PROGRESS-AWARE VISUAL TOKEN PRUNING FOR EFFICIENT VISION-LANGUAGE-ACTION MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models have shown great promise for embodied AI, yet repeatedly processing hundreds of visual tokens during robot execution incurs substantial computational overhead. Existing pruning methods reduce this cost by retaining task-relevant visual tokens and discarding irrelevant ones. However, the retention priority of task-relevant evidence changes with execution progress, particularly in long-horizon manipulation involving multiple subtasks. To address this issue, we introduce ProgPrune, a training-free method that adapts visual-token pruning to execution progress. Specifically, ProgPrune first parses instructions into ordered subtasks and uses execution feedback to track the active subtask and its execution phase. Guided by this progress, ProgPrune dynamically prioritizes visual tokens relevant to the active subtask over those associated with completed or future subtasks. This allows task-relevant but currently unnecessary tokens to be pruned, while full-instruction guidance preserves global context. As decoder representations evolve, we introduce Fresh-QK to recompute progress-aware token relevance at successive pruning layers. This keeps later pruning decisions from relying on outdated relevance estimates, without additional forward passes. Experiments on LIBERO, SimplerEnv, and CALVIN across multiple VLA architectures show consistent 1.20–1.44$\times$ speedups improvement with competitive task success rate. Notably, ProgPrune achieves particularly strong efficiency on long-horizon tasks (e.g., LIBERO-Long and CALVIN), demonstrating its effectiveness of progress modeling in visual token pruning.

1 INTRODUCTION

Vision-language-action (VLA) models have emerged as a promising paradigm for general-purpose robotic manipulation by transferring the semantic knowledge of large vision-language models to visuomotor control (Brohan et al., 2023; Kim et al., 2025b; Black et al., 2025; Kim et al., 2025a). Leveraging large-scale robot datasets (O'Neill et al., 2024), VLA models jointly process visual observations and natural-language instructions to generate low-level actions, enabling robots to perform complex tasks across diverse scenarios. However, during closed-loop execution, this multimodal design requires VLA models to repeatedly process hundreds of visual tokens, many of which are only weakly relevant to the current manipulation step. Propagating the complete token sequence through the model therefore incurs considerable redundant computation and memory overhead, ultimately increasing inference latency (Pei et al., 2026; Wang et al., 2025; Liu et al., 2025).

Existing methods accelerate VLA inference via model lightweighting, quantization, early exiting, caching, and efficient action decoding (Yue et al., 2024; Park et al., 2024; Xu et al., 2025; Kim et al., 2025a). Among these approaches, visual-token pruning provides a promising solution that directly shortens the multimodal input sequence and can be applied without retraining the VLA backbone. Typically, visual-token pruning methods reduce visual redundancy by retaining task-relevant tokens while discarding less relevant ones. Existing methods guide token selection or pruning decisions using spatial-semantic salience (Li et al., 2026), text relevance combined with motion-based gating (Pei et al., 2026), global history and current-layer relevance (Wang et al., 2025), or semantic relevance augmented with temporally smoothed action-decoding attention (Liu et al., 2025).

Despite these advances, existing visual-token pruning methods remain progress-agnostic and do not explicitly model how the retention priorities of task-relevant visual tokens evolve with task progress,

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

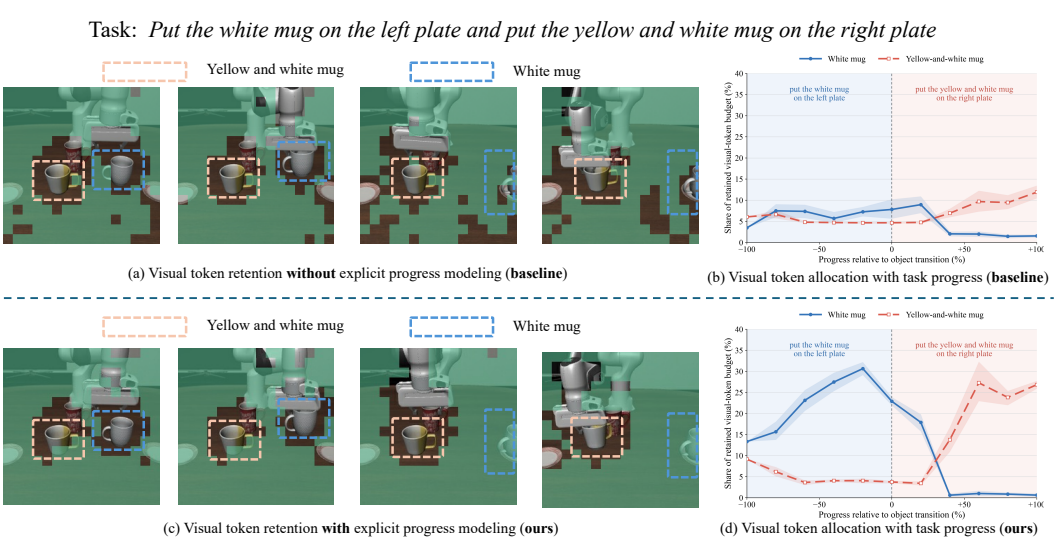


Figure 1: **Progress-agnostic versus progress-aware pruning.** For a two-subgoal task, the robot first places the white mug on the left plate and then the yellow-and-white mug on the right plate. Dashed boxes mark object regions; masked observations show retained visual tokens. **(a,b)** Pruning conditioned on the full instruction retains evidence for both subgoals and responds weakly to their transition. **(c,d)** By conditioning on the tracked active subtask, our method reallocates the fixed token budget from the completed white mug to the active yellow-and-white mug. Curves show each region's share of the retained-token budget, with episode-level variation shaded.

particularly in long-horizon tasks comprising multiple subtasks. Consider the task in Figure 1(a): place the white mug on the left plate, then the yellow-and-white mug on the right plate. Visual tokens depicting the white mug and its interaction region should receive higher retention priority during the first subtask, with priority shifting to the yellow-and-white mug during the second. However, Figure 1(b) shows little change in the two object regions' shares of the retained-token budget around the subtask transition: the completed white mug initially retains a larger share than the newly active yellow-and-white mug. Under full-instruction scoring, evidence for the completed subtask may remain highly ranked because it is still relevant to the overall instruction, occupying budget needed by the active subtask. We refer to this mismatch as *progress-induced retention staleness*: task-relevant visual evidence remains prioritized after it is no longer needed for the active subtask. Addressing this problem requires a pruning mechanism that tracks task progress online and allocates the limited token budget primarily to visual evidence relevant to the active subtask.

To address this challenge, we introduce **ProgPrune**, a training-free, plug-and-play method that explicitly achieves progress-aware visual-token pruning for efficient VLA manipulation. ProgPrune first parses the instruction into ordered manipulation subtasks and uses gripper, end-effector, and action feedback to track the active subtask and its execution phase. The tracked progress determines which instruction context guides token selection, giving priority to visual evidence for the active subtask over evidence associated only with completed or future subtasks. Full-instruction relevance complements this local guidance to preserve broader task context. In Figure 1(c,d), ProgPrune initially prioritizes tokens depicting the white mug, then reduces their share and shifts emphasis to the yellow-and-white mug as the second subtask progresses. This redistribution is more pronounced than under full-instruction scoring, allowing the limited token budget to follow the changing visual requirements of the current manipulation. Since decoder representations also evolve across layers, relevance estimates computed at an earlier layer can become outdated. ProgPrune therefore introduces Fresh-QK to recompute both relevance scores at successive pruning points using the current query and key projections. Progress tracking thus updates the selection context across policy queries, while Fresh-QK updates token relevance across decoder layers, without additional forward passes.

We conduct extensive experiments on three manipulation benchmarks, i.e., LIBERO (Liu et al., 2023), SimplerEnv (Li et al., 2025), and CALVIN (Mees et al., 2022), using OpenVLA-OFT (Kim et al., 2025a) and CogACT (Li et al., 2024). Across these benchmarks, ProgPrune achieves con-

sistent 1.20–1.44× speedups relative to dense inference while maintaining competitive task success rate. Notably, ProgPrune achieves particularly strong efficiency on long-horizon tasks, matching the dense policy's 94.2% success rate on LIBERO-Long while achieving a 1.39× overall speedup on LIBERO. Additional experiments on long-horizon benchmark RoboCasa365 (Nasiriany et al., 2026) further demonstrate the generalization of our method (Appendix B).

In a nutshell, our main contributions are summarized as follows:

- We propose ProgPrune, a plug-and-play pruning method for VLA models that aligns visual token retention with task progress. This allows task-relevant but currently unnecessary visual tokens to be pruned, leaving more capacity for evidence needed by the active subtask.
- We introduce Fresh-QK to maintain progress-aware token selection as decoder representations evolve. It refreshes relevance from live query–key projections at each pruning layer without additional forward passes.
- Extensive experiments across diverse manipulation benchmarks and VLA models demonstrate that ProgPrune achieves consistent speedups improvement over dense inference while maintaining competitive task performance.

2 Related Work

Vision-Language-Action (VLA) Models. Vision-language-action (VLA) models extend pretrained vision-language models to robotic manipulation by generating actions conditioned on visual observations and natural-language instructions. Early works such as RT-1 and RT-2 (Brohan et al., 2023; Zitkovich et al., 2023) demonstrate the scalability and semantic generalization of large robot policies, while Open X-Embodiment (O'Neill et al., 2024) enables cross-task and cross-embodiment learning from large-scale robot demonstrations. Recent models further explore different action representations and decoding mechanisms, including discretized action tokens in OpenVLA (Kim et al., 2025b), continuous parallel action decoding in OpenVLA-OFT (Kim et al., 2025a), and diffusion-based action generation in CogACT (Li et al., 2024). Despite these advances, repeatedly processing dense visual-token sequences during closed-loop execution incurs substantial computational cost, motivating visual-token pruning for more efficient VLA inference in this work.

Efficient Robotic Manipulation. Recent studies have improved the efficiency of robotic manipulation policies by exploiting redundancy in action generation and sequential observations. OpenVLA-OFT accelerates action generation through parallel decoding and action chunking (Kim et al., 2025a), while VLA-Cache reuses the representations of visual tokens that remain stable across consecutive observations (Xu et al., 2025). EfficientVLA jointly reduces redundant language-model layers, visual tokens, and intermediate computations in diffusion-based action heads (Yang et al., 2025). Other methods adapt computation to the current control state: SP-VLA combines action-aware model scheduling with spatial-semantic token pruning (Li et al., 2026); ADP adjusts token retention ratios according to recent robot motion (Pei et al., 2026); SpecPrune-VLA integrates historical and current information with layer-wise pruning (Wang et al., 2025); and VLA-Pruner combines semantic relevance with temporally smoothed action-level importance (Liu et al., 2025). However, these methods do not explicitly account for task progress, so task-relevant but currently unnecessary visual tokens may continue to occupy the limited retention budget. Our method addresses this limitation by tracking execution progress and prioritizing visual evidence for the active subtask.

Progress Awareness in VLA Models. Recent work has introduced progress awareness to improve long-horizon embodied reasoning and control. PALM (Liu et al., 2026) learns affordance representations and continuous within-subtask progress, ProgressLM (Zhang et al., 2026) investigates progress reasoning in vision-language models, and See, Plan, Rewind (Dai et al., 2026) grounds instructions into spatial milestones for progress monitoring and failure recovery. These methods use progress primarily to improve task reasoning or policy behavior rather than inference efficiency. For efficient VLA inference, ADP (Pei et al., 2026) uses robot motion to determine when and how aggressively visual tokens should be pruned, while continuing to evaluate their relevance against the full instruction. In contrast, ProgPrune uses execution feedback to track the active subtask and its execution phase, prioritizing the corresponding visual evidence within the available token budget.

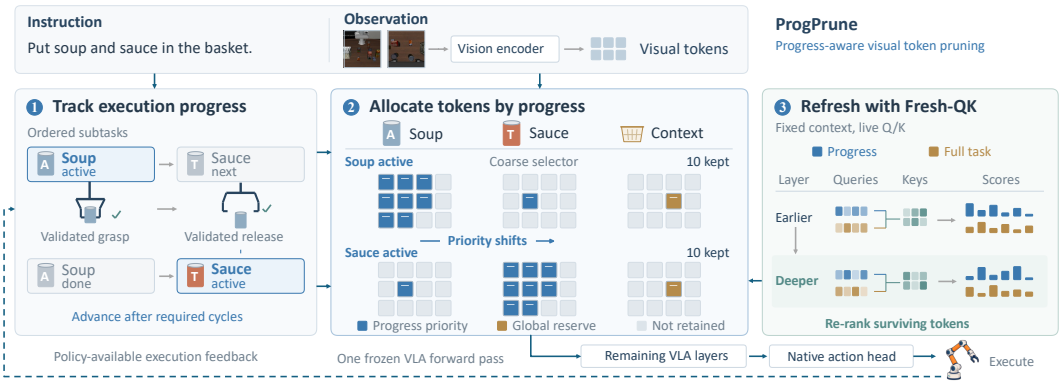


Figure 2: **Progress-aware Visual Token Pruning (ProgPrune).** ProgPrune uses execution feedback to prioritize visual tokens for the active subtask over those for completed or future subtasks, while full-instruction guidance preserves global context. Fresh-QK refreshes relevance across pruning layers to maintain this progress-aware selection as decoder representations evolve within the same forward pass. The soup-to-sauce example uses an illustrative 10-token budget.

3 METHODOLOGY

Figure 2 illustrates our proposed **Progress-aware Visual Token Pruning (ProgPrune)**. We first parse the instruction into ordered subtasks and use execution feedback to track the active subtask and its execution phase. Guided by this progress, we prioritize visual tokens relevant to the active subtask over those associated with completed or future subtasks. This allows task-relevant but currently unnecessary tokens to be pruned, while full-instruction guidance preserves global context. As decoder representations evolve, Fresh-QK recomputes progress-conditioned and full-instruction relevance from current query–key projections at successive pruning layers. This keeps later pruning decisions from relying on outdated relevance estimates, without additional forward passes.

3.1 PROBLEM FORMULATION

Consider a frozen VLA at a pruning point with N surviving visual tokens. Their hidden states form $\mathbf{V} \in \mathbb{R}^{N \times d}$, where d is the hidden dimension and $\mathcal{I} = \{1, \ldots, N\}$ indexes its rows at this pruning point. Given a retention budget $K \in \{1, \ldots, N\}$, the selector constructs a reduced visual sequence for the downstream decoder:

$$\mathbf{V}_{\text{keep}} = \mathbf{V}[\mathcal{S}], \qquad \mathcal{S} \subseteq \mathcal{I}, \qquad |\mathcal{S}| = K. \tag{1}$$

The subset $\mathcal{S}$ should preserve the evidence required for the current action within the available budget. Indexing by $\mathcal{S}$ gathers the corresponding rows in their original sequence order. All nonvisual tokens are retained, and successive pruning stages operate on the surviving visual tokens.

The budget is specified for each policy query and pruning layer by the routing schedule. Unless otherwise stated, quantities refer to the current pruning point, with query and layer indices omitted. The index t denotes policy queries, which may predict multi-step action chunks.

3.2 EXECUTION-GROUNDED PROGRESS TRACKING

Execution state. Visual requirements change both between subtasks and within a manipulation cycle. ProgPrune therefore tracks the active subtask together with its execution phase. A deterministic parser maps each instruction to an ordered subtask sequence $\mathcal{G}$. Each subtask specifies a manipulation type and cycle count, together with source, target, and context spans in the instruction. Parsing and initialization of z_0 occur at each new instruction; ambiguous parses or unsupported transition structures yield a single whole-instruction subtask.

Let e_t denote execution evidence available before policy query t, obtained from temporally filtered gripper commands, gripper width, and motion feedback. The progress state evolves according to

$$z_t = (p_t, \phi_t, c_t), \qquad z_t = \mathcal{T}(z_{t-1}, e_t; \mathcal{G}), \tag{2}$$

where p_t identifies the current subtask during execution, ϕ_t denotes its execution phase, and c_t counts completed cycles within that subtask. The phase takes values in SOURCE, TARGET, ACTION, CONTINUOUS, and COMPLETE. The deterministic transition function $\mathcal{T}$ encodes the rules associated with the parsed subtasks. For pick-and-place operations, a validated grasp changes the phase from SOURCE to TARGET; a subsequent validated release increments the cycle count. Advancement to the next subtask requires completion of the specified number of cycles. Action subtasks use conservative motion-and-settle events, and COMPLETE is absorbing. Ambiguous evidence delays advancement or yields coarser context. Tracking relies on policy-available feedback without access to rewards, privileged object poses, or simulator success predicates. These transitions reflect feedback-based event estimates, and the state is fixed within each forward pass.

Progress-conditioned instruction context. The state z_t selects instruction spans associated with the active subtask and its execution phase. The default context also includes the full previous subtask, when available; progress conditioning therefore does not exclude all completed-subtask evidence. Spatial relations and follow-up actions may retain joint source–target spans or the complete current-subtask context. Character offsets align these spans with the original instruction tokens. Let $\mathcal{R}$ denote the selected character positions, $\mathcal{P}_j$ the positions covered by token j, and $\mathcal{J}_{\text{all}} = \{1, \dots, L\}$ the indices of all $L \geq 1$ instruction tokens. The active text set is

$$\mathcal{J}_{\text{act}} = \begin{cases} \{j \in \mathcal{J}_{\text{all}} : \mathcal{P}_j \cap \mathcal{R} \neq \varnothing\}, & \text{valid context,} \\ \mathcal{J}_{\text{all}}, & \text{otherwise.} \end{cases} \tag{3}$$

Validity requires successful parsing and nonempty token alignment; otherwise, selection reverts to full-instruction guidance. Thus, both text sets are nonempty, and scoring leaves the original instruction tokens unchanged.

3.3 FRESH-QK: LAYER-WISE RELEVANCE REFRESH

Even with the progress state fixed during a policy query, multimodal representations continue to evolve across decoder layers. Reusing an early ranking can therefore leave later pruning decisions based on outdated relevance estimates. Fresh-QK recomputes progress-conditioned and full-instruction relevance at subsequent pruning points using the current query and key projections.

Let $\mathbf{q}_j^h, \mathbf{k}_i^h \in \mathbb{R}^{d_h}$ denote the query of instruction token j and the key of visual token i for query head h. There are H query heads; in grouped-query attention, $\mathbf{k}_i^h$ denotes the corresponding key, which may be shared across heads. The mean query vector for text context r is

$$\bar{\mathbf{q}}_r^h = \frac{1}{|\mathcal{J}_r|} \sum_{j \in \mathcal{J}_r} \mathbf{q}_j^h, \qquad r \in \{\text{act}, \text{all}\}. \tag{4}$$

The relevance of visual token i to this context is defined by the mean scaled query–key similarity across heads:

$$s_i^r = \frac{1}{H} \sum_{h=1}^{H} \frac{\langle \bar{\mathbf{q}}_r^h, \mathbf{k}_i^h \rangle}{\sqrt{d_h}}, \qquad i \in \mathcal{I}. \tag{5}$$

The scores s_i^{act} and s_i^{all} share the same visual keys and differ in their textual conditioning. By linearity, Equation 5 averages raw scaled QK similarities over the selected instruction tokens and attention heads. These are unnormalized scores, not averages of softmax attention weights.

At deeper pruning points, the mean queries and scores are computed from live Q/K projections in the ongoing decoder pass. Their computation costs $\mathcal{O}((L+N)Hd_h)$, excluding the existing projections and token selection, and avoids materializing a full text-to-vision score matrix. No learned scorer or additional decoder pass is required. Progress tracking thus adapts the textual condition across policy queries, while Fresh-QK updates relevance estimates at subsequent decoder pruning points.

3.4 ROUTED BUDGETS AND TOKEN SELECTION

ProgPrune follows ADP's action-aware routing principle (Pei et al., 2026) to choose between two pruning schedules. The coarse branch prunes at entry and again at a deeper layer, whereas the coverage-preserving branch retains the full visual sequence until a later pruning point. Within this schedule, instruction structure controls entry protection, and the two relevance scores determine which tokens are retained under the assigned budgets.

Instruction-dependent entry budget. The entry budget $K_{\text{ent}} = \text{round}(\kappa_t N_{\text{ent}})$ scales the visual-token count N_{ent} at the first pruning point. On the coarse branch, the keep ratio depends on instruction structure:

$$\kappa_t = \begin{cases} \kappa_{\text{base}}, & \text{reliable multi-subtask or multi-cycle parse,} \\ \kappa_{\text{atomic}}, & \text{reliable single-subtask, single-cycle parse,} \\ \kappa_{\text{fallback}}, & \text{unreliable parse with whole-instruction fallback.} \end{cases} \tag{6}$$

Reliability requires the parser to support the instruction's subtask and transition structure. We impose $0 < \kappa_{\text{base}} < \kappa_{\text{atomic}} < \kappa_{\text{fallback}} \leq 1$, assigning larger entry budgets when structured progress guidance is limited. Throughout the reported evaluations, we use the fixed default $(\kappa_{\text{base}}, \kappa_{\text{atomic}}, \kappa_{\text{fallback}}) = (0.60, 0.75, 0.90)$. This tuple is treated as an empirical accuracy–compute operating point and is not adapted per task.

Progress-conditioned selection with a global reserve. To account for instruction-wide context alongside the active subtask and phase, the coarse branch reserves a fraction $\beta \in [0, 1]$ of the budget for full-instruction relevance, with $\beta = 0.1$ by default. For a single candidate pool, the active and reserve allocations are

$$K_{\text{act}} = \text{round}((1-\beta)K), \qquad K_{\text{res}} = K - K_{\text{act}}. \tag{7}$$

Let $\text{TopK}(s, \mathcal{I}, k)$ return exactly k distinct highest-scoring indices from $\mathcal{I}$ for $0 \leq k \leq |\mathcal{I}|$, with $\varnothing$ for $k = 0$. The resulting selection is

$$\begin{aligned} \mathcal{S}_{\text{act}} &= \text{TopK}(s^{\text{act}}, \mathcal{I}, K_{\text{act}}), \\ \mathcal{S} &= \mathcal{S}_{\text{act}} \cup \text{TopK}(s^{\text{all}}, \mathcal{I} \setminus \mathcal{S}_{\text{act}}, K_{\text{res}}). \end{aligned} \tag{8}$$

Drawing the reserve from unselected candidates allocates capacity to instruction-wide relevance within the prescribed budget. The endpoints $\beta = 0$ and $\beta = 1$ recover pure progress-conditioned and pure full-instruction selection, respectively. On the coarse branch, multi-view candidates are partitioned into camera-specific sets $\mathcal{I}_v$, with integer budgets K_v and selected sets $\mathcal{S}_v$ satisfying

$$\sum_v K_v = K, \qquad 0 \leq K_v \leq |\mathcal{I}_v|, \qquad \mathcal{S} = \bigcup_v \mathcal{S}_v. \tag{9}$$

Equations 7–8 are applied independently to each pair $(\mathcal{I}_v, K_v)$, with allocation rounding performed within each view before the selected sets are combined.

Coverage-preserving selection. For contact-sensitive motion, the coverage branch protects a spatially distributed core $\mathcal{C} \subseteq \mathcal{I}$, with $|\mathcal{C}| \leq K$, then fills the remaining slots by relevance. Let $R = K - |\mathcal{C}|$, $R_{\text{act}} = \text{round}((1-\beta)R)$, and $R_{\text{res}} = R - R_{\text{act}}$. The selected set is

$$\begin{aligned} \mathcal{S}^{\mathcal{C}}_{\text{act}} &= \text{TopK}(s^{\text{act}}, \mathcal{I} \setminus \mathcal{C}, R_{\text{act}}), \\ \mathcal{S} &= \mathcal{C} \cup \mathcal{S}^{\mathcal{C}}_{\text{act}} \cup \text{TopK}(s^{\text{all}}, \mathcal{I} \setminus (\mathcal{C} \cup \mathcal{S}^{\mathcal{C}}_{\text{act}}), R_{\text{res}}). \end{aligned} \tag{10}$$

Selection and quota rounding are independent per view. Both rules retain exactly K distinct tokens; on the coverage branch, β allocates the full-instruction reserve from the residual budget outside the core. Both branches thus prioritize visual evidence for the active subtask while preserving global task context within their assigned token budgets.

4 Experiments

4.1 Experimental setup

Simulation Benchmarks and Platforms. We evaluate ProgPrune on three widely-used simulated robotic manipulation benchmarks: LIBERO (Liu et al., 2023), SimplerEnv (Li et al., 2025), and CALVIN (Mees et al., 2022). On LIBERO, we evaluate suite-specific OpenVLA-OFT checkpoints (Kim et al., 2025a) across the Spatial, Object, Goal, and Long suites, with 500 episodes per suite. LIBERO-Long serves as the primary testbed for progress-aware pruning because its long-horizon tasks involve multiple subtasks. On SimplerEnv, we evaluate OpenVLA-OFT on the Bridge/WidowX tasks and CogACT (Li et al., 2024), using the released CogACT-Base checkpoint, on the Google Robot tasks as an additional architecture control. On CALVIN, we evaluate the RLinf policy path over 1,000 five-task sequences. For every protocol, we preserve the checkpoint-specific visual inputs, processor, action representation, and decoding.

Table 1: **Comparison results on LIBERO and CALVIN.** Success rates cover four LIBERO suites or 1–5 consecutive CALVIN tasks; Avg. Len. is the mean completed sequence length. Speedup/FLOPs use dense-relative analytical LLM compute; latency is decoder-prefill CUDA ms. †: SR/FLOPs from Wang et al. (2025, v3, Table 1), with speedup derived from FLOPs. **Bold**/underlined mark best/second-best per panel, including ties.

LIBERO benchmark	Success Rate (%) ↑				Speedup ↑	Latency (ms) ↓	Avg. SR ↑	FLOPs ↓
Method	Spatial	Object	Goal	Long			(%)	(%)
OpenVLA-OFT	98.2	**98.2**	97.6	94.2	1.00×	98.82	97.1	100.0
+ FastV†	94.6	95.8	94.0	88.8	1.75×	–	93.3	57.0
+ DivPrune†	92.4	91.2	89.0	84.8	**1.85**×	–	89.4	**54.0**
+ SparseVLM†	96.8	94.2	97.6	93.6	1.30×	–	95.6	77.0
+ VLA-Cache	98.2	98.0	97.2	**94.4**	1.20×	100.40	97.0	83.0
+ VLA-ADP	**99.4**	98.0	96.4	91.2	1.23×	83.99	96.3	81.3
+ **ProgPrune**	98.8	97.6	**98.0**	94.2	1.39×	**83.42**	**97.2**	71.8

CALVIN benchmark	Success Rate (%) ↑					Avg. Len. ↑	Speedup ↑	Latency (ms) ↓	FLOPs ↓
Method	SR@1	SR@2	SR@3	SR@4	SR@5				(%)
OpenVLA-OFT	92.8	**83.6**	**73.5**	**62.8**	**55.1**	**3.678**	1.00×	84.24	100.0
+ VLA-Cache	92.5	82.2	72.1	61.7	54.6	3.631	1.21×	108.36	82.9
+ SpecPrune-VLA	93.3	81.4	68.5	56.0	49.6	3.488	**1.71**×	97.00	**58.5**
+ VLA-ADP	92.4	81.5	70.3	60.9	52.5	3.576	1.14×	77.12	88.0
+ **ProgPrune**	**93.8**	83.5	72.8	62.2	53.7	3.660	1.20×	**73.53**	83.1

Baselines. We compare ProgPrune against the corresponding dense policy and representative training-free acceleration methods. General visual-token pruning baselines include FastV (Chen et al., 2024), DivPrune (Alvar et al., 2025), and SparseVLM (Zhang et al., 2025); VLA-specific baselines include VLA-Cache (Xu et al., 2025), EfficientVLA (Yang et al., 2025), SpecPrune-VLA (Wang et al., 2025), and VLA-ADP (Pei et al., 2026). VLA-ADP is our closest baseline: it uses robot motion to control pruning aggressiveness but estimates token relevance from the full instruction throughout execution.

Evaluation Metrics. We report success rates on LIBERO and SimplerEnv, and chain-completion rates (SR@1–SR@5) and average completed sequence length on CALVIN. Following VLA-ADP (Pei et al., 2026), analytical speedup is computed as the ratio of dense to pruned LLM FLOPs. We also report CUDA-synchronized decoder-prefill latency on an RTX A6000; timing details and complete action-call measurements are provided in Appendix A.1.

4.2 Main Results

Performance on diverse benchmarks. Tables 1 and 2 compare ProgPrune with the dense policy and representative baselines across diverse benchmarks, including LIBERO, SimplerEnv, and CALVIN. Generally, ProgPrune achieves 1.20–1.44× speedups relative to the dense policy (i.e., OpenVLA-OFT and CogACT) with competitive task performance, demostrating better performance–efficiency trade-offs. Compared with VLA-ADP, it improves average task performance at lower FLOPs on LIBERO, CALVIN, and Bridge/WidowX. These joint gains show that the improved task performance is achieved with less LLM computation, supporting a better performance–efficiency trade-off than motion-aware pruning with full-instruction guidance. Measured decoder-prefill speedups of 1.15–1.27× further demonstrate lower decoder latency across these policies, supporting applicability across distinct VLA architectures and action decoders.

Performance on long-horizon tasks. As shown in Table 1, ProgPrune achieves notable performance gains on long-horizon tasks, e.g., LIBERO-Long and CALVIN. On LIBERO-Long, ProgPrune achieves 94.2% success rate, matching Dense and exceeding VLA-ADP's 91.2%, while the aggregate LIBERO analytical speedup is 1.39×. The improvement over ADP is largest on Long among the four LIBERO suites, consistent with the benefit of updating retention priorities as execution moves between subtasks. On CALVIN's chains of evaluator-provided atomic instructions, ProgPrune achieves 1.20× analytical speedup with the highest average completed sequence length among the evaluated accelerated policies (3.660), compared with 3.576 for ADP and 3.678 for Dense. These results suggest that explicit progress modeling is particularly beneficial for long-horizon manipula-

Table 2: **SimplerEnv results on Bridge/WidowX and Google Robot VM.** Avg. SR denotes unweighted mean over task success rates. Speedup/FLOPs use dense-relative analytical LLM compute; latency is decoder-prefill CUDA time. †: SR from Wang et al. (2025, v3, Table 6); other rows are local evaluations. **Bold**/underlined mark best/second-best per panel, including ties.

Bridge/WidowX	Success Rate (%) ↑				Speedup ↑	Latency (ms) ↓	Avg. SR ↑ (%)	FLOPs ↓ (%)
Method	PutCarrot	PutSpoon	StackCube	PutEggplant				
OpenVLA-OFT	33.3	41.7	4.2	**95.8**	1.00×	75.07	43.8	100.0
+ FastV	37.5	**50.0**	**12.5**	54.2	**1.53**×	**57.86**	38.5	**65.3**
+ VLA-Cache	**45.8**	45.8	8.3	62.5	1.12×	97.77	40.6	89.6
+ VLA-ADP	41.7	41.7	4.2	66.7	1.21×	66.64	38.5	83.0
+ **ProgPrune**	37.5	**50.0**	**12.5**	**95.8**	1.32×	64.54	**49.0**	76.0
Google Robot VM	Success Rate (%) ↑				Speedup ↑	Latency (ms) ↓	Avg. SR ↑ (%)	FLOPs ↓ (%)
Method	PickCan	MoveNear	Drawer	DrawerApple				
CogACT	92.0	**86.7**	72.2	**52.8**	1.00×	64.07	**75.9**	100.0
+ FastV†	92.6	81.4	69.8	52.4	–	–	74.1	–
+ VLA-Cache†	92.0	83.3	70.5	51.6	–	–	74.4	–
+ SpecPrune-VLA†	91.3	83.8	71.6	50.2	–	–	74.2	–
+ **ProgPrune**	**93.7**	85.8	**72.7**	45.4	**1.44**×	**50.33**	74.4	**69.2**

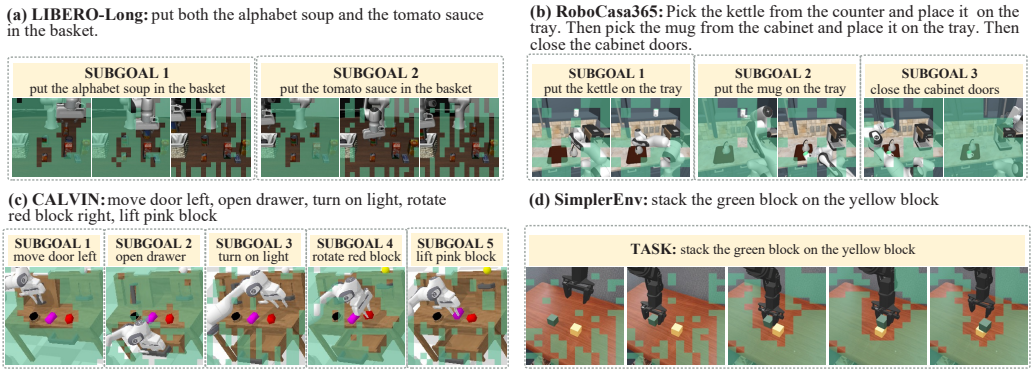


Figure 3: Progress-conditioned visual-token retention over four complete, successful rollouts spanning LIBERO-Long, RoboCasa365, CALVIN, and SimplerEnv. Mint cells are masked patches; dashed groups separate active subtasks. RoboCasa365 contains three subgoals, CALVIN contains five atomic transitions, and SimplerEnv is the single-task control.

tion, where visual-token retention must adapt to the changing requirements of successive subtasks. We also evaluate ProgPrune on an additional long-horizon benchmark, RoboCasa365 (Nasiriany et al., 2026), and report results in Appendix B.

Qualitative Analysis. Figure 3 illustrates the visual token retention behavior of our method. Retained regions follow the transition from soup to tomato sauce in LIBERO-Long and from object placement to cabinet closure in RoboCasa365. The instruction remains unchanged, but the visual evidence needed for ongoing manipulation changes. These examples illustrate how progress conditioning directs retention toward current visual requirements within the overall task.

4.3 Ablation Studies and Analysis

Effect of progress and global context. Table 3 compares guidance strategies under the same pruning schedule. Combining active-context and full-instruction guidance achieves 94.2% on Long, versus 92.0% with full-instruction-only and 92.6% with active-only guidance. The gain over full-instruction scoring supports adapting retention priorities to task progress, while the gain over active-only scoring supports retaining a global reserve.

Effect of the pruning schedule. We investigate different pruning schedules and report the results in Table 3. As shown in the table, adding the delayed coverage branch with progress guidance and Fresh-QK fixed raises Long success from 91.0% to 94.2% and analytical speedup from 1.244× to 1.393×, highlighting the benefit of adaptive pruning schedules.

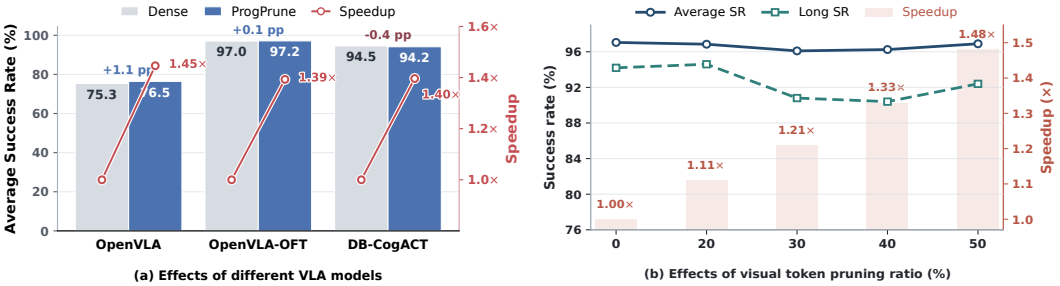


Figure 4: **VLA backbone transfer and pruning-ratio analysis.** **(a)** Average success rates and analytical speedups across VLA backbones; pp denotes percentage points. **(b)** Success rates and speedups at different visual-token pruning ratios.

Effect of Fresh-QK. Table 3 compares Fresh-QK with entry-layer Q/K replay under the same token budgets. Fresh-QK improves Long success from 91.0% to 94.2%, while both variants achieve a 1.393× analytical speedup. These results demonstrate the benefit of refreshing relevance at successive pruning layers.

Table 3: **Controlled component ablations on LIBERO.** Each block varies one factor with others fixed. Results average four suites of 500 episodes each; bold marks the best success rates per block.

Variant	Long SR ↑ (%)	Avg. SR ↑ (%)	Speedup ↑
Progress conditioning			
Full instruction	92.0	96.65	1.393×
Active instruction	92.6	96.90	1.394×
Active + full instruction	**94.2**	**97.15**	1.393×
Pruning schedule (Progress + Fresh-QK fixed)			
Deep coarse only	91.0	96.55	1.244×
+ Delayed delicate	**94.2**	**97.15**	1.393×
Layer-wise scoring (final schedule)			
Replayed entry-QK	91.0	96.50	1.393×
Fresh-QK (ProgPrune)	**94.2**	**97.15**	1.393×

Effect of the VLA backbone. In Figure 4 (a), we analyze the generalization of our method across different VLA backbones, including OpenVLA, OpenVLA-OFT, and DB-CogACT. Our method achieves consistent 1.39–1.45× speedups across these backbones. Notably, our pruned models even improve success rates on OpenVLA and OpenVLA-OFT with faster inference, demonstrating the plug-and-play design across distinct VLA backbones and action decoders.

Effect of the pruning ratio. Figure 4 (b) investigates the effects of different visual-token pruning ratios. Increasing the pruning ratio from 20% to 50% raises speedup from 1.11× to 1.48×, while average success remains between 96.10% and 96.90%. Among the tested pruned settings, 50% achieves the highest average success (96.90%), with 92.40% success on Long.

Limitations. Despite its promising performance in aforementioned benchmarks, ProgPrune has two limitations. First, limited access to robotic hardware has restricted our experiments to simulation, and the effectiveness of ProgPrune on physical robots remains to be validated. Second, its rule-based instruction parser and progress tracker may require further adaptation to handle diverse language inputs and noisy real-world feedback. Future work will focus on evaluating ProgPrune on physical robots and improving the robustness of instruction parsing and progress tracking.

5 Conclusion

We presented ProgPrune, a training-free method that aligns visual-token pruning with task progress in VLA models. Using execution feedback to track the active subtask and its execution phase, ProgPrune prioritizes the corresponding visual evidence while preserving global task context through full-instruction guidance. Fresh-QK refreshes these relevance estimates at successive pruning layers as decoder representations evolve, without additional forward passes. Experiments on LIBERO, SimplerEnv, and CALVIN with multiple VLA architectures demonstrate competitive task performance at 1.20–1.44× speedups over dense inference. On long-horizon tasks, ProgPrune matches the dense policy's success rate on LIBERO-Long and achieves the highest average completed sequence length among the evaluated accelerated policies on CALVIN. Accounting for task progress allows visual computation to follow the changing needs of manipulation.

AI USE DISCLOSURE

The authors used generative AI tools to assist with language editing, manuscript organization, and consistency checking. All technical claims, methodological choices, code, experimental protocols, numerical results, citations, figures, and final text were reviewed and verified by the authors, who take full responsibility for the submitted work.

REFERENCES

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. DivPrune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9392–9401, 2025. URL https://arxiv.org/abs/2503.02175.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert 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, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics transformer for real-world control at scale. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://roboticsproceedings.org/rss19/p025.html.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35, 2024. doi: 10.1007/978-3-031-73004-7_2.

Tingjun Dai, Mingfei Han, Tingwen Du, Zhiheng Liu, Zihao Zhang, Zhihui Li, Salman Khan, Jun Yu, and Xiaojun Chang. See, plan, rewind: Progress-aware vision-language-action models for robust robotic manipulation. *arXiv preprint arXiv:2603.09292*, 2026.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025a. doi: 10.15607/RSS.2025.XXI.017. URL https://www.roboticsproceedings.org/rss21/p017.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

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

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su,

Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3705–3728. PMLR, 2025. URL https://proceedings.mlr.press/v270/li25c.html.

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shu-Tao Xia, Zhi Wang, and Wenwu Zhu. SP-VLA: A joint model scheduling and token pruning approach for VLA model acceleration. In *International Conference on Learning Representations*, volume 2026, pp. 38155–38184, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4072543747a14bbed76284cf2c04b9e9-Paper-Conference.pdf.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2306.03310.

Yuanzhe Liu, Jingyuan Zhu, Yuchen Mo, Gen Li, Xu Cao, Jin Jin, Yifan Shen, Zhengyuan Li, Tianjiao Yu, Wenzhen Yuan, Fangqiang Ding, and Ismini Lourentzou. PALM: Progress-aware policy learning via affordance reasoning for long-horizon robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28096–28110, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Liu_PALM_Progress-Aware_Policy_Learning_via_Affordance_Reasoning_for_Long-Horizon_Robotic_CVPR_2026_paper.html.

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Bridging the semantic-action gap in visual token pruning for efficient VLA inference. *arXiv preprint arXiv:2511.16449*, 2025. doi: 10.48550/arXiv.2511.16449. URL https://arxiv.org/abs/2511.16449.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108. URL https://arxiv.org/abs/2112.03227.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a05003fdb1e9562ab0c0a9719ea4de10-Paper-Conference.pdf.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open X-embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024. doi: 10.1109/ICRA57147.2024.10611477. URL https://arxiv.org/abs/2310.08864.

Seongmin Park, Hyungmin Kim, Wonseok Jeon, Juyoung Yang, Byeongwook Jeon, Yoonseon Oh, and Jungwook Choi. Quantization-aware imitation-learning for resource-efficient robotic control. *arXiv preprint arXiv:2412.01034*, 2024. doi: 10.48550/arXiv.2412.01034. URL https://arxiv.org/abs/2412.01034.

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 10832–10851, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1223ca2b14513106ad55db6a537337f9-Paper-Conference.pdf.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. SpecPrune-VLA: Accelerating vision-language-action models via action-aware self-speculative pruning. *arXiv preprint arXiv:2509.05614*, 2025. doi: 10.48550/arXiv.2509.05614. URL https://arxiv.org/abs/2509.05614. Accepted to ICML 2026.

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 *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 164448–164473. Curran Associates, Inc., 2025. doi: 10.52202/085713-5484. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/f062da1973ac9ac61fc6d44dd7fa309f-Abstract-Conference.html.

Yantai Yang, Yuhao Wang, Zichen Wen, Zhongwei Luo, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. EfficientVLA: Training-free acceleration and compression for vision-language-action models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1365. URL https://arxiv.org/abs/2506.10100.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. DeeR-VLA: Dynamic inference of multimodal large language models for efficient robot execution. In *Advances in Neural Information Processing Systems*, volume 37, pp. 56619–56643, 2024. URL https://arxiv.org/abs/2411.02359.

Jianshu Zhang, Chengxuan Qian, Haosen Sun, Haoran Lu, Dingcheng Wang, Letian Xue, and Han Liu. ProgressLM: Towards progress reasoning in vision-language models. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11243–11271, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.516. URL https://aclanthology.org/2026.acl-long.516/.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A. Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 74840–74857. PMLR, 2025. URL https://proceedings.mlr.press/v267/zhang25s.html.

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