When and What to Prune? Stage-Aware Visual Token Pruning for Efficient VLA

Tianjun Shi[1], Haotian Xiong[1], Ziyu Gong[1], Qi Lu[1], Li Li[1]*
[1]SKL-IOTSC, ISCT, University of Macau
{mc55413,mc55441,mc55143,yc57960,llili}@um.edu.mo

Abstract

Visual token pruning is an effective way to accelerate vision-language models and is especially useful for vision-language-action (VLA) inference, where many visual tokens must be processed before predicting robot actions. Existing pruning methods usually estimate which tokens can be pruned based on attention scores or feature diversity, retaining tokens that are either highly attended or visually different from others. However, most of them use fixed pruning schedules, such as pruning once at a preset layer or pruning at uniformly spaced layers. Such schedules can be risky for VLA models, because the model may not know which visual regions matter for the action in early layers. Tokens that look unimportant at first may become useful after the model combines visual observations with the language instruction. In this work, we propose **SAPrune**, a training-free visual token pruning framework for efficient VLA inference. Instead of pruning at fixed layers, SAPrune uses a small calibration set to observe how action-to-visual attention changes across layers, and chooses pruning layers only after the attention pattern becomes more reliable. At each selected layer, SAPrune applies a dual-path pruning rule: one path protects strongly attended visual tokens from pruning, while the other prevents useful surrounding context from being discarded. Experiments on LIBERO, SIMPLER, and real-world robotic tasks show that SAPrune prunes 87.5% of visual tokens and achieves up to **1.718**$\times$ inference speedup while maintaining competitive task success rates.

1 Introduction

Vision-language-action (VLA) models have recently shown strong potential for embodied decision making by mapping visual observations and language instructions directly to robot actions [1–6]. Despite their effectiveness, VLA inference remains computationally expensive because a large number of visual tokens must be propagated through deep Transformer layers [5, 7–9]. Visual token pruning is therefore a natural direction for reducing latency and computation, as shown in prior studies on efficient Vision Transformers and large vision-language models [10–17].

Although recent studies have shown substantial redundancy in VLA inference, most pruning-oriented acceleration methods still decide token removal from local importance cues, visual saliency, feature-space diversity, or their fixed combinations [8, 9, 13–18]. This treats pruning mainly as a static token-selection problem. Such a formulation is insufficient for action generation, where visual evidence does not remain fixed across depth: it gradually emerges, redistributes, and stabilizes as the model forms an action decision. Moreover, visual distinctiveness alone does not imply decision usefulness, since tokens that are geometrically diverse may still be irrelevant to the current action. These limitations motivate us to reconsider VLA pruning from two questions: *when* pruning becomes reliable, and *what* action-relevant evidence should be preserved.

*Corresponding author.

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

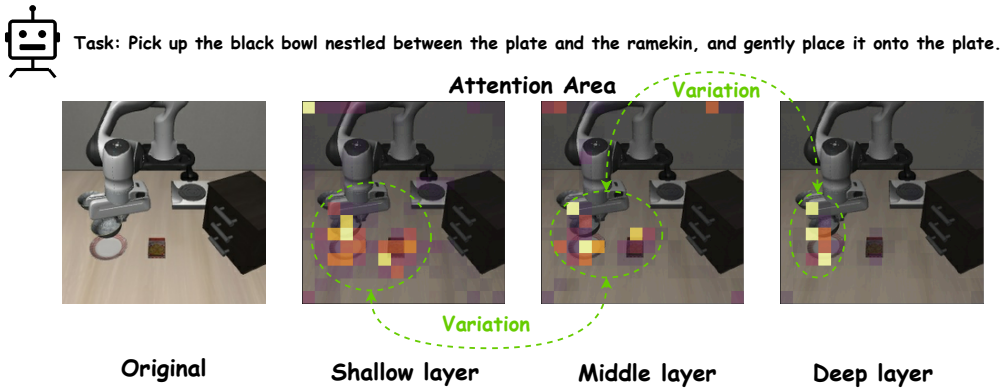
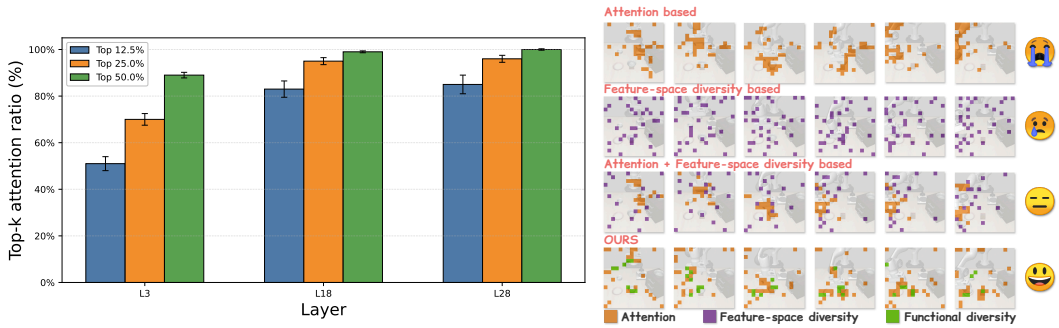


Figure 1: Layer-wise visualization of attention in VLA.

To understand when visual tokens can be safely pruned during VLA inference, we analyze the attention from action-related states to visual tokens across Transformer layers. Figure 1 shows that early layers attend to a broad set of visual tokens, while middle and deep layers gradually concentrate on tokens related to the target action. Figure 2a further quantifies this trend using the top-k attention mass: the mass increases quickly from shallow to middle layers and then changes more slowly. This suggests that early layers have not yet clearly determined which visual tokens are useful for the action. Therefore, pruning too early may remove tokens that become important later, motivating us to delay pruning until the attention pattern becomes more stable. We further compare different pruning criteria in Fig. 2b. Attention-only selection tends to concentrate on dominant regions and may miss supportive context, whereas feature-space diversity-based selection may retain tokens that are visually distinctive but weakly related to the current action. These observations suggest that attention-only selection, feature-space diversity-based selection, and their fixed combination are all insufficient for aggressive VLA pruning. Based on these principles, we propose **SAPrune**,



(a) Visualization of Top-k Importance Image Token Received Attention Ratio.

(b) Comparison of Performance under Different Token Pruning Strategies.

Figure 2: Preliminary observations on top-k attention concentration and token pruning strategy comparison.

a training-free visual token pruning framework for efficient VLA inference. SAPrune first selects pruning layers from rollout-level action-attention dynamics, and then applies a dual-path retention mechanism at each selected layer. The attention-core path preserves directly action-relevant tokens, while the functional-diversity path selects complementary tokens using query-conditioned functional signatures. Experiments on LIBERO [19], SIMPLER [20], and real-world robotic tasks show that SAPrune retains only 32 visual tokens, prunes 87.5% of the original visual tokens, and achieves up to 1.718× speedup while maintaining competitive task success rates.

2 Preliminary

2.1 Related Works

Efficient inference for VLA models. Vision-Language-Action (VLA) models extend large vision-language models by introducing an action modality, enabling robots to predict executable actions from

visual observations and language instructions. Representative generalist robot policies include RT-1 [1], RT-2 [2], Open X-Embodiment / RT-X [3], Octo [4], OpenVLA [5], π_0 [6]. Although effective for general-purpose manipulation, these models repeatedly process dense visual tokens through deep Transformer layers, making inference costly for real-time robot deployment [7–9, 21, 22].

Visual token pruning for VLMs. Visual token pruning reduces visual redundancy by retaining a compact subset of informative visual tokens. Early efficient Vision Transformer methods, such as DynamicViT [10], EViT [11], and ToMe [12], reduce computation through token sparsification or merging. For large vision-language models, attention-based methods such as FastV [13], SparseVLM [15], FoPru [23], HiPrune [24], LVPruning [16], IVTP [14], and ZSPA-Prune [25] rank tokens by prefill attention, text-to-vision attention, or prompt-conditioned relevance. AOT [26] further explores context-preserving token reduction for video LLMs through local-global optimal transport. Other methods preserve diversity or global representativeness [17, 18, 27, 28], while calibration or layer-adaptive methods determine pruning layers, ratios, or progressive schedules from calibration statistics or cross-layer information flow [29–33]. However, these methods are mainly designed for VLM understanding and do not explicitly model action-dependent visual evidence in VLA inference.

Training-free acceleration for VLA models. Training-free VLA acceleration mainly follows temporal reuse, adaptive computation, and pruning-based acceleration. VLA-Cache [7] reuses visual features across adjacent timesteps, while VLA-ADP [22] dynamically reduces computation during manipulation. EfficientVLA [8], SP-VLA [21], SpecPrune-VLA [34], and VLA-Pruner [9] reduce visual-token computation during VLA inference. These methods reveal substantial redundancy in VLA models, but often rely on fixed pruning locations, semantic relevance, temporal reuse, or additional acceleration modules. They leave underexplored how to choose pruning layers and preserve action-relevant evidence under aggressive token reduction.

2.2 VLA Inference and Action-to-Visual Attention

VLA inference. Vision-language-action (VLA) models map visual observations and language instructions to executable robot actions. At rollout step t, the model takes an image observation I_t and a language instruction x_t as input, and predicts an action a_t. For notational simplicity, we omit the timestep index in the following. The vision encoder produces N visual tokens $V^{(0)} = \{v_1^{(0)}, \dots, v_N^{(0)}\}$, while the text encoder produces M text tokens $X^{(0)} = \{x_1^{(0)}, \dots, x_M^{(0)}\}$. Depending on the backbone, action-related states can be implemented as action tokens, action placeholders, or action query states, denoted by $Q_a^{(0)}$, as in OpenVLA [5], π_0 [6]. The initial multimodal sequence is

$$H^{(0)} = V^{(0)} \oplus X^{(0)} \oplus Q_a^{(0)}, \tag{1}$$

where $\oplus$ denotes sequence concatenation. The sequence is then propagated through L Transformer layers:

$$H^{(l)} = F^{(l)}(H^{(l-1)}), \qquad H^{(l)} = V^{(l)} \oplus X^{(l)} \oplus Q_a^{(l)}. \tag{2}$$

The final robot action is predicted from the action-related output states. Since visual tokens usually occupy a large portion of the multimodal sequence, reducing $V^{(l)}$ is an effective way to lower VLA inference cost [7–9, 21, 22].

Action-to-visual attention. For visual token pruning, we use the attention from action-related states to visual tokens as the decision-conditioned relevance signal. At layer l, the action-to-visual attention score of visual token i is defined as

$$a_i^{(l)} = \frac{1}{|Q_a^{(l)}|H} \sum_{q \in Q_a^{(l)}} \sum_{h=1}^{H} \alpha_{q,v_i^{(l)}}^{(l,h)}, \tag{3}$$

where $\alpha_{q,v_i}^{(l,h)}$ denotes the attention weight from action-related query q to visual token v_i in head h, H is the number of attention heads, and $Q_a^{(l)}$ is the set of action-related states.

To characterize the concentration of action-relevant visual evidence at layer l, we compute the top-κ attention mass:

$$S_l(\kappa) = \sum_{i \in \mathrm{Top}_{\lfloor \kappa N \rfloor}(a^{(l)})} a_i^{(l)}, \tag{4}$$

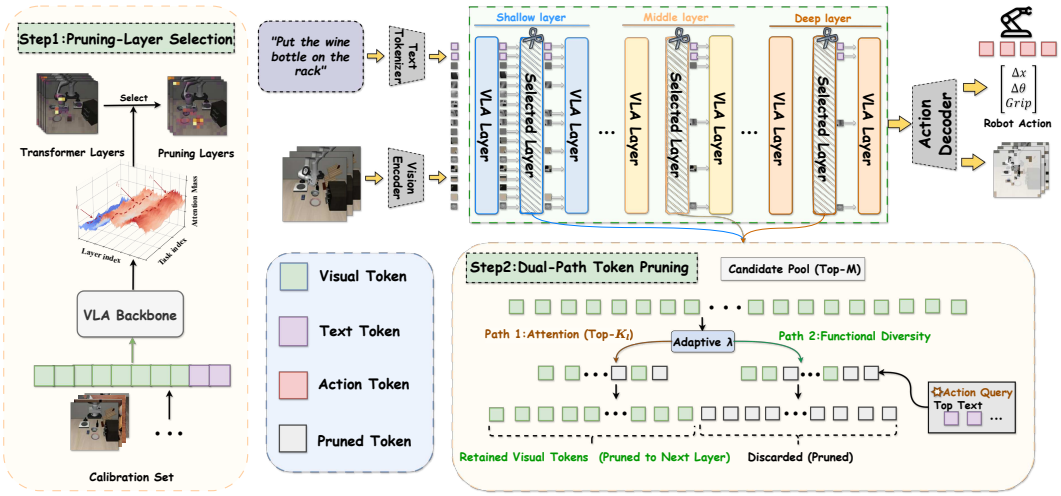


Figure 3: Overview of the framework.

where $\mathrm{Top}_{\lfloor \kappa N \rfloor}(a^{(l)})$ denotes the indices of the top $\lfloor \kappa N \rfloor$ visual tokens ranked by action-to-visual attention. A larger $S_l(\kappa)$ indicates that action evidence is concentrated on fewer visual tokens. We use this signal for both pruning-layer selection and action-relevant token retention.

3 Method

3.1 Overview

Figure 3 gives an overview of **SAPrune**. SAPrune uses layer-wise action-to-visual attention statistics to decide where pruning is applied. It first selects pruning layers from rollout-level action-attention dynamics, and then applies a layer-local dual-path pruning rule.

3.2 Pruning-layer selection from action-attention dynamics

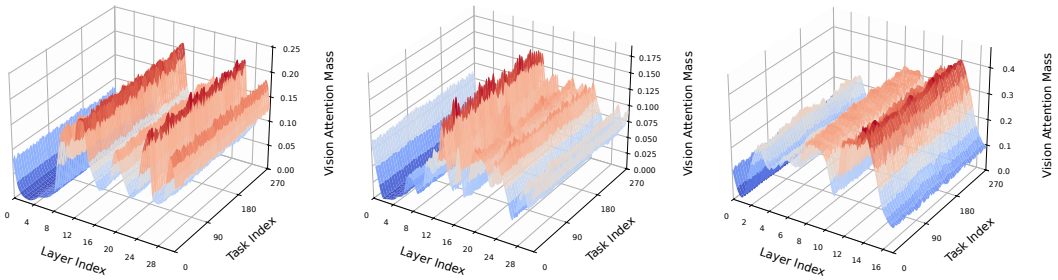
Given a VLA backbone, we select pruning layers by detecting stable stages in rollout-level action-to-visual attention dynamics. We measure the layer-wise top-κ action-to-visual attention mass and its inter-layer redistribution, and then choose stable layers from the shallow, middle, and deep regions. Calibration only logs attention statistics and uses no rewards, success labels, or task outcomes. By default, we use LIBERO [19] for calibration; we also consider SIMPLER, real-world, and mixed simulation-real splits with 30, 40, and 270 rollouts, respectively. Figure 4 visualizes task-level averages and shows that the depth-wise attention trend is largely consistent across calibration tasks. This visualization is used only for stability analysis, while pruning-layer selection uses the rollout-averaged statistic defined below.

Using the action-to-visual attention defined in Sec. 2.2, we compute the top-κ attention mass at each layer and rollout step:

$$S_l(t) = \sum_{i \in \mathcal{T}_\kappa^{(l,t)}} a_i^{(l,t)}, \qquad \mu_l = \frac{1}{T} \sum_{t=1}^{T} S_l(t), \tag{5}$$

where $\mathcal{T}_\kappa^{(l,t)}$ contains the indices of the top $\lfloor \kappa N \rfloor$ visual tokens ranked by $\mathbf{a}^{(l,t)}$, N is the number of visual tokens, t indexes all calibration rollout steps, and μ_l is the rollout-averaged top-κ attention mass used for pruning-layer selection. Here, κ denotes the fraction of visual tokens used to compute the top-κ attention mass. We use $\kappa = 0.125$ by default. For Figure 4, we group rollout steps by calibration task n and plot the task-averaged mass $S_l^{(n)} = |\mathcal{D}_n|^{-1} \sum_{t \in \mathcal{D}_n} S_l(t)$, where $\mathcal{D}_n$ is the set of rollout steps from task n.

To make the rule consistent across backbones of different depths, we partition candidate pruning layers into three contiguous relative-depth regions: $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$, corresponding to shallow, middle, and deep layers. The last Transformer layer is excluded from candidate pruning locations because pruning there would not reduce subsequent computation. We select one pruning layer from each region, denoted by $l_1 \in \mathcal{R}_1$, $l_2 \in \mathcal{R}_2$, and $l_3 \in \mathcal{R}_3$.



(a) OpenVLA (b) OpenVLA-OFT (c) π_0

Figure 4: Task-level variation of layer-wise action-to-visual attention mass. The x-axis denotes the layer index, the y-axis denotes the calibration task index, and the z-axis denotes the vision attention mass.

To suppress single-layer noise, we smooth $\{\mu_l\}$ by a three-point moving average and obtain $\{\tilde{\mu}_l\}$. We then compute the inter-layer shift and two quantile-based thresholds:

$$d_l = |\tilde{\mu}_{l+1} - \tilde{\mu}_l|, \qquad \tau_s = Q_{q_s}(\{d_l\}_{l=0}^{L-2}), \qquad \tau_r = Q_{q_r}(\{d_l\}_{l=0}^{L-2}). \tag{6}$$

Here, d_l measures the attention shift between adjacent layers. We use τ_s for plateau detection and τ_r for redistribution-event detection. A layer is regarded as a plateau start if the following m inter-layer shifts are all below τ_s; the set of plateau starts is denoted by $\mathcal{P}$. A redistribution event is a maximal consecutive segment of indices for which $d_l > \tau_r$.

For the middle and deep regions, we identify the dominant redistribution event. Let $\mathcal{E}_r$ denote the set of redistribution events within $\mathcal{R}_r$, where $r \in \{2, 3\}$. When $\mathcal{E}_r$ is non-empty, we define

$$[a_r^\star, b_r^\star] = \arg\max_{[a,b]\in\mathcal{E}_r} \sum_{u=a}^{b} d_u, \qquad r \in \{2, 3\}. \tag{7}$$

Here, $[a_r^\star, b_r^\star]$ is the redistribution interval with the largest cumulative attention shift in region $\mathcal{R}_r$.

We define the valid search regions as

$$\bar{\mathcal{R}}_r = \begin{cases} \{p \in \mathcal{R}_1 \mid p \geq l_{\min}\}, & r = 1,\ \{p \in \mathcal{R}_1 \mid p \geq l_{\min}\} \neq \varnothing, \\ \mathcal{R}_1, & r = 1,\ \{p \in \mathcal{R}_1 \mid p \geq l_{\min}\} = \varnothing, \\ \{p \in \mathcal{R}_r \mid p > b_r^\star\}, & r \in \{2, 3\},\ \mathcal{E}_r \neq \varnothing,\ \{p \in \mathcal{R}_r \mid p > b_r^\star\} \neq \varnothing, \\ \mathcal{R}_r, & r \in \{2, 3\} \text{ otherwise.} \end{cases} \tag{8}$$

Here, $l_{\min}$ is the earliest allowed pruning layer. For the middle and deep regions, we search after the dominant redistribution interval when it exists; otherwise, we fall back to the full region.

Within each valid search region, we choose the earliest plateau start if available:

$$l_r = \begin{cases} \min\left(\mathcal{P} \cap \bar{\mathcal{R}}_r\right), & \text{if } \mathcal{P} \cap \bar{\mathcal{R}}_r \neq \varnothing, \\ \arg\min_{p\in\bar{\mathcal{R}}_r} \frac{\sum_{u=p}^{\min(p+m-1,L-2)} d_u}{\min(p+m-1,L-2)-p+1}, & \text{otherwise,} \end{cases} \qquad r \in \{1, 2, 3\}. \tag{9}$$

In the fallback case, ties are resolved by choosing the earliest layer. Thus, l_r is either the earliest plateau start in the valid region or the layer with the smallest local average shift. In practice, this deterministic selection also makes the offline calibration outcome reproducible across runs with the same recorded attention statistics. The complete procedure is summarized in Algorithm 1. Unless otherwise stated, we use $q_s = 0.4$, $q_r = 0.8$, $m = 2$, and $l_{\min} = 1$.

3.3 Layer-Local Dual-Path Pruning

At each selected pruning layer l, the module receives the current visual token set $V^{(l)} = \{z_i^{(l)}\}_{i=1}^{N_l}$ and keeps K_l tokens. Let $q_a^{(l)}$ denote the action query and $X^{(l)} = \{x_1^{(l)}, \ldots, x_M^{(l)}\}$ denote the text-token representations. A simple baseline is to keep the top-K_l tokens ranked by action attention. However, this rule often collapses the retained set onto a small highly attended region and may remove nearby objects, spatial anchors, or background cues needed for control. SAPrune therefore splits token retention into two paths. The attention path keeps the highest-attention tokens, while the coverage path selects additional candidates whose query-response patterns are not already represented by the retained set.

Candidate pool and reliability-adaptive budget. We first restrict token selection to an expanded action-guided candidate pool and construct an attention-ranked probe set:

$$C_l = \mathrm{TopM}\Big(a^{(l)}, \lfloor \gamma K_l \rfloor\Big), \qquad A_l^{\mathrm{probe}} = \mathrm{TopK}\Big(a^{(l)}, \lfloor \rho K_l \rfloor\Big), \tag{10}$$

where $a_i^{(l)}$ is the attention from the action query to visual token i. The candidate pool C_l is larger than the final budget, allowing the second path to search beyond the top-K_l tokens while still avoiding the full visual sequence. The probe set A_l^{probe} is used to test whether the high-attention tokens already span the main candidate subspace.

To determine the budget split, we compute two reliability signals. First, from the attention-ranked probe tokens, we construct an orthonormal basis U_l using thin QR decomposition over their normalized features. For each candidate $j \in C_l$, let $w_j^{(l)}$ be its action-attention weight normalized within C_l. We measure how well the probe basis explains the weighted candidate space by

$$R_l = \frac{\sum_{j \in C_l} w_j^{(l)} \left\| z_j^{(l)} - U_l U_l^\top z_j^{(l)} \right\|_2^2}{\sum_{j \in C_l} w_j^{(l)} \left\| z_j^{(l)} \right\|_2^2}, \qquad P_l = 1 - R_l. \tag{11}$$

Here, P_l is a preservation score. A larger P_l means that the attention-ranked probe tokens already capture more of the dominant action-relevant candidate structure.

Second, we estimate how concentrated the current action attention is within the candidate pool using normalized entropy:

$$H_l = -\sum_{j \in C_l} w_j^{(l)} \log w_j^{(l)}, \qquad A_l = 1 - \frac{H_l}{\log |C_l|}. \tag{12}$$

A larger A_l indicates that attention is concentrated on a few candidates, while a smaller value suggests a more diffuse distribution and therefore leaves more room for coverage-based selection.

The preservation score P_l and the concentration score A_l jointly determine the budget allocation:

$$\eta_l = \sqrt{P_l A_l}, \qquad \lambda_l = (1-\eta_l)\lambda_{\min} + \eta_l \lambda_{\max}, \qquad K_l^{\mathrm{attn}} = \lfloor \lambda_l K_l \rfloor, \quad K_l^{\mathrm{cov}} = K_l - K_l^{\mathrm{attn}}. \tag{13}$$

Thus, SAPrune allocates more tokens to the attention path when the high-attention candidates are both representative and concentrated. Otherwise, it assigns more tokens to the coverage path.

Attention-core path. The attention path keeps the top-K_l^{attn} tokens in C_l ranked by action attention, forming S_l^{attn}. These tokens provide the high-confidence subset that the model currently queries most strongly.

Functional-diversity path. The coverage path fills the remaining budget with tokens that are not necessarily top-ranked by action attention but respond differently to the action and instruction queries. Since text tokens are kept unpruned, we use them to construct instruction-conditioned queries. Specifically, we rank non-special text tokens by their head-averaged attention from the action query and keep the top-K_x tokens to form $Q_x^{(l)}$. The query family used by the coverage path is

$$Q_l = \{q_a^{(l)}\} \cup Q_x^{(l)}. \tag{14}$$

The action query captures the control state, while the selected text queries provide instruction-specific context for measuring token coverage.

For each candidate visual token $i \in C_l$, we define a query-conditioned functional signature:

$$r_i^{(l)} = \big[A(q,i)\big]_{q \in Q_l} \in \mathbb{R}^{|Q_l|}, \tag{15}$$

where $A(q,i)$ denotes the head-averaged attention from query q to visual token i. Unlike feature-space diversity, this signature measures how a visual token functionally responds to the current action-language query family. Therefore, two tokens are regarded as redundant if they provide similar query-response patterns, even if their hidden-state features are not identical.

Table 1: **Comparison under reported/default token settings on LIBERO [19].** Red bold marks the best result among pruned methods within each backbone.

Method	Tokens	LIBERO Success Rate (%, ↑)					Latency (ms, ↓)	Speedup (↑)	FLOPs Ratio (%, ↓)
		Spatial	Object	Goal	Long	Avg.			
OpenVLA [5]									
OpenVLA [5]	256	85.3	85.9	76.1	52.3	74.9	178.46	1.000×	100.0
+ FastV [13]	128	81.7	80.5	73.1	47.9	70.8	139.88	1.276×	59.2
+ DivPrune [17]	128	81.1	78.8	71.7	47.6	69.8	136.10	1.311×	62.3
+ SP-VLA [21]	128	83.5	82.6	74.1	48.7	72.2	127.07	1.404×	68.6
+ VLA-ADP [22]	76	84.2	**85.2**	74.9	50.3	73.7	130.18	1.371×	45.7
+ EfficientVLA(T-only) [8]	58	80.7	80.6	70.0	46.4	69.4	115.64	1.543×	42.1
+ **SAPrune**	32	**84.9**	85.1	**75.3**	**50.8**	**74.0**	**110.96**	**1.608×**	**39.5**
OpenVLA-OFT [35]									
OpenVLA-OFT [35]	256	95.5	98.8	96.6	90.7	95.4	103.59	1.000×	100.0
+ FastV [13]	128	94.3	96.5	92.8	87.1	92.7	72.42	1.430×	59.9
+ DivPrune [17]	128	91.8	92.1	89.6	84.1	89.4	73.83	1.403×	58.5
+ SP-VLA [21]	128	93.8	96.5	94.4	85.1	92.5	68.85	1.505×	61.7
+ VLA-ADP [22]	76	94.5	**98.3**	95.3	**90.0**	94.5	71.11	1.457×	46.3
+ EfficientVLA(T-only) [8]	58	92.0	92.7	91.0	84.1	90.0	63.89	1.621×	41.0
+ **SAPrune**	32	**95.2**	98.1	**95.9**	88.9	**94.5**	**60.30**	**1.718×**	**37.8**
π_0 [6]									
π_0 [6]	256	96.9	96.3	95.2	87.2	93.9	78.49	1.000×	100.0
+ FastV [13]	128	93.3	92.5	89.6	80.0	88.8	58.19	1.349×	60.0
+ DivPrune [17]	128	91.0	92.1	87.2	79.8	87.5	60.40	1.300×	55.8
+ SP-VLA [21]	128	94.1	94.2	91.7	82.9	90.7	57.74	1.359×	64.2
+ VLA-ADP [22]	76	**96.3**	95.7	**94.9**	84.7	92.9	58.60	1.339×	46.7
+ **SAPrune**	32	96.2	**95.8**	94.4	**86.0**	**93.1**	**51.22**	**1.532×**	**38.6**

We initialize the retained set as $R = S_l^{\text{attn}}$. After ℓ_2 normalizing the functional signatures, we aggregate the total functional evidence in the candidate pool as $r_{\text{full}}^{(l)}$ and the evidence already covered by R as $r_R^{(l)}$. Their difference defines the residual evidence $m_R^{(l)} = r_{\text{full}}^{(l)} - r_R^{(l)}$, which represents the portion of action-language evidence not yet covered by the retained tokens. The coverage path then greedily selects the candidate that best explains this residual while avoiding redundancy:

$$i^\star = \arg\max_{i \in C_l \setminus R} \left[\left\langle \hat{r}_i^{(l)}, \hat{m}_R^{(l)} \right\rangle - \beta \max_{j \in R} \cos\left(r_i^{(l)}, r_j^{(l)} \right) \right]. \tag{16}$$

The first term favors tokens aligned with the missing functional evidence, while the second term suppresses tokens redundant with the already retained set. After each selection, the chosen token is added to both S_l^{cov} and R, and the process continues until K_l^{cov} coverage tokens are selected.

Finally, the retained visual tokens are obtained by merging the two paths: $S_l = S_l^{\text{attn}} \cup S_l^{\text{cov}}$. If duplicate removal leads to underfilling, the remaining slots are refilled from unused candidates according to action-attention rank until $|S_l| = K_l$. The retained set S_l is then forwarded to subsequent Transformer layers. Unless otherwise specified, we set $\gamma = 1.75$, $\rho = 0.8$, $\lambda_{\min} = 0.1$, $\lambda_{\max} = 0.9$, $K_x = 6$, and $\beta = 0.3$.

4 Experiment

We evaluate SAPrune on two simulation benchmarks, LIBERO [19] and SIMPLER [20], as well as on four real-world manipulation tasks. We compare our method with several representative efficient inference baselines, including FastV [13], DivPrune [17], SP-VLA [21], VLA-ADP [22], and EfficientVLA [8], and instantiate them on three VLA backbones: OpenVLA [5], OpenVLA-OFT [35], and π_0 [6]. For real-world evaluation, we deploy the system on an AgileX Piper-X. All latency results are measured on an NVIDIA RTX 5090 GPU. Unless stated otherwise, latency is measured with PyTorch SDPA for both the full model and SAPrune, without model compilation. FlashAttention-2 results are reported in Appendix B.4.3.

4.1 Experimental Setup

Compared Methods. We compare SAPrune with representative token-pruning baselines for efficient VLA inference, including FastV [13], DivPrune [17], SP-VLA [21], VLA-ADP [22], and EfficientVLA [8]. These methods cover the major pruning paradigms relevant to our setting: FastV [13] performs attention-based top-token selection, DivPrune [17] preserves visually diverse tokens to reduce redundancy, SP-VLA [21] applies structured pruning tailored to VLA inference, VLA-ADP [22] adaptively adjusts token retention during inference, and EfficientVLA [8] combines token relevance with feature-space diversity in a progressive pruning framework. For the main comparison, we follow the reported or default token settings of these methods rather than enforcing a unified token budget for all baselines. Specifically, the original models use 256 visual tokens, while different baselines adopt their own retained-token settings; our method keeps only 32 final visual tokens. To eliminate the effect of unequal token budgets, we further provide controlled comparisons under the same 32-token setting.

Evaluation Metrics. We report success rate, FLOPs Ratio, latency, and speedup. Success rate measures task performance. FLOPs Ratio measures the relative theoretical computation cost with respect to the original model. Unless labeled as E2E latency, latency measures the VLA model inference time in milliseconds, including pruning overhead. E2E latency measures the policy pipeline from observation preprocessing to action-command transmission, excluding physical actuation and low-level servo time. Speedup is computed as the ratio between the latency of the original model and that of the accelerated model. Lower FLOPs Ratio and latency indicate better efficiency, while higher success rate and speedup indicate better overall performance. For consistency, the quantity previously denoted as *inference time* in the tables is referred to as latency throughout this paper.

Table 2: Comparison of SAPrune within the OpenVLA [5] model in the SIMPLER [20] environment. Red bold marks the best result among pruned methods.

Method	Tokens	Success Rate (%, ↑)			
		PickCoke	MoveNear	Open/Close	Avg.
OpenVLA [5]					
OpenVLA [5]	256	54.7	57.8	48.1	53.5
+ FastV [13]	32	19.6	20.1	16.5	18.7
+ SP-VLA [21]	32	35.4	36.1	35.4	35.6
+ VLA-ADP [22]	32	39.4	42.5	38.2	40.0
+ EfficientVLA [8]	32	43.3	44.0	39.7	42.3
+ **SAPrune**	32	**53.2**	**56.3**	**47.9**	**52.5**

Table 3: Comparison on $\pi_{0.5}$ under reported/default token settings on LIBERO. Red bold marks the best result among pruned methods.

Method	Spatial SR	Object SR	Goal SR	Long SR	Avg. SR	Visual tokens / image	Retention	E2E latency (ms)
$\pi_{0.5}$ [36]	97.4	98.4	97.6	93.0	96.6	256	100%	91.47
+ FastV [13]	95.8	95.6	94.1	89.3	93.7	128	50%	70.84
+ VLA-Pruner [9]	95.8	96.2	**97.1**	**91.8**	95.2	64	25%	62.73
+ **SAPrune**	**97.2**	**98.0**	96.9	91.5	**95.9**	32	12.5%	**59.68**

4.2 Results on Simulation Environment

Results on LIBERO. Table 1 reports the main results on LIBERO [19]. Across OpenVLA [5], OpenVLA-OFT [35], and π_0 [6], SAPrune maintains success rates close to the full-token models while using a much smaller visual-token budget. It also brings clear efficiency gains, achieving 1.608×, 1.718×, and 1.532× speedups on the three backbones, respectively, with substantial FLOPs reduction. Compared with existing pruning baselines, SAPrune achieves higher average success under a more aggressive final pruning ratio, indicating that aggressive pruning can remain reliable when token reduction is applied at more stable action-attention stages. The main degradation appears on the Long suite, where long-horizon manipulation is more sensitive to accumulated execution errors and extreme token reduction. Figure 11 provides additional LIBERO visualizations that qualitatively support the observed robustness under high-ratio pruning. We also evaluate SAPrune on $\pi_{0.5}$ [36] on LIBERO. The complete results are shown in Table 3. SAPrune reaches 95.9% average success versus 96.6% for the full model, while reducing E2E latency from 91.47 ms to 59.68 ms with 32 visual tokens versus 256 for the full model. This result extends the evaluation to a newer VLA backbone.

Results on SIMPLER. Table 2 reports the results on the visible tasks in SIMPLER [20] under a compact 32-token budget. Under this aggressive pruning setting, SAPrune maintains substantially stronger task performance than other pruning baselines on the OpenVLA [5] backbone. While attention-based, structured, adaptive, and feature-diversity-based baselines suffer clear success drops after reducing the visual tokens to 32, SAPrune remains comparatively stable across PickCoke,

Table 4: **Real-world manipulation results on AgileX Piper-X with the π_0 [6] backbone.** Each task is evaluated over 100 trials. Bold marks the best result among pruned methods.

Method	Success Rate ↑					Latency (ms)↓	FLOPs Ratio (%)↓
	Cube Grasping	Pick&Place	Select-Pick&Place	Multi-Cube Stacking	Average		
π_0 [6]	75.0	63.0	54.0	30.0	55.5	78.16	100.0
+ FastV [13]	69.0	59.0	47.0	23.0	49.5	58.92	60.6
+ DivPrune [17]	70.0	60.0	47.0	24.0	50.3	61.38	64.8
+ SP-VLA [21]	72.0	61.0	52.0	25.0	52.5	58.40	63.4
+ VLA-ADP [22]	**74.0**	64.0	**54.0**	**26.0**	**54.5**	59.30	47.3
+ **SAPrune**	73.0	**65.0**	53.0	25.0	54.0	**51.45**	**39.3**

MoveNear, and Open/Close. These results further indicate that, under high-ratio token reduction, selecting pruning stages from action-attention dynamics and preserving complementary action-language evidence are critical for maintaining reliable manipulation performance.

4.3 Results on Real-World Environment

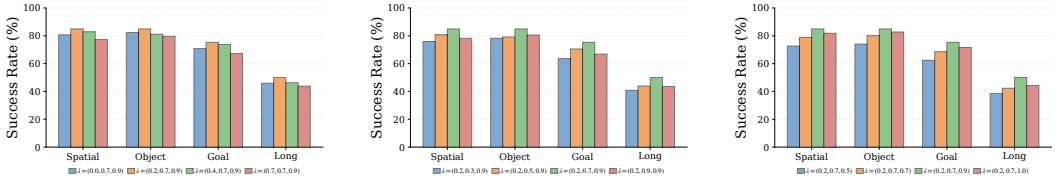
Table 4 reports the real-world manipulation results on AgileX Piper-X with the π_0 [6] backbone. The results show that SAPrune achieves the lowest inference cost among pruned methods while maintaining competitive task success rates. Figure 12 provides qualitative rollouts on the AgileX Piper-X. We also measure end-to-end latency on the same real-robot runs. The complete comparison is shown in Table 9. SAPrune reduces latency from 88.68 ms to 61.97 ms and raises the policy-pipeline frequency from 11.28 to 16.14 Hz, a 30.1% latency reduction. Physical actuation and low-level servo time are excluded.

4.4 Ablation Studies

Table 5: **Ablation on pruning-location strategy under OpenVLA [5] on LIBERO [19].** Red bold marks the best result among pruning variants.

Method	Pruning location	LIBERO Success Rate (%, ↑)					Latency (ms, ↓)	FLOPs Ratio (%, ↓)
		Spatial	Object	Goal	Long	Avg.		
OpenVLA [5]	–	85.3	85.9	76.1	52.3	74.9	178.46	100.0
Uniform-A	[1, 11, 23]	78.2	77.1	67.5	40.2	65.8	104.40	34.7
Uniform-B	[5, 19, 28]	79.1	78.4	69.6	42.5	67.4	117.85	42.7
Single-layer	[2]	68.2	67.1	58.5	26.7	55.1	**95.64**	**25.5**
Random three layers	–	71.5	72.3	64.1	32.9	60.2	119.19	41.1
SAPrune	[2, 16, 26]	**84.9**	**85.1**	**75.3**	**50.8**	**74.0**	110.96	39.5

Effect of pruning-location strategy. Table 5 ablates the choice of pruning locations under OpenVLA [5] on LIBERO [19]. Uniform, random, and single-layer pruning all lead to lower success rates, showing that pruning-layer selection is critical under high-ratio token reduction. In contrast, SAPrune achieves the best success rate among pruning-location variants while still providing substantial acceleration. This indicates that pruning layers should be determined by action-attention dynamics rather than manually fixed or randomly selected.



(a) Ablation at Shallow Layer (b) Ablation at Middle Layer (c) Ablation at Deep Layer

Figure 5: Ablation study on balance factor.

Effect of the balance factor. Figure 5 ablates the layer-wise balance factor λ under OpenVLA [5] on LIBERO [19]. The results show that the preferred allocation changes with depth. A small λ works better in the shallow layer, suggesting that early pruning should keep more surrounding visual context. As the layer becomes deeper, larger λ values become more effective, indicating that attention

becomes a more reliable cue for selecting action-relevant tokens. However, the deep-layer results also show that $\lambda = 1.0$ is not optimal, meaning that fully relying on attention still discards useful supporting information. These trends support the adaptive layer-wise allocation used in SAPrune.

5 Conclusion

We present SAPrune, a training-free visual token pruning framework for efficient VLA inference under aggressive token reduction. Instead of treating pruning as a static token-ranking problem, SAPrune addresses two practical questions for VLA deployment: when pruning should be applied, and which visual tokens can be safely removed. The proposed pruning-layer selection avoids arbitrary layer choices by using rollout-derived action-to-visual attention statistics to identify more reliable pruning stages. At each selected layer, the dual-path pruning rule removes redundant visual tokens while reducing the risk of pruning strongly attended action cues or useful surrounding context. This design requires no retraining or architectural modification, making SAPrune compatible with different VLA backbones.

Acknowledgments and Disclosure of Funding

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), and the 2024 Nansha District Science and Technology Program, Guangzhou, China (Grant No. 2024ZD005).

Competing Interests. The authors declare no competing interests.

References

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

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

[3] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[4] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, 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*, Delft, Netherlands, 2024.

[5] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

[7] Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. *arXiv preprint arXiv:2502.02175*, 2025.

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

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

[10] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

[11] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. *arXiv preprint arXiv:2202.07800*, 2022.

[12] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

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

[14] Kai Huang, Hao Zou, Ye Xi, BoChen Wang, Zhen Xie, and Liang Yu. Ivtp: Instruction-guided visual token pruning for large vision-language models. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XVII*, pages 214–230, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-72642-2. doi: 10.1007/978-3-031-72643-9_13. URL `https://doi.org/10.1007/978-3-031-72643-9_13`.

[15] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. In *International Conference on Machine Learning*, 2025.

[16] Yizheng Sun, Yanze Xin, Hao Li, Jingyuan Sun, Chenghua Lin, and Riza Theresa Batista-Navarro. Lvpruning: An effective yet simple language-guided vision token pruning approach for multi-modal large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4299–4308, 2025.

[17] Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9392–9401, 2025.

[18] Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting visual cues for effective token pruning in vlms. *arXiv preprint arXiv:2412.01818*, 2025.

[19] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[20] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

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

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

[23] Lei Jiang, Weizhe Huang, Tongxuan Liu, Yuting Zeng, Jing Li, Lechao Cheng, and Xiaohua Xu. Fopru: Focal pruning for efficient large vision-language models. *arXiv preprint arXiv:2411.14164*, 2024.

[24] Jizhihui Liu, Guangdao Zhu, and Feiyi Du. Hiprune: Training-free visual token pruning via hierarchical attention in vision-language models (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 41275–41277, 2026.

[25] Pu Zhang, Yuwei Li, Xingyuan XIAN, and Guoming Tang. ZSPAPrune: Zero-shot prompt-aware token pruning for vision-language models, 2026. URL `https://openreview.net/forum?id=5Y8PMEeAkv`.

[26] Jinlong Li, Liyuan Jiang, Haonan Zhang, and Nicu Sebe. Token reduction via local and global contexts optimization for efficient video large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10451–10461, 2026.

[27] Xin Zou, Di Lu, Yizhou Wang, Yibo Yan, Yuanhuiyi Lyu, Xu Zheng, Linfeng Zhang, and Xuming Hu. Don't just chase" highlighted tokens" in mllms: Revisiting visual holistic context retention. *arXiv preprint arXiv:2510.02912*, 2025.

[28] Xiwen Chen, Wenhui Zhu, Gen Li, Xuanzhao Dong, Yujian Xiong, Hao Wang, Peijie Qiu, Qingquan Song, Zhipeng Wang, Shao Tang, et al. Otprune: Distribution-aligned visual token pruning via optimal transport. *arXiv preprint arXiv:2602.20205*, 2026.

[29] Weihao Ye, Qiong Wu, Wenhao Lin, and Yiyi Zhou. Fit and prune: Fast and training-free visual token pruning for multi-modal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 22128–22136, 2025.

[30] Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 5334–5342, 2025.

[31] Jaewoo Lee, Keyang Xuan, Chanakya Ekbote, Sandeep Polisetty, Yi R Fung, and Paul Pu Liang. Tamp: Token-adaptive layerwise pruning in multimodal large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6892–6908, 2025.

[32] Kaiyuan Li, Xiaoyue Chen, Chen Gao, Yong Li, and Xinlei Chen. Balanced token pruning: Accelerating vision language models beyond local optimization. *arXiv preprint arXiv:2505.22038*, 2025.

[33] Sihan Yang, Runsen Xu, Chenhang Cui, Tai Wang, Dahua Lin, and Jiangmiao Pang. Vflowopt: A token pruning framework for lmms with visual information flow-guided optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23924–23934, 2025.

[34] 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.

[35] 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*, 2025. doi: 10.15607/RSS.2025.XXI.017.

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

[37] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 38574–38583, 2026.

A Method Details

A.1 Algorithm Details

We detail SAPrune in Algorithm 1 and Algorithm 2

Algorithm 1 Stage-Aware Pruning-Layer Selection

Input: Calibration action-to-visual attentions $\{a^{(l,t)}\}$, $\kappa, q_s, q_r, m, l_{\min}$
Output: Pruning layers $\mathcal{L}_{\text{prune}} = \{l_1, l_2, l_3\}$

Compute rollout-averaged top-κ attention masses $\{\mu_l\}_{l=0}^{L-1}$
Smooth $\{\mu_l\}_{l=0}^{L-1}$ to obtain $\{\tilde{\mu}_l\}_{l=0}^{L-1}$
Compute inter-layer shifts $d_l = |\tilde{\mu}_{l+1} - \tilde{\mu}_l|$ for $l = 0, \ldots, L-2$
$\tau_s \leftarrow Q_{q_s}(\{d_l\}_{l=0}^{L-2}), \quad \tau_r \leftarrow Q_{q_r}(\{d_l\}_{l=0}^{L-2})$
Detect plateau starts $\mathcal{P}$ using threshold τ_s and length m
Detect redistribution events $\mathcal{E}$ using threshold τ_r
Partition candidate pruning layers into $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$
for $r = 1, 2, 3$ **do**
    if $r = 1$ **then**
        $\bar{\mathcal{R}}_r \leftarrow \{p \in \mathcal{R}_1 \mid p \geq l_{\min}\}$
        if $\bar{\mathcal{R}}_r = \emptyset$ **then**
            $\bar{\mathcal{R}}_r \leftarrow \mathcal{R}_1$
        end if
    else
        $\mathcal{E}_r \leftarrow \{[a, b] \in \mathcal{E} \mid [a, b] \cap \mathcal{R}_r \neq \emptyset\}$
        if $\mathcal{E}_r \neq \emptyset$ **then**
            $[a_r^\star, b_r^\star] \leftarrow \arg\max_{[a,b] \in \mathcal{E}_r} \sum_{u=a}^{b} d_u$
            $\bar{\mathcal{R}}_r \leftarrow \{p \in \mathcal{R}_r \mid p > b_r^\star\}$
            if $\bar{\mathcal{R}}_r = \emptyset$ **then**
                $\bar{\mathcal{R}}_r \leftarrow \mathcal{R}_r$
            end if
        else
            $\bar{\mathcal{R}}_r \leftarrow \mathcal{R}_r$
        end if
    end if
end for
for $r = 1, 2, 3$ **do**
    if $\mathcal{P} \cap \bar{\mathcal{R}}_r \neq \emptyset$ **then**
        $l_r \leftarrow \min(\mathcal{P} \cap \bar{\mathcal{R}}_r)$
    else
        $l_r \leftarrow \arg\min_{p \in \bar{\mathcal{R}}_r} \frac{\sum_{u=p}^{\min(p+m-1, L-2)} d_u}{\min(p+m-1, L-2) - p + 1}$
    end if
end for
return $\mathcal{L}_{\text{prune}} = \{l_1, l_2, l_3\}$

A.2 Complexity Analysis

We first estimate the multi-head attention and FFN FLOPs in the Transformer backbone. The equations below describe this backbone cost; the reported total FLOPs also include routing and token selection.

Let L be the number of Transformer layers, N_0 the original number of visual tokens, and M_{nv} the number of non-visual tokens, including text and action-related tokens. The initial sequence length is

$$n_0 = M_{\text{nv}} + N_0. \tag{17}$$

The selected pruning layers l_1, l_2, l_3 in the main text use 0-based indices. For FLOPs accounting, we convert them to 1-based layer indices:

$$b_i = l_i + 1, \qquad i \in \{1, 2, 3\}, \tag{18}$$

with $1 \leq b_1 < b_2 < b_3 < L$. Let $K_{l_1}, K_{l_2}, K_{l_3}$ denote the retained numbers of visual tokens after the three pruning stages, and define

$$n_1 = M_{\text{nv}} + K_{l_1}, \qquad n_2 = M_{\text{nv}} + K_{l_2}, \qquad n_3 = M_{\text{nv}} + K_{l_3}. \tag{19}$$

Algorithm 2 Layer-Local Dual-Path Visual Token Pruning

Input: Visual tokens $V_t^{(l)}$, text states $X^{(l)}$, action query $q_a^{(l)}$, action attention $a^{(l)}$, keep budget K_l
Output: Retained visual-token set S_l

Construct candidate pool $C_l \leftarrow \mathrm{TopM}(a^{(l)}, \lfloor \gamma K_l \rfloor)$
Construct probe set $A_l^{\mathrm{probe}} \leftarrow \mathrm{TopK}(a^{(l)}, \lfloor \rho K_l \rfloor)$
Compute preservation score P_l from the probe subspace
Compute attention-concentration score A_l over C_l
$\eta_l \leftarrow \sqrt{P_l A_l}$
$\lambda_l \leftarrow (1-\eta_l)\lambda_{\min} + \eta_l \lambda_{\max}$
$K_l^{\mathrm{attn}} \leftarrow \lfloor \lambda_l K_l \rfloor, \quad K_l^{\mathrm{cov}} \leftarrow K_l - K_l^{\mathrm{attn}}$
Attention-core path
$S_l^{\mathrm{attn}} \leftarrow \mathrm{TopK}_{C_l}(a^{(l)}, K_l^{\mathrm{attn}})$
$R \leftarrow S_l^{\mathrm{attn}}$
Functional-diversity path
Select top text queries $Q_x^{(l)}$ by action-to-text attention
$Q_l \leftarrow \{q_a^{(l)}\} \cup Q_x^{(l)}$
Compute functional signatures $\{r_i^{(l)}\}_{i \in C_l}$ using Q_l
$S_l^{\mathrm{cov}} \leftarrow \emptyset$
while $|S_l^{\mathrm{cov}}| < K_l^{\mathrm{cov}}$ and $C_l \setminus R \neq \emptyset$ **do**
    Compute residual functional evidence $m_R^{(l)}$
    Select

$$i^\star \leftarrow \arg\max_{i \in C_l \setminus R} \left[\langle \hat{r}_i^{(l)}, \hat{m}_R^{(l)} \rangle - \beta \max_{j \in R} \cos\left(r_i^{(l)}, r_j^{(l)} \right) \right]$$

    $S_l^{\mathrm{cov}} \leftarrow S_l^{\mathrm{cov}} \cup \{i^\star\}$
    $R \leftarrow R \cup \{i^\star\}$
end while
$S_l \leftarrow S_l^{\mathrm{attn}} \cup S_l^{\mathrm{cov}}$
if $|S_l| < K_l$ **then**
    Refill remaining slots from unused candidates by action-attention rank
end if
return S_l

For a Transformer layer with sequence length n, hidden size d, and FFN intermediate size d_{ff}, we approximate its FLOPs as

$$C(n) = 4nd^2 + 2n^2 d + 2nd\, d_{\mathrm{ff}}. \tag{20}$$

Without pruning, the total backbone FLOPs are

$$\mathrm{FLOPs}_{\mathrm{full}} = L\, C(n_0). \tag{21}$$

In SAPrune, pruning is applied after the selected layer outputs. Therefore, the pruning at layer l_i affects only subsequent layers. The first b_1 layers use n_0, layers b_1+1 to b_2 use n_1, layers b_2+1 to b_3 use n_2, and layers b_3+1 to L use n_3. Thus,

$$\mathrm{FLOPs}_{\mathrm{prune}} = b_1 C(n_0) + (b_2 - b_1) C(n_1) + (b_3 - b_2) C(n_2) + (L - b_3) C(n_3). \tag{22}$$

The theoretical FLOPs ratio is

$$r_{\mathrm{FLOPs}} = \frac{\mathrm{FLOPs}_{\mathrm{prune}}}{\mathrm{FLOPs}_{\mathrm{full}}}. \tag{23}$$

In the main setting, SAPrune uses stage-wise 50% pruning:

$$K_{l_1} = 128, \qquad K_{l_2} = 64, \qquad K_{l_3} = 32, \tag{24}$$

starting from $N_0 = 256$ visual tokens. The final retained-token ratio is therefore $32/256 = 12.5\%$, corresponding to an overall pruning ratio of 87.5%. Token selection is performed only at the three pruning layers and over restricted candidate sets.

The reported inference latency includes the selected action- and text-query attention computation, routing, token selection, indexing, and gather operations. On OpenVLA, SAPrune uses 39.5% of the full-model FLOPs, including pruning costs, while latency falls from 178.46 ms to 110.96 ms. Thus, a 60.5% FLOPs reduction gives a 37.8% latency reduction (1.608× speedup). Memory access, kernel launches, and other runtime costs do not decrease in direct proportion to FLOPs. The attention-backend comparison and real-robot policy-pipeline breakdown are reported in Appendix B.4.3 and Table 9, respectively.

B Experimental Setup

B.1 VLA Model Details

We evaluate SAPrune on representative VLA backbones, including OpenVLA, OpenVLA-OFT, π_0, and $\pi_{0.5}$. These models cover both autoregressive action-token policies and flow-based continuous-action policies, allowing us to examine whether the proposed pruning strategy generalizes across different action-generation mechanisms.

OpenVLA. OpenVLA [5] is a 7B open-source vision-language-action model adapted from a pretrained vision-language model for robotic control. It encodes visual observations and language instructions, and predicts discretized robot actions through autoregressive token generation. We use OpenVLA as the representative token-based VLA backbone.

OpenVLA-OFT. OpenVLA-OFT [35] is an optimized fine-tuning variant of OpenVLA. It improves the original action-generation recipe with parallel decoding, action chunking, continuous action representation, and regression-based training objectives. Compared with the original OpenVLA, it provides stronger control performance and higher inference efficiency while preserving the same general VLA formulation. We use OpenVLA-OFT as a stronger OpenVLA-family backbone.

π_0 **Model.** π_0 [6] is a flow-based vision-language-action model for general robot control. Instead of decoding discretized action tokens autoregressively, it augments a pretrained vision-language model with a continuous-action flow-matching module. This makes π_0 representative of continuous-action VLA policies and complements the OpenVLA-family models in our evaluation.

$\pi_{0.5}$ **Model.** $\pi_{0.5}$ [36] is a newer VLA model designed for open-world generalization. We evaluate it on LIBERO and LIBERO-Plus using the same calibration and pruning procedure, without manual pruning-layer tuning.

B.2 Acceleration Baseline Details

We compare SAPrune with representative training-free or plug-and-play acceleration baselines, including FastV, DivPrune, SP-VLA, VLA-ADP, and EfficientVLA. These methods cover attention-based pruning, diversity-based pruning, structured VLA pruning, adaptive pruning, and progressive VLA acceleration.

FastV. FastV [13] is a training-free attention-based pruning method for large vision-language models. It ranks visual tokens by attention-derived importance and removes low-score tokens after a selected layer. Since it relies mainly on attention relevance, we use it as an attention-only pruning baseline.

DivPrune. DivPrune [17] selects visual tokens by encouraging feature-space diversity among the retained set. It reduces redundancy by preserving visually dissimilar tokens, but its diversity criterion is not explicitly conditioned on action generation. We use it as a representative diversity-based pruning baseline.

SP-VLA. SP-VLA [21] is a VLA-oriented acceleration method that combines model scheduling with token pruning. Compared with generic VLM pruning methods, it is more tailored to VLA inference. We include it as a structured VLA pruning baseline.

VLA-ADP. VLA-ADP [22] performs action-aware dynamic pruning for efficient VLA manipulation. It adaptively adjusts token retention during inference rather than using a completely fixed pruning rule. We use it as a representative adaptive VLA pruning baseline.

EfficientVLA. EfficientVLA [8] is a recent training-free acceleration framework for VLA models. Its pruning component combines token relevance with feature-space diversity in a progressive pruning pipeline. We include EfficientVLA, especially its text-guided variant EfficientVLA(T-only), as a strong relevance-plus-diversity baseline.

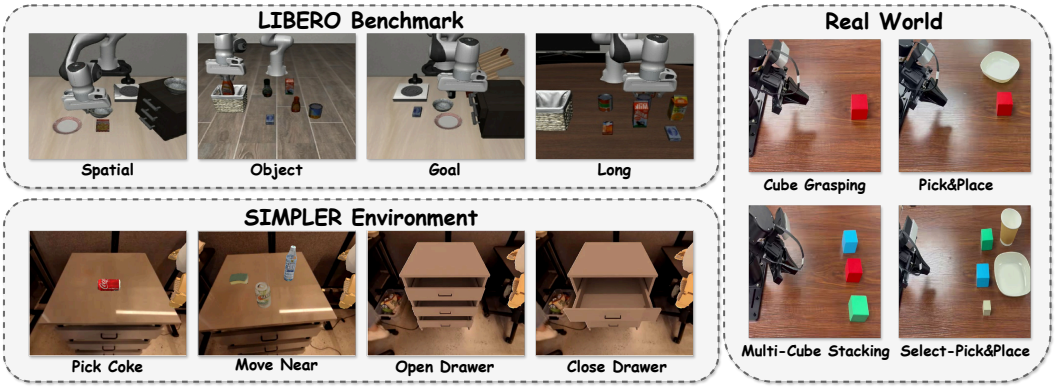


Figure 6: Tasks on LIBERO Benchmark, SIMPLER and Real World Environment.

Evaluation protocol. We report two groups of comparisons. In the default-budget setting, each baseline is evaluated under its reported or recommended token budget, while the original models use 256 visual tokens and SAPrune keeps 32 visual tokens. This setting reflects the practical configuration of each method. In the controlled-budget setting, we fix the final retained-token budget to 32 for all applicable pruning methods, so that performance differences mainly reflect the token-selection strategy rather than the number of remaining visual tokens.

B.3 Benchmarks and Tasks

We evaluate SAPrune on three groups of benchmarks: LIBERO [19], SIMPLER [20], and a set of real-world robotic manipulation tasks, as shown in Figure 6. These benchmarks cover both simulation and real-robot settings, and include tasks with different horizons, scene complexity, and semantic requirements.

LIBERO. LIBERO [19] is a language-conditioned manipulation benchmark built on Robosuite. We follow the standard four-suite protocol and evaluate on **Spatial**, **Object**, **Goal**, and **Long**, with 10 tasks in each suite. Spatial focuses on spatial relations under varying object configurations. Object changes the target object while keeping the task template relatively fixed. Goal evaluates goal-conditioned manipulation with different final states. Long contains multi-stage long-horizon tasks and is more sensitive to accumulated execution errors.

SIMPLER. SIMPLER [20] is a simulation benchmark designed to better reflect real-robot manipulation performance. We evaluate on four Google-Robot-style tasks: **Pick Coke**, **Move Near**, **Open Drawer**, and **Close Drawer**. These tasks cover common short-horizon manipulation primitives, including grasping, local object repositioning, and articulated-object interaction. SIMPLER complements LIBERO by providing visually grounded tabletop tasks with a closer connection to real-robot evaluation.

Real-world tasks. We further conduct real-world experiments on an AgileX Piper-X. The real-world suite includes **Cube Grasping**, **Pick&Place**, **Select-Pick&Place**, and **Multi-Cube Stacking**. Cube Grasping evaluates basic target localization and grasp execution. Pick&Place tests object transport and placement. Select-Pick&Place introduces distractor objects and requires target selection before manipulation. Multi-Cube Stacking requires repeated grasping and placement, and is the most sequential task in this suite. These real-world tasks test whether aggressive visual-token pruning remains reliable under practical sensing and execution conditions.

B.4 More results

B.4.1 Controlled Comparison Under a Unified 32-Token Budget

Table 6 compares all applicable pruning methods on LIBERO under the same final budget of 32 visual tokens. SAPrune consistently achieves the strongest task performance across OpenVLA, OpenVLA-OFT, and π_0, indicating that its advantage comes from stage-aware pruning-layer placement and higher-quality token retention rather than from using more final tokens. SAPrune can be slightly slower than some baselines because it delays pruning until more reliable stages and reduces visual tokens progressively across selected layers, so earlier layers still process more tokens than early

one-step pruning methods. Nevertheless, it preserves substantially higher success rates while still accelerating the full-token model, yielding the best overall efficiency-performance balance under the same token budget.

Table 6: **Controlled comparison under a unified 32-token budget on LIBERO [19].** All applicable pruning methods are evaluated with the same final retained-token budget of 32 visual tokens. Red bold marks the best result among pruned methods within each backbone.

Method	Tokens	LIBERO Success Rate (%, ↑)					Latency (ms, ↓)	Speedup (↑)	FLOPs Ratio (%, ↓)
		Spatial	Object	Goal	Long	Avg.			
OpenVLA [5]									
OpenVLA	256	85.3	85.9	76.1	52.3	74.9	178.46	1.000×	100.0
+ FastV [13]	32	55.9	52.0	51.2	18.1	44.3	102.84	1.735×	29.9
+ DivPrune [17]	32	50.1	49.8	47.8	14.9	40.7	103.76	1.720×	33.0
+ SP-VLA [21]	32	68.0	69.5	63.4	30.0	57.7	101.32	1.761×	32.1
+ VLA-ADP [22]	32	72.2	68.4	62.9	34.0	59.4	100.66	1.773×	**28.4**
+ EfficientVLA(T-only) [8]	32	70.6	69.0	60.4	31.6	57.9	**99.58**	**1.792×**	32.0
+ **SAPrune**	32	**84.9**	**85.1**	**75.3**	**50.8**	**74.0**	110.96	1.608×	39.5
OpenVLA-OFT [35]									
OpenVLA-OFT	256	95.5	98.8	96.6	90.7	95.4	103.59	1.000×	100.0
+ FastV [13]	32	63.8	60.7	61.0	31.1	54.2	56.75	1.825×	25.5
+ DivPrune [17]	32	60.9	62.2	62.9	29.3	53.8	55.88	1.854×	27.9
+ SP-VLA [21]	32	72.0	65.3	71.8	43.7	63.2	54.23	1.910×	26.3
+ VLA-ADP [22]	32	80.3	73.5	79.4	51.7	71.2	54.71	1.893×	**24.6**
+ EfficientVLA(T-only) [8]	32	76.2	74.9	77.1	54.2	70.6	**53.68**	**1.930×**	25.7
+ **SAPrune**	32	**95.2**	**98.1**	**95.9**	**88.9**	**94.5**	60.30	1.718×	37.8
π_0 **[6]**									
π_0	256	96.9	96.3	95.2	87.2	93.9	78.49	1.000×	100.0
+ FastV [13]	32	67.2	61.4	51.5	29.3	52.4	47.15	1.665×	**30.1**
+ DivPrune [17]	32	60.0	65.5	64.8	32.0	55.6	47.53	1.651×	35.0
+ SP-VLA [21]	32	78.2	69.7	60.6	47.8	64.1	46.66	1.682×	32.2
+ VLA-ADP [22]	32	78.0	77.8	80.0	56.3	73.0	**46.18**	**1.700×**	31.4
+ **SAPrune**	32	**96.2**	**95.8**	**94.4**	**86.0**	**93.1**	51.22	1.532×	38.6

B.4.2 Routing Overhead

Table 7 reports the wall-clock cost of Algorithm 2 at the three pruning stages. The total routing overhead is 4.76 ms on OpenVLA, with the functional-signature construction and greedy coverage-selection component accounting for the largest portion (3.09 ms). The stage-wise cost decreases from 2.72 ms to 1.35 ms and 0.69 ms as the visual-token set is progressively reduced. This indicates that the additional computation introduced by the dual-path selection becomes increasingly lightweight at later pruning stages.

Table 7: **Detailed wall-clock latency breakdown of Algorithm 2 across the three pruning stages of SAPrune on OpenVLA.** We report the computational overhead of candidate ranking, reliability estimation, functional-signature construction and greedy coverage selection, and token indexing at each pruning stage. All latency values are measured in milliseconds, with the final column reporting the cumulative overhead across all three stages.

Component	**Shallow 256→128**	**Middle 128→64**	**Deep 64→32**	**Total**
Candidate ranking / Top-M	0.08	0.05	0.04	0.17
Reliability computation (P_l, A_l, λ_l)	0.49	0.32	0.17	0.98
Functional signature + greedy selection	1.90	0.82	0.37	3.09
Token indexing / gather	0.25	0.16	0.11	0.52
Algorithm 2 total	**2.72**	**1.35**	**0.69**	**4.76**

B.4.3 Attention Backend

Table 8 compares eager attention, PyTorch SDPA, and FlashAttention-2. SAPrune achieves similar speedups across the three tested backends, showing that its gains can be retained with standard attention optimizations.

Table 8: **Compatibility and inference efficiency of SAPrune across different attention backends.** We report the inference latency and speedup on OpenVLA, OpenVLA-OFT, and π_0 using Eager, SDPA, and FlashAttention-2. All latency values are measured in milliseconds.

Backbone	Attention backend	Full-token latency (ms)	SAPrune latency (ms)	Speedup
OpenVLA	Eager	182.92	112.62	1.624×
OpenVLA	SDPA (default)	178.46	110.96	1.608×
OpenVLA	FlashAttention-2	177.75	110.74	1.605×
OpenVLA-OFT	Eager	105.87	61.20	1.730×
OpenVLA-OFT	SDPA (default)	103.59	60.30	1.718×
OpenVLA-OFT	FlashAttention-2	103.18	60.18	1.715×
π_0	Eager	80.06	51.83	1.545×
π_0	SDPA (default)	78.49	51.22	1.532×
π_0	FlashAttention-2	78.18	51.12	1.529×

B.4.4 End-to-end latency on real robots.

Table 9 provides a breakdown of the real-robot policy pipeline latency on π_0. SAPrune reduces the VLA inference time from 78.16 ms to 51.45 ms, while the remaining pipeline components introduce similar overhead across methods. This leads to an E2E latency reduction from 88.68 ms to 61.97 ms and increases the policy-pipeline frequency from 11.28 Hz to 16.14 Hz, showing that token pruning provides practical acceleration beyond isolated model inference speedups.

Table 9: **Real-robot policy-pipeline latency and E2E frequency on π_0.** E2E frequency is computed as $f_{\mathrm{E2E}} = 1000/T_{\mathrm{E2E}}$. E2E latency also includes 0.61–0.62 ms of orchestration and synchronization overhead. Red bold marks the best result among pruned methods.

Method	Preprocess (ms)	VLA inference (ms)	Action decoding (ms)	Communication (ms)	E2E latency (ms)	E2E frequency (Hz)
π_0 [6]	6.08	78.16	2.46	1.36	88.68	11.28
+ FastV [13]	6.05	58.92	2.44	1.35	69.37	14.42
+ SP-VLA [21]	6.04	58.40	2.47	1.37	68.89	14.52
+ VLA-ADP [22]	6.07	59.30	2.48	1.36	69.83	14.32
+ **SAPrune**	6.10	51.45	2.46	1.35	**61.97**	**16.14**

B.4.5 Calibration Transfer

We further examine whether the calibration procedure transfers across environments. Specifically, we calibrate OpenVLA using SIMPLER or real-world rollouts and directly evaluate the resulting layer selections on LIBERO-Long. No LIBERO calibration samples are used in these transfer experiments, and all other settings are kept unchanged; in particular, we perform neither retraining nor additional recalibration.

Table 10: **Calibration transfer on OpenVLA.** Calibration is performed on different data sources and evaluated on LIBERO-Long without LIBERO-specific recalibration.

Calibration	Layers	Long SR (%)
LIBERO (original)	[2,16,26]	50.8
SIMPLER ($N = 30$)	[2,15,26]	49.3
Real-world ($N = 60$)	[2,17,26]	**51.1**
Real-world ($N = 30$)	[2,17,27]	47.6

As shown in Table 10, calibration transferred from SIMPLER or real-world data yields success rates of 47.6–51.1%, compared with 50.8% when calibration is performed directly on LIBERO. Notably, calibration with 60 real-world rollouts achieves 51.1% success, showing that the selected pruning layers can remain effective when the calibration and evaluation environments differ. Here, N denotes the number of calibration rollouts, and all layer indices are 0-based.

When increasing the calibration set to 120 rollouts, LIBERO, SIMPLER, real-world, and mixed-source data all select the same layers, $[2, 16, 26]$. The mixed-source calibration also selects $[2, 16, 26]$ with only 30 or 60 rollouts. These results indicate that, for a fixed model, the selected pruning layers

are relatively stable across the tested calibration sources and can be reused across downstream tasks. A new model still requires a separate calibration procedure, and transfer to previously unseen settings should be validated empirically.

B.4.6 Robustness on LIBERO-Plus

We evaluate $\pi_{0.5}$ and SAPrune on LIBERO-Plus [37] under seven distribution shifts. SAPrune is calibrated once on 300 trajectories from standard LIBERO, with no LIBERO-Plus data used for calibration or recalibration. Both methods are evaluated once on each of the same 10,030 task variants under identical settings.

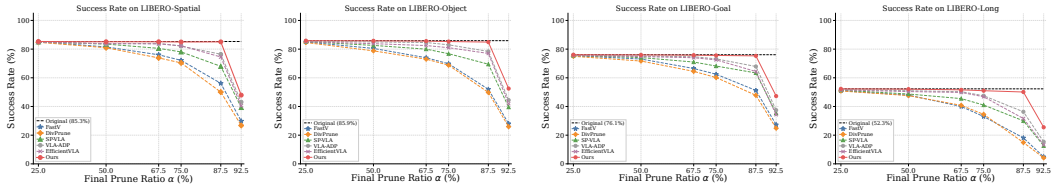
Table 11: **Success rates (%) under LIBERO-Plus distribution shifts.** Overall is computed over all 10,030 task variants, accounting for different category sizes. Tokens denotes visual tokens per image.

Method	Tokens	Camera	Robot	Language	Light	Background	Noise	Layout	Overall
$\pi_{0.5}$ [36]	256	75.7	77.1	85.9	96.4	94.9	90.1	85.2	85.7
+ **SAPrune**	32	74.5	76.1	84.8	95.6	93.8	88.6	83.9	84.5

Table 11 shows that SAPrune achieves 84.5% overall success with 32 tokens, compared with 85.7% for the full model with 256 tokens. The largest category gap is 1.5 percentage points under noise, indicating only a small additional loss under the tested shifts.

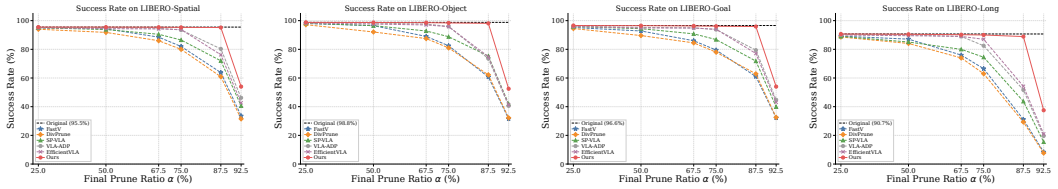
B.4.7 Performance under Different Final Prune Ratios

Figures 7, 8, and 9 report the success rates on the four LIBERO suites as the final prune ratio increases on OpenVLA, OpenVLA-OFT, and π_0, respectively. Across all three backbones, a consistent trend can be observed: as pruning becomes more aggressive, the performance of all methods gradually degrades, with the most pronounced drops appearing in the high-pruning regime. Compared with the competing baselines, **SAPrune** maintains stronger performance over a wide range of prune ratios and shows better robustness especially under aggressive token reduction.



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

Figure 7: **Performance under different final prune ratios on OpenVLA.** Success rates on the four LIBERO suites are reported as the final prune ratio increases. Our method consistently preserves stronger performance, especially under aggressive pruning regimes.

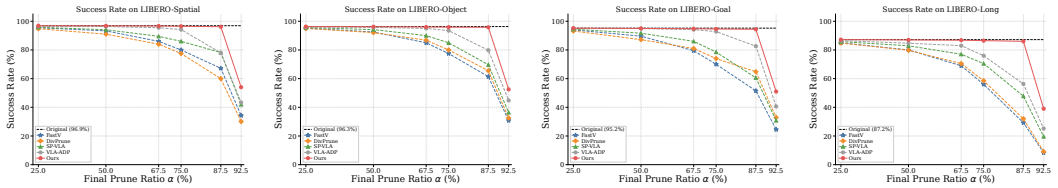


(a) LIBERO-Spatial (b) LIBERO-Object (c) LIBERO-Goal (d) LIBERO-Long

Figure 8: **Performance under different final prune ratios on OpenVLA-OFT.** Success rates on the four LIBERO suites are reported as the final prune ratio increases. Our method remains the most robust among the compared pruning methods across a wide range of pruning levels.

B.5 More Ablation

Table 12 ablates the two token-retention paths under the same pruning layers and 32-token budget. Removing either path consistently degrades performance, showing that both components are necessary



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

Figure 9: **Performance under different final prune ratios on π_0.** Success rates on the four LIBERO suites are reported as the final prune ratio increases. The results show that our method better maintains policy performance as pruning becomes more aggressive.

for aggressive visual-token pruning. The attention-core path contributes the larger gain, indicating that highly action-attended tokens are the most reliable evidence for control. The functional-diversity path still provides clear additional benefit by preserving complementary context that attention-only selection may miss, such as surrounding objects, spatial anchors, and instruction-conditioned cues.

Table 12: **Ablation study of token-retention paths on LIBERO [19].** All variants use the same stage-aware pruning layers and the same 32-token budget; only the retention rule is changed. Red bold marks the best success rate among pruning variants.

Variant	LIBERO Success Rate (%, ↑)					Latency (ms, ↓)	FLOPs Ratio (%, ↓)
	Spatial	Object	Goal	Long	Avg.		
Full SAPrune	**84.9**	**85.1**	**75.3**	**50.8**	**74.0**	110.96	39.5
w/o Attention core	75.9	75.1	65.7	39.0	63.9	111.47	40.2
w/o Functional diversity	78.1	77.6	68.9	43.2	67.0	107.93	39.1

B.5.1 More analysis

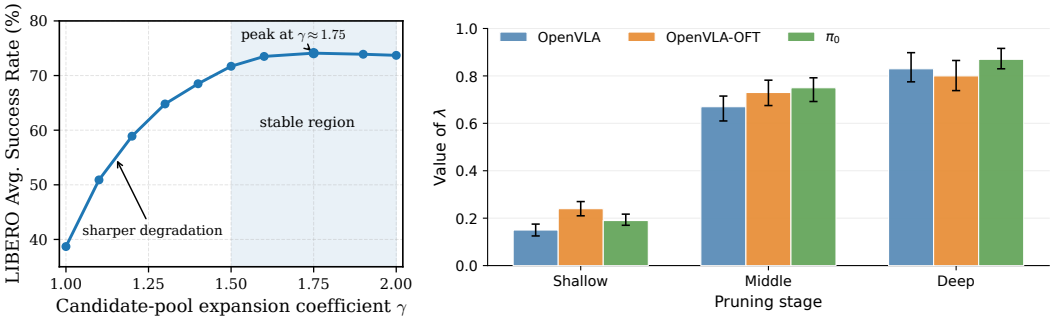
Figure 10a shows the effect of the candidate-pool expansion coefficient γ on the average LIBERO success rate. When γ is too small, the candidate pool is overly restrictive and may exclude potentially useful visual tokens, leading to clear performance degradation. As γ increases, the performance improves steadily, since a larger candidate pool provides more room for complementary evidence selection. The curve reaches its peak around $\gamma \approx 1.75$ and then remains relatively stable, suggesting that once the candidate pool is sufficiently expressive, further expansion brings only marginal benefit. Based on this observation, we use $\gamma = 1.75$ in our experiments.

B.5.2 Sensitivity and allocation analysis.

Figure 10 shows two overall trends. First, the average LIBERO success rate increases as the candidate-pool expansion coefficient γ grows, and then gradually enters a stable region when γ becomes moderately large. Second, the allocation coefficient λ shows a clear depth-wise increasing pattern across different VLA backbones: it remains small in shallow layers, becomes larger in middle layers, and reaches the highest values in deep layers.

C Limitation

SAPrune is built on the empirically supported premise that action-to-visual attention can serve as a useful proxy for action-relevant visual evidence. Our experiments suggest that this premise holds well for sufficiently trained VLA backbones whose action-related states are grounded in task-relevant visual regions. Under such conditions, SAPrune is better aligned with VLA inference than generic visual-token pruning methods because it delays pruning until action-conditioned attention becomes stable and preserves both dominant and complementary visual evidence. However, this premise may become weaker when the backbone attention is poorly grounded, remains diffuse, or is dominated by spurious background regions. In such cases, the attention signal may not accurately reflect the causal contribution of visual tokens, and SAPrune may still introduce performance degradation, especially in scenarios involving out-of-distribution objects, fine-grained contact regions, heavy occlusion, or long-horizon manipulation.



(a) Sensitivity to candidate-pool size

(b) Layer-wise allocation coefficient

Figure 10: Sensitivity and allocation analysis of SAPrune. (a) The average LIBERO success rate increases with the candidate-pool expansion coefficient γ and becomes stable when γ is moderately large. (b) The layer-wise allocation coefficient λ remains small in shallow layers and increases in middle and deep layers.

SAPrune also requires visual and action-related representations to interact within the Transformer being pruned. This holds for the implementations evaluated here. In separated vision-language model (VLM) and diffusion Transformer (DiT) systems, action-conditioned queries may be available only in the downstream action module. Direct use in the VLM then requires an adaptation across modules, which we have not evaluated. Pruning the VLM conditioning sequence would not remove the remaining DiT denoising computation, limiting the possible end-to-end speedup.

D Impact Statement

This work aims to improve the efficiency of vision-language-action models by reducing visual-token computation during inference. More efficient VLA inference can lower latency and deployment cost, potentially making robotic manipulation systems more practical on resource-constrained hardware. At the same time, efficiency improvements may also make robotic automation easier to deploy at larger scale, which raises broader concerns about safety, accountability, and labor displacement. Since token pruning may remove visual evidence needed for reliable control in rare or out-of-distribution situations, SAPrune should be used with appropriate validation, monitoring, and task-specific safety constraints before real-world deployment.

E More Visualization

Real-World Robot Evaluation. Figure 12 shows qualitative real-world rollout comparisons between the full π_0 baseline and SAPrune on four manipulation tasks. Across cube grasping, pick-and-place, multi-cube stacking, and select-pick-and-place, SAPrune maintains similar execution behavior to the baseline under aggressive visual token pruning.

Qualitative Analysis. Figure 11 visualizes SAPrune token pruning results across representative tasks from the LIBERO Spatial, Object, Goal, and Long suites. The retained visual tokens remain concentrated around task-relevant objects and interaction regions across shallow, middle, and deep pruning stages, indicating that SAPrune preserves action-critical visual evidence under aggressive pruning.

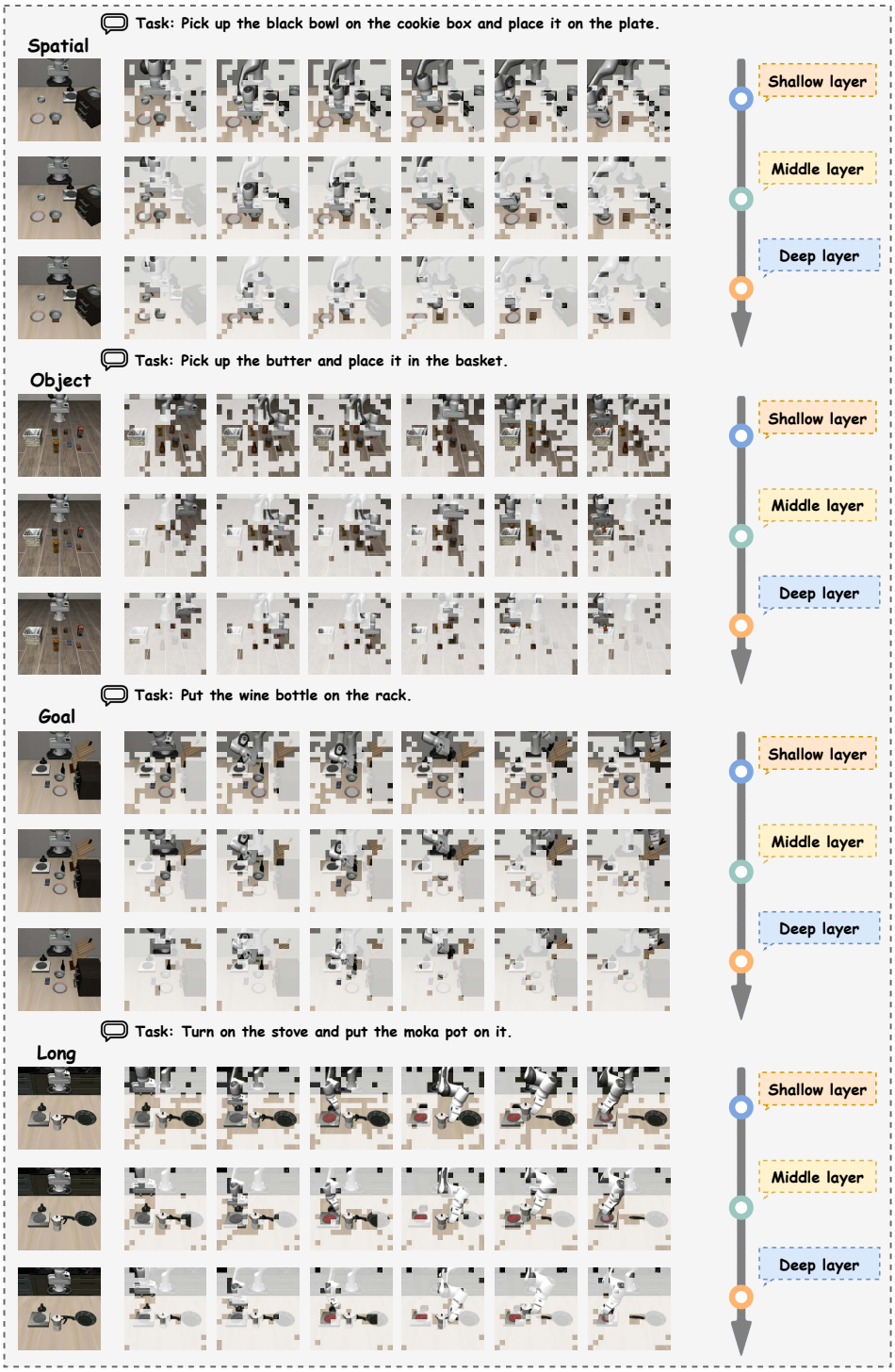


Figure 11: Visualization of SAPrune token pruning results across LIBERO tasks .

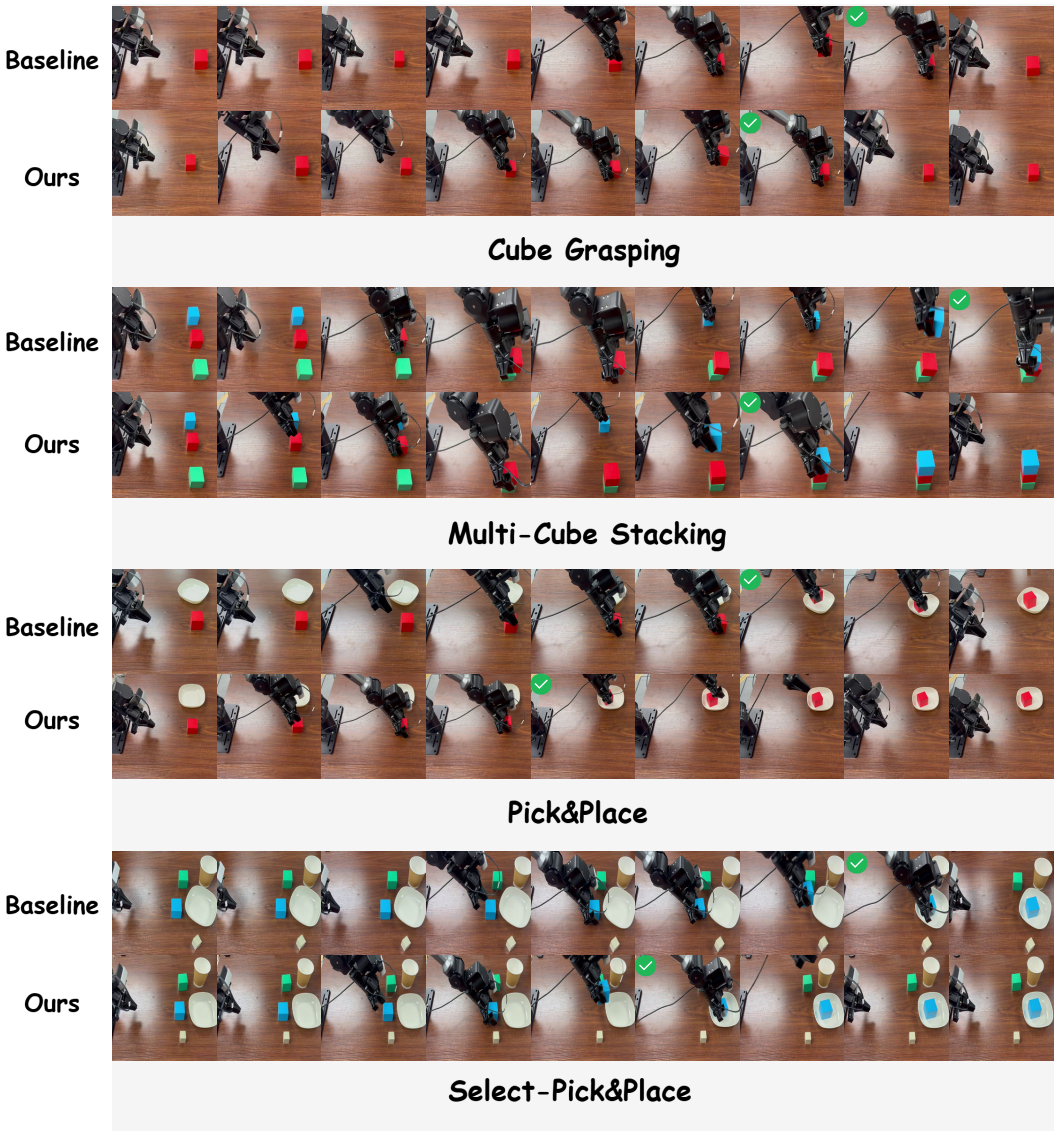


Figure 12: **Real-world rollout comparison between the full π_0 baseline and SAPrune.** We compare baseline and SAPrune rollouts on four real-robot manipulation tasks, including cube grasping, pick-and-place, multi-cube stacking, and select-pick-and-place. Green check marks indicate successful task completion.

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 accurately summarize the paper's main contribution: an inference-time visual token pruning method for efficient VLA inference. The stated claims are limited to the evaluated VLA backbones and manipulation benchmarks, and are supported by the experimental results in Sec. 4.

   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 limitations in Sec. C.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not present formal theoretical results, theorems, or proofs. The equations are used only to describe the overall procedure and implementation workflow.

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 provides the main information needed to reproduce the experimental results, including the evaluated VLA backbones, benchmarks, token budgets, calibration protocol, pruning configuration, and evaluation metrics. These details are described in the method and experiment sections.

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 implementation is not publicly released at the camera-ready stage. We provide the algorithmic procedure and experimental details to support reproducibility.

   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, including the evaluated benchmarks, task suites, VLA models, token budgets, pruning layers, and evaluation metrics. The proposed method is a plug-and-play inference-time pruning approach and does not require additional training or optimizer settings.

   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 success rates and efficiency metrics under fixed evaluation protocols, but does not include error bars, confidence intervals, or statistical significance tests. We omit exhaustive repeated evaluations due to the high cost of VLA rollout evaluation and real-world robot experiments.

   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 reports the compute resource used for latency measurement and describes the runtime evaluation protocol. Since the proposed method is plug-and-play and training-free, it does not require additional training compute; its extra cost mainly comes from the lightweight calibration stage used to select pruning layers.

   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 uses existing VLA backbones, public manipulation benchmarks, and controlled real-world robot experiments, and does not involve human subjects, private data, or sensitive personal information. We have reviewed the NeurIPS Code of Ethics and do not identify any deviation from it.

   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 paper discusses positive impacts such as reducing the computational cost and latency of VLA inference, which can improve energy efficiency and deployment accessibility. It also notes that aggressive pruning may lead to a partial drop in task success rates, especially on long-horizon tasks, and therefore should be deployed with appropriate task-level validation and safety checks.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release a new high-risk pretrained model, image generator, scraped dataset, or general-purpose foundation model. The work proposes a plug-and-play inference-time pruning method evaluated on existing VLA backbones and manipulation benchmarks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The paper uses only publicly available existing assets cited in the experimental section and appendix, including public benchmarks, pretrained VLA backbones, and baseline implementations. We credit the original creators through the corresponding citations, follow the released licenses and terms of use of these assets, and use them only for academic evaluation and comparison. No existing datasets, pretrained models, or third-party code are redistributed as part of this submission.

Guidelines:

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

13. **New assets**

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

Answer: [N/A]

Justification: The paper does not release a new dataset, benchmark, pretrained model, or other new research asset. The contribution is a plug-and-play inference-time pruning method evaluated on existing VLA backbones and manipulation benchmarks.

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, user studies, or research with human subjects. The experiments are conducted on simulation benchmarks and controlled robot manipulation tasks.

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 human subjects, crowdsourcing participants, or user studies, so IRB approval or equivalent human-subjects review is not applicable.

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development does not use LLMs as an important, original, or non-standard research component. The paper evaluates existing language-conditioned VLA backbones, but does not use LLMs to generate data, labels, experimental results, or scientific conclusions.

    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.