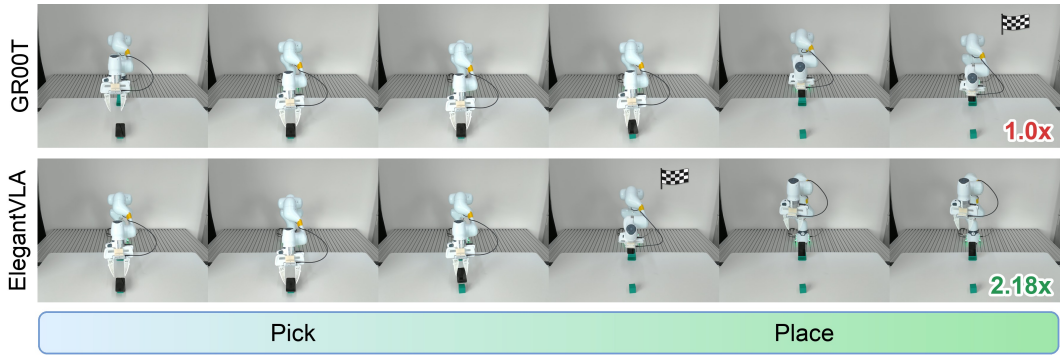

Figure 6: **Real-world rollout on Phone Stand.** ElegantVLA preserves precise placement and release on a stationary task, indicating that acceleration does not degrade fine-grained execution.

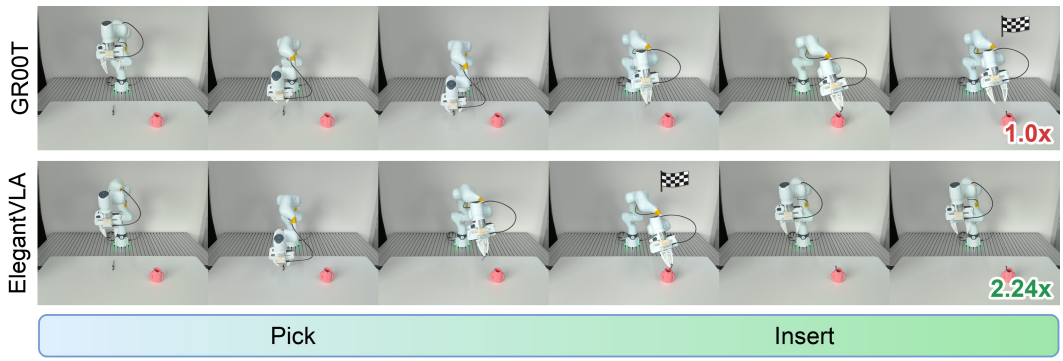

Figure 7: **Real-world rollout on Pen Holder.** Adaptive reuse keeps the contact-sensitive insertion phase, providing a qualitative check that temporal reuse does not remove precision-critical behavior.



Figure 8: **Real-world rollout on Stack Bowls.** The sequence preserves grasping, alignment, and release, showing that acceleration does not collapse the manipulation into a coarse action sequence.

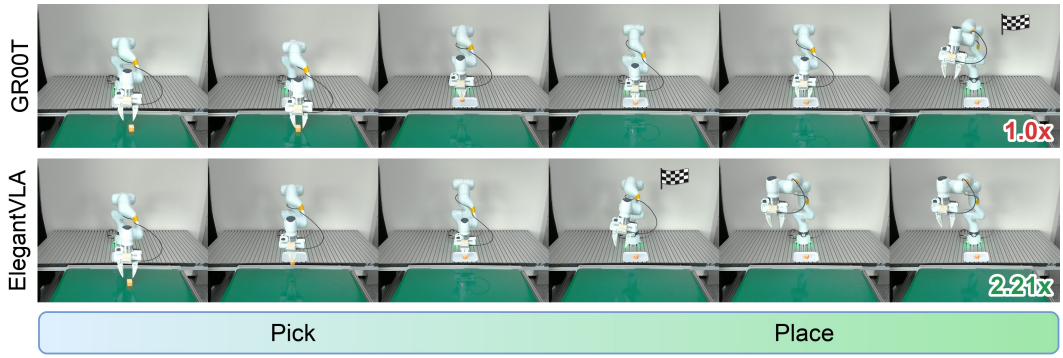

Figure 9: **Real-world rollout on Toast.** The conveyor-belt setting tests whether lower latency lets the policy keep correcting the end-effector motion around a moving target.

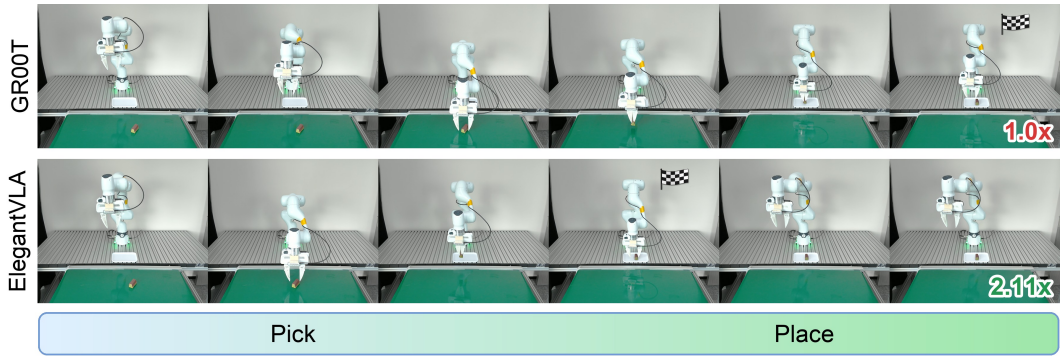

Figure 10: **Real-world rollout on Chocolate Bun.** This additional moving-target rollout shows the same responsiveness regime as the main pineapple-bun example, where timely corrections are important.

D.2 Simulation Rollouts and Scheduler Behavior

The simulation rollouts visualize the task-complexity adaptivity of ElegantVLA. Rather than applying a fixed lightweight policy throughout an episode, the scheduler changes the amount and type of computation as the task state evolves. During simple and stable phases, such as smooth transport or visually unchanged approach, the policy can use more aggressive reuse. When the task becomes more complex, such as object contact, drawer-state change, grasp adjustment, or final placement, the scheduler reduces reuse and allocates more fresh computation.

This adaptation happens at two semantic levels. For the Vision–LLM backbone, the scheduler mainly uses CKA to estimate high-level semantic stability. A high CKA value indicates that the current visual-language representation remains close to the cached representation, so backbone reuse is likely safe. A low CKA value suggests that the task semantics may have changed, for example when the object pose, drawer state, or spatial relation changes, and the scheduler therefore refreshes the high-level representation. For the action head, the scheduler mainly uses robot speed to estimate low-level control complexity. Smooth and continuous motion usually corresponds to a stable control phase where action-head reuse is sufficient. In contrast, rapid speed changes often occur near contact, correction, or release, where fine-grained control matters and aggressive reuse can be risky.

Fig. 11 illustrates this two-level adaptation on a representative rollout. The remaining simulation rollouts in Figs. 12–23 show the same principle across Google Robot and WidowX Robot tasks. Across different task demands, ElegantVLA uses lightweight inference during easy phases while preserving computation for semantically or physically complex phases. This behavior is the key qualitative evidence that our acceleration is task-complexity adaptive rather than a fixed reduction in model computation.

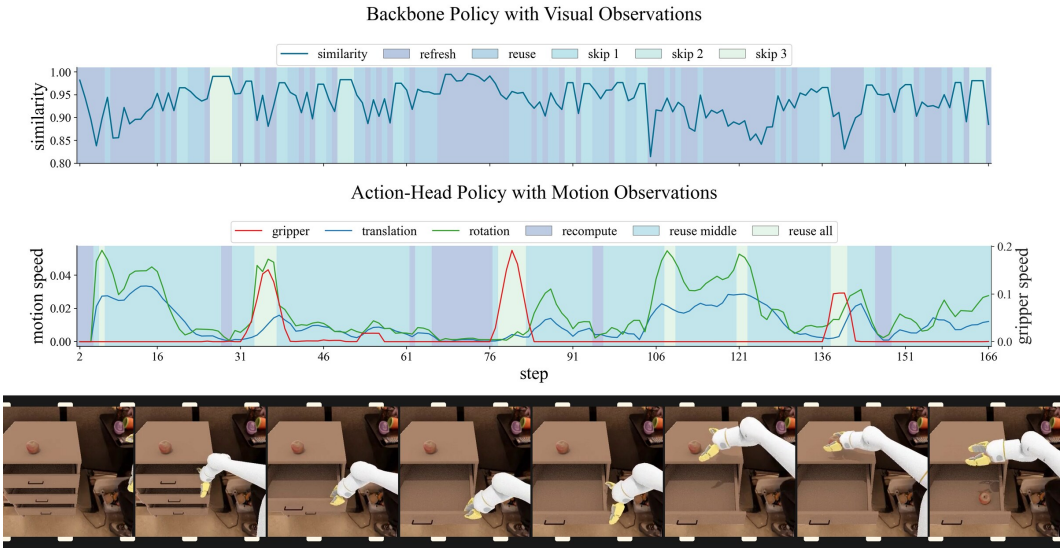

Figure 11: **Representative rollout-level scheduling diagnostic.** The background colors show selected execution modes for the Vision–LLM backbone and the action head, while the curves show CKA-based representation similarity and robot motion speeds along the same trajectory. The backbone follows high-level semantic stability, whereas the action head follows low-level control stability.

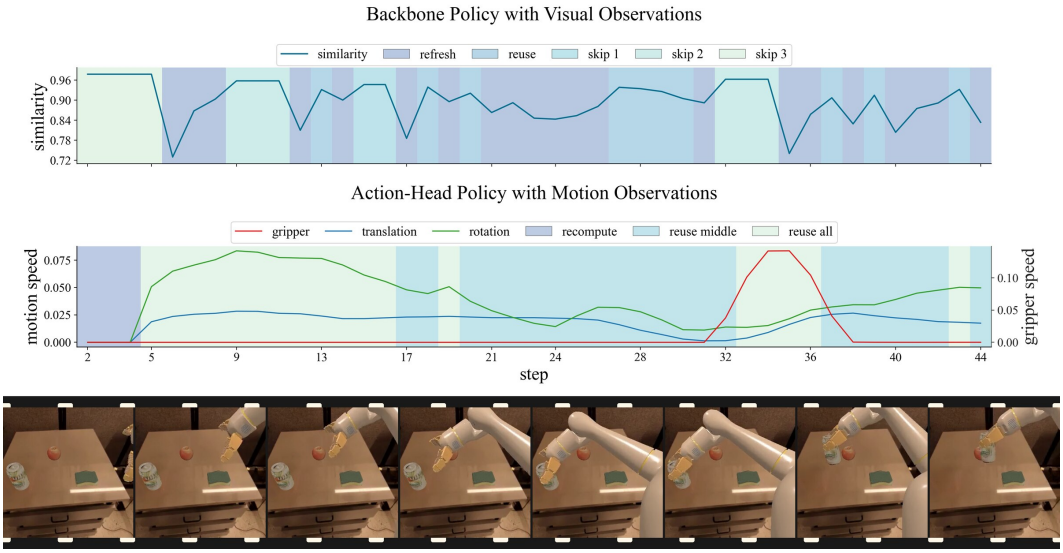

Figure 12: **Google Robot simulation rollout on Move Near.** The spatial transport task shows how the scheduler changes lightweight modes as high-level task complexity changes.

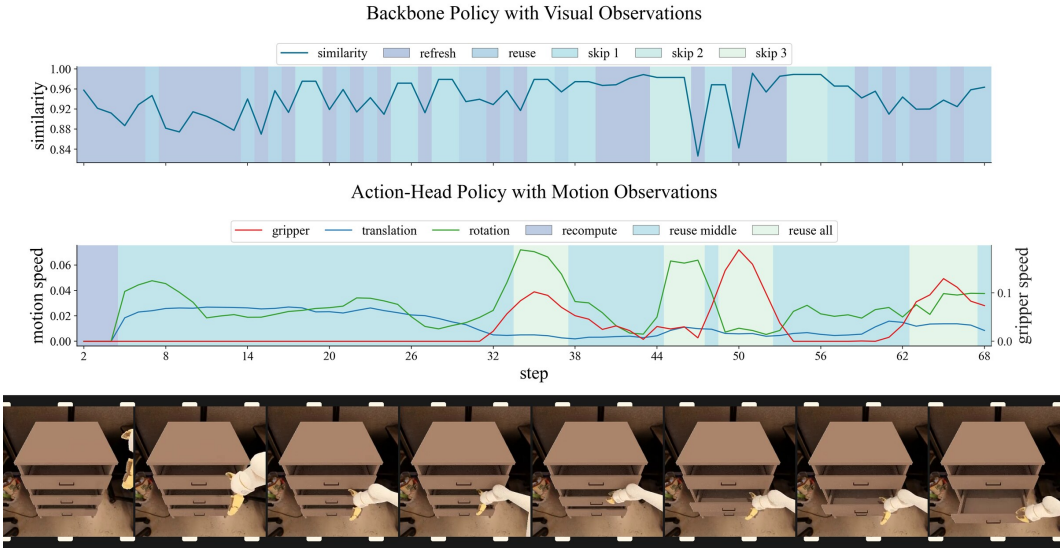

Figure 13: **Google Robot simulation rollout on Open Drawer.** The drawer manipulation task illustrates that ElegantVLA adapts computation to semantic state changes rather than following one fixed reuse pattern.

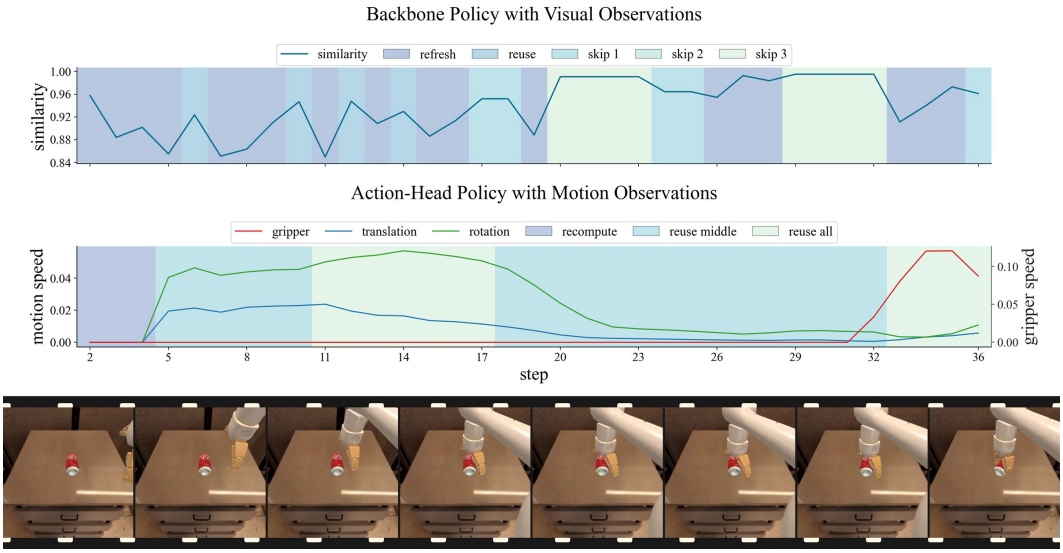

Figure 14: **Google Robot simulation rollout on Pick Coke Can.** The object pickup task shows how high-level object state and low-level grasp control jointly affect the scheduler's lightweight choices.

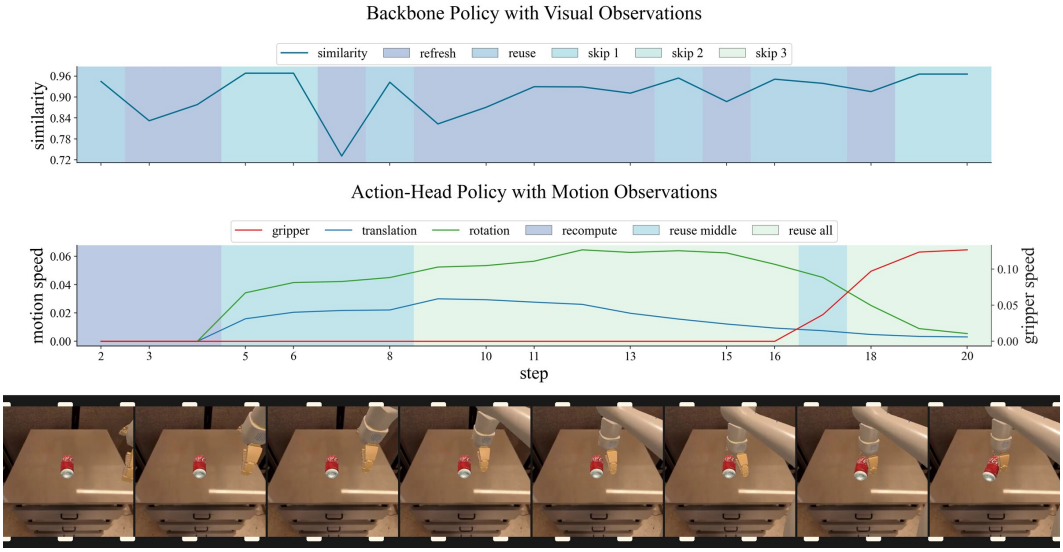

Figure 15: **Google Robot simulation rollout on Pick Object.** The generic pickup task provides another example where ElegantVLA adjusts computation as semantic and control complexity vary over time.

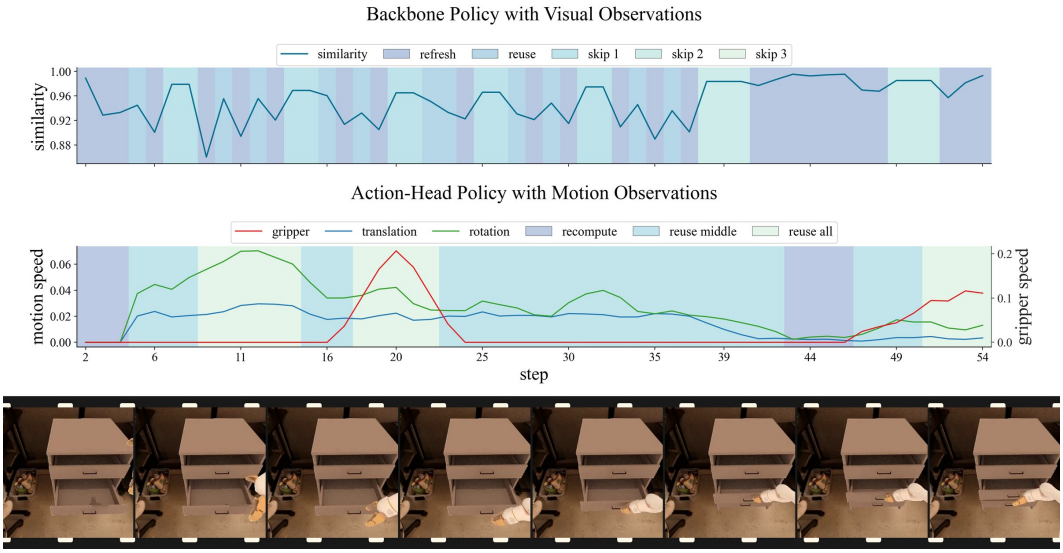

Figure 16: **Google Robot simulation rollout on Close Drawer.** Closing the drawer requires the scheduler to preserve computation around drawer-state changes while allowing reuse during stable motion.

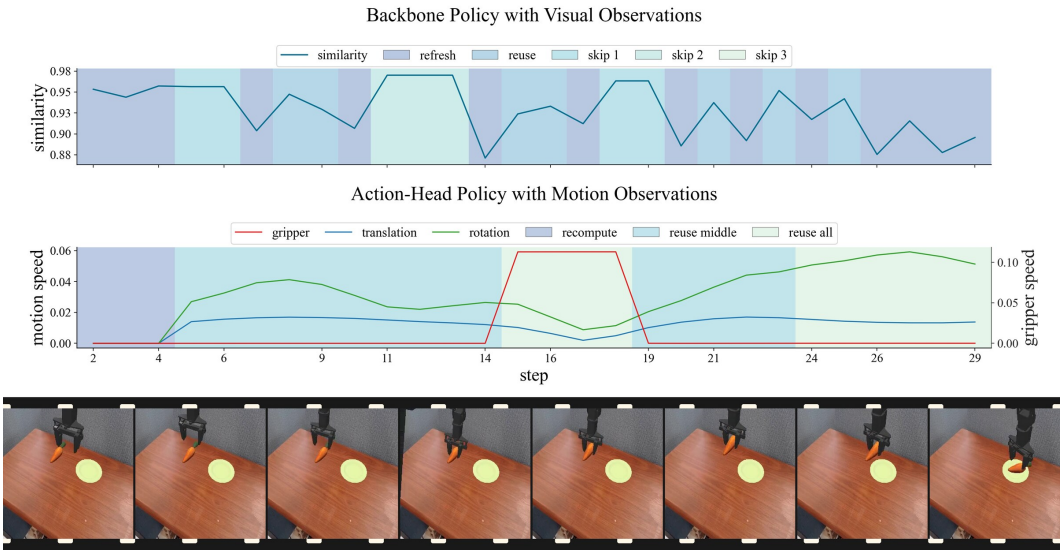

Figure 17: **WidowX Robot simulation rollout on Carrot on Plate.** The placement task tests whether the two-level scheduler remains adaptive under the WidowX visual domain.

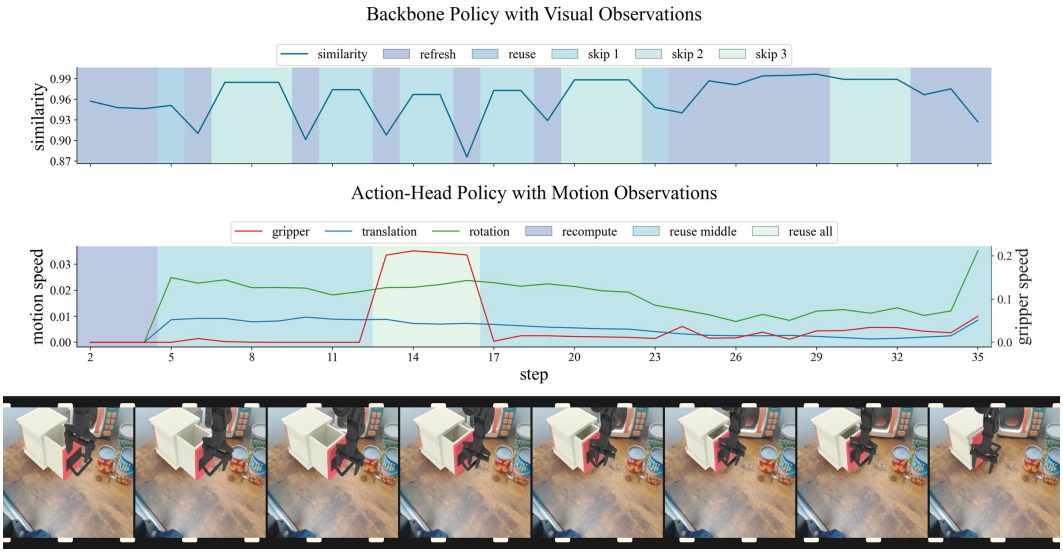

Figure 18: **WidowX Robot simulation rollout on Close Drawer.** The contact-rich drawer interaction shows how low-level control complexity affects action-head reuse.

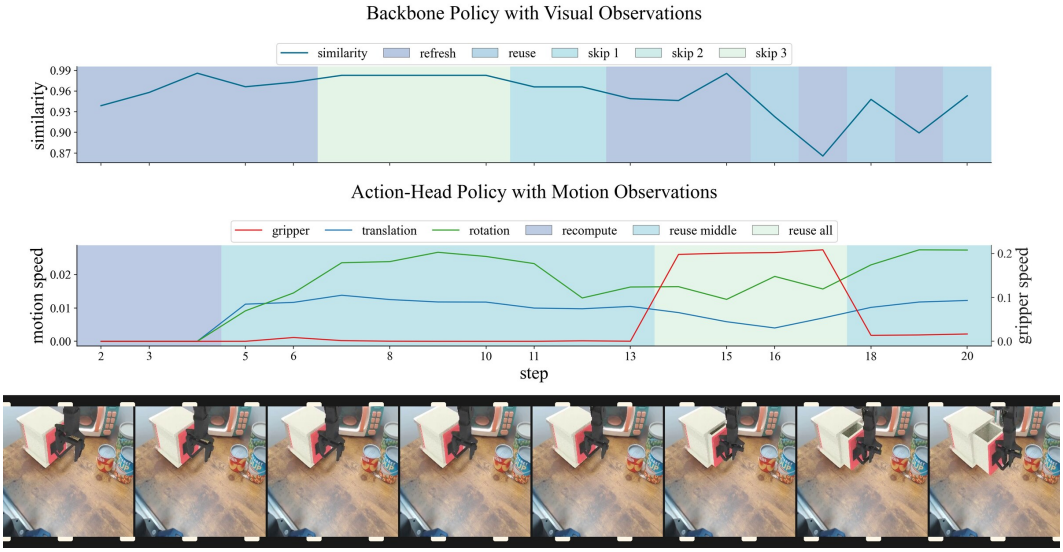

Figure 19: **WidowX Robot simulation rollout on Open Drawer.** Opening the drawer provides another contact-sensitive example where the scheduler should reduce aggressive reuse around physical interaction.

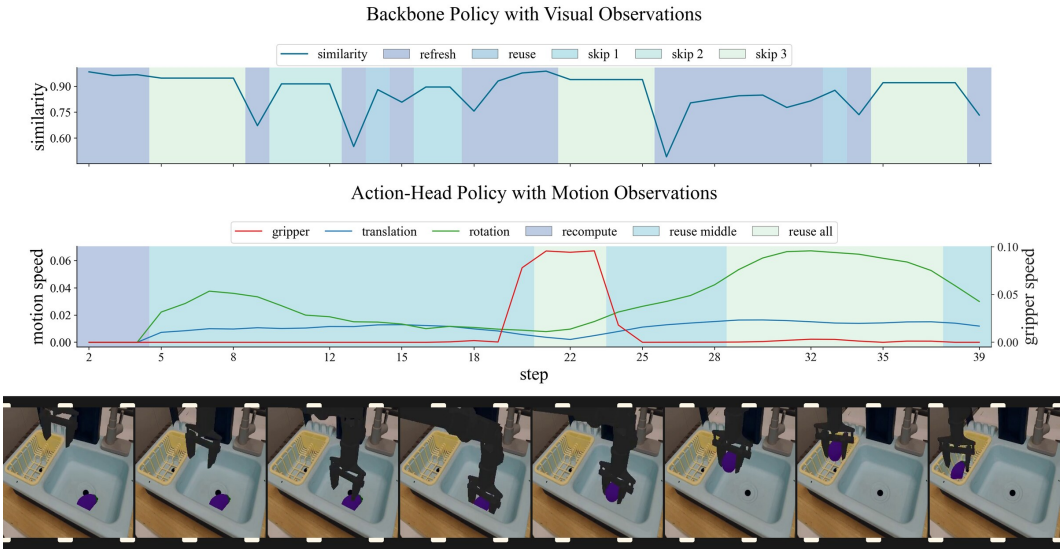

Figure 20: **WidowX Robot simulation rollout on Put Eggplant in Basket.** The final placement changes the required object configuration, so the scheduler preserves computation during the more complex control phase.

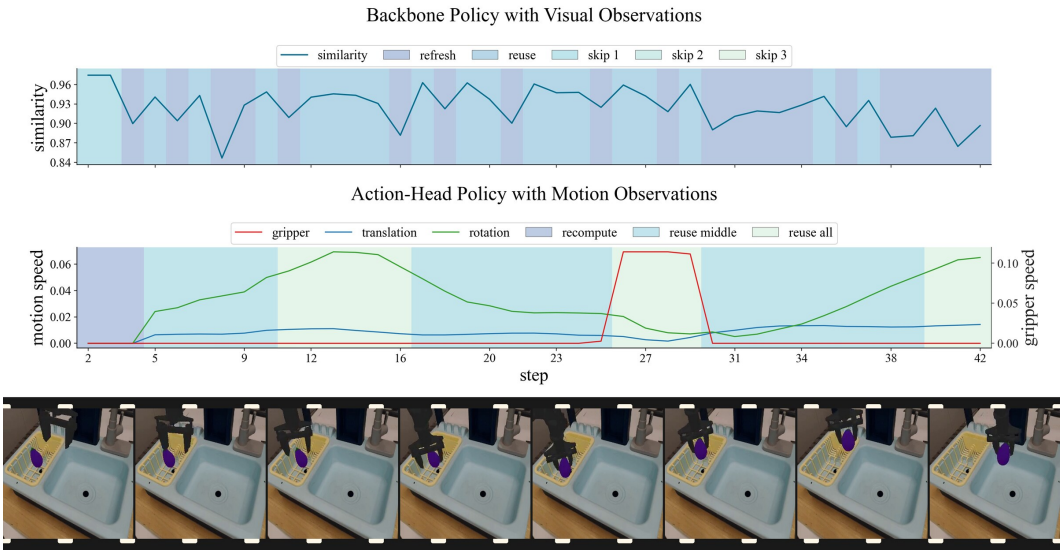

Figure 21: **WidowX Robot simulation rollout on Put Eggplant in Sink.** This placement task further shows task-complexity-adaptive computation under changing final object configurations.

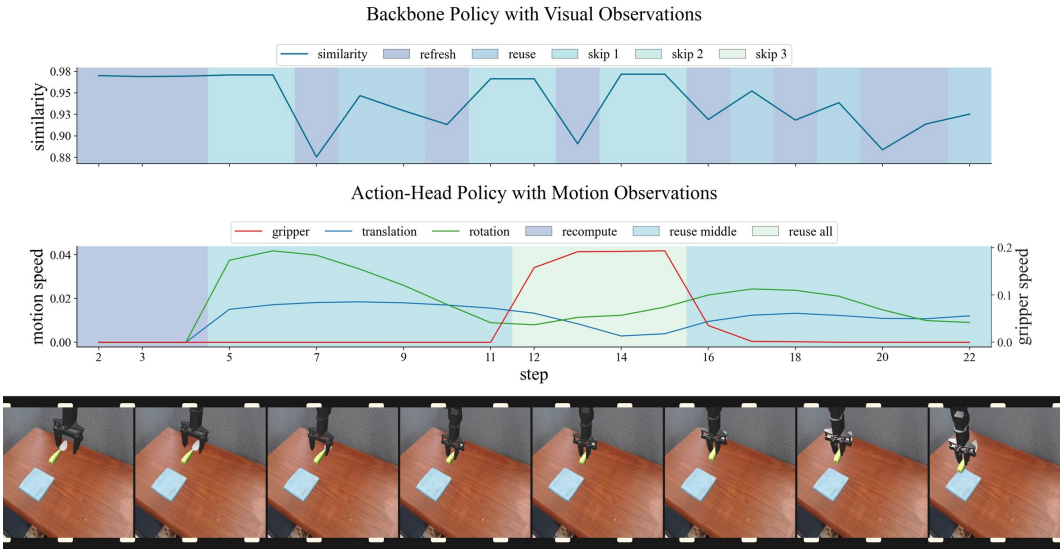

Figure 22: **WidowX Robot simulation rollout on Spoon on Towel.** The scheduler keeps lightweight inference for stable motion while preserving computation near placement and release.

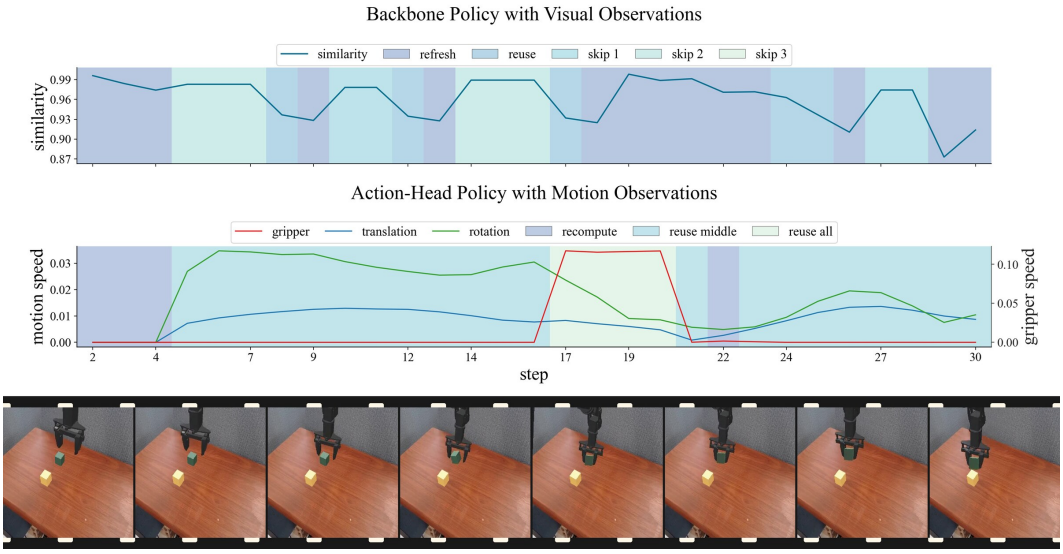

Figure 23: **WidowX Robot simulation rollout on Stack Cube.** The stacking task provides an additional contact-sensitive case where low-level alignment and release require less aggressive action-head reuse.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes].

   Justification: The abstract and introduction state that ElegantVLA is a plug-in inference-time acceleration framework for frozen VLA policies, and they report the same speedup and success-rate claims evaluated in Sections 4 and B.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes].

   Justification: The appendix includes a "Limitations and Responsible Use" discussion that clarifies the scope of the work as inference-time acceleration for already trained robot policies and notes that deployment still requires task validation, robot safety checks, and fallback execution.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

Answer: [N/A].

Justification: The paper does not present formal theoretical results or proofs; the mathematical content in Section 3 defines the scheduler and execution paths used by the method.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes].

Justification: Sections 3 and 4 describe the algorithm, scheduler observations, action spaces, baselines, metrics, and main evaluation protocol. Appendix B and Appendix A further specify task groups, episode counts, real-robot trials, hardware, and success criteria.

Guidelines:

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

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes].

   Justification: The code and reproduction instructions are publicly available at https://github.com/ChildTang/ElegantVLA.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes].

   Justification: Section 3 specifies the frozen-policy setting, scheduler observations, compute actions, reward structure, and Maskable PPO training procedure. Section 4 and the appendix describe the benchmark tasks, baselines, episode counts, real-robot trials, metrics, and hardware.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes].

Justification: The paper reports the number of evaluation episodes or trials behind the main results, including 200 GR00T episodes per simulation task, CogACT task-group episode counts, and 10 real-robot trials per task. Appendix A additionally reports real-world success counts rather than only percentages.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes].

   Justification: The experiments report profiling on an NVIDIA RTX 4090 GPU, and Appendix A specifies the real-robot platform and GPU used by the GR00T policy server. The main paper and appendix report per-step latency, control frequency, FLOPs speedup, and wall-clock speedup for the evaluated settings.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes].

   Justification: The work is a technical study of inference-time acceleration for robot policies and does not involve human-subject data, personal data, or deployment decisions, as discussed in the appendix. The real-robot experiments are controlled manipulation tasks and are framed with standard robot safety and fallback considerations.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes].

    Justification: The appendix discusses responsible use of the proposed acceleration layer, noting that it is a technical efficiency method without direct human-subject or personal-data concerns. It also states that real-world robot use should still follow task validation, safety checks, and fallback execution for the base policy.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A].

    Justification: The paper does not release a new high-risk model, scraped dataset, image generator, or pre-trained language model. The proposed scheduler is an inference-time acceleration layer evaluated on robot-control policies.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes].

    Justification: The paper cites the existing models, benchmarks, and baselines used in the evaluation, including GR00T, CogACT, SimplerEnv-style tasks, and acceleration baselines. No existing asset is redistributed as part of this submission.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes].

    Justification: The released GR00T implementation and Google Robot scheduler checkpoint are documented in the public repository linked from `https://childtang.github.io/ElegantVLA/`, including setup, evaluation, and training instructions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A].

    Justification: The paper does not involve crowdsourcing, human participants, or human-subject data collection. All experiments are simulation or controlled robot manipulation evaluations.

    Guidelines:

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

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

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

    Answer: [N/A].

    Justification: No human subjects are involved in the research, so IRB or equivalent approval is not applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A].

    Justification: LLMs were used solely for language polishing of the manuscript.

    Guidelines:

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