SynerVLA: Exploiting Embodied Execution Phases for On-Device Dual-System VLA Acceleration

Qi Lu[1] **Haotian Xiong**[1] **Ziyu Gong**[1] **Tianjun Shi**[1]
Lei Xie[2] **Chengzhong Xu**[1] **Li Li**[1*]

[1] SKL-IOTSC, ISCT, University of Macau
[2] Tsinghua University
{yc57960,llili}@um.edu.mo

Abstract

Dual-system visual-language-action models, which integrate high-level planning (System 2) with instant control (System 1), are promising for embodied AI, but their deployment is hindered by the high computational cost of processing continuous visual streams. Existing acceleration methods are one-sided, focusing only on optimizing the VLM (System 2), which not only limits speed gains but also risks compromising the deep reasoning capabilities it is meant to provide. In this paper, we introduce SynerVLA, a plug-and-play framework for accelerating dual-system VLA models. Its core idea is to leverage the distinct phase information in embodied execution, rapid approaching and fine-grained manipulation, to exploit both spatial and temporal redundancy in visual streams throughout the entire execution pipeline. SynerVLA accelerates dual-system VLA inference by first selecting key visual tokens via text-action fusion, then dynamically adjusting the token pruning-reuse ratio through phase-aware feedback, and finally accelerating both System-2 VLM and System-1 diffusion Transformer with a dual cache reuse mechanism. Evaluations on representative platforms demonstrate that SynerVLA delivers up to a 1.93× speedup and a 26% higher control frequency, with only a negligible impact on task success rate.

1 Introduction

Drawing inspiration from the human "fast and slow thinking" cognitive model [1], the dual-system Vision-Language-Action (VLA) architecture splits a unified model into two relatively independent yet cooperative subsystems[2–5]. The "slow system" (System 2), built upon a large vision-language model, handles complex scene understanding, semantic reasoning, and high-level task planning. The lightweight "fast system" (System 1) is responsible for translating System 2's outputs, such as symbolic goals or object coordinates, into real-time, low-level actions. This design seeks to balance robust semantic generalization with precise real-time performance[6–8]. Nevertheless, deploying this architecture on embodied AI systems that run directly on resource-constrained edge devices remains challenging, as both subsystems must continuously process streaming visual data in dynamic environments. This leads to high computational overhead and synchronization delays[9, 10].

To enhance inference efficiency, several acceleration strategies have been introduced. For instance, methods such as DTP[11], Light-VLA[12], and VLA-Pruner[13] target spatial redundancy by aggressively removing visual tokens that contribute minimally to task performance. In contrast, another notable line of research, typified by VLA-Cache[14], exploits temporal similarity across consecutive frames to reuse key-value (KV) caches-in order to eliminate redundant computation for unchanged

*Corresponding author.

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

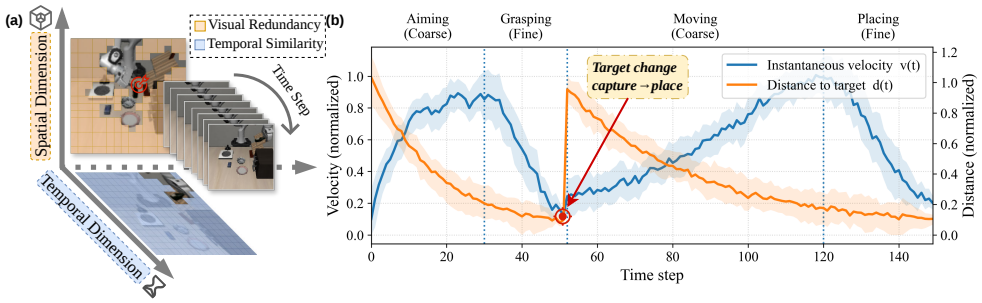


Figure 1: Visual redundancy and phase-dependent dynamics in robotic manipulation. (a) Spatial redundancy within visual observations and temporal similarity across consecutive frames. (b) Normalized velocity and target distance across coarse-grained (aiming/moving) and fine-grained (grasping/placing) execution phases.

information while maintaining accuracy and further accelerating inference. However, prevailing approaches are predominantly one-sided, optimizing only traditional VLM (System 2). This not only limits the acceleration gain but also compromises the high-level reasoning capability of complex scenes.

In embodied execution, the following two key features create a unique opportunity to optimize the latency-performance trade off for dual-system VLA throughout the entire execution pipeline. First, the visual information processing is consistently characterized by the coexistence of temporal similarity and spatial redundancy. As illustrated in Figure 1(a), for inter-frame redundancy, there is high content similarity between consecutive video frames (inter-frame redundancy). For intra-frame redundancy, individual frames also contain a large amount of non-critical background information (intra-frame redundancy). These two forms of redundancy persist across the entire embodied execution pipeline, providing a structural foundation for dynamically optimizing computational efficiency. Second, embodied execution exhibits a pronounced phase-wise characteristic, typically manifested in two distinct granularity phases: *rapid approaching* and *fine-grained manipulation*, as shown in Figure 1(b). In the *rapid approaching* phase, the robot moves toward the target at high speed, where precision requirements are low and redundant information dominates the visual input. In contrast, during the *fine-grained manipulation* phase, the robot performs slow adjustments and precise alignment, demanding highly complete local geometry and subtle visual cues. This phase-wise distinction further guides how to leverage redundancy differently across phases. In the rapid approaching phase, rapid scene changes and high redundancy make direct pruning of unimportant visual tokens a better choice for drastically cutting computational cost. In contrast, the fine-grained manipulation phase, characterized by continuous motion and high inter-frame similarity, favors caching important tokens to maintain detail integrity. *A key challenge, therefore, is to judiciously select the pruning and caching visual tokens across different phases in order to strike an well balance between inference latency and performance across the entire execution process.*

Based on the aforementioned opportunity, we propose SynerVLA, a training-free, plug-and-play inference acceleration framework that well accelerate VLA inference for on-device embodied AI by jointly exploiting spatial and temporal redundancies according to the execution phases. In particular, SynerVLA comprises the following three core components.

Fusion-based Token Selection. It selects key visual tokens based on spatiotemporal consistency. By fusing attention mechanisms from textual semantics and action generation, our approach filters the most critical visual information to enable real-time inference without compromising task performance, while deferring less essential data for subsequent optimization.

Phase-Aware Feedback Optimization. By integrating external execution states (e.g., end-effector velocity, target distance) and internal attention distributions, it adaptively determines the optimal token pruning-reuse ratio at each timestep based on real-time feedback, thereby achieving a dynamic balance between inference efficiency and motion quality across different execution stages.

Dual Cache Reuse. Employing the optimal pruning-reuse ratio, we further introduce a two-level caching mechanism to accelerate the dual-system inference pipeline. This involves reusing visual

token key-value pairs in System-2 VLM and caching conditional token key-value projections in System-1 diffusion Transformer based on a latent drift threshold.

To the best of our knowledge, SynerVLA is the *first* acceleration framework specifically designed for dual-system VLA models, judiciously exploiting different execution phases across the entire operation cycle of embodied AI. We comprehensively evaluate SynerVLA on robotic manipulation tasks across two simulation benchmarks, LIBERO[15] and CALVIN[16], using four dual-system VLA models: OpenHelix[4], RoboDual[3], Hume[7], and DexVLA[8]. Experiments show that SynerVLA achieves up to 1.93× inference speedup with negligible degradation in task success rate. Additionally, we deploy SynerVLA on Agilex PiPER-X robot (6-DoF), validating its practical feasibility in real-time control scenarios, where it maintains substantial acceleration under realistic operating conditions.

2 Related Work

Dual-System Vision-Language-Action Models. End-to-end VLA and generalist robot policies, such as RT-1, RT-2, RT-X, Octo, OpenVLA, and the π-series models, have shown strong promise in mapping visual observations and language instructions directly to robot actions [17–23]. However, sustaining high-frequency control with these monolithic models remains challenging due to the computational cost of large-scale visual-language reasoning and action generation. Dual-System VLA architectures have emerged as a practical alternative. For example, LCB [24] and HiRT[2] leverage slow VLMs for high-level semantic reasoning and delegate execution to fast, lightweight controllers, while RoboDual[3] explicitly decouples planning and execution by coupling a large VLA planner with a diffusion-based controller via compact latent exchanges. OpenHelix[4] systematically analyzes and modularizes dual-system VLA frameworks, demonstrating that well-matched slow–fast model pairs can jointly balance semantic expressiveness and execution efficiency. Nevertheless, Dual-System VLA methods are inherently computationally intensive. VLMs rely on continuous autoregressive generation[9] due to temporal inference over consecutive frames, while Diffusion Transformer[10] requires iterative multi-step denoising during action generation. Collectively, these computational demands constrain control frequency and pose significant challenges for real-time deployment.

Acceleration for Vision-Language-Action Models. These methods exploit redundancy within VLA pipelines and can be broadly categorized into pruning-based and key-value (KV) cache-based approaches. Pruning-based methods reduce computational cost by eliminating redundant structures during inference. Specifically, FastV[25] improves efficiency by adaptively pruning visual tokens across layers; VLA-Pruner[13] combines two-stage attention guidance for token pruning; Light-VLA[12] employs a differentiable strategy to retain task-relevant visual tokens. Although effective in reducing computation, such methods may discard critical visual information in fine-grained manipulation scenarios, resulting in degraded action accuracy. In contrast, key–value cache–based approaches exploit temporal consistency across consecutive frames. VLA-Cache[14] reuses KV caches for unchanged tokens to reduce redundant attention computation; however, cache reuse typically eliminates less than 25% of all FLOPs in VLM[26], leading to limited overall acceleration in latency-critical robotic applications.

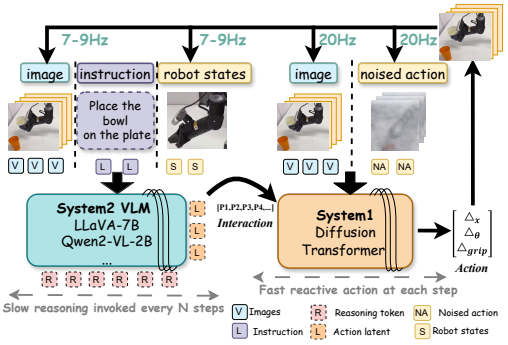
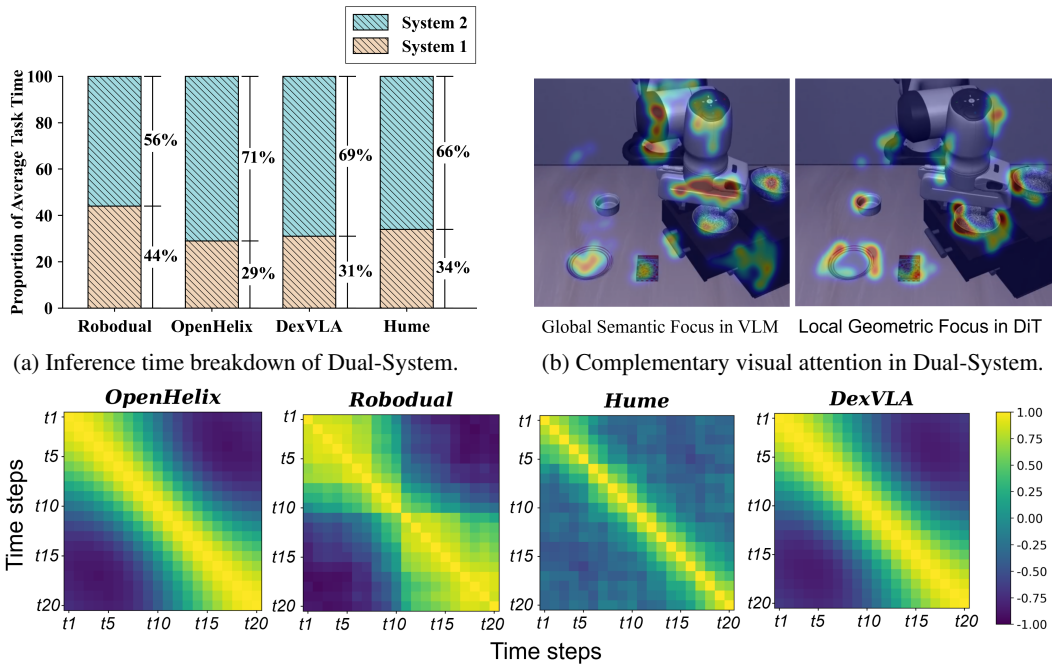


Figure 2: Dual-system VLA architecture.

3 Preliminary Analysis

3.1 Dual-System VLA inference

As shown in Figure 2, Dual-System VLA models decompose inference into a slow semantic system 2 and a fast action system 1[3–6]. At time step t, System 2 applies a Vision-Language Model to inputs (V^t, L^t). The language input L^t is tokenized as $E_T^t = \{\tau_1^t, \ldots, \tau_N^t\}$, and the vision encoder



(a) Inference time breakdown of Dual-System.

(b) Complementary visual attention in Dual-System.

(c) Temporal similarity of action latents across timesteps.

Figure 3: Motivating observations for dual-system VLA acceleration.

projects V^t into visual tokens $E_V^t = \{v_1^t, \ldots, v_M^t\}$. Conditioned on these inputs, the VLM produces a set of action latents:

$$Z^t = \{z_1^t, \ldots, z_P^t\} = f_{\text{VLM}}(V^t, L^t). \tag{1}$$

which encodes high-level task intent. Receiving the task intent, System 1 then employs a Diffusion Transformer conditioned on (Z^t, E_V^t). Starting from a noised action $a_0^t \sim \mathcal{N}(0, I)$, the DiT performs iterative denoising:

$$a_k^t = f_{\text{DiT}}\left(a_{k-1}^t \mid Z^t, E_V^t\right), \quad k = 1, \ldots, K. \tag{2}$$

Both subsystems repeatedly condition on visual inputs across timesteps, while the DiT performs multi-step denoising per step, resulting in substantial redundant computation.

3.2 Motivating Observation for SynerVLA

SynerVLA is motivated by a set of key observations derived from an in-depth analysis of existing dual-system VLA structures and embodied task characteristics.

Observation 1: *Unified optimization is necessary for Dual-System VLA inference.* To investigate the inference latency of dual-system VLAs, we conducted extensive multi-run analyses across four representative models from the LIBERO and CALVIN benchmarks. As shown in Figure 3a, the results reveal that latency bottlenecks are not limited to System 2: System 1 contributes 29%–44% of the total task latency. Consequently, optimizing System 2 alone is insufficient to enhance overall performance; effective improvements require jointly considering the two systems.

Observation 2: *The two subsystems focus on distinct types of visual information.* Figure 3b shows the attention patterns of the two sub systems, System 2 VLM primarily focuses on global visual cues that facilitate semantic understanding, whereas System 1 DiT emphasizes local geometric structures and edge information that are critical for action generation. Retaining only the semantically important visual tokens for the VLM leads to the loss of action-relevant visual information; in contrast, jointly accounting for the distinct visual regions attended by both subsystems enables more effective token selection and provides principled guidance for accelerating dual-system inference.

Observation 3: *Action Latents exchanged between the two systems also exhibit strong temporal similarity.* We extract intermediate action latents from four dual-system VLA inference pipelines at

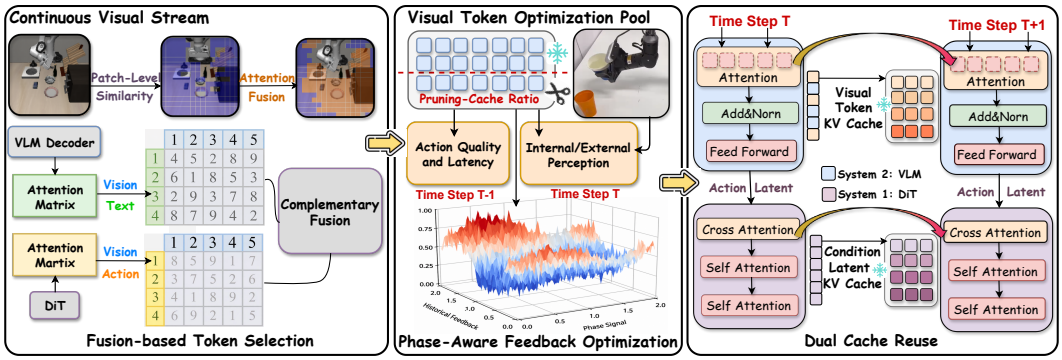


Figure 4: Overview of the SynerVLA framework.

consecutive time steps and map them into a common normalized latent representation to compute cosine similarity. As shown in Figure 3c, observe that the similarity between action latents in adjacent iterations reaches 0.8–0.9, indicating that System 1 repeatedly performs substantial redundant computation under highly similar conditions across successive steps. This further reveals that the temporal similarity inherent in embodied tasks offers new opportunities for unified acceleration of the dual-system VLA inference pipeline.

4 Methodology

4.1 Problem Formulation

The acceleration problem of Dual-System VLA is formulated as follows. Specifically, at each time step t, an RGB image I_t is encoded into a set of visual tokens $E_v^t = \{v_1^t, \ldots, v_M^t\}$. We first identify visual tokens that are eligible for optimization by excluding: (i) dynamically changing tokens detected via temporal similarity, and (ii) Text-Action-Attention tokens that must participate in the current inference step. The remaining tokens form an optimization pool: $E_{\text{pool}}^t \subset E_v^t, \quad |E_{\text{pool}}^t| = M_t$. Our goal is to partition E_{pool}^t into two sets E_{prune}^t and E_{cache}^t . The Tokens in E_{prune}^t are directly pruned, while those in E_{cache}^t have their token key–value representations cached and reused at I_{t+1} . We parameterize this partition by a pruning–reuse ratio $\alpha_t \in [0, 1]$, such that:

$$|E_{\text{prune}}^t| = \lfloor \alpha_t M_t \rfloor , \quad |E_{\text{cache}}^t| = M_t - \lfloor \alpha_t M_t \rfloor . \tag{3}$$

An aggressive pruning-reuse ratio reduces computation but may degrade action quality, while conservative pruning limits acceleration. Therefore, at each frame we determine α_t by solving:

$$\alpha_t^* = \arg \min_{\alpha \in [0,1]} J_t(\alpha) \tag{4}$$

where $J_t(\alpha)$ balances action quality and inference latency based on the current execution state and feedback from the previous timestep (Sec. 4.2). The optimized ratio α_t^* uniquely determines E_{prune}^t and E_{cache}^t enabling adaptive, frame-wise pruning and cache reuse for dual-system VLA inference. Figure 4 represents the overall workflow of SynerVLA which mainly consists of the following three key components: (i) Fusion-based Token Selection, (ii) Phase-Aware Feedback Optimization, and (iii) Dual Cache Reuse.

4.2 Fusion-based Token Selection

In embodied execution, consecutive visual observations exhibit limited changes, where most visual tokens remain static except for regions related to the manipulator or task-relevant objects. We exploit this property to identify a compact set of visual tokens that are critical for both semantic understanding and action generation, while deferring the remaining tokens to a subsequent optimization stage.

Patch-Level Temporal Similarity. Given two consecutive RGB frames I_t and I_{t-1}, we partition each image into a grid of visual patches according to the vision encoder's tokenization scheme. Let

$\mathbf{p}_i^t \in \mathbb{R}^d$denote the feature embedding of the i-th visual patch at time step t. Patch-wise cosine similarity between adjacent frames is defined as:

$$\mathrm{Sim}\big(\mathbf{p}_i^t, \mathbf{p}_i^{t-1}\big) = \frac{\mathbf{p}_i^t \cdot \mathbf{p}_i^{t-1}}{\|\mathbf{p}_i^t\|_2 \left\|\mathbf{p}_i^{t-1}\right\|_2}. \tag{5}$$

Patches with similarity below a predefined threshold are treated as dynamically changing and are always retained for the current inference step. Patches exhibiting high similarity are considered optimization candidates and are further evaluated by semantic-action relevance.

Task Relevance and Action Relevance. To assess the importance of visual tokens, we leverage two complementary sources of cross-modal attention: The Text–Vision cross-attention from the System-2 VLM decoder, which reflects the relevance of visual tokens to high-level semantic understanding, and the Action–Vision cross-attention from the System-1 Diffusion Transformer (DiT), which captures their contribution to low-level action generation. Let $\mathcal{T} = \{t_1, \ldots, t_{N_T}\}$ denote the set of text tokens describing the task, and let $\mathcal{A} = \{a_1, \ldots, a_{N_A}\}$ denote the action tokens in the DiT. Let $A_{\mathrm{TV}}^{h,l}(v_i, t_j)$ and $A_{\mathrm{AV}}^{h,l}(v_i, a_k)$ denote the corresponding cross-attention weights at head h and layer l. We use the last two cross-attention layers of each subsystem as representative layers, denoted $\mathcal{L}_{\mathrm{TV}}$ and $\mathcal{L}_{\mathrm{AV}}$. The Text–Vision and Action-Vision attention score for v_i is computed as:

$$S_{\mathrm{TV}}(v_i) = \frac{1}{|\mathcal{L}_{\mathrm{TV}}| H N_T} \sum_{l \in \mathcal{L}_{\mathrm{TV}}} \sum_{h=1}^{H} \sum_{j=1}^{N_T} A_{\mathrm{TV}}^{h,l}(v_i, t_j), \tag{6}$$

$$S_{\mathrm{AV}}(v_i) = \frac{1}{|\mathcal{L}_{\mathrm{AV}}| H N_A} \sum_{l \in \mathcal{L}_{\mathrm{AV}}} \sum_{h=1}^{H} \sum_{k=1}^{N_A} A_{\mathrm{AV}}^{h,l}(v_i, a_k). \tag{7}$$

Complementary Fusion and Token Selection. The two attention scores encode complementary information: Text–Vision attention emphasizes semantic relevance, while Action–Vision attention highlights control-critical visual cues. We fuse them using a complementary fusion strategy:

$$S_{\mathrm{TA}}(v_i) = \lambda \cdot S_{\mathrm{TV}}(v_i) + (1 - \lambda) \cdot S_{\mathrm{AV}}(v_i), \tag{8}$$

Based on the fused score S_{TA}, we select the top-K visual tokens as Text-Action-important tokens, which are guaranteed to participate in the current inference step. The remaining static visual tokens are placed into an optimization pool for pruning or reuse decisions. This design preserves task-critical semantic and action-relevant information while enabling flexible downstream acceleration.

4.3 Phase-Aware Feedback Optimization

Given the optimization pool of visual tokens obtained in Section 4.2, our goal is to dynamically determine the optimal pruning–reuse ratio $\alpha_t \in [0, 1]$ at each time step t. Intuitively, an overly aggressive pruning strategy may reduce computation but degrade action quality, while conservative reuse limits acceleration benefits. We therefore formulate the selection of α_t as a per-step online optimization problem that adapts to the external execution state and the internal inference dynamics.

External Phase Perception. We first characterize the external execution phase of the robot at time step t using two task-relevant signals: the end-effector velocity and the Euclidean distance to the target object. Let $\mathbf{p}^t \in \mathbb{R}^3$ denote the end-effector position, and $\mathbf{g} \in \mathbb{R}^3$ denote the target position. Phase signal are defined as:

$$v_t = \frac{\|\mathbf{p}_t - \mathbf{p}_{t-1}\|_2}{\Delta t}, \quad d_t = \|\mathbf{p}_t - \mathbf{g}\|_2 . \tag{9}$$

These signals reflect the current execution phase: large v_t and large d_t typically correspond to coarse motion phases, while small v_t and small d_t indicate fine-grained manipulation phases as demonstrated in Figure 1(b).

Internal Attention Aggregation. In addition to external signals, we capture the internal inference state by measuring the aggregation degree of cross-modal attention over the optimization pool. Let ν_t^{pool} denote the set of visual tokens eligible for pruning or reuse at time step t and let $S_{\mathrm{TA}}(v_i)$ be

the fused Text–Action attention score defined in Eq. (8). We quantify attention concentration using a normalized entropy metric:

$$\mathcal{H}_t = -\sum_{v_i \in \mathcal{V}_t^{\text{pool}}} \tilde{S}_t(v_i) \log \tilde{S}_t(v_i), \quad \tilde{S}_t(v_i) = \frac{S_{\text{TA}}(v_i)}{\sum_{v_j \in \mathcal{V}_t^{\text{pool}}} S_{\text{TA}}(v_j)}. \tag{10}$$

Low entropy indicates that attention is highly concentrated on a small subset of tokens, suggesting strong redundancy among the remaining tokens and permitting more aggressive pruning. Conversely, high entropy implies dispersed attention and necessitates conservative pruning.

Action Quality and Latency Feedback. To close the feedback loop, we incorporate the outcomes of the previous inference step. Let ℓ_{t-1} denote the measured inference latency at time $t-1$. We define a action quality metric based on effective motion. Let $\Delta\mathbf{p}_{t-1}$ be the executed displacement and $\Delta\mathbf{p}_{t-1}^*$ is the expected displacement obtained by solving a short-horizon trajectory prediction problem under a classical minimum-jerk formulation [27]. The action quality score is defined as:

$$q_{t-1} = \min\left\{1, \max\left\{0, \frac{\Delta\mathbf{p}_{t-1}^\top \Delta\mathbf{p}_{t-1}^*}{\|\Delta\mathbf{p}_{t-1}^*\|_2^2 + \epsilon}\right\}\right\}. \tag{11}$$

where ϵ is a small constant for numerical stability. This score reflects whether the generated action leads to effective progress toward the task objective.

Optimization Objective. We formulate the selection of the pruning–reuse ratio as a one-dimensional closed-form quadratic optimization problem that balances action quality and inference latency:

$$J_t(\alpha) = w_q(t)\alpha^2 + w_\ell(t)(1-\alpha)^2, \quad \alpha \in [0,1]. \tag{12}$$

The first term penalizes aggressive pruning in quality-sensitive phases, while the second term penalizes overly conservative pruning when latency reduction is needed. The weights $w_q(t)$ and $w_\ell(t)$ are dynamically adjusted based on the current phase and previous-step feedback:

$$r_t = \frac{1}{3}\left(\bar{v}_t + \bar{d}_t + 1 - \bar{H}_t\right), \quad w_q(t) = (1-r_t) + (1-q_{t-1}), \quad w_\ell(t) = r_t \bar{\ell}_{t-1}. \tag{13}$$

where $\bar{v}_t$, $\bar{d}_t$, $\bar{H}_t$, and $\bar{\ell}_{t-1}$ denote the normalized end-effector velocity, target distance, attention entropy, and latency pressure, respectively. The score r_t estimates the degree of coarse-phase execution. Thus, $w_q(t)$ increases in fine-grained manipulation phase or low previous action quality, while $w_\ell(t)$ increases when the system is in a rapid approaching phase with high latency pressure. Since Eq. (12) is convex in α, the optimal pruning–reuse ratio has the following closed-form solution:

$$\alpha_t^* = \text{clip}\left(\frac{w_\ell(t)}{w_q(t) + w_\ell(t) + \epsilon}, 0, 1\right). \tag{14}$$

Online Feedback Update. Finally, the selected α_t^* is applied to the optimization pool at time t, which determines the proportion of tokens to be pruned or reused. The resulting action quality q_t and latency ℓ_t are then observed and fed back into the next-step optimization at time $t+1$. This online feedback update enables SynerVLA to continuously adapt the pruning–reuse ratio across different execution phases without requiring offline training or task-specific tuning. The complete algorithm is provided in Appendix A.1.

4.4 Dual Cache Reuse

Given the optimal pruning–reuse ratio α_t^* (Sec. 4.3), the visual token set is partitioned into E_{prune}^t and E_{cache}^t. For tokens $v_i \in E_{\text{cache}}^t$, we reuse their key-value (KV) representations in the System 2 VLM across consecutive frames, while the remaining tokens are recomputed. We further exploit temporal coherence of the action latents Z_t in System 1 by caching the key-value projections of condition tokens, thereby reducing redundant cross-attention computation. For each layer l, the visual and condition KV caches are maintained as:

$$\mathcal{C}_v^{(l)}(t) = \left\{K_v^{(l)}(t),\, V_v^{(l)}(t)\right\}, \qquad \mathcal{C}_c^{(l)}(t) = \left\{K_c^{(l)}(t),\, V_c^{(l)}(t)\right\}. \tag{15}$$

Condition reuse is controlled by the drift:

$$\Delta_c(t) = \|\Phi(Z_t) - \Phi(Z_{t-1})\|_2 \tag{16}$$

Table 1: **Comprehensive Comparison on CALVIN and LIBERO Benchmarks.** **Red** highlights the best efficiency result per block.

Method (L)	CALVIN BENCHMARK								Method (R)	LIBERO BENCHMARK							
	L=1	L=2	L=3	L=4	L=5	Avg	Len	SpeedUp		Spat.	Obj.	Goal	Long	Avg	TFLOPS	Lat.(ms)	SpeedUp
OpenHelix [4]									***DexVLA*** [8]								
Baseline	92.7	80.8	66.9	55.7	45.2	68.3	3.41	1.00×	Baseline	95.6	98.3	95.2	94.1	95.8	2.125	51.6ms	1.00×
+ FastV [25]	91.5	78.2	66.1	52.5	43.2	66.3	3.32	1.26×	+ FastV [25]	94.5	96.9	94.5	92.8	94.7	1.813	42.7ms	1.21×
+ VLA-Cache [14]	92.8	80.5	66.8	54.2	44.5	67.8	3.39	1.36×	+ VLA-Cache [14]	95.2	97.5	94.8	93.6	95.3	1.647	39.3ms	1.31×
+ Efficient-VLA [28]	91.8	79.0	65.9	53.1	44.0	66.8	3.34	1.71×	+ Efficient-VLA [28]	94.7	97.2	94.3	93.2	94.9	1.347	31.2ms	1.65×
+ SP-VLA [29]	87.3	76.1	61.8	50.6	42.1	63.6	3.18	1.65×	+ SP-VLA [29]	92.1	95.9	92.6	91.4	93.0	1.379	31.9ms	1.62×
+ Ours	92.5	80.2	66.5	54.0	44.2	67.5	3.37	**1.85×**	**+ Ours**	95.0	97.4	94.6	93.5	95.1	**1.205**	**29.2ms**	**1.77×**
RoboDual [3]									***Hume*** [7]								
Baseline	94.4	82.7	72.1	62.4	54.4	73.2	3.66	1.00×	Baseline	97.8	98.3	98.1	96.0	97.6	1.982	55.8ms	1.00×
+ FastV [25]	91.8	80.9	71.3	58.8	52.2	71.0	3.55	1.23×	+ FastV [25]	96.7	97.1	97.2	94.9	96.5	1.654	45.3ms	1.23×
+ VLA-Cache [14]	94.3	82.5	72.1	62.3	54.1	73.1	3.65	1.35×	+ VLA-Cache [14]	97.3	97.6	97.8	95.6	97.1	1.573	41.6ms	1.34×
+ Efficient-VLA [28]	92.6	80.3	72.9	58.4	52.9	71.4	3.57	1.67×	+ Efficient-VLA [28]	96.9	97.5	97.4	95.2	96.8	1.251	34.2ms	1.63×
+ SP-VLA [29]	90.2	80.1	69.3	55.7	50.8	69.2	3.46	1.62×	+ SP-VLA [29]	94.1	95.8	95.3	92.7	94.5	1.218	33.4ms	1.67×
+ Ours	94.2	81.9	71.7	61.8	54.2	72.8	3.64	**1.93×**	**+ Ours**	97.0	97.8	97.5	95.7	97.0	**1.133**	**31.9ms**	**1.75×**

Here, $\Phi(\cdot)$ denotes a training-free latent summarization operator. Given action latents $Z_t = \{z_1^t, \ldots, z_P^t\}$, we compute $\Phi(Z_t) = \mathrm{Norm}\left(\frac{1}{P}\sum_{k=1}^{P} z_k^t\right)$, where $\mathrm{Norm}(\cdot)$ denotes ℓ_2 normalization. The drift $\Delta c(t)$ therefore measures the normalized change of the action-conditioning representation between two consecutive steps. The reuse rule is defined as follows:

$$\mathcal{C}_c^{(l)}(t) = \begin{cases} \mathcal{C}_c^{(l)}(t-1), & \Delta_c(t) < \epsilon_c, \\ \mathrm{Reuse}\left(\mathcal{C}_c^{(l)}(t)\right), & \text{otherwise.} \end{cases} \tag{17}$$

Only condition KV projections are cached, while queries are always recomputed. Together, visual and condition KV reuse enable unified acceleration of the dual-system.

5 Experiments

5.1 Experimental Setup

To validate the effectiveness of SynerVLA, we conduct comprehensive evaluations in both simulation environments and real-world robotic settings. In simulation, we evaluate SynerVLA on four open-source VLA models-OpenHelix[4], DexVLA[8], Hume[7], and RoboDual[3]-using the CALVIN[16] and LIBERO[15] benchmarks, and compare it against four state-of-the-art VLA acceleration methods: FastV[25], VLA-Cache[14], Efficient-VLA[28], and SP-VLA[29] under the same experimental configurations. All experiments are conducted on an NVIDIA RTX 4090 GPU. In real-world settings, we present both quantitative and qualitative evaluations of SynerVLA on real-world manipulation tasks, including single-arm control across different robot configurations.

5.2 Main Results

Table 1 reports the main results on CALVIN and LIBERO. SynerVLA consistently improves inference efficiency across all four dual-system VLA backbones while maintaining competitive task performance, demonstrating a favorable speed–accuracy trade-off under both long-horizon and goal-conditioned manipulation settings. These results indicate that the proposed joint optimization of Fusion-based token selection, phase-aware Feedback Optimization, and dual cache reuse generalizes across different model architectures and task distributions.

On CALVIN, SynerVLA achieves the highest speedups on both OpenHelix and RoboDual, reaching 1.85× and 1.93×, respectively, while keeping the average success rate close to the baseline with less than 2 percentage-point degradation. Compared with pruning-dominated methods such as SP-VLA, which suffer larger degradation at longer horizons, SynerVLA preserves more stable performance across increasing chain lengths. This indicates that phase-aware pruning–reuse better supports long-horizon execution by adapting to changing task phases than static or aggressive token removal.

On LIBERO, SynerVLA delivers the lowest latency and computational cost across DexVLA and Hume, reducing latency to 29.2 ms and 31.9 ms with 1.77× and 1.75× speedups, respectively, while maintaining high average success rates of 95.1% and 97.0%. Compared with VLA-Cache and Efficient-VLA, SynerVLA further lowers TFLOPs and latency without noticeable performance loss. This confirms that jointly exploiting token pruning and cross-frame cache reuse across the semantic and action subsystems is more effective than applying either optimization alone.

Overall, the results show that SynerVLA achieves consistent acceleration across different benchmarks, model backbones, and task horizons. Unlike generic acceleration methods that often trade accuracy for speed, SynerVLA maintains task success while substantially reducing inference cost, validating its suitability for real-time embodied control. A more comprehensive performance analysis across LIBERO sub-tasks is provided in Appendix B.2.

5.3 Real-World Experiments

To validate deployment feasibility on physical hardware, we evaluate SynerVLA on an AgileX PiPER robotic arm across four manipulation tasks: *Pick*, *Grasp*, *Select*, and *Pour*. Table 2 reports the success rate, computational cost, and end-to-end latency.

Table 2: **Performance on real robot tasks.**

Method	Success Rate (%, ↑)					FLOPs (T)	Lat. (ms)
	Pick	Grasp	Select	Pour	Avg.		
RoboDual	89.0	83.3	92.5	69.7	83.6	1.814	64.16
+ VLA-cache	88.2	**82.4**	**92.3**	**68.6**	**82.9**	1.535	46.53
+ Efficient-VLA	87.1	79.2	91.1	65.3	80.7	1.239	38.42
+ SP-VLA	86.5	76.6	88.7	64.2	79	1.256	39.61
+ **Ours**	**89.7**	79.8	91.9	65.1	81.6	**1.028**	**33.77**

SynerVLA achieves the lowest computational cost and latency among all methods, reducing FLOPs from 1.814T to 1.028T and latency from 64.16 ms to 33.77 ms, corresponding to a 1.90× speedup over RoboDual. Compared with VLA-Cache, Efficient-VLA, and SP-VLA, SynerVLA further lowers latency while maintaining a competitive average success rate of 81.6%. Although the average success rate drops by 2.0 percentage points compared with the baseline, the substantial latency reduction enables more responsive real-world control. These results demonstrate that SynerVLA effectively reduces inference cost on physical robotic platforms while preserving practical manipulation performance, as shown in Figure 10. Table 8 reports the speedup on the JAKA S5, and Figure 5 shows a pick-and-place rollout on the Franka robot.

5.4 Ablation Study

System-Level Analysis. To isolate system component, we compare SynerVLA with two single-system variants. As shown in Table 3, optimizing System 2 alone achieves a 1.53× speedup with a 72.0% average success, while optimizing System 1 alone yields a 1.39× speedup with 72.7%. This shows that visual token pruning–reuse provides larger acceleration, whereas condition KV reuse is more conservative.

Table 3: **Ablation Study on Cache Strategy.** Impact of disabling Visual KV and Action Latents.

Method	Success Rate (%, ↑)						Speedup
	L=1	L=2	L=3	L=4	L=5	Avg.	
Only SYS2	93.6	81.7	70.9	61.2	52.4	72.0	1.53×
Only SYS1	94.2	82.2	71.8	61.6	53.5	72.7	1.39×
SynerVLA	**94.2**	**81.9**	**71.7**	**61.8**	**54.2**	**72.8**	**1.93×**

Combining both components, SynerVLA achieves the largest speedup of 1.93× while maintaining a competitive 72.8% average success rate, confirming that coordinated dual-cache reuse provides stronger end-to-end acceleration than optimizing either subsystem alone.

Effect of Pruning–Reuse Ratio α. Table 4 compares SynerVLA with fixed pruning ratios. Static policies show a clear accuracy–efficiency trade-off: larger α improves speedup up to 2.49× but sharply reduces the average success rate to 23.7%. Even moderate pruning (α=50%) drops the average success rate to 53.1%, indicating poor robustness to varying execution phases.

By dynamically adjusting α, SynerVLA achieves a stronger balance, reaching a 1.93× speedup while maintaining a 72.8% average suc-

Table 4: **Comparison of static pruning ratios and dynamic optimization.**

Pruning Strategy	Success Rate (%, ↑)						Speedup
	L=1	L=2	L=3	L=4	L=5	Avg.	
Baseline	94.4	82.7	72.1	62.4	54.4	73.2	1.00×
Static 15%	92.8	**82.3**	71.2	**62.3**	**54.4**	72.6	1.12×
Static 35%	87.5	76.4	63.5	53.7	43.6	64.9	1.39×
Static 50%	71.2	63.5	56.8	42.3	31.5	53.1	1.73×
Static 75%	56.3	49.1	40.9	29.2	20.7	39.2	1.97×
Static 90%	40.9	32.1	23.3	16.9	5.2	23.7	**2.49×**
SynerVLA	**94.2**	81.9	**71.7**	61.8	54.2	**72.8**	1.93×

Table 5: Effect of target-distance feedback on RoboDual/CALVIN. Only the scheduler's distance input is varied. Unaccelerated RoboDual achieves 73.2% average success at 1.00× speedup.

Distance setting	**Avg. success (%) ↑**	**Speedup ↑**	$\bar{\alpha}$ **(%)**
Full signals (simulator target position)	**72.8**	**1.93×**	47.8
10% contiguous dropout	72.0	1.92×	48.0
20% contiguous dropout	70.6	1.94×	48.5
Dropout during final approach/manipulation	68.9	1.93×	49.1
Last valid distance retained	68.1	1.95×	50.3
No target position (w/o d_t)	**66.7**	**1.95×**	49.7

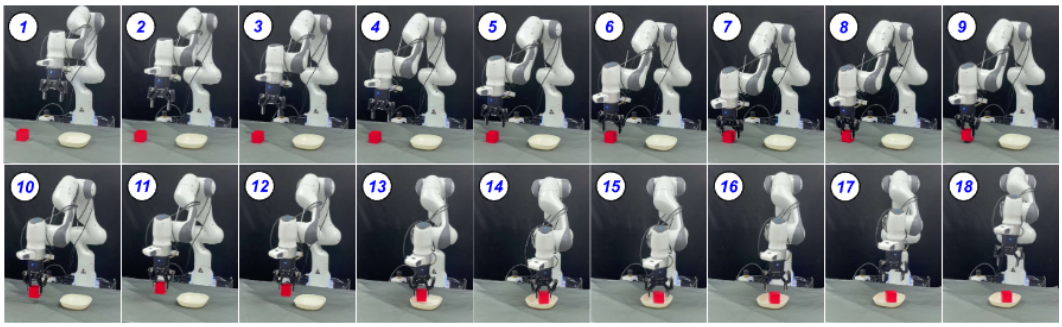


Figure 5: Real-world pick-and-place rollout on a Franka robot.

cess rate. This confirms that phase-aware pruning–reuse better adapts to changing visual redundancy and control requirements than fixed pruning strategies.

Target-Distance Feedback. We next examine how this adaptation depends on target distance. The scheduler computes the distance between the end-effector position, obtained via forward kinematics, and the target position. In CALVIN and LIBERO, it queries the simulator for the position of the relevant object or destination. These privileged coordinates are available only to SynerVLA's scheduler. On the real robot, we estimate the target position by back-projecting depth pixels from calibrated fixed and wrist-mounted RGB-D cameras into the robot base frame. On RoboDual/CALVIN, removing target distance reduces average success from 72.8% to 66.7%, a drop of 6.1 percentage points, while speedup remains similar (1.93× with distance and 1.95× without it; Table 5). The remaining signals thus sustain acceleration, but distance feedback helps the scheduler preserve task success during precision-sensitive motion.

6 Conclusion

This paper presents SynerVLA, a plug-and-play acceleration framework for dual-system VLA models on resource-constrained edge devices. Motivated by execution-phase characteristics in embodied tasks, SynerVLA jointly exploits spatial and temporal redundancy in continuous visual streams. It selects key visual tokens through text-action fusion, adaptively determines the pruning–reuse ratio via phase-aware feedback, and accelerates both the System-2 VLM and System-1 diffusion Transformer through dual cache reuse. Experiments across representative dual-system VLA models and robotic benchmarks show that SynerVLA achieves up to 1.93× speedup and 26% higher control frequency with negligible task-success degradation, demonstrating its effectiveness for real-time embodied AI.

Acknowledgments and Disclosures

This work was supported in part by the Science and Technology Development Fund of Macao (Grant Nos. 0107/2024/RIA2 and 0061/2025/RIB2), the Multi-Year Research Grants of the University of Macau (Grant Nos. MYRG-GRG2023-00211-IOTSC-UMDF and MYRG-GRG2024-00180-IOTSC), the 2024 Nansha District Science and Technology Program, Guangzhou, China (Grant No. 2024ZD005), and Huawei (No. TC20260605012).

Competing Interests. The authors declare no competing interests.

References

[1] Daniel Kahneman. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

[2] Jianke Zhang, Yanjiang Guo, Xiaoyu Chen, Yen-Jen Wang, Yucheng Hu, Chengming Shi, and Jianyu Chen. HiRT: Enhancing robotic control with hierarchical robot transformers. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 933–946, 2024.

[3] Qingwen Bu, Hongyang Li, Li Chen, Jisong Cai, Jia Zeng, Heming Cui, Maoqing Yao, and Yu Qiao. Towards synergistic, generalized, and efficient dual-system for robotic manipulation, 2025.

[4] Can Cui, Pengxiang Ding, Wenxuan Song, Shuanghao Bai, Xinyang Tong, Zirui Ge, Runze Suo, Wanqi Zhou, Yang Liu, Bofang Jia, Han Zhao, Siteng Huang, and Donglin Wang. Openhelix: A short survey, empirical analysis, and open-source dual-system vla model for robotic manipulation, 2025.

[5] Hao Chen, Jiaming Liu, Chenyang Gu, Zhuoyang Liu, Renrui Zhang, Xiaoqi Li, Xiao He, Yandong Guo, Chi-Wing Fu, Shanghang Zhang, and Pheng-Ann Heng. Fast-in-slow: A dual-system VLA model unifying fast manipulation within slow reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[6] Figure AI. Helix: A vision-language-action model for generalist humanoid control, 2025. Accessed May 7, 2025.

[7] Haoming Song, Delin Qu, Yuanqi Yao, Qizhi Chen, Qi Lv, Yiwen Tang, Modi Shi, Guanghui Ren, Maoqing Yao, Bin Zhao, Dong Wang, and Xuelong Li. Hume: Introducing system-2 thinking in visual-language-action model, 2025.

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

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[10] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4195–4205, October 2023.

[11] Chenyang Li, Jieyuan Liu, Bin Li, Bo Gao, Yilin Yuan, Yangfan He, Yuchen Li, and Jingqun Tang. Dtp: A simple yet effective distracting token pruning framework for vision-language action models, 2026.

[12] 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, 2025.

[13] 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, 2025.

[14] Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-cache: Efficient vision-language-action manipulation via adaptive token caching. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[15] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning, 2023.

[16] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2022.

[17] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, and Brian Ichter. Rt-1: Robotics transformer for real-world control at scale, 2023.

[18] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, and Pete Florence. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[19] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic learning datasets and RT-X models, 2023.

[20] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[21] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model, 2024.

[22] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024.

[23] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 2025.

[24] Yide Shentu, Philipp Wu, Aravind Rajeswaran, and Pieter Abbeel. From LLMs to actions: Latent codes as bridges in hierarchical robot control. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8539–8546, 2024. doi: 10.1109/IROS58592.2024.10801683.

[25] 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, 2024.

[26] Hanzhen Wang, Jiaming Xu, Jiayi Pan, Yongkang Zhou, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning, 2025.

[27] Abdel-Nasser Sharkawy. Minimum jerk trajectory generation for straight and curved movements: Mathematical analysis, 2021.

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

[29] 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, 2025.

A Method Details

A.1 Algorithm Details

Algorithm 1: Phase-Aware Feedback Optimization

Input: End-effector positions p_t, p_{t-1}, target g, time interval Δt; token pool $\mathcal{V}_{\text{pool}}^t$ with fused scores $\text{STA}(v_i)$; previous-step feedback $(\ell_{t-1}, \Delta p_{t-1}, \Delta p_{t-1}^*)$; normalization constants $(v_{\max}, d_{\max}, \tau)$ and ϵ.

Output: Optimal ratio α_t^* and token partitions $\mathcal{E}_{\text{prune}}^t, \mathcal{E}_{\text{cache}}^t$.

$M_t \leftarrow |\mathcal{V}_{\text{pool}}^t|$;

if $M_t = 0$ **then**

  $\alpha_t^* \leftarrow 0$; $\mathcal{E}_{\text{prune}}^t, \mathcal{E}_{\text{cache}}^t \leftarrow \emptyset$;

  return $\alpha_t^*, \mathcal{E}_{\text{prune}}^t, \mathcal{E}_{\text{cache}}^t$;

`// External phase perception`

$v_t \leftarrow \|p_t - p_{t-1}\|_2 / \Delta t$; $d_t \leftarrow \|p_t - g\|_2$;

$\bar{v}_t \leftarrow \text{clip}(v_t / (v_{\max} + \epsilon), 0, 1)$;

$\bar{d}_t \leftarrow \text{clip}(d_t / (d_{\max} + \epsilon), 0, 1)$;

`// Internal attention aggregation`

$A_t \leftarrow \sum_{v_i \in \mathcal{V}_{\text{pool}}^t} \text{STA}(v_i) + \epsilon$;

foreach $v_i \in \mathcal{V}_{\text{pool}}^t$ **do**

  $\tilde{S}_t(v_i) \leftarrow \text{STA}(v_i) / A_t$;

$H_t \leftarrow -\sum_{v_i \in \mathcal{V}_{\text{pool}}^t} \tilde{S}_t(v_i) \log(\tilde{S}_t(v_i) + \epsilon)$;

if $M_t > 1$ **then**

  $\bar{H}_t \leftarrow H_t / \log M_t$;

else

  $\bar{H}_t \leftarrow 0$;

`// Previous-step quality and latency feedback`

$q_{t-1} \leftarrow \text{clip}\left(\frac{\Delta p_{t-1}^\top \Delta p_{t-1}^*}{\|\Delta p_{t-1}^*\|_2^2 + \epsilon}, 0, 1\right)$;

$\bar{\ell}_{t-1} \leftarrow \text{clip}(\ell_{t-1} / (\tau + \epsilon), 0, 1)$;

`// Closed-form pruning-reuse optimization`

$r_t \leftarrow (\bar{v}_t + \bar{d}_t + 1 - \bar{H}_t)/3$;

$w_q(t) \leftarrow (1 - r_t) + (1 - q_{t-1})$;

$w_\ell(t) \leftarrow r_t \bar{\ell}_{t-1}$;

$\alpha_t^* \leftarrow \text{clip}\left(\frac{w_\ell(t)}{w_q(t) + w_\ell(t) + \epsilon}, 0, 1\right)$;

`// Token partitioning`

$M_t^{\text{prune}} \leftarrow \lfloor \alpha_t^* M_t \rfloor$;

Sort $\mathcal{V}_{\text{pool}}^t$ by $\text{STA}(v_i)$ in ascending order;

$\mathcal{E}_{\text{prune}}^t \leftarrow$ first M_t^{prune} tokens;

$\mathcal{E}_{\text{cache}}^t \leftarrow \mathcal{V}_{\text{pool}}^t \setminus \mathcal{E}_{\text{prune}}^t$;

return $\alpha_t^*, \mathcal{E}_{\text{prune}}^t, \mathcal{E}_{\text{cache}}^t$;

Algorithm 2: SynerVLA Inference Workflow

Input: Visual stream $\{I_t\}$, instruction L, models f_{VLM} and f_{DiT}; similarity threshold τ_s, fusion weight λ, protected-token budget K; VLM refresh interval N, cache threshold ϵ_c; robot state $\mathbf{x}_t$ and previous feedback $\mathbf{f}_{t-1}$.

Output: Action sequence $\{a_t\}$ and adaptive ratios $\{\alpha_t^*\}$.

Initialize $\mathcal{C}^V, \mathcal{C}^Z \leftarrow \emptyset$;
Initialize the previous-step feedback $\mathbf{f}_0$;
for $t = 1, 2, \ldots$ **do**
  $\mathcal{V}^t \leftarrow \mathrm{EncodeVisual}(I_t)$;
  `// Fusion-based token selection`
  if $t = 1$ **then**
    $\alpha_t^* \leftarrow 0$;
    $\mathcal{E}_{\mathrm{prune}}^t, \mathcal{E}_{\mathrm{cache}}^t \leftarrow \emptyset$;
  else
    $\mathcal{V}_{\mathrm{static}}^t \leftarrow \{v_i^t : \mathrm{Sim}(p_i^t, p_i^{t-1}) \geq \tau_s\}$;
    Obtain available S_{TV} and S_{AV} attention scores;
    $\mathrm{STA}_t(v_i) \leftarrow \lambda S_{\mathrm{TV}}(v_i) + (1-\lambda) S_{\mathrm{AV}}(v_i)$;
    $\mathcal{V}_{\mathrm{key}}^t \leftarrow \mathrm{TopK}_K(\mathcal{V}^t, \mathrm{STA}_t)$;
    $\mathcal{V}_{\mathrm{pool}}^t \leftarrow \mathcal{V}_{\mathrm{static}}^t \setminus \mathcal{V}_{\mathrm{key}}^t$;
    `// Phase-aware feedback optimization`
    $(\alpha_t^*, \mathcal{E}_{\mathrm{prune}}^t, \mathcal{E}_{\mathrm{cache}}^t) \leftarrow \mathcal{O}(\mathbf{x}_t, \mathbf{f}_{t-1}, \mathcal{V}_{\mathrm{pool}}^t, \mathrm{STA}_t)$;
  $\mathcal{V}_{\mathrm{fresh}}^t \leftarrow \mathcal{V}^t \setminus (\mathcal{E}_{\mathrm{prune}}^t \cup \mathcal{E}_{\mathrm{cache}}^t)$;
  `// System-2 visual KV reuse`
  if $t = 1$ ***or*** $(t-1) \bmod N = 0$ **then**
    $KV_t^V \leftarrow \mathrm{VisualKV}(\mathcal{V}_{\mathrm{fresh}}^t, \mathcal{E}_{\mathrm{cache}}^t, \mathcal{C}^V)$;
    $Z_t \leftarrow f_{\mathrm{VLM}}(I_t, L; KV_t^V)$;
    $\mathcal{C}^V \leftarrow \mathrm{Update}(\mathcal{C}^V, KV_t^V)$;
  else
    $Z_t \leftarrow Z_{t-1}$;
  `// System-1 condition KV reuse`
  if $\mathcal{C}^Z = \emptyset$ ***or*** $\|\Phi(Z_t) - \Phi(Z_{\mathrm{ref}})\|_2 \geq \epsilon_c$ **then**
    $KV_t^Z \leftarrow \mathrm{ProjectKV}(Z_t)$;
    $\mathcal{C}^Z \leftarrow KV_t^Z$;
    $Z_{\mathrm{ref}} \leftarrow Z_t$;
  else
    $KV_t^Z \leftarrow \mathcal{C}^Z$;
  $a_t \leftarrow f_{\mathrm{DiT}}(I_t, Z_t; KV_t^Z)$;
  Execute a_t and record inference latency ℓ_t;
  Observe Δp_t and estimate Δp_t^* for the next feedback step;
  $\mathbf{f}_t \leftarrow (\ell_t, \Delta p_t, \Delta p_t^*)$;
return $\{a_t\}, \{\alpha_t^*\}$;

At each timestep, SynerVLA identifies candidate visual tokens through temporal similarity and fused text–action attention, then dynamically determines the pruning–reuse ratio using phase-aware feedback. The resulting token partitions guide visual KV reuse in the VLM, while condition KV reuse in the DiT is independently controlled by latent drift. Execution feedback is incorporated into subsequent optimization steps.

A.2 Complexity Analysis

Patch Similarity Computation To identify visually static regions across consecutive frames, SynerVLA computes patch-wise similarity between aligned visual tokens. Let N_{patch} denote the number of image patches and D_{patch}. the patch embedding dimension. The computational cost is

$$\mathrm{FLOPs}_{\text{static-sim}} = N_{\text{patch}}^2 D_{\text{patch}} \approx H^2, \tag{18}$$

where H is the spatial resolution of the input feature map.

Complementary Token Importance Scoring For each token in the optimization pool, SynerVLA computes complementary importance scores by fusing text–vision attention and action–vision attention.

Assuming L_v visual tokens,L_t text tokens, and hidden dimension D, the cost of attention score extraction is

$$\mathrm{FLOPs}_{\text{fusion}} \approx L_t L_v D + L_a L_v D, \tag{19}$$

where L_a denotes the number of action latent tokens. The subsequent linear fusion introduces only negligible additional cost.

Phase-Aware Entropy and Feedback Computation To capture the concentration of fused attention scores, SynerVLA computes an entropy statistic over the token pool:

$$\mathrm{FLOPs}_{\text{entropy}} \approx L_v D. \tag{20}$$

In addition, phase-aware feedback signals—including end-effector velocity, distance-to-goal, and action-quality estimation—incur only constant-time overhead per step and do not scale with token length.

Online Pruning–Reuse Ratio Optimization At each timestep, SynerVLA computes the pruning–reuse ratio α_t^* using the closed-form solution in Eq. (14). This computation only involves scalar normalization, weight estimation, and a constant number of arithmetic operations. Therefore, its computational overhead is independent of model size and token length:

$$\mathrm{FLOPs}_{\alpha} = O(1). \tag{21}$$

Dual Cache Reuse Overhead SynerVLA performs cache reuse for both visual token KV pairs in the VLM and condition projection KV pairs in the DiT. Cache lookup and reuse operations scale linearly with the number of reused tokens:

$$\mathrm{FLOPs}_{\text{cache}} \approx L_{\text{reuse}} D, \tag{22}$$

which is significantly smaller than recomputing full attention and projection layers.

Final FLOP Reduction Let Ω denote the number of Transformer layers. The total FLOP reduction achieved by SynerVLA across all layers is

$$\Delta\mathrm{FLOPs}_{\text{total}} = \sum_{l=1}^{\Omega} \Delta\mathrm{FLOPs}_{\text{layer}}^{(l)}, \tag{23}$$

Overall, the additional overhead introduced by SynerVLA remains modest relative to the full forward-pass cost, while enabling substantial computation savings through adaptive token pruning and cross-frame cache reuse.

A.3 Derivation of the Closed-Form Pruning–Reuse Ratio

The phase-aware optimization in Eq. (12) is a one-dimensional quadratic problem:

$$J_t(\alpha) = w_q(t)\alpha^2 + w_\ell(t)(1-\alpha)^2, \quad \alpha \in [0,1], \tag{24}$$

where $w_q(t) \geq 0$ and $w_\ell(t) \geq 0$ are dynamically determined by the current execution phase and previous-step feedback. Expanding the objective gives

$$J_t(\alpha) = \big(w_q(t) + w_\ell(t)\big)\alpha^2 - 2w_\ell(t)\alpha + w_\ell(t). \tag{25}$$

Since

$$\frac{\partial^2 J_t(\alpha)}{\partial \alpha^2} = 2\big(w_q(t) + w_\ell(t)\big) \geq 0, \tag{26}$$

the objective is convex with respect to α. Setting the first-order derivative to zero yields

$$\frac{\partial J_t(\alpha)}{\partial \alpha} = 2w_q(t)\alpha - 2w_\ell(t)(1-\alpha) = 0, \tag{27}$$

which gives the unconstrained optimum:

$$\alpha_t^{\mathrm{unc}} = \frac{w_\ell(t)}{w_q(t) + w_\ell(t) + \epsilon}. \tag{28}$$

Finally, because the pruning–reuse ratio is constrained to $[0, 1]$, we project the solution onto the feasible interval:

$$\alpha_t^* = \mathrm{clip}\left(\frac{w_\ell(t)}{w_q(t) + w_\ell(t) + \epsilon}, 0, 1\right). \tag{29}$$

This closed-form solution avoids iterative search and introduces only constant-time overhead at each timestep.

A.4 Layer selection for attention fusion.

SynerVLA computes the Text–Vision and Action–Vision scores using only a small set of representative cross-attention layers rather than aggregating attention over the entire network. Specifically, we use the last two decoder cross-attention layers of the System-2 VLM to compute S_{TV}, because these layers encode stronger task-level semantic alignment between the language instruction and visual tokens. In contrast, earlier layers are more biased toward local visual patterns and are less directly related to high-level task semantics. Similarly, we use the last two cross-attention layers of the System-1 DiT to compute S_{AV}, since these layers are closer to the final denoised action prediction and therefore better capture control-critical visual cues. For each selected layer, attention maps are first averaged across heads, and the resulting token scores are then averaged across the selected layers. This layer-selection strategy reduces attention aggregation overhead while preserving the complementary semantic and action-relevant information required by Eq. (6)–(8).

A.5 Limitations

Although SynerVLA demonstrates effective acceleration for on-device dual-system VLA inference, it still has several limitations. First, our current phase-aware feedback optimization characterizes external execution phases using two lightweight signals: end-effector velocity and distance-to-target. These signals are easy to obtain during real-time robot execution and provide practical indicators for distinguishing rapid approaching from fine-grained manipulation phases. However, embodied execution phases are often affected by richer physical and task-level states, including contact states, force/torque feedback, gripper status, object motion, joint-space smoothness, scene dynamics, and safety constraints. The current design does not comprehensively analyze how these additional external signals may guide phase identification or influence the pruning–reuse decision. Incorporating such signals could further improve the robustness of phase-aware feedback optimization, especially in contact-rich manipulation, deformable-object interaction, or highly dynamic environments.

Second, SynerVLA is evaluated mainly on representative dual-system VLA models and robotic manipulation benchmarks. Although the results demonstrate consistent speedup with negligible performance degradation across the evaluated settings, the generality of the proposed phase-aware optimization may still depend on the structure of the underlying VLA model, the type of action expert, and the characteristics of the embodied task. For example, tasks involving long-term contact, multi-object interaction, or highly dexterous manipulation may exhibit more complex phase transitions than the rapid approaching and fine-grained manipulation pattern considered in this work. Extending SynerVLA to a broader range of robot platforms, task categories, and real-world environments remains an important direction for future work.

Finally, our method focuses on training-free inference acceleration and does not jointly optimize model parameters or task policies. This design makes SynerVLA plug-and-play and easy to deploy, but it may limit the achievable accuracy–efficiency trade-off compared with methods that co-train the policy with acceleration-aware objectives. Future work could explore combining phase-aware token optimization and cache reuse with lightweight adaptation or policy fine-tuning, enabling the model to better tolerate aggressive pruning and reuse decisions while maintaining reliable control performance.

A.6 Impact Statement

SynerVLA aims to improve the efficiency of on-device dual-system VLA inference, which can make embodied AI systems more responsive, accessible, and practical on resource-constrained robotic

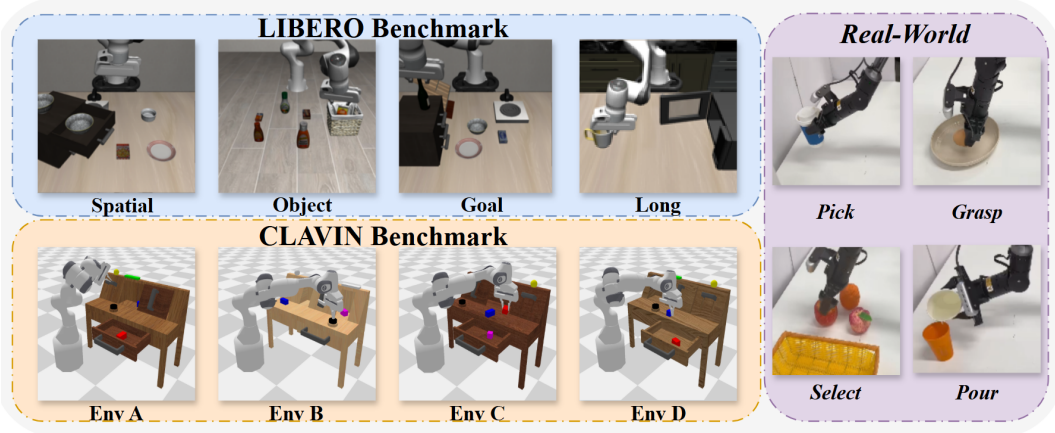


Figure 6: Our experimental environment.

platforms. By reducing inference latency and computational cost, this work may benefit applications such as assistive robots, household service robots, and edge-deployed autonomous agents, where real-time interaction and local processing are important. In particular, on-device acceleration can reduce reliance on cloud computation, thereby lowering communication latency and potentially improving data privacy. At the same time, more efficient robotic agents may also increase the risk of unsafe or unintended deployment if they are used without sufficient safety validation, human oversight, and physical safeguards. Therefore, future deployment of such systems should be accompanied by rigorous testing, task-specific safety constraints, and appropriate human control mechanisms.

B Experiments Details

B.1 Benchmark Details

LIBERO Benchmark LIBERO [15] is a Robosuite-based benchmark for lifelong robot manipulation, comprising 130 language-conditioned tasks organized into four suites: *Spatial*, *Object*, *Goal*, and *Long*, as illustrated in Figure 6. *LIBERO-Spatial* introduces variations only in spatial configurations while keeping the object set and task template unchanged (e.g., picking up the black bowl on the stove and placing it on the plate). *LIBERO-Object* changes the manipulated object while fixing the scene layout and goal template (e.g., picking up the BBQ sauce and placing it in the basket). *LIBERO-Goal* keeps both the objects and spatial layout fixed but modifies the goal description (e.g., opening the middle drawer of the cabinet). *LIBERO-Long* corresponds to the long-horizon subset (10 tasks) of LIBERO-100, where each episode requires executing multi-stage instructions, such as opening a drawer, inserting an object, and subsequently rearranging other items (e.g., placing both the alphabet soup and the cream cheese box into the basket).

CALVIN Benchmark CALVIN [16] is a language-conditioned tabletop manipulation benchmark consisting of 34 distinct tasks. It features unconstrained natural-language instructions that span a broad spectrum of manipulation skills, ranging from basic pick-and-place to articulated object interactions. The benchmark comprises four different environments as shown in Figure 6. each centered on a Franka Panda arm operating in a tabletop workspace. In this work, we follow the challenging evaluation protocol: policies are trained using demonstrations from environments A, B, and C, and are then evaluated zero-shot in environment D. The evaluation set contains 1,000 unique instruction chains, each comprising five consecutive tasks. Baseline results are taken directly from the official CALVIN benchmark. For implementation, we train the generalist model with LoRA for 50 epochs. The specialist model is trained for 100k iterations with a batch size of 64 (approximately 8 epochs). We set the action chunk sizes to $k_g = k_s = 8$ for both the generalist and specialist policies. Unless stated otherwise, the specialist takes as input RGB and depth images from a static third-view camera and a gripper-view camera, and both policies operate on 224×224 image observations.

Real-world Benchmark To validate the deployability of our method on physical hardware, we conduct real-world experiments on an AgileXPiPER robotic arm. The platform is a 6-DoF manipulator

with a 1.5,kg end-effector payload. Its six revolute joints provide full-range manipulation flexibility, enabling high precision and repeatability. The arm adopts a lightweight design that supports fast motions while maintaining relatively high payload capacity, making it suitable for real-scene data collection in embodied intelligence. We evaluate our method on four representative manipulation tasks: **Pick**, lifting a blue paper cup and placing it back down stably; **Grasp**, picking up an egg from the tabletop and placing it into a plate without disturbance; **Select**, selecting the apple from a set of an orange, a peach, and an apple, and placing it into a basket; and **Pour**, lifting a water-filled cup and pouring the water into another cup, as summarized in Figure 6. These tasks jointly cover basic pick-and-place, fragile-object grasping, object selection under distractors, and liquid pouring, providing a practical testbed for real-world execution.

B.2 More Experiment Results

Contribution of individual feedback signals. We evaluate the phase-aware scheduler on RoboDual/CALVIN by removing one signal at a time (Table 6). When an execution-phase signal is removed, we renormalize the weights of the remaining signals. When an action-quality or latency feedback term is removed, we replace it with its rollout mean. Removing target distance or end-effector velocity reduces average success by 6.1 or 5.3 percentage points, respectively, at similar speedups. Removing both action-quality and latency feedback increases the mean pruning ratio from 47.8% to 52.8% and reduces success from 72.8% to 69.0%.

Table 6: **Scheduler signal ablations on RoboDual/CALVIN.** $\bar{\alpha}$ is the mean pruning ratio over rollout steps.

Scheduler setting	Avg. success (%)	Speedup	$\bar{\alpha}$ (%)
RoboDual (unaccelerated)	73.2	1.00×	–
SynerVLA (all signals)	**72.8**	**1.93×**	47.8
w/o target distance	66.7	1.95×	49.7
w/o end-effector velocity	67.5	1.94×	49.0
w/o attention entropy	69.6	1.94×	49.1
w/o action-quality feedback	70.3	1.98×	51.4
w/o latency feedback	71.4	1.96×	50.2
w/o both feedback terms	69.0	**2.00×**	52.8

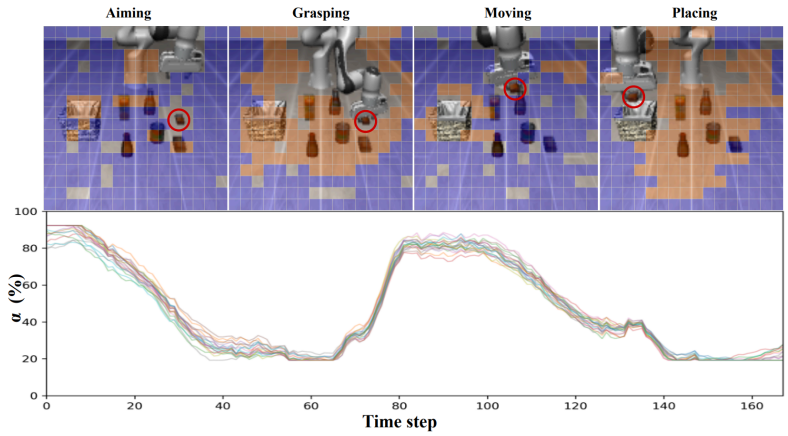
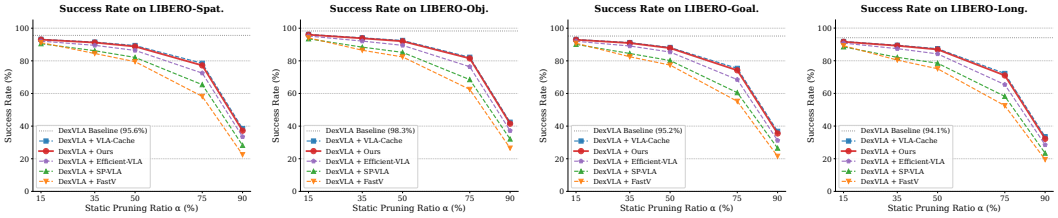


Figure 7: Variation in the pruning ratio α across task stages.

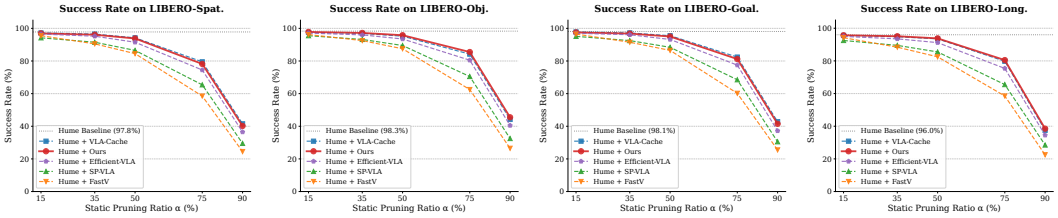
Alignment with phase-specific pruning tolerance. To test whether the scheduler allocates computation according to the demands of each phase, we replay held-out RoboDual/CALVIN states at pruning ratios of 15%, 35%, 50%, 75%, and 90%. For each state, we define an oracle reference as the largest tested ratio that preserves the gripper decision and keeps short-horizon end-effector pose deviation below 5%. Simulator events provide phase labels for this analysis only. The oracle

ratio averages 70.4% during coarse motion and 28.6% during fine manipulation. With uniform phase-signal weights, the scheduler selects corresponding mean ratios of 67.2% and 31.4%, with a frame-wise correlation of 0.73 with the oracle. Across seven weight settings, average success ranges from 71.8% to 72.8% and speedup from $1.89\times$ to $1.97\times$.



(a) LIBERO-Spatial (b) LIBERO-Object (c) LIBERO-Goal (d) LIBERO-Long

Figure 8: **Detailed Performance Comparison on DexVLA.** Evaluation of success rates across four LIBERO sub-tasks. Our method (DexVLA + Ours) demonstrates superior robustness compared to baselines, particularly maintaining higher performance at extreme pruning ratios.



(a) LIBERO-Spatial (b) LIBERO-Object (c) LIBERO-Goal (d) LIBERO-Long

Figure 9: **Performance Comparison of Methods on Hume.** We extend our evaluation to the **Hume** backbone, comparing the success rates of our method against FastV, SP-VLA, Efficient-VLA, and VLA-Cache. The results indicate that our approach maintains high robustness on the stronger Hume model, showing a more gradual performance decline in early pruning stages and competitive stability at high pruning ratios.

B.3 Error Analysis of Approximate KV Reuse

Evaluation protocol. We compare SynerVLA with full recomputation on RoboDual/CALVIN and Hume/LIBERO. At each cache-hit step, we run a reference forward pass with the same inputs, initial noise, and random seed. We measure relative KV errors at the layers and token positions where reuse occurs, and compare the resulting action latents and predicted actions with the reference outputs. The reference pass is used only for error analysis and is excluded from the reported latency.

Table 7: **Deviation from full recomputation at cache-hit steps.** KV errors and action deviations are relative to the reference pass. Speedup is measured against the unaccelerated backbone.

		Relative KV error (%)			
Backbone / benchmark	**Latent cosine sim.**	**Visual**	**Condition**	**Action dev. (%)**	**Speedup**
RoboDual / CALVIN	0.985 ± 0.005	3.3 ± 1.1	2.5 ± 0.9	1.8 ± 0.6	$1.93\times$
Hume / LIBERO	0.979 ± 0.007	4.0 ± 1.4	3.1 ± 1.1	2.4 ± 0.9	$1.75\times$

Layer-wise error and action deviation. At each System-2 update, the current image is re-encoded and queries are recomputed. Cross-frame reuse applies only to selected visual-token keys and values in decoder self-attention. Even when a local patch changes little, changes elsewhere in the image can affect its deeper representations. To examine this effect, we group RoboDual's 32 decoder layers into four consecutive blocks. Mean visual-KV error increases from 2.2% in layers 1–8 to 3.0%, 3.8%,

and 4.2% in layers 9–16, 17–24, and 25–32, respectively. At the action output, mean latent cosine similarity is 0.985 on RoboDual and 0.979 on Hume, with mean normalized action deviations of 1.8% and 2.4%, respectively (Table 7). These measurements characterize the approximation at the evaluated cache-hit steps.

B.4 Generalization and Additional Efficiency Comparisons

Transfer to an end-to-end VLA. We adapt SynerVLA to $\pi_{0.5}$ by mapping its System-2 and System-1 operations to the VLM prefix and the flow-matching action expert, respectively. We reuse eligible visual and context states and recompute action- and timestep-dependent states. This adaptation requires access to the model's internal interfaces but no retraining. We keep the scheduler objective and coefficients fixed across LIBERO and SIMPLER. As shown in Table 8, inference latency decreases from 103.0 to 64.4 ms on LIBERO and from 80.0 to 54.4 ms on SIMPLER Google-VM. The corresponding average success rates decrease by 1.6 and 2.6 percentage points, respectively.

Table 8: **Transfer to an end-to-end policy and a second robot platform.** Speedup is relative to the unaccelerated policy in the same setting.

Policy / setting	Method	Avg. success (%)	Latency (ms)	Speedup
$\pi_{0.5}$ / LIBERO	Baseline	96.9	103.0	1.00×
	+ SynerVLA	95.3	64.4	1.60×
$\pi_{0.5}$ / SIMPLER Google-VM	Baseline	72.4	80.0	1.00×
	+ SynerVLA	69.8	54.4	1.47×
RoboDual / JAKA S5	Baseline	85.7	64.16	1.00×
	+ SynerVLA	82.2	35.20	1.82×

Transfer to a second robot platform. We also evaluate RoboDual on a JAKA S5 equipped with Intel RealSense D435i and D455 cameras. We use the same four manipulation tasks as on the PiPER-X and retain the scheduler coefficients and update rule. On the JAKA S5, SynerVLA reduces latency from 64.16 to 35.20 ms (1.82× speedup), with average success decreasing from 85.7% to 82.2% (Table 8).

Controlled comparisons on LIBERO. Within each block of Table 9, we use the same checkpoint, numerical precision, evaluation protocol, and RTX 4090 GPU. On DexVLA, timings include token-selection overhead. SynerVLA achieves 29.2 ms latency and 95.1% average success, compared with 36.4 ms and 94.9% for VLA-Pruner. On π_0, FLASH achieves slightly lower latency than SynerVLA (34.9 vs. 35.8 ms). FLASH uses a separately trained 110M-parameter draft model, whereas SynerVLA reaches 1.62× speedup with no additional training or parameters. Their observed average success rates are 93.4% and 93.7%, respectively.

Table 9: **Controlled comparisons on LIBERO.** Each block uses its own unaccelerated backbone as the reference.

Method	Avg. success (%)	Latency (ms)	Speedup
DexVLA	95.8	51.6	1.00×
+ VLA-Pruner	94.9	36.4	1.42×
+ SynerVLA	95.1	29.2	1.77×
Torch-π_0	94.1	58.0	1.00×
+ FLASH	93.4	34.9	1.66×
+ SynerVLA	93.7	35.8	1.62×

Absolute inference efficiency on CALVIN. Table 10 reports absolute latency and computational cost for OpenHelix and RoboDual on an RTX 4090. Within each backbone, all methods use the same precision, batch size, inputs, and action-generation settings. SynerVLA reduces latency from 64.00 to 34.59 ms on OpenHelix and from 82.00 to 42.49 ms on RoboDual. Inference rate is computed as $1000/T$, where T is the per-inference latency in milliseconds.

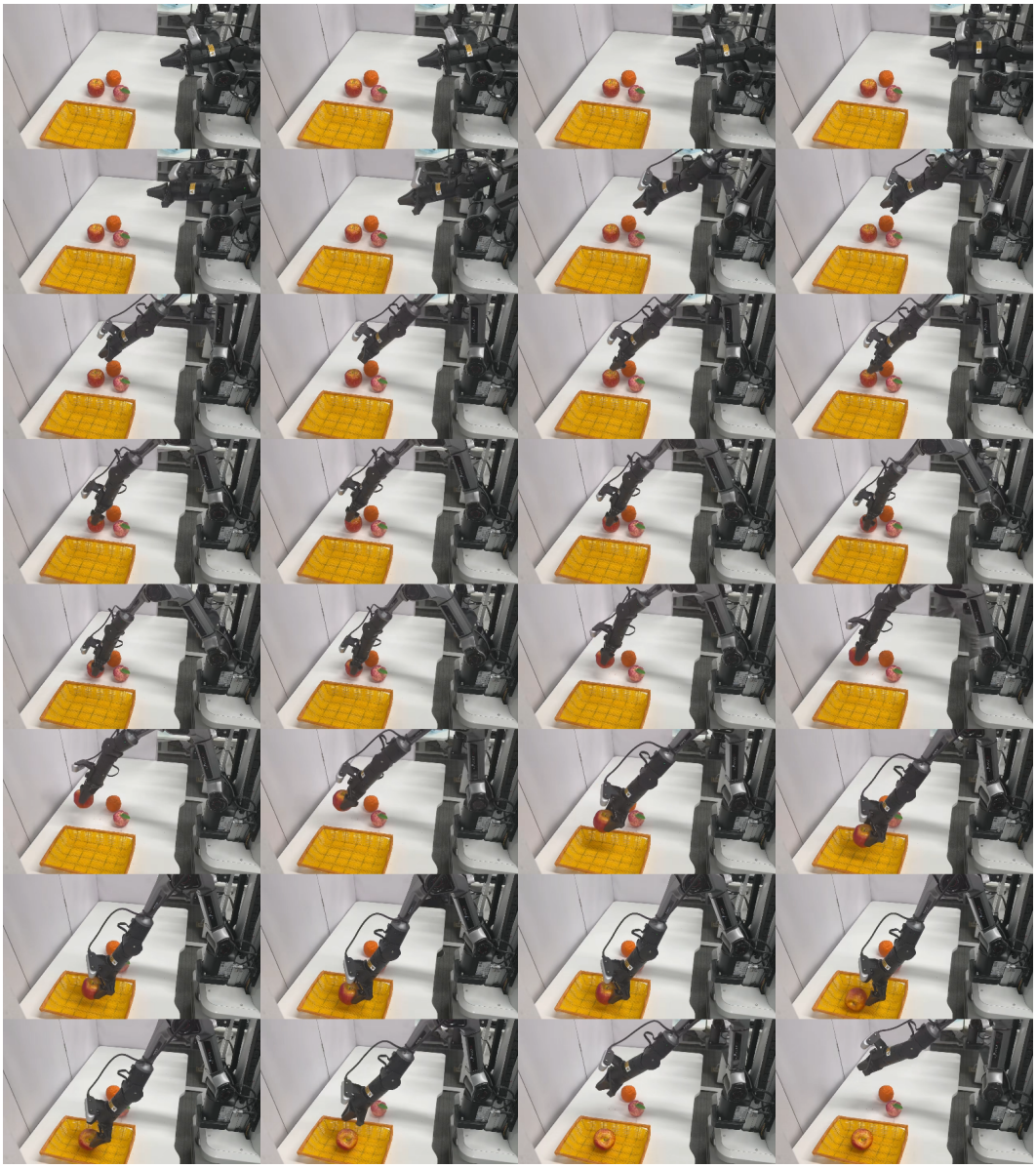


Figure 10: **Real-world robot trajectories for orange picking.** Which show independent trials with randomized initial positions, and columns illustrate the temporal progression of the pick-and-place task.

Table 10: **Absolute inference efficiency on CALVIN (RTX 4090).** Computational cost is reported in TFLOPs per inference. Speedup is relative to the baseline within each backbone.

Method	Avg. success (%)	Latency (ms)	TFLOPs / inf.	Inf. rate (Hz)	Speedup
OpenHelix	68.3	64.00	1.814	15.62	1.00×
+ FastV	66.3	50.79	1.596	19.69	1.26×
+ VLA-Cache	67.8	47.06	1.488	21.25	1.36×
+ Efficient-VLA	66.8	37.43	1.324	26.72	1.71×
+ SP-VLA	63.6	38.79	1.361	25.78	1.65×
+ SynerVLA	67.5	34.59	1.303	28.91	1.85×
RoboDual	73.2	82.00	2.320	12.20	1.00×
+ FastV	71.0	66.67	2.088	15.00	1.23×
+ VLA-Cache	73.1	60.74	1.925	16.46	1.35×
+ Efficient-VLA	71.4	49.10	1.716	20.37	1.67×
+ SP-VLA	69.2	50.62	1.786	19.76	1.62×
+ SynerVLA	72.8	42.49	1.624	23.54	1.93×

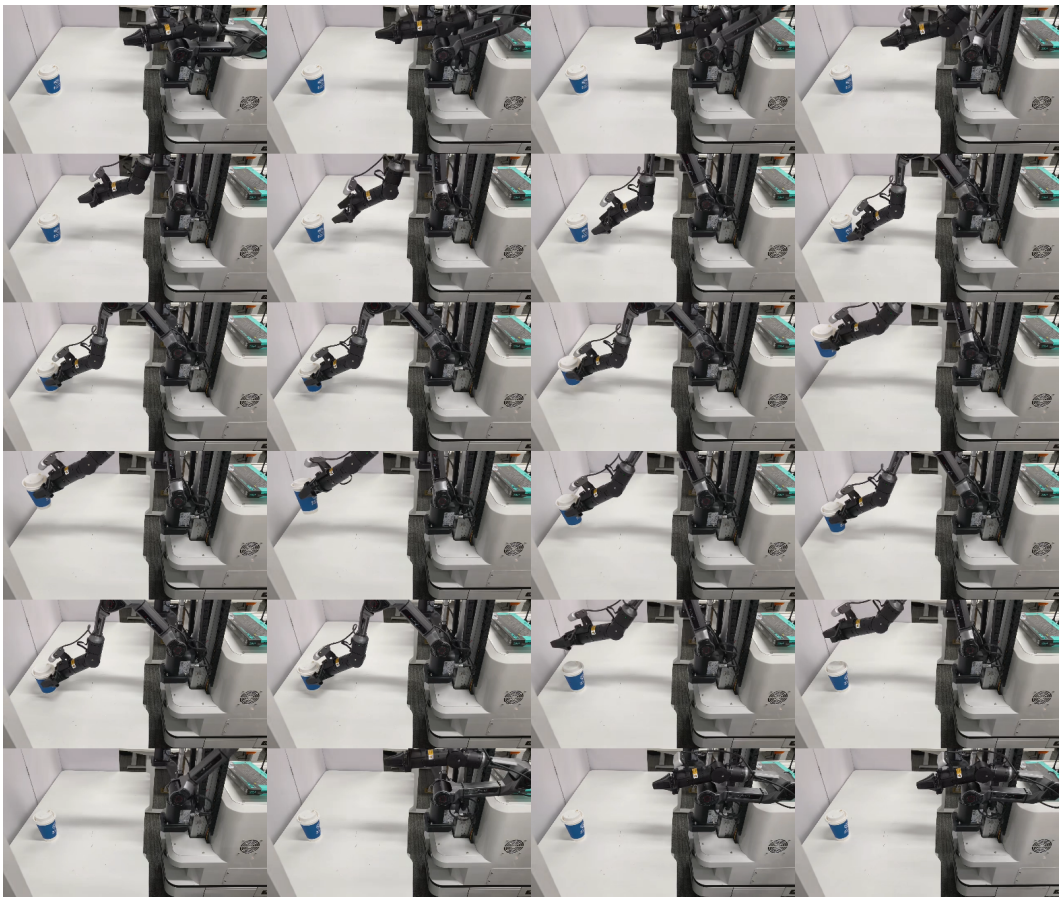


Figure 11: **Real-world cup grasping task.** The robotic arm approaches the target cup, aligns the gripper based on visual feedback, executes a stable grasp, and successfully lifts the object. The experiment demonstrates reliable closed-loop perception–action coordination in a real-world setting.

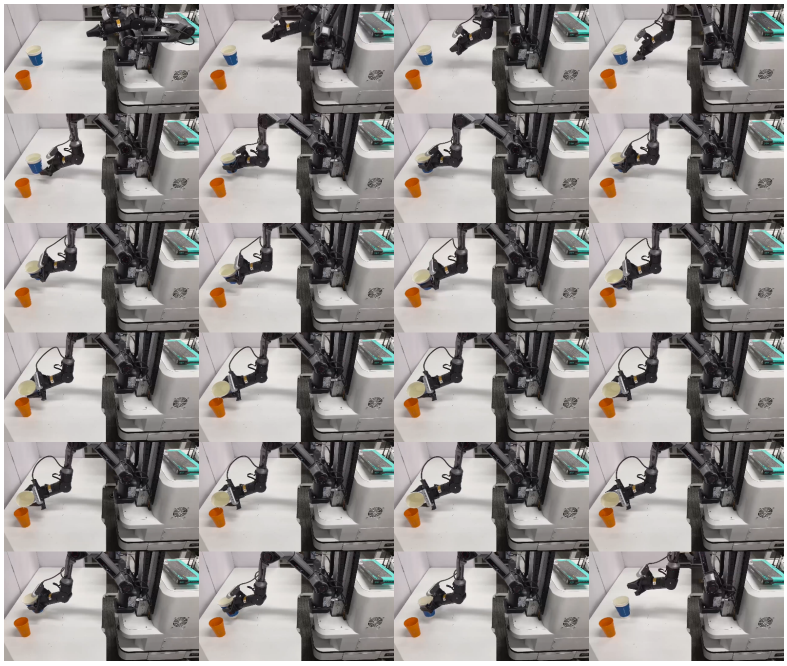


Figure 12: **Real-world Pour Rollout Trajectories.** Columns show time steps and rows denote different trials. The robot grasps and lifts a water-filled cup, aligns it over a second cup, and tilts to pour, demonstrating consistent real-world execution.

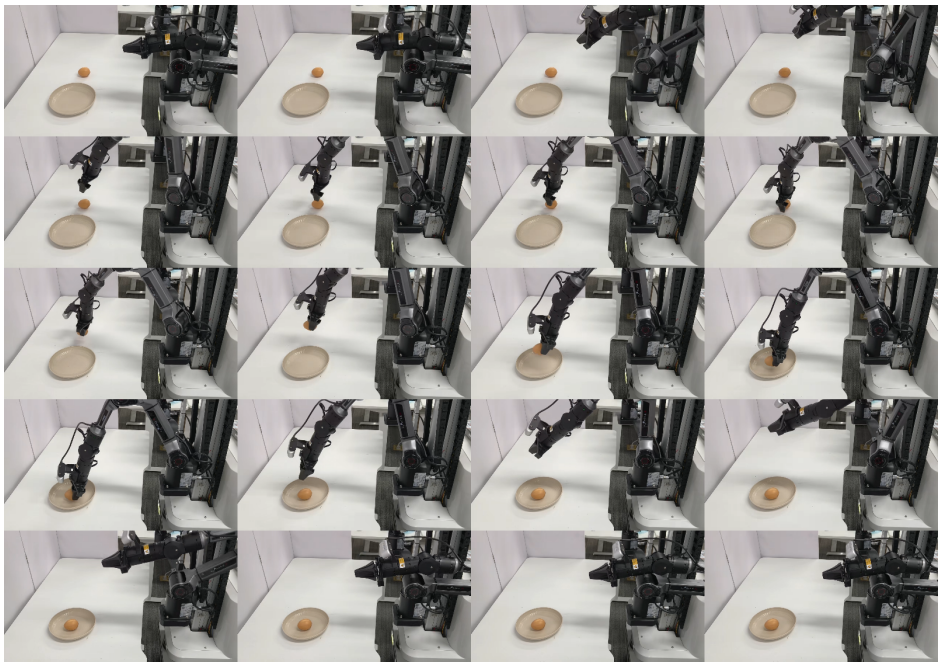


Figure 13: **Qualitative visualization of real-world egg manipulation.** Each row displays a successful rollout of the egg picking task under randomized initial poses. The columns represent the temporal progression, demonstrating the stability and precision of our policy in handling fragile objects.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's main claims: SynerVLA is a plug-and-play framework for dual-system VLA acceleration that exploits execution phases, spatial-temporal visual redundancy, Fusion-based token selection, phase-aware feedback optimization, and dual cache reuse. These claims are supported by the methodology and in Section 5 on CALVIN, LIBERO, four representative dual-system VLA models, and real-world PiPER-X robotic-arm evaluation, where SynerVLA achieves up to 1.93× speedup and 26% higher control frequency with negligible task-success degradation.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses its limitations in Appendix A. Specifically, we acknowledge that SynerVLA currently uses only end-effector velocity and distance-to-target as external phase signals, while richer physical and task-level feedback, such as contact states, force/torque feedback, gripper status, object motion, and scene dynamics, may further improve phase perception and pruning–reuse decisions. We also discuss the scope of evaluation and the limitation of focusing on training-free inference acceleration without jointly optimizing model parameters or task policies.

   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: [Yes]

   Justification: The paper provides the assumptions and derivations for the mathematical formulations used in SynerVLA. In particular, the phase-aware pruning–reuse optimization is defined as a one-dimensional quadratic objective, and its closed-form solution is derived in the appendix. The appendix also provides a complexity analysis for the main components, including patch similarity computation, complementary token importance scoring, entropy-based feedback computation, pruning–reuse ratio optimization, and dual cache reuse.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper discloses the key information needed to reproduce the main experimental results, including the evaluated benchmarks (CALVIN and LIBERO), baseline dual-system VLA models, compared acceleration methods, hardware platform, evaluation metrics, and the detailed design of SynerVLA in Sections 4 and 5. The main claims are supported by the reported simulation and real-world robotic experiments, together with ablation studies that isolate the effects of the proposed components.

   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: [No]

   Justification: The paper does not currently provide open access to the implementation code or full reproduction scripts. However, the paper uses public benchmarks and open-source dual-system VLA models, and provides the main algorithmic details, experimental settings, compared methods, hardware platform, and evaluation metrics needed to support reproducibility; we plan to release anonymized code and configuration files upon publication.

   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: The paper specifies the experimental settings necessary to understand the results, including the evaluated benchmarks, baseline dual-system VLA models, compared acceleration methods, hardware platform, evaluation metrics, and real-world robotic setup in Section 5. Since SynerVLA is a training-free inference acceleration framework, no

training optimizer or training hyperparameters are required; the relevant inference-time design choices and optimization objectives are described in Section 4.

Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper reports quantitative results across multiple benchmarks, models, and real-world robotic tasks, but does not currently include error bars, confidence intervals, or formal statistical significance tests. We report aggregate metrics such as success rate, latency, FLOPs, and speedup to compare the proposed method with baselines, and leave statistical significance analysis as future work.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper provides the main compute-resource information needed to understand and reproduce the experiments. Specifically, Section 5 states that the simulation experiments are conducted on an NVIDIA RTX 4090 GPU, and the experimental results report inference-side compute and runtime metrics including FLOPs, latency, and speedup across CALVIN, LIBERO, and real-world robotic evaluations. These reported resources and efficiency metrics characterize the compute requirements of the evaluated methods.

   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 research conducted in this paper conforms to the NeurIPS Code of Ethics. Our experiments are limited to public robotic manipulation benchmarks and controlled real-world robotic evaluations, and do not involve human subjects, private or personally identifiable data, sensitive attributes, or deceptive data collection procedures.

   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 the broader impacts of efficient on-device dual-system VLA acceleration. On the positive side, SynerVLA can reduce inference latency and computational cost, enabling more responsive and accessible embodied AI systems on resource-constrained robotic platforms. On the negative side, the appendix acknowledges that more efficient robotic agents may increase the risk of unsafe or unintended deployment if they are used without sufficient safety validation, human oversight, and physical safeguards.

    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 new high-risk models, pre-trained language models, image generators, or scraped datasets that would require additional safeguards for responsible release. SynerVLA is a training-free inference acceleration framework evaluated on public robotic benchmarks and controlled robotic manipulation experiments.

    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 datasets and benchmarks used in this paper are publicly available for research purposes. We follow their license terms and usage conditions, and properly cite the corresponding sources in the paper.

    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: [N/A]

    Justification: The paper does not introduce or release new assets such as datasets, model checkpoints, or benchmark suites. SynerVLA is a training-free inference acceleration framework evaluated using existing public benchmarks and existing dual-system VLA models.

Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects. All experiments are conducted using public robotic manipulation benchmarks and controlled robotic evaluation tasks, so participant instructions, screenshots, and compensation details are not applicable.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects. Therefore, IRB approval, participant risk disclosure, and related human-subject review procedures are 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: The core method development in this paper does not involve the use of external LLMs as an important, original, or non-standard component. The paper studies inference acceleration for dual-system VLA models, and any use of LLMs is limited to the evaluated VLA/VLM backbones or standard editing assistance, which does not affect the core methodology, scientific rigor, or originality of the research.

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.