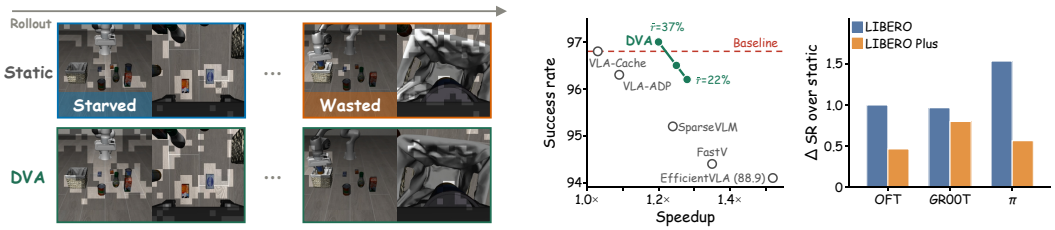
NOT EVERY ACTION NEEDS EVERY TOKEN: DYNAMIC VISUAL TOKEN ALLOCATION FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models repeatedly execute the vision-language backbone at every control step, where processing visual tokens accounts for a dominant portion of the inference cost. Existing visual-token pruning methods accelerate inference by applying a fixed retention ratio throughout deployment. However, we observe that sensitivity to sustained token reduction varies across tasks and control steps within a rollout. A uniform ratio can therefore miss opportunities to reduce computation while preserving task performance. To address this issue, we propose **D**ecision-importance-driven **V**isual-token **A**llocation (**DVA**), a dynamic token allocation framework that determines the retention ratio online based on the current observation and instruction. The key idea is to introduce **D**ecision **I**mportance (**DI**), a per-step supervision signal derived from counterfactual suffix rollbacks, which quantifies the downstream degradation caused by sustained token reduction starting at a given control step. A lightweight importance predictor learns to estimate DI from features already available in VLA models, and a monotone calibrator converts the predicted importance into the deployed retention ratio. DVA serves as a plug-in module for existing token pruners, introducing only about 1% per-step latency overhead while keeping both the pruning strategy and the policy parameters unchanged. Experiments show improved performance-efficiency trade-offs over fixed-ratio pruning. On OpenVLA-OFT, retaining 37% of visual tokens yields a success rate comparable to the full-token baseline at a 1.20× speedup, while retaining 22% maintains a 96.2% success rate at a 1.28× speedup. At matched average retention targets, DVA outperforms fixed-ratio pruning across three action heads, three pruning methods, and two benchmarks, demonstrating its effectiveness and generalization ability. Code and models will be released publicly.



(a) (b)

Figure 1: **DVA adapts visual budgets across control steps.** (a) DVA reallocates a fixed average token budget from the contact phase to cue-distributed search. (b) *Left*: DVA achieves success rates comparable to those of full-token OpenVLA-OFT on LIBERO at a 1.20× speedup. *Right*: DVA improves success rates over static pruning across three action heads at equal average retention ratios.

1 INTRODUCTION

Vision-Language-Action (VLA) models have emerged as a general paradigm for robotic manipulation. By coupling a pretrained vision-language backbone with an action decoder, they map raw visual

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

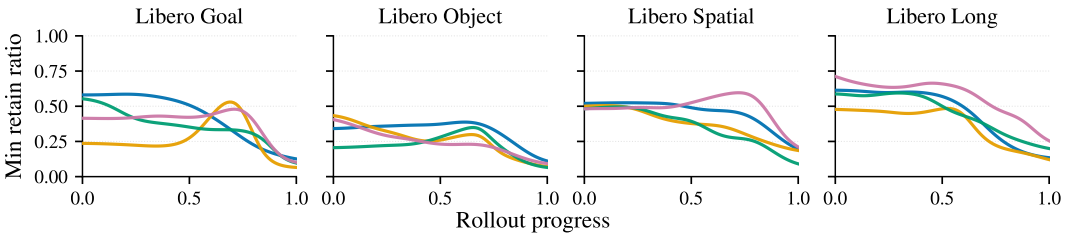


Figure 2: **Minimum viable fixed-ratio suffix retention across control steps.** Each panel corresponds to a LIBERO suite, and each curve to one task. Each candidate ratio is held from the indicated step until termination. Variation across control steps reflects differences in the remaining rollout's sensitivity to sustained pruning.

observations and language instructions directly to control actions (Kim et al., 2025b;a; Black et al., 2025; Physical Intelligence et al., 2025). Because manipulation operates in a closed loop, the backbone must be re-executed at every control step on a newly captured frame, making its inference latency a direct bottleneck for control frequency. In each forward pass, visual tokens produced by the vision encoder account for a substantial fraction of the sequence processed by the language model, making them a natural target for reducing per-step computation.

Visual-token pruning therefore offers a natural way to accelerate VLA inference. By ranking visual tokens and retaining only the most informative ones, it shortens the sequence processed by the language model with minimal architectural modification, following the broader line of token reduction in ViTs and VLMs (Chen et al., 2024; Zhang et al., 2025; Rao et al., 2021; Bolya et al., 2023; Shang et al., 2025). However, existing methods allocate tokens based on the current observation and cannot anticipate how the chosen budget will affect future control decisions and task outcomes. Consequently, when adapted to VLA models, prior methods either fix the retention ratio in advance (Xu et al., 2025; Yang et al., 2025) or adjust it using hand-crafted heuristics (Pei et al., 2025; Cheng et al., 2026). Neither approach grounds the token budget in the actual cost of pruning for closed-loop control.

In practice, different tasks and control steps can tolerate different degrees of visual-token reduction (Sec. 2.2). This motivates dynamically selecting the retention ratio at each control step based on the current observation and instruction (Fig. 1a). Hand-crafted cues such as gripper state or motion magnitude do not directly measure the downstream consequences of pruning.

A natural supervision signal is obtained by pruning at a single step and checking whether the rollout succeeds. However, subsequent full-token actions can compensate for an isolated perturbation, potentially obscuring sensitivity to repeated pruning. We therefore use sustained-pruning rollouts to construct a signal that captures cumulative closed-loop consequences. Even when a rollout succeeds under sustained pruning, it may require more control steps than full-token execution. Task success alone therefore provides an incomplete measure of the consequences of pruning.

Based on this insight, we introduce **D**ecision-importance-driven **V**isual-token **A**llocation (**DVA**), which grounds the per-step token budget in a new supervision signal called **D**ecision **I**mportance (**DI**). Decision importance is a per-step signal estimated through counterfactual suffix rollback. Starting at each selected control step, we hold a candidate retention ratio until termination and compare the outcome with a full-token reference. DI aggregates failures and additional control steps across candidate ratios to measure the remaining trajectory's sensitivity to sustained pruning.

With this signal, DVA adopts a lightweight predictor to estimate decision importance online and a monotone calibrator to map the predicted importance to a per-step retention ratio, operating as a plug-in module that supplies dynamic budgets to existing static pruners, without modifying their token-scoring rules or the underlying policy weights. Experiments on LIBERO and LIBERO-PLUS with OpenVLA-OFT and StarVLA show that DVA achieves a more favorable performance–speed trade-off than fixed-ratio pruning methods (Fig. 1b). In summary, our contributions are as follows:

- We observe that sensitivity to sustained pruning varies across tasks and control steps within a rollout, motivating per-step token budgets beyond fixed-ratio allocation.
- To address this issue, we propose **DVA**, a dynamic token allocation framework built on **DI**, which measures the sustained downstream impact of pruning through counterfactual suffix rollback. DVA converts this signal into per-step retention ratios, enabling plug-and-play adaptation of existing static pruners for enhancing efficiency.
- Extensive experiments show that DVA outperforms fixed-ratio pruning across different VLA backbones, pruners and benchmarks. It even surpasses the full-token baseline while retaining only 37% of visual tokens, demonstrating its effectiveness and generalizability.

2 Background and Motivation

This section reviews the general VLA inference pipeline and presents the key observations that motivate our method. Further discussion of related work is provided in Appendix A.

2.1 VLA Inference Pipeline

A Vision-Language-Action (VLA) model maps multimodal observations to executable robot actions in three stages (Yu et al., 2025). Given an RGB observation I and a tokenized instruction $\mathbf{l}$, a vision encoder (Dosovitskiy et al., 2021; Zhai et al., 2023; Oquab et al., 2024; Radford et al., 2021) produces visual tokens $\mathbf{v} = E_{\text{img}}(I) \in \mathbb{R}^{N_v \times D_v}$; a lightweight projector $P(\cdot)$ maps them into the embedding space of an LLM backbone (Touvron et al., 2023; Beyer et al., 2024; Karamcheti et al., 2024), which reasons over the concatenated sequence; and an action decoder (Lipman et al., 2023; Chi et al., 2025; Brohan et al., 2022) turns the resulting hidden states into a chunk of T low-level actions:

$$\mathbf{a}_{1:T} \;=\; D_{\text{act}}\Big(\text{LLM}\big([\,P(\mathbf{v})\,;\,\mathbf{l}]\big)\Big) \;\in\; \mathbb{R}^{T \times D_a}. \tag{1}$$

During closed-loop deployment, the full pipeline is re-executed at every control step on a new frame, so its per-step latency directly limits the achievable control frequency, and the N_v visual tokens are the primary target for reduction.

2.2 Key Observations

Visual context changes throughout manipulation: search may involve cues spread across the scene, while contact may involve a smaller region. This motivates adaptive allocation, but the effect of pruning also depends on subsequent closed-loop actions. We therefore examine sensitivity to sustained pruning across tasks and across control steps within a rollout.

Task-Level Retention. Table 1 reports the success rates of StarVLA-OFT, an OFT-style variant of StarVLA, on the four LIBERO suites under different fixed retention ratios. Visual tokens are ranked by task-query attention averaged across all text and action queries, and the highest-scoring fraction r is retained. Specifically, at a 20% ratio, performance on LIBERO-Long drops to 43.4%, a decrease of 51.6 points from the full-token baseline, whereas LIBERO-Object still achieves 91.2%, with only an 8.2-point drop. Even at 40%, LIBERO-Object matches the full-token baseline, while LIBERO-Goal remains 11.6 points lower. These results indicate that *different tasks tolerate very different levels of pruning.*

Table 1: Success rate (%) of StarVLA-OFT on the four LIBERO task suites under different fixed retention ratios.

StarVLA-OFT	Goal	Object	Spatial	Long	Avg.
Baseline	98.6	99.4	99.2	95.0	98.1
Retain 20%	69.0	91.2	71.6	43.4	68.8
Retain 40%	87.0	99.4	96.0	90.8	93.3
Retain 60%	94.8	99.6	96.8	95.2	96.6

Step-Level Retention. To measure sensitivity at each control step, we hold each candidate retention ratio from step t until termination and evaluate the remaining trajectory. The minimum viable fixed-ratio suffix retention is the smallest ratio at which the suffix completes without degradation. Fig. 2 shows how sensitivity to sustained pruning varies across control steps.

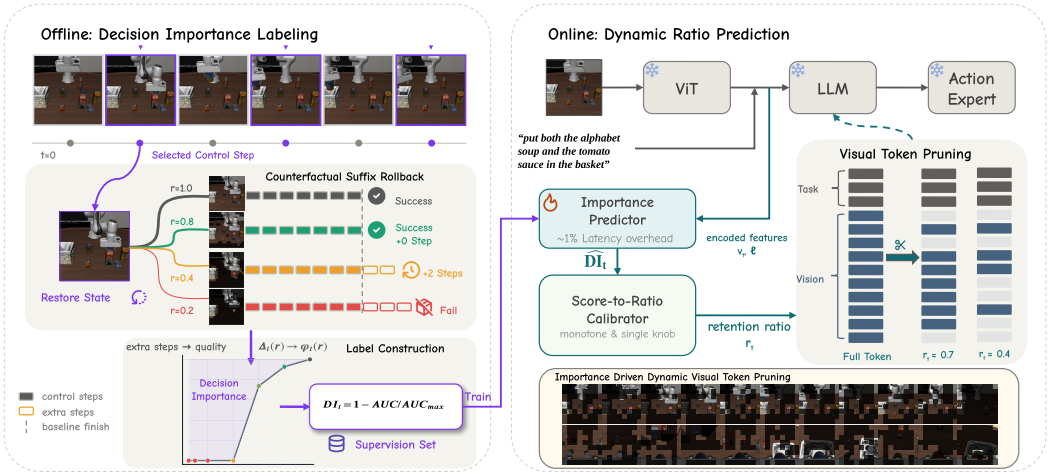


Figure 3: **Overview of the DVA framework.** *Offline (left)*: the simulator is restored at selected steps and each candidate ratio $r \in \mathcal{R}$ is held until termination; failures and delays relative to full-token execution yield DI_t. *Online (right)*: a lightweight predictor reuses VLA features to estimate $\hat{\mathrm{DI}}_t$, and a monotone calibrator maps it to a retention ratio. This budget is recomputed at each control step, with EMA smoothing as specified in Sec. 3.2.

These observations motivate a central question: *Can per-step sensitivity to sustained pruning guide efficient online token allocation?*

3 METHOD

Building on the observation in Sec. 2.2, we present DVA in detail. As illustrated in Fig. 3, DVA consists of two components: **(i) Decision-importance labeling (Sec. 3.1).** For each control step, we measure how sustained pruning affects the remaining rollout and summarize this effect as a scalar label $\mathrm{DI} \in [0, 1]$. A higher DI indicates greater degradation under sustained pruning from that step. **(ii) Dynamic ratio prediction (Sec. 3.2).** A lightweight predictor f_θ estimates decision importance from features already computed by the VLA, and a monotone calibrator g_τ maps the estimate to a per-step retention ratio r_t. This ratio is then passed to the token pruner with little additional latency.

3.1 OFFLINE: DECISION-IMPORTANCE LABELING

To guide the pruning level at each control step, we introduce **D**ecision **I**mportance (**DI**), which measures the remaining rollout's sensitivity to sustained visual-token pruning starting at the current control step. DVA uses this signal to learn a per-step budget allocation rule.

Decision Importance. At a selected control step t, we restore the simulator to its saved state s_t and evaluate counterfactual rollouts in which each candidate retention ratio is held until success or timeout. We measure failures and additional control steps relative to full-token execution. The index t specifies the control step at which pruning begins, and the remaining suffix defines the measurement horizon. DI therefore captures cumulative pruning effects across subsequent actions and can reflect upcoming task difficulty. Appendix B.4 compares labels obtained with different intervention horizons.

Let $\mathcal{R} = \{r_1, \ldots, r_K\} \subset (0, 1)$ denote a set of candidate retention ratios. For each ratio r, let $T_t^{(r)}$ be the number of control steps from step t to termination when r is used for the remainder of the rollout. We use $T_t^{(1)}$ as the full-token baseline. The additional steps caused by pruning are

$$\Delta_t(r) \;=\; T_t^{(r)} - T_t^{(1)}. \tag{2}$$

We normalize by the reference suffix length with a floor, $N_t = \max(T_t^{(1)}, n_{\text{floor}})$ (Appendix B.1), and define the quality score:

$$\varphi_t(r) \;=\; \begin{cases} \text{clip}\Big(1 - \dfrac{\Delta_t(r)}{N_t},\; 0,\; 1\Big) & \text{if successful,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Thus, $\varphi_t(r) = 1$ when pruning introduces no additional steps, while $\varphi_t(r) = 0$ when the suffix fails. Lower quality across candidate ratios indicates greater sensitivity of the remaining rollout to sustained pruning. We aggregate this response over the ratio axis:

$$\text{AUC}(\varphi_t) = \int_{r_{\min}}^{1} \varphi_t(r)\, dr, \qquad \text{DI}_t = 1 - \frac{\text{AUC}(\varphi_t)}{\text{AUC}_{\max}} \;\in\; [0,1], \tag{4}$$

where $r_{\min} = \min \mathcal{R}$ and $\text{AUC}_{\max} = 1 - r_{\min}$ corresponds to the ideal case $\varphi_t(r) \equiv 1$. A high DI_t indicates greater downstream degradation under sustained pruning starting at step t.

Integrating the quality curve combines the magnitude of degradation with the range of retention ratios over which it occurs. This yields a continuous supervision target that summarizes sensitivity across pruning strengths.

Label Construction via Counterfactual Rollback. We collect the DI supervision data in three phases, using the full-suffix intervention defined above.

Phase 1 (Baseline rollout). We execute each episode with full tokens ($r = 1$), saving states and predictor inputs at selected control steps: projected visual tokens v_t, instruction token embeddings ℓ, and optional proprioceptive state p_t.

Phase 2 (Counterfactual suffix rollback). From each restored state, we first run an independent full-token reference to measure $T_t^{(1)}$, excluding states where it fails. We then evaluate a subset of candidate ratios, holding each until success or timeout. The evaluated ratios, including 1, form $\mathcal{R}_t \subseteq \mathcal{R} \cup \{1\}$. Their suffix lengths yield $\varphi_t(r)$ via Eqs. (2)–(3).

Phase 3 (Aggregation). We evaluate $\text{AUC}(\varphi_t)$ by the trapezoidal rule on $\mathcal{R}_t$, anchored by $\varphi_t(1) = 1$, and obtain DI_t from Eq. (4). Each selected control step with a successful reference contributes a training pair $\big((v_t, \ell, p_t), \text{DI}_t\big)$. Appendix B.1 specifies the grid and normalizer.

3.2 Online: Dynamic Retention-Ratio Prediction

Given the supervision set collected in Sec. 3.1, we train a lightweight predictor f_θ on the fixed-ratio suffix labels. At each control step during deployment, DVA predicts DI from the latest observation and uses a monotone calibrator fitted offline to determine the retention ratio. The resulting budget sequence adapts as the trajectory evolves, with EMA smoothing as described below.

Decision Importance Predictor. At each control step t, f_θ takes projected visual tokens v_t, instruction token embeddings ℓ, and optional proprioceptive state p_t as input, all already available from the VLA. The visual branch uses channel-wise max pooling, while a single learnable query pools the language tokens by cross-attention. The summaries, including encoded proprioception when available, are concatenated and fused by an MLP. A sigmoid head predicts decision importance (Appendix C.1):

$$\hat{\text{DI}}_t \;=\; f_\theta(v_t, \ell, p_t) \;\in\; [0,1]. \tag{5}$$

We use visual max pooling because it yields better validation performance than the tested learnable and pyramid pooling alternatives (Appendix C.2).

We train the predictor f_θ using the following objective:

$$\mathcal{L} = \mathcal{L}_{\text{reg}} + \lambda_{\text{rank}}\mathcal{L}_{\text{rank}} + \lambda_{\text{var}}\mathcal{L}_{\text{var}}. \tag{6}$$

Regression $\mathcal{L}_{\text{reg}}$ fits DI labels with smooth-L_1. Ranking $\mathcal{L}_{\text{rank}}$ penalizes signed score gaps below $m_0 + |\text{DI}_t - \text{DI}_{t'}|$ over sampled non-tied pairs, balancing separation for calibration against numerical fit. Variance matching $\mathcal{L}_{\text{var}}$ discourages compressed scores by matching minibatch sample standard deviations: it controls global dispersion, while ranking distinguishes individual control steps. Appendix C.3 specifies the full definitions, weights, and pair sampling.

Score-to-Ratio Mapping. The predicted score $\hat{\mathrm{DI}}_t$ represents relative sensitivity. We convert it to a retention ratio using a lightweight, deployment-specific monotone calibrator g_τ, fitted at a tolerance τ for suffix-quality degradation in the offline rollouts.

For each labeled control step, we reuse the fixed-ratio suffix quality curve $\varphi_t(r) \in [0, 1]$ from Sec. 3.1 and obtain the *critical ratio*, the smallest ratio whose suffix quality stays within τ of baseline:

$$r_t^\star(\tau) \;=\; \min\{\, r : \varphi_t(r) \;\geq\; 1 - \tau \,\}, \tag{7}$$

where we interpolate the threshold crossing on the non-decreasing envelope of φ_t to obtain a continuous estimate $r_t^\star$ from the coarse ratio grid. A larger τ permits greater degradation and thus produces a smaller critical ratio. We then map predictor scores to retention ratios with a monotone calibrator g_τ, fitted by isotonic regression (Busing, 2022):

$$g_\tau = \arg\min_{g \in \mathcal{G}_\uparrow} \sum_t \big(g(\hat{\mathrm{DI}}_t) - r_t^\star(\tau)\big)^2, \qquad r_t = g_\tau(\hat{\mathrm{DI}}_t), \tag{8}$$

where $\mathcal{G}_\uparrow$ denotes the set of non-decreasing functions. Isotonic fitted values at calibration samples are invariant to strictly increasing score transformations that preserve order and ties; the interpolated deployment map need not be. The tolerance τ controls the deployment trade-off without a manually designed score-to-ratio rule.

For a fixed labeling protocol and deployment setting, the DI labels and predictor are shared across tolerances. We fit a calibrator for each tolerance using the stored quality curves (Appendix B.5). For a given backbone and action head, DVA shares one predictor across pruning methods and fits a separate calibrator for each pruner. Appendix C provides the predictor architecture and calibration procedure.

Deployment Configuration. The tolerance τ is the only user-specified parameter. It is selected to satisfy a target drop δ in the overall success rate (SR). On a fixed held-out set, we choose the largest tolerance that remains within the allowed SR-drop budget:

$$\tau^\star \;=\; \max\{\, \tau : \Delta\mathrm{SR}(\tau) \;\leq\; \delta \,\}. \tag{9}$$

Decision importance DI typically changes gradually across manipulation phases, whereas the per-step predictions may contain high-frequency noise. We therefore optionally smooth the predicted ratios using a first-order exponential moving average: $\tilde{r}_t = \alpha\, r_t + (1-\alpha)\, \tilde{r}_{t-1}$. This smoothing reduces step fluctuations while preserving the trajectory-level variation needed for dynamic allocation (Table 6).

4 Experiments

4.1 Experimental Setup

We conduct the main comparison on **OpenVLA-OFT** (Kim et al., 2025a), a parallel-decoding variant of OpenVLA (Kim et al., 2025b), on which we benchmark against VLA-ADP (Pei et al., 2025), FastV (Chen et al., 2024), VLA-Cache (Xu et al., 2025), SparseVLM (Zhang et al., 2025), and EfficientVLA (Yang et al., 2025). To assess whether dynamic token allocation generalizes across architectures, we additionally use three variants from the StarVLA codebase (StarVLA Community, 2026), all built on the same Qwen3-VL (Bai et al., 2025) backbone: **StarVLA-OFT**, **StarVLA-π**, and **StarVLA-GR00T**, covering parallel-decoding, flow-matching, and dual-system paradigms.

Benchmarks. **LIBERO** (Liu et al., 2023) provides 130 tasks; we follow the standard per-suite protocol on the four evaluation suites (*Spatial*, *Object*, *Goal*, and *Long*, 10 tasks each). **LIBERO-PLUS** (Fei et al., 2025) extends these tasks into 10,030 episodes spanning seven perturbation dimensions (layout, viewpoint, lighting, instruction, etc.) and thus measures robustness rather than only in-distribution success.

Static Pruning Methods. We compare against three representative visual-token pruners, all operating as plug-in modules at a fixed retention ratio. **FastV** prunes visual tokens after an early LLM layer, scoring each token by the attention it receives from the last prefill token, and **SparseVLM** is a training-free sparsifier that uses selected text tokens as raters of visual-token significance with a layer-wise adaptive ratio. **Task-Query Attention** (TQA) is a simple VLA-tailored baseline we introduce: each visual token is scored by the mean attention it receives from all text and action queries during the LLM prefill, and the highest-scoring fraction r is retained. It provides a strong reference point and matches the scoring used in our data collection and analysis.

Table 2: **Comparison with prior pruners on OpenVLA-OFT (LIBERO).** Success rate (%) per suite, together with the average latency of one control step and the resulting end-to-end speedup over the full-token baseline OpenVLA-OFT (7B). For DVA, the *value* in parentheses is the latency contributed by the predictor and calibrator, which is already included in the reported total. $\bar{r}$ denotes the average retention ratio produced by DVA.

Method	CKPT	Success Rate (%)					Efficiency	
		Spatial	Object	Goal	Long	**Avg.**	Latency (ms)	Speedup
OpenVLA	OpenVLA (7B)	84.7	88.4	79.2	53.7	76.5	–	–
OpenVLA-OFT	OFT (7B)	98.6	97.8	96.6	94.2	96.8	107.0	1.00×
FastV	OFT (7B)	99.0	97.2	95.6	85.6	94.4	79.2	1.35×
VLA-Cache	OFT (7B)	98.4	98.4	96.8	93.6	96.8	103.9	1.03×
SparseVLM	OFT (7B)	99.4	92.6	96.8	92.0	95.2	86.3	1.24×
EfficientVLA	OFT (7B)	96.5	91.1	96.0	72.1	88.9	70.4	1.52×
VLA-ADP ($Ratio = 0.50$)	OFT (7B)	99.4	98.0	96.4	91.2	96.3	98.2	1.09×
VLA-ADP ($Ratio = 0.30$)	OFT (7B)	97.6	98.4	97.4	84.2	94.4	96.3	1.11×
DVA-TQA ($\bar{r} = 0.37$)	OFT (7B)	99.0	98.0	97.0	94.0	**97.0**	88.8 *(1.12)*	1.20×
DVA-TQA ($\bar{r} = 0.28$)	OFT (7B)	98.2	98.2	96.6	92.8	96.5	85.4 *(1.08)*	1.25×
DVA-TQA ($\bar{r} = 0.22$)	OFT (7B)	98.6	98.2	95.6	92.4	96.2	83.4 *(1.09)*	1.28×

Implementation Details. Predictor supervision and configuration-specific calibration data are collected offline using the protocol of Sec. 3.1. Since adjacent control steps yield largely redundant supervision, we label only a compact subset of each baseline rollout: 5–10 key steps flagged by gripper-state transitions, abrupt changes between adjacent action chunks, or high ViT patch-attention entropy, plus at most 8 uniformly sampled steps from the remainder for temporal coverage. Each selected control step is rolled back under $\mathcal{R} = \{0.05, 0.1, 0.2, 0.4, 0.6, 0.8\}$. For the real-robot experiments, the simulator-trained predictor is fine-tuned on 4 labeled steps per task. Statistics of the collected supervision set are reported in Appendices B.5 and B.6.

Predictor and Calibration Reuse. Predictor training and fine-tuning use only TQA-derived DI labels; FastV and SparseVLM batches provide calibration data. The predictors for StarVLA-π and GR00T are initialized from the OFT predictor and fine-tuned on their own TQA batches. On LIBERO-PLUS, the OFT predictor is reused unchanged, the π/GR00T predictors are fine-tuned again, and all nine calibrators are refitted. OpenVLA-OFT uses one TQA predictor and calibrator, with the calibrator fitted on the predictor training set.

Deployment Settings. At deployment, unless otherwise stated, we prune at LLM layer $L^\star = 1$, and smooth the deployed ratio with an EMA coefficient $\alpha = 0.5$. Operating points are obtained by sweeping the tolerance $\tau \in \{0.02, 0.05, 0.1\}$; for the comparisons in Table 3 we select $\tau^\star$ under a success rate budget of $\delta = 0.03$ on LIBERO and $\delta = 0.05$ on the harder LIBERO-PLUS. All experiments run on NVIDIA A800 GPUs. Additional experimental implementation details are provided in Appendix D.

Offline Labeling Cost. Across the StarVLA and OpenVLA-OFT main sets, DI collection averages 5.8 and 4.6 suffix rollbacks per selected control step, respectively, including full-token references. Appendix B.5 reports the total collection workload and adaptation costs.

4.2 Main Results

Comparison on OpenVLA-OFT. Table 2 compares DVA with visual-token pruning baselines on OpenVLA-OFT. Instantiated with our TQA scoring, dynamic allocation retains an average of only 37% of the visual tokens yet reaches 97.0% average success at a 1.20× speedup, achieving the highest success rate (SR) in the table. Reducing the average budget to 22% still yields 96.2% average success at a 1.28× speedup. Among the other pruners, only VLA-Cache preserves baseline SR (96.8%), but at a marginal 1.03× speedup, whereas FastV, SparseVLM, and EfficientVLA trade SR for speed. These speedups are net of DVA's own cost: the predictor and calibrator add 1.1 ms per control step, about 1% of the per-step inference latency, so the allocation machinery consumes

Table 3: **Dynamic versus static token pruning on LIBERO and LIBERO-PLUS.** For each pruner, the static ratio is set separately for each suite to match the mean retention ratio from a reference DVA run. *Baseline* is the uncompressed policy ($\bar{r}$=1). Results report success rate (%), with green numbers giving the gain of DVA over the static pruner.

		LIBERO						LIBERO-PLUS					
Model	Method	$\bar{r}$	Spatial	Object	Goal	Long	Avg.	$\bar{r}$	Spatial	Object	Goal	Long	Avg.
StarVLA-OFT	*Baseline*	1.000	99.2	99.4	98.6	95.0	**98.1**	1.000	84.0	78.1	72.9	69.6	**76.1**
	FastV	0.399	95.0	99.5	88.3	90.2	93.3	0.520	77.9	76.1	69.6	67.6	72.7
	+ **DVA**		96.3	99.6	88.9	91.3	**94.1** (+0.8)		77.6	75.7	70.2	68.7	**73.0** (+0.3)
	TQA	0.377	95.7	99.4	87.1	89.7	92.9	0.498	77.4	75.9	66.8	68.3	72.0
	+ **DVA**		97.3	99.4	88.3	93.7	**94.7** (+1.8)		79.0	75.8	69.3	67.4	**72.8** (+0.8)
	SparseVLM	0.220	95.0	99.6	96.2	91.4	95.6	0.396	78.6	76.2	72.1	68.0	73.6
	+ **DVA**		94.8	99.6	96.8	93.2	**96.1** (+0.5)		79.9	75.9	72.3	67.9	**73.9** (+0.3)
StarVLA-GR00T	*Baseline*	1.000	99.8	99.0	97.8	96.4	**98.3**	1.000	83.9	80.9	72.7	71.5	**77.1**
	FastV	0.591	98.6	98.6	94.6	95.4	96.8	0.610	84.1	79.5	72.1	69.6	76.2
	+ **DVA**		98.0	98.6	97.6	95.2	**97.4** (+0.6)		84.4	80.5	72.9	69.8	**76.8** (+0.6)
	TQA	0.599	98.6	99.2	95.2	96.4	97.4	0.467	80.8	76.7	64.0	67.4	72.1
	+ **DVA**		98.8	99.6	98.2	97.8	**98.6** (+1.2)		81.3	77.2	69.3	67.3	**73.6** (+1.5)
	SparseVLM	0.253	95.2	96.8	92.8	92.8	94.4	0.395	79.4	75.6	71.4	69.5	73.9
	+ **DVA**		95.4	97.6	94.8	94.0	**95.5** (+1.1)		80.6	76.0	72.0	68.8	**74.2** (+0.3)
StarVLA-π	*Baseline*	1.000	98.6	98.6	95.8	94.6	**96.9**	1.000	87.6	80.4	73.7	72.1	**78.3**
	FastV	0.389	97.2	98.8	90.8	93.8	95.2	0.469	83.9	79.2	66.7	71.1	75.1
	+ **DVA**		96.4	99.0	96.4	95.8	**96.9** (+1.7)		84.3	80.4	68.2	69.6	**75.5** (+0.4)
	TQA	0.485	98.0	98.2	95.8	95.2	96.8	0.458	83.1	80.1	65.1	69.5	74.3
	+ **DVA**		97.8	99.0	97.8	95.8	**97.6** (+0.8)		83.3	80.8	67.9	69.2	**75.1** (+0.8)
	SparseVLM	0.262	94.2	98.6	95.2	89.0	94.3	0.363	81.9	78.1	73.0	67.5	75.0
	+ **DVA**		95.6	99.4	95.8	94.8	**96.4** (+2.1)		83.0	78.1	73.4	68.1	**75.5** (+0.5)

a small fraction of what it saves. A per-module breakdown of the control-step latency is given in Appendix D.2 (Table D.2).

Dynamic vs. Static. Table 3 compares *dynamic* allocation with static pruning using matched average retention targets. Across three StarVLA variants (OFT, π, GR00T), three pruners (FastV, TQA, SparseVLM), and both LIBERO and the harder LIBERO-PLUS, DVA improves average success in all 18 settings, by $+0.3$ to $+2.1$ points, and in several cases lifts a pruned policy to or above its full-token baseline (e.g., StarVLA-GR00T with TQA reaches 98.6% versus a 98.3% baseline on LIBERO). The dynamic and static variants also have comparable estimated FLOPs under the accounting described in Appendix D.6. The largest average gain occurs with StarVLA-π and SparseVLM on LIBERO, where DVA improves success by 2.1 percentage points. This gain includes a 5.8-point improvement on the Long suite (89.0% to 94.8%), highlighting the benefit of adapting the budget on extended manipulation tasks. On LIBERO-PLUS, average gains range from 0.3 to 1.5 points across all nine head–pruner pairs, although individual suite scores do not improve uniformly.

Real-robot Evaluation. We further deploy DVA-TQA on a physical Franka arm, evaluating on two tasks, *Pick Carrot Out* and *Cube Up*, over 10 trials each (Table 4), against static TQA pruning at comparable per-task retention ratios $\bar{r}$. DVA improves success on both tasks, most clearly on the contact-rich *Cube Up*, where it raises the success rate from 3/10 to 7/10, confirming that our simulation findings transfer to the physical world.

4.3 Ablation Study

We ablate the pruning layer, retention-ratio smoothing, and predictor transferability. The first two use OpenVLA-OFT on LIBERO with the first 10 initial states of every task; the third spans the three StarVLA action-head variants.

Pruning Layer. Table 5 studies the LLM layer L at which pruning is applied. Because shallower pruning avoids more full-sequence computation, it generally yields greater speedup. Across all tolerances, L=1 remains within 1.8 points of the best success rate and within $0.01\times$ of the highest speedup. We therefore use L=1 in all main experiments.

Table 4: **Real-robot evaluation on a Franka arm.** For each task (10 trials), we report SR, the average retention ratio $\bar{r}$, and the FLOPs relative to the full-token baseline.

Method	Pick Carrot Out			Cube Up		
	SR	$\bar{r}$	FLOPs	SR	$\bar{r}$	FLOPs
Baseline	10/10	1.00	100%	5/10	1.00	100%
Static	8/10	0.20	55%	3/10	0.40	66%
DVA-TQA	9/10	0.19	55%	7/10	0.42	67%

Table 5: **Pruning-layer ablation on OpenVLA-OFT.** Success rate (%) and end-to-end speedup (×) across pruning layers L and tolerances τ.

	τ=0.02		τ=0.05		τ=0.10	
Layer	SR	×	SR	×	SR	×
$L^\star$=1	97.5	1.19	96.0	1.23	96.0	1.26
L=2	97.0	1.18	96.5	1.22	92.5	1.27
L=4	98.0	1.16	97.8	1.20	93.8	1.23
L=8	97.8	1.14	97.3	1.16	97.3	1.21

Table 6: **EMA smoothing ablation.** Success rate (%) and normalized std of the deployed per-step ratio under EMA coefficient α at tolerance τ. We use α=0.5 in all main experiments.

	Success rate (%)			Ratio std		
$\alpha\backslash\tau$	0.02	0.05	0.10	0.02	0.05	0.10
1.00	97.0	97.3	95.8	1.000	1.000	1.000
0.75	96.8	95.0	94.8	0.905	0.918	0.922
0.50	97.3	96.3	96.0	0.808	0.833	0.846
0.25	96.8	97.0	96.0	0.686	0.738	0.751

Ratio Smoothing. Table 6 varies the EMA coefficient α used to smooth per-step retention ratios. Stronger smoothing reduces the normalized ratio dispersion to 0.69–0.75 at α=0.25, while the success rate remains similar. This suggests that the predictor primarily captures slow, phase-level variation rather than high-frequency fluctuations, although excessive smoothing weakens the adaptivity of the token budget. We adopt α=0.5, which preserves over 80% of the unsmoothed dispersion while maintaining success rates similar to those of the unsmoothed variant.

Cross-Head Transfer. Fig. 4 examines whether decision importance transfers across action heads sharing the StarVLA backbone. We train the predictor on StarVLA-OFT and apply it *zero-shot* to π and GR00T, measuring Spearman correlation between predicted scores and per-step DI across all three pruners. Zero-shot transfer yields positive correlation for every head–pruner pair, although below the source-head level. Lightweight mini-batch fine-tuning closes most of this gap on both π and GR00T. These results indicate that suffix-sensitivity information transfers to the evaluated target heads, with adaptation improving agreement with their DI labels.

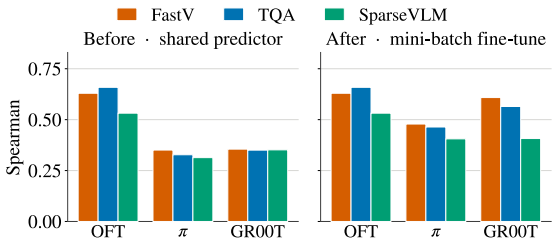


Figure 4: **Cross-head transfer of the importance predictor.** Spearman correlation between predicted score and measured per-step DI, for three pruners and three StarVLA action heads. *Left*: a StarVLA-OFT predictor applied *zero-shot*. *Right*: after mini-batch fine-tuning on target-head data.

5 Conclusion

We observe that sensitivity to sustained pruning varies across tasks and control steps within a rollout, motivating per-step visual-token budgets. Based on this finding, we introduce DVA, a dynamic allocation framework built on Decision Importance (DI), which quantifies the downstream impact of sustained pruning through counterfactual suffix rollback. A lightweight predictor estimates DI from existing VLA features, and a monotone calibrator fitted offline maps its predictions to retention ratios, enabling dynamic allocation with existing pruners while preserving their token-scoring rules and the policy weights. Experiments on LIBERO and LIBERO-PLUS show improved performance-efficiency trade-offs across the evaluated backbones, action heads, and pruners. Comparisons at equal average retention ratios support the benefit of adapting token budgets across control steps, while experiments on a Franka arm demonstrate the feasibility of deploying DVA beyond simulation.

Limitations and Future Work. Our labels rely on simulator-based suffix rollbacks, which require resettable environments and are costly to approximate on real robots. Extending importance estimation to reset-free or offline data is therefore an important direction. Label quality also depends on the underlying policy, since failures may reflect both step difficulty and backbone limitations. Stronger VLA policies should provide cleaner importance signals.

AI USE STATEMENT

We used generative AI tools for language polishing and LaTeX formatting, including table layouts and cross-reference checks. We take full responsibility for the content and conclusions of this paper.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. PaliGemma: A versatile 3B VLM for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations*, 2023.

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.

Frank M. T. A. Busing. Monotone regression: A simple and fast O(n) PAVA implementation. *Journal of Statistical Software, Code Snippets*, 102(1):1–25, 2022. doi: 10.18637/jss.v102.c01. URL https://www.jstatsoft.org/index.php/jss/article/view/v102c01.

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 Ales Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part LXXXI*, Lecture Notes in Computer Science, pp. 19–35. Springer, 2024. doi: 10.1007/978-3-031-73004-7_2. URL https://doi.org/10.1007/978-3-031-73004-7_2.

Jintao Cheng, Haozhe Wang, Weibin Li, Gang Wang, Yipu Zhang, Xiaoyu Tang, Jin Wu, Xieyuanli Chen, Yunhui Liu, and Wei Zhang. VLA-IAP: Training-free visual token pruning via interaction alignment for vision-language-action models. *arXiv preprint arXiv:2603.22991*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion Policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Hengyu Fang, Yijiang Liu, Yuan Du, Li Du, and Huanrui Yang. SQAP-VLA: A synergistic quantization-aware pruning framework for high-performance vision-language-action models. *arXiv preprint arXiv:2509.09090*, 2025.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. LIBERO-PLUS: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

Titong Jiang, Xuefeng Jiang, Yuan Ma, Xin Wen, Bailin Li, Kun Zhan, Peng Jia, Yahui Liu, Sheng Sun, and Xianpeng Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning, 2025. URL https://arxiv.org/abs/2509.12594.

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

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025a. doi: 10.15607/RSS.2025.XXI.017.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. CogVLA: Cognition-aligned vision-language-action models via instruction-driven routing & sparsification. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 137646–137675. Curran Associates, Inc., 2025a. doi: 10.52202/085713-4594. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c9028f7874df04843e7bf435ee4cd3c3-Paper-Conference.pdf.

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*, 2025b.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: benchmarking knowledge transfer for lifelong robot learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

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.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=a68SUt6zFt.

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. *CoRR*, abs/2509.22093, 2025. doi: 10.48550/ARXIV.2509.22093. URL https://doi.org/10.48550/arXiv.2509.22093.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pp. 8748–8763. PMLR, 2021. URL http://proceedings.mlr.press/v139/radford21a.html.

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.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. LLaVA-PruMerge: Adaptive token reduction for efficient large multimodal models. In *ICCV*, 2025.

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

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Yuxin Huang, Han Zhao, Donglin Wang, and Haoang Li. CEED-VLA: Consistency vision-language-action model with early-exit decoding. *arXiv preprint arXiv:2506.13725*, 2025a.

Wenxuan Song, Jiayi Chen, Pengxiang Ding, Han Zhao, Wei Zhao, Zhide Zhong, Zongyuan Ge, Zhijun Li, Donglin Wang, Lujia Wang, Jun Ma, and Haoang Li. PD-VLA: Accelerating vision-language-action model integrated with action chunking via parallel decoding. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13162–13169, 2025b. doi: 10.1109/IROS60139.2025.11247519.

StarVLA Community. StarVLA: A Lego-like codebase for vision-language-action model developing. *CoRR*, abs/2604.05014, 2026. doi: 10.48550/ARXIV.2604.05014. URL https://doi.org/10.48550/arXiv.2604.05014.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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*, 2025a.

Hongyu Wang, Chuyan Xiong, Ruiping Wang, and Xilin Chen. BitVLA: 1-bit vision-language-action models for robotics manipulation. *arXiv preprint arXiv:2506.07530*, 2025b.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. TinyVLA: Toward fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 10(4):3988–3995, 2025. doi: 10.1109/LRA.2025.3544909. URL https://doi.org/10.1109/LRA.2025.3544909.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-Cache: Efficient vision-language-action manipulation via adaptive token caching. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 164448–164473. Curran Associates, Inc., 2025. doi: 10.52202/085713-5484. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f062da1973ac9ac61fc6d44dd7fa309f-Paper-Conference.pdf.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. EfficientVLA: Training-free acceleration and compression for vision-language-action models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 40891–40914. Curran Associates, Inc., 2025. doi: 10.52202/085713-1365. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3a2ef31a1e45908901adc0ca853a8faf-Paper-Conference.pdf.

Zhaoshu Yu, Bo Wang, Pengpeng Zeng, Haonan Zhang, Ji Zhang, Lianli Gao, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on efficient vision-language-action models. *CoRR*, abs/2510.24795, 2025. doi: 10.48550/ARXIV.2510.24795. URL https://doi.org/10.48550/arXiv.2510.24795.

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

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 11941–11952. IEEE, 2023. doi: 10.1109/ICCV51070.2023.01100. URL https://doi.org/10.1109/ICCV51070.2023.01100.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Dan Wang, Yuan Du, and Shanghang Zhang. MoLe-VLA: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18764–18772, 2026. doi: 10.1609/aaai.v40i22.38945. URL https://ojs.aaai.org/index.php/AAAI/article/view/38945.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A. Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/zhang25s.html.

APPENDIX CONTENTS

A Related Work **15**

A.1 Vision-Language-Action Models 15

A.2 Visual Token Reduction in ViTs and VLMs 15

A.3 Token Reduction for VLA Models 15

A.4 Other Axes of VLA Efficiency 16

B Decision-Importance Labeling **17**

B.1 Measurement Protocol and Derivation. 17

B.2 Key Step Selection. 19

B.3 Single-Step Probe vs. Suffix Rollback. 21

B.4 Effect of the Intervention Horizon 21

B.5 Supervision Data, Reuse, and Offline Cost. 22

B.6 Label Distribution 24

C Predictor and Calibrator **24**

C.1 Architecture of f_θ 25

C.2 Pooling Operators 26

C.3 Training Objective 27

C.4 Calibrator Details 28

D Experimental Implementation Details **30**

D.1 Benchmarks 31

D.2 Backbone Configurations. 31

D.3 Baseline Configurations and Reproduction 32

D.4 Task-Query Attention (TQA) 33

D.5 Hardware, Software and Seeds 33

D.6 FLOPs Accounting and Retention Variability 34

A Related Work

A.1 Vision-Language-Action Models

VLA models cast robotic manipulation as conditional sequence prediction: a pretrained vision-language backbone consumes an RGB observation and a language instruction, and an action decoder turns the resulting hidden states into low-level commands. RT-1 (Brohan et al., 2022) established the recipe of training a single transformer policy across large multi-task demonstration data, and OpenVLA (Kim et al., 2025b) scaled it by fine-tuning a visually conditioned language model (Karamcheti et al., 2024) and emitting discretised action tokens autoregressively. Because autoregressive decoding makes the per-step cost grow with the action dimension and the chunk length, subsequent work replaced it with parallel prediction heads (Kim et al., 2025a; Song et al., 2025b) or with continuous action experts trained by diffusion (Chi et al., 2025) and flow matching (Lipman et al., 2023; Black et al., 2025; Physical Intelligence et al., 2025).

These variants differ in how actions are produced but share the same front end: a ViT-family encoder (Dosovitskiy et al., 2021; Zhai et al., 2023; Oquab et al., 2024; Radford et al., 2021) followed by a projector into an LLM backbone (Beyer et al., 2024; Bai et al., 2025). Modular codebases such as StarVLA (StarVLA Community, 2026) make the shared structure explicit by assembling different paradigms over a common backbone. They also share a deployment property that shapes the efficiency literature discussed below: manipulation runs in a closed loop, so the backbone is re-executed at every control step on a freshly captured frame, and per-step latency bounds the achievable control frequency directly.

A.2 Visual Token Reduction in ViTs and VLMs

Because visual tokens dominate the sequence length of multimodal transformers, reducing their number is an established route to faster inference. Two mechanisms recur.

Pruning discards tokens judged uninformative. DynamicViT (Rao et al., 2021) inserts a lightweight prediction module between ViT stages that emits a keep/drop decision per token, trained end-to-end with a Gumbel relaxation and a loss that holds the keep rate near a target. In multimodal LLMs the corresponding observation is that visual tokens attract disproportionately little attention beyond the first few layers; FastV (Chen et al., 2024) exploits this by ranking tokens according to the attention they receive at a shallow layer and discarding the low-ranked ones for the remaining depth, which leaves the early layers intact and therefore preserves the representation on which the ranking is computed. SparseVLM (Zhang et al., 2025) sharpens the ranking by first selecting the instruction tokens that are actually relevant to the image and using only those as raters of visual significance, rather than averaging attention over the whole text prefix; it further adapts how much to prune per layer instead of applying one ratio uniformly across depth, and is training-free.

Merging preserves information by fusing redundant tokens rather than removing them. ToMe (Bolya et al., 2023) merges a fixed number of token pairs per layer via bipartite soft matching on key features, requiring no training and applying to off-the-shelf ViTs. LLaVA-PruMerge (Shang et al., 2025) combines both mechanisms: it selects tokens using the concentration of class-token attention, adapting the count per image rather than fixing it, and folds the unselected tokens into the retained ones by clustering so that discarded content is summarised rather than lost.

These methods primarily decide *which* tokens to remove or merge within an input. Depending on the method, the retained count is fixed or adapts to layer depth and image content. Such input-dependent adaptation can also change the count across successive frames, but does not explicitly allocate computation across control steps according to measured closed-loop consequences.

A.3 Token Reduction for VLA Models

Adapting these techniques to closed-loop control is an active area; see Yu et al. (2025) for a survey. The VLA-specific work divides according to what governs the amount of compression.

Fixed Compression Settings. The first group selects a configuration once and holds it throughout deployment. VLA-Cache (Xu et al., 2025) targets redundancy between consecutive frames rather

Table A.1: Axes along which VLA inference is accelerated, and whether existing methods set the amount once or vary it per control step. The axes are largely independent, and several methods combine two or more of them.

Axis	Mechanism	Representative	Allocation
Token count	drop or merge visual tokens	FastV, SparseVLM, ToMe, VLA-ADP	fixed or per-step
Temporal reuse	cache computation for static regions	VLA-Cache, SP-VLA	per-step
Depth	skip layers or exit early	DeeR-VLA, MoLe-VLA	per-step
Precision	quantise or binarise weights	BitVLA, SQAP-VLA	fixed
Model scale	train or distil a compact policy	TinyVLA, SmolVLA	fixed
Decoding	parallel or few-step generation	OpenVLA-OFT, CEED-VLA	fixed

than within one: it identifies tokens whose corresponding image regions are unchanged since the previous control step, reuses their cached computation, and recomputes only the tokens that changed or that the task marks as relevant, with the reuse governed by preset thresholds. EfficientVLA (Yang et al., 2025) is training-free and composes three reductions at once – pruning functionally redundant language-model layers, selecting visual tokens in a task-aware manner, and caching intermediate features in the action head across denoising steps – which is itself evidence that the axes multiply rather than substitute. Both accelerate inference substantially at a compression level fixed before the rollout begins.

Action-Aware Modulation. A second group varies compression over the rollout, motivated by the observation that manipulation phases differ in redundancy. VLA-ADP (Pei et al., 2025) reports that redundancy is higher during coarse phases such as relocating, where global movement dominates, than during fine-grained phases such as grasping, where local geometry and contact detail matter, and that this redundancy correlates with action dynamics. It accordingly combines text-driven token selection with an action-aware trajectory gate, conditioning the pruning signal on recent action history so that patch relevance is treated as action-conditioned as well as text-conditioned. VLA-IAP (Cheng et al., 2026) pursues the same intuition without training, scoring tokens by their alignment with the interaction region. SpecPrune-VLA (Wang et al., 2025a) introduces a speculative variant, using action-aware self-speculation to prune aggressively while retaining a mechanism to recover when the aggressive path is unreliable, and VLA-Pruner (Liu et al., 2025) makes the temporal dimension explicit with a dual-level criterion. A further variant learns the selection rather than scoring it heuristically, making the pruning decision differentiable and training it jointly with the policy (Jiang et al., 2025). What these share is that the amount of compression is read off a proxy computed from the current step, such as gripper state, motion magnitude, attention concentration, or recent action trajectory, rather than from the consequence of compressing at that step.

A.4 Other Axes of VLA Efficiency

Token count is one of several knobs; Table A.1 summarises the landscape.

Dynamic Depth. Closest in spirit to per-step allocation is work that varies network depth with the situation. DeeR-VLA (Yue et al., 2024) starts from the observation that the easier situations make up the bulk of a control episode and generally require a far smaller model to produce the correct action, and equips the multimodal backbone with multiple exits so that processing terminates once enough of the model has been activated for the case at hand. Its termination criteria are derived from explicit resource specifications such as average computational cost, peak cost, and GPU memory, so the user constrains the budget and the exit rule spends it. MoLe-VLA (Zhang et al., 2026) realises the same axis differently, routing each input through a selected subset of layers via a mixture-of-layers formulation, and CEED-VLA (Song et al., 2025a) combines early-exit decoding with a consistency objective. CogVLA (Li et al., 2025a) couples instruction-driven routing with sparsification, coordinating the two axes with a single mechanism.

Weight-Level Compression. An orthogonal family reduces the cost of each operation rather than the number of operations issued. Quantisation has been pushed to the extreme of binary weights for VLA policies (Wang et al., 2025b), and SQAP-VLA (Fang et al., 2025) argues that quantisation and pruning interact and should be designed jointly rather than stacked. Compact policies trained from the outset (Wen et al., 2025; Shukor et al., 2025) occupy the same space from the architecture side.

Decoding and Scheduling. On the decoder side, parallel decoding removes the sequential dependency in action-chunk generation (Kim et al., 2025a; Song et al., 2025b), while consistency-style objectives cut the number of integration steps a flow or diffusion expert requires (Song et al., 2025a). Others reuse computation across control steps at the action level rather than the token level (Tan et al., 2025), or schedule which model runs at which step alongside token pruning (Li et al., 2025b).

B Decision-Importance Labeling

This appendix completes Sec. 3.1. Appendix B.1 fixes the notation, states what *termination* and *success* mean operationally, derives Eqs. (2)–(4) from the measurement protocol, and records two properties of the resulting label. Appendix B.2 gives the key step selection rule and its thresholds. Appendix B.3 compares the signals induced by single-step and full-suffix interventions and examines action divergence as an alternative signal. Appendix B.4 analyzes how the intervention horizon affects the resulting labels. Appendices B.5 and B.6 describe the supervision workload, reuse across configurations, and label distribution. Table B.1 collects the symbols.

B.1 Measurement Protocol and Derivation

Rollout Conventions. A rollout is a sequence of *control steps*. At a control step the backbone runs once and emits an action chunk $a^{(t)}$, of which the environment executes c actions before the next control step; we use $c = 8$ for OpenVLA-OFT and the full predicted chunk for StarVLA. A control step is therefore the unit at which the retention ratio can change, and also the unit of the label: $T_t^{(r)}$, $\Delta_t(r)$ and N_t all count control steps, as in Sec. 3.1. Simulator transitions, or *environment steps*, are the unit in which LIBERO expresses its episode caps, and occur c times per control step.

A rollout ends either when the environment returns `done`, the BDDL goal predicate of the task, or when its step budget is exhausted. `done` is tested after every environment step, so a rollout can end part way through a chunk; the control step in progress is counted, which makes $T_t^{(r)}$ the number of backbone executions begun between t and termination. `success` is exactly this `done` flag, and a rollback that exhausts its budget without `done` is a timeout, recorded as a failure. A baseline episode is capped at $T_{\text{suite}}^{\text{cap}}$ environment steps, equal to 220 (Spatial), 280 (Object), 300 (Goal) and 520 (Long). A rollback needs a cap of its own, since it starts partway into an episode rather than at the beginning. The natural choice is whatever the baseline had left at that point, written $T_{\text{suite}}^{\text{cap}} - t^{\text{env}}$ for a checkpoint taken at environment step t^{env}. Taken literally, however, that remainder shrinks towards zero for a checkpoint near the cap, and the rollback would then time out for want of time rather than because pruning had cost it anything. We therefore floor it:

$$B_t \;=\; \max\bigl(T_{\text{suite}}^{\text{cap}} - t^{\text{env}},\; 50\bigr). \tag{B.1}$$

Within 50 steps of the cap the floor is active and the rollback receives more budget than the baseline episode had left. Because the $r = 1$ reference is measured under the same floor, that surplus cancels in $\Delta_t(r)$.

Two consequences matter for the label. *Failure is absorbing*: $\varphi_t(r) = 0$ whenever the rollback at ratio r fails, however far it progressed, so the step-count term of Eq. (3) resolves only the successful part of the curve and the failing part is flat at zero. *Unlabelable control steps are gated out*: before any pruned rollback runs we roll the suffix back once at $r = 1$, and if that unpruned reference itself fails, the state is not solvable by the policy at full budget, so DI_t is set to `NaN` and the step is discarded during aggregation (Phase 3). This conditioning is what makes DI a measure of the *cost of pruning* rather than of intrinsic task difficulty: every labeled step is one the unpruned policy demonstrably solves from the restored state.

Table B.1: Principal symbols. Symbols introduced only inside a single derivation are defined where they appear.

Symbol	Meaning	Symbol	Meaning
Rollout			
t	control step index within a rollout	$\ell,\ p_t$	instruction token embeddings; proprioceptive state
n	number of control steps in a baseline rollout	$a^{(t)}$	action chunk at step t, $a^{(t)} \in \mathbb{R}^{c\times D_a}$
c	actions executed open loop per control step (chunk length)	s_t	restorable execution state (simulator + policy) at t
$I_t,\ l$	RGB observation at step t; tokenized instruction	f_r	closed-loop transition at ratio r, $f_r(\cdot)=f(\cdot,\pi_r(\cdot))$
v_t	visual tokens, $v_t\in\mathbb{R}^{N_v\times D_v}$; N_v their number	n_{wait}	settling actions before the first control step
Counterfactual measurement			
r	retain ratio (fraction of visual tokens kept), $r\in(0,1]$	$T_t^{(r)}$	*control* steps from t to termination, r held
$\mathcal{R}$	candidate ratio grid $\{r_1<\ldots<r_K\}$; $r_{\min}=r_1$	$\Delta_t(r)$	extra control steps caused by pruning, Eq. (2)
$\mathcal{R}_t,\ m_t$	ratios actually replayed at t, $\mathcal{R}_t\subseteq\mathcal{R}\cup\{1\}$, and $m_t=\lvert\mathcal{R}_t\rvert$	$N_t,\ n_{\text{floor}}$	normalizer and its floor, in control steps
H	intervention window: control steps the ratio is held	$\varphi_t(r)$	normalized progress (quality) score, Eq. (3)
$T^{\text{cap}}_{\text{suite}}$	per-suite episode cap, in *environment* steps	$\bar\varphi_t$	non-decreasing envelope, $\bar\varphi_t(r)=\max_{r'\le r}\varphi_t(r')$
B_t	environment-step budget of a rollback launched at t	$\text{AUC}_{\max}$	$1-r_{\min}$, area of the ideal curve $\varphi\equiv1$
t^{env}	environment step at which a checkpoint was taken	$\text{DI}_t,\ \text{DI}_{\max}$	decision importance, Eq. (4), and its grid ceiling, Eq. (B.3)
Key step selection			
$\bar g_i$	mean gripper channel of chunk i over the chunk horizon	$S,\ C$	flagged step set; its complement $C=\{1,\ldots,n\}\setminus S$
d_i	chunk-to-chunk displacement $\lVert\text{vec}(a^{(i)})-\text{vec}(a^{(i-1)})\rVert_2$	$k_{\min},k_{\max}$	floor and ceiling on $\lvert S\rvert$
$A_i,\ p_i,\ \mathcal{H}_i$	patch-attention map; its normalization; its entropy	$K_e,\ m$	top-K budget of the entropy trigger; uniform quota
Prediction and deployment			
$\widehat{\text{DI}}_t$	predicted importance $f_\theta(v_t,\ell,p_t)$, Eq. (5)	g_τ	isotonic score-to-ratio calibrator, Eq. (8)
z_t	transformed regression target DI_t^κ	α	EMA coefficient of Sec. 3.2
$\tau,\ r_t^\star(\tau)$	quality-degradation tolerance; critical ratio, Eq. (7)	$\bar r$	realized mean retain ratio of a run

Table B.2: Key step selection statistics over $8{,}531$ key steps with defined DI from $1{,}163$ unperturbed StarVLA episodes. *steps* counts steps selected by a trigger; *excl.* counts those selected exclusively by it. Steps with a failed full-token reference rollback (NaN DI) are excluded from every column. Time coverage fires only when no other trigger does, so its counts coincide.

Trigger	steps	excl.	$\overline{\text{DI}}$	std	DI>0.5 (%)
gripper change	2634	79	0.239	0.173	5.7
action-chunk discontinuity	3410	730	0.266	0.177	7.0
attn entropy	5073	4197	0.322	0.183	11.6
time coverage	778	778	0.335	0.192	13.1

The Replayed Grid. We detail the replay protocol of Sec. 3.1 for one selected control step t with s_t restored.

First, for a labelable control step, the replayed set $\mathcal{R}_t$ can be a strict subset of $\mathcal{R} \cup \{1\}$. Coarse-to-fine search evaluates a coarse grid and then the untested candidates within the bracketed success boundary; uniformly sampled steps may terminate after a successful lowest-ratio probe (Appendix B.2). A labelable step therefore costs two to $1 + |\mathcal{R}|$ rollbacks, including the full-token reference. Let H denote the number of control steps for which a ratio is held. All labels used for predictor training, fine-tuning, and calibration use $H = \infty$. The $H = 1$ and $H = 16$ interventions in Appendices B.3 and B.4 are diagnostic comparisons only.

Second, $T_t^{(1)}$ comes from a *separate* $r = 1$ rollback launched from the same restored state, not from the original episode. One extra rollback per labeled step buys a $\Delta_t(r)$ that differences two quantities produced by an identical protocol, so constant offsets cancel and nothing need be assumed about the restored suffix reproducing the baseline suffix. The numerator is clipped at zero, which matches the upper clip in Eq. (3): a pruned suffix finishing marginally faster than the reference counts as undegraded, not as better.

Third, N_t carries a floor,

$$\begin{aligned} N_t &= \max\big(T_t^{(1)},\ n_{\text{floor}}\big), \\ n_{\text{floor}} &= \max\big(5,\ q_{0.25}\{T^{(1)}\}\big), \end{aligned} \tag{B.2}$$

both in control steps, with $q_{0.25}$ the lower quartile of reference suffix lengths over the suite's labelable steps. This is the normalizer used in Eq. (3). Without a floor a checkpoint a few steps from termination would saturate φ to zero on a two-step delay and read as maximally important. With it, a delay still costs proportionally more late than early, which is intended, since late steps leave less horizon for recovery.

Aggregation and the Grid Ceiling. The trapezoidal quadrature of Phase 3 runs over the non-uniform evaluated grid $\mathcal{R}_t$. Writing $r_{(1)} < \cdots < r_{(m_t)}$ for its sorted elements, the reference rollback and the lowest-ratio evaluation ensure $r_{(1)} = r_{\min}$ and $r_{(m_t)} = 1$ at every labelable step, so $\text{AUC}_{\max} = 0.95$ throughout.

That ceiling is the one consequence of the anchor $\varphi_t(1) \equiv 1$ worth making explicit, since it bounds the label away from 1. The final trapezoid spans $[r_{(m_t-1)}, 1]$, where $r_{(m_t-1)} \le \max \mathcal{R}$ and $\varphi_t(1) = 1$. It therefore contributes at least $\frac{1}{2}(1 - \max \mathcal{R})$, giving the uniform bound

$$\text{DI}_t \ \le\ \text{DI}_{\max} \ =\ 1 - \frac{1 - \max \mathcal{R}}{2\,(1 - r_{\min})}, \tag{B.3}$$

which is 0.895 at $\max \mathcal{R} = 0.8$ and $r_{\min} = 0.05$. This is a property of the candidate grid. It is attained when the evaluated grid includes $\max \mathcal{R}$ and all evaluated pruned ratios have zero quality, as occurs when all pruned candidates fail. The label distribution carries an atom at $\text{DI}_{\max}$ instead of a tail reaching 1 (Appendix B.6).

(1) Range and anchors. $\text{DI}_t \in [0, \text{DI}_{\max}]$ with $\text{DI}_{\max}$ given by Eq. (B.3). $\text{DI}_t = 0$ iff every evaluated ratio has $\varphi_t(r) = 1$. The upper bound is attained iff $\max \mathcal{R} \in \mathcal{R}_t$ and every evaluated pruned ratio has $\varphi_t(r) = 0$. Both bounds are attained in practice (Appendix B.6).

(2) Monotonicity. DI_t is non-increasing in each $\varphi_t(r_{(j)})$, so a pointwise more degraded suffix never receives a lower importance.

B.2 Key Step Selection

Adjacent control steps yield near-duplicate supervision, so each baseline rollout is labeled at a compact subset of its n control steps. The rule below runs *after* the rollout completes, on the cached per-step action chunks and attention maps.

Trigger Set and Thresholds. Let $a^{(i)} \in \mathbb{R}^{c \times D_a}$ be the chunk predicted at control step i, and $\bar{g}_i$ the mean of its gripper channel over the chunk horizon.

1. **Gripper transition**. Flag i if $|\bar{g}_i - \bar{g}_{i-1}| > 0.5$, in the policy's native gripper range ($[0, 1]$ for OpenVLA-OFT, before the $[-1, 1]$ remap applied to the executed action). A half-range jump is a commanded open↔close event, not jitter. These are the grasp and release moments, where the visual evidence collapses onto a small contact region.

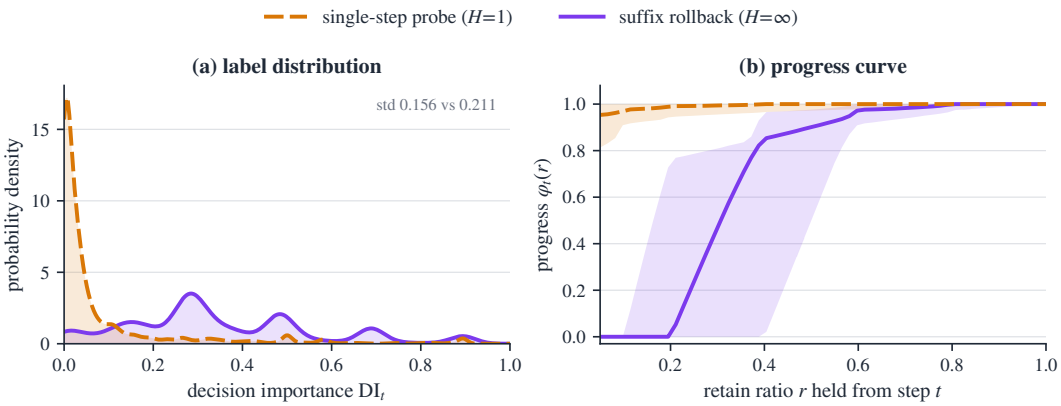


Figure B.1: Single-step probe vs. suffix rollback on the same control steps. (a) DI distributions: single-step labels are more concentrated near zero. (b) Mean quality curves $\varphi_t(r)$: the single-step intervention followed by full-token execution produces higher mean quality at reduced retention ratios, while sustained pruning reveals larger downstream degradation.

2. **Action-chunk discontinuity**. With $d_i = \|\text{vec}(a^{(i)}) - \text{vec}(a^{(i-1)})\|_2$, flag i if $d_i > 2 \cdot \text{median}\{d_1, \ldots, d_{i-1}\}$. This trigger is stored under the historical key `action_variance_spike`.

3. **ViT patch-attention entropy**. Select the $K_e = 5$ steps of highest Shannon entropy $\mathcal{H}_i = -\sum_j p_{ij} \log p_{ij}$ of the vision-encoder patch attention, normalized to a distribution by $p_i = \text{relu}(A_i)/\|\text{relu}(A_i)\|_1$. High entropy means the evidence is spread over many patches, which is precisely the regime a fixed budget starves.

Coverage and Quantity Control. Let S be the set flagged so far and $C = \{1, \ldots, n\} \setminus S$ its complement.

4. **Time coverage**. For each quantile $q \in \{0.2, 0.4, 0.6, 0.8\}$, if no member of S lies within $\pm 0.05n$ of qn, add $\lfloor qn \rfloor$. This guarantees the label set spans the rollout even for tasks whose events cluster in one phase.

5. **Padding**. If $|S| < k_{\min} = 5$, add evenly spaced steps from C until $|S| = k_{\min}$. This rule was never activated in the data of Table B.2, because the entropy trigger already selected five steps per episode.

6. **Trimming.** If $|S| > k_{\max} = 10$, rank by (has a visual trigger, number of triggers, earlier index) and keep the top $k_{\max}$.

Finally, up to $m = 8$ *uniform* steps are drawn from C with a fixed stride $\lfloor |C|/m \rfloor$ (deterministic, not random) for temporal coverage of the non-event portion of the rollout. Each episode therefore contributes between $k_{\min}$ and $k_{\max} + m = 18$ labeled control steps. Remaining steps are stored state-only: their simulator state is kept so they *could* be labeled later, but no features are cached and no rollback is run.

Key steps undergo coarse-to-fine frontier search. Uniformly sampled steps first evaluate the full-token reference ($r = 1$). If it fails, the DI label is undefined and the step is excluded from aggregation. Otherwise, they evaluate the lowest candidate ratio ($r = r_{\min}$). If this probe succeeds, collection terminates after two rollbacks; if it fails, the step is upgraded to coarse-to-fine search, reusing both probe results. Thus the number of rollbacks depends on the search outcome, rather than being fixed at two for uniformly sampled steps. Both step classes use full-suffix interventions ($H = \infty$), and their DI labels are computed from the ratios actually evaluated, as specified in Appendix B.1. Table B.2 reports trigger counts and DI statistics for the defined-DI key steps of unperturbed StarVLA episodes, excluding the separately sampled uniform steps.

Table B.3: Single-step probe ($H{=}1$) vs. suffix rollback ($H{=}\infty$)

Statistic	$H{=}1$	$H{=}\infty$
$\overline{\mathrm{DI}}$	0.079	0.347
std	0.156	0.211
DI$<$0.02 (%)	49.6%	4.8%
DI$>$0.5 (%)	3.5%	18.1%

B.3 Single-Step Probe vs. Suffix Rollback

Sec. 2 motivates sustained pruning as a way to measure cumulative closed-loop consequences. This subsection compares the label distributions induced by single-step and full-suffix interventions.

Protocol. The two arms differ in exactly one collection parameter, the intervention window H: the retain ratio is applied for the first H control steps after the restore and then reverted to $r = 1$.

- **Single-step probe**, $H = 1$: the ratio applies to one control step, after which the controller sees unpruned observations again.
- **Suffix rollback**, $H = \infty$: the ratio is held until success or timeout. This is the setting used for all predictor-training, fine-tuning, and calibration labels; the finite-horizon arms are diagnostic only.

Everything else is shared: the same baseline episodes, the same selected control steps, the same restored states, the same $\mathcal{R}$, the same budget Eq. (B.1), the same seed per step, and the same $\varphi \to \mathrm{DI}$ pipeline.

Result. Fig. B.1 and Table B.3 report the comparison. Under the single-step probe, 49.6% of labels fall below $\mathrm{DI} = 0.02$. Subsequent full-token execution can compensate for an isolated perturbation, so this intervention can yield a different sensitivity signal from repeated pruning. Under the suffix intervention, the fraction with $\mathrm{DI} > 0.5$ increases from 3.5% to 18.1%. These distributions motivate measuring sustained effects; their difference alone does not establish that single-step labels are unsuitable for supervision.

Action Divergence as an Alternative Signal. The cheapest alternative to a suffix rollback is to prune at the current step and measure how far the action moves, $\|a_r^{(t)} - a_1^{(t)}\|$. It costs one forward pass and no simulator reset, and it is the kind of surface cue that heuristic pruners use to modulate their budgets. Fig. B.2 examines how this signal varies across suites and retention ratios.

Divergence tracks how far pruning moves the network's output, not what it costs the task. That displacement depends mainly on the architecture and the ratio, both shared across suites, so the curves in Fig. B.2a lie on top of one another, separated by far less than their own dispersion. Fig. B.2b shows the consequence: mapped to a retention ratio, the four suite means fall within 0.03 of each other, whereas Table 1 has these suites tolerating fixed ratios that differ by tens of points. This suite-level comparison does not establish how well action divergence ranks control steps within a task.

B.4 Effect of the Intervention Horizon

We compare the available StarVLA-OFT labels on LIBERO under three intervention horizons. For $H = 1$ or $H = 16$, each candidate ratio is applied for the first H control steps after restoration and full-token execution resumes thereafter; $H = \infty$ holds the ratio until termination. Labeled control steps are matched by suite, task, episode, and control-step index. Each comparison uses only control steps with labels available under both horizons, so the sample count depends on the pair. These paired summaries are reported separately from the aggregate distributions in Table B.3.

Table B.4 shows that $H = 1$ and full-suffix labels have low rank agreement ($\rho = 0.10$), while $H = 16$ and full-suffix labels are more closely associated ($\rho = 0.64$). For the latter pair, control steps with longer reference suffixes exhibit a larger mean difference and lower rank agreement. These observations show that the measured sensitivity depends on the intervention horizon and includes

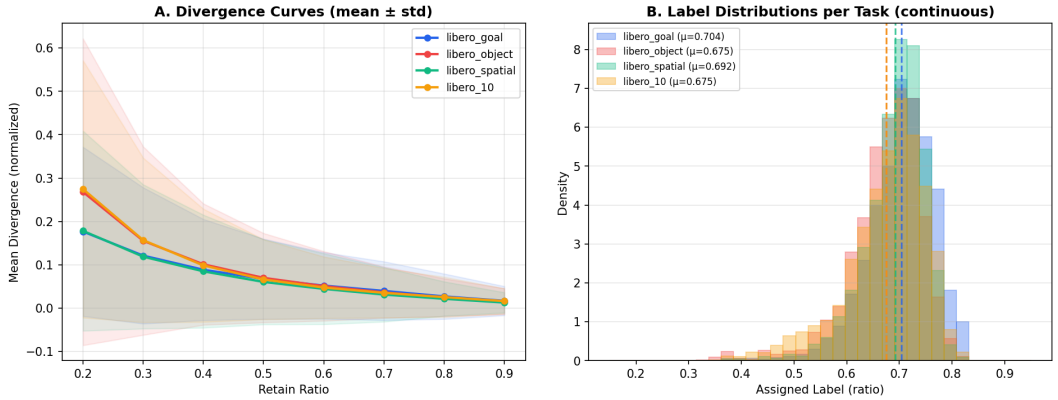


Figure B.2: Action divergence as a candidate supervision signal. (a) Normalised divergence between pruned and unpruned behaviour against retain ratio, per LIBERO suite, mean and one standard deviation. (b) Distribution of the retention ratio each suite would receive if the label were derived from that divergence, with suite means dashed.

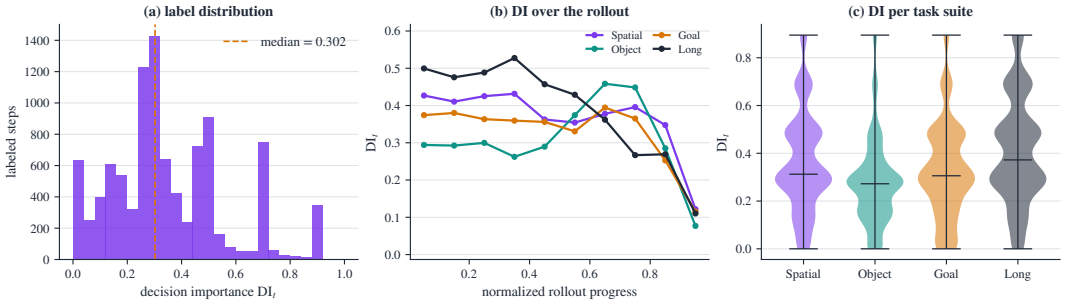


Figure B.3: Decision-importance labels from fixed-ratio suffix interventions. (a) Marginal distribution over labeled control steps. (b) DI against normalized rollout progress. (c) DI per LIBERO suite. Each value summarizes consequences over the remaining rollout, rather than the isolated current action.

later rollout consequences. A full-token reference suffix of at most 16 steps does not imply identical interventions: pruning can prolong the rollout beyond that horizon.

B.5 Supervision Data, Reuse, and Offline Cost

Main Supervision Sets. We collect two main sets of counterfactual suffix rollbacks, both using TQA: one for StarVLA with its OFT action head, and one for OpenVLA-OFT. The StarVLA set contains 2,151 episode records (1,163 original and 988 perturbed) and 32,250 selected control-step records, including 3,620 with undefined DI. The remaining 28,630 are valid DI labels. OpenVLA-OFT contains 962 episode records (381 original and 581 perturbed), with 12,259 valid labels among 14,354 selected control-step records; 2,095 have undefined DI. The separate diagnostic collection has 9,946 StarVLA-OFT full-suffix labels; Appendix B.4 reports the horizon comparisons on pairwise intersections.

Perturbed Reference Trajectories. In addition to full-token trajectories, the main sets include trajectories collected under pruning to broaden the visited-state coverage. Each selected control step is evaluated using a fresh full-token suffix rollback from its saved simulator state, so a perturbed reference trajectory changes which states are sampled without changing the DI definition. Invalid labels are discarded from predictor supervision, while their executed rollbacks remain part of the collection cost.

Table B.4: Paired DI comparisons across intervention horizons on StarVLA-OFT (LIBERO). ρ denotes Spearman rank correlation. Panel (b) groups the $H = 16$ and $H = \infty$ pairs by the remaining length of the full-token reference rollout.

(a) Pairwise horizon comparisons				
Horizon pair	Paired steps	Mean DI (first)	Mean DI (second)	ρ
$H = 1$ vs. $H = \infty$	4,882	0.073	0.337	0.10
$H = 16$ vs. $H = \infty$	4,866	0.318	0.336	0.64
$H = 1$ vs. $H = 16$	4,873	0.072	0.319	0.13
(b) $H = 16$ vs. $H = \infty$ by reference suffix length				
Reference steps	Paired steps	Mean DI ($H = 16$)	Mean DI ($H = \infty$)	ρ
≤ 16	3,910	0.311	0.312	0.69
> 16	956	0.346	0.432	0.44

Predictor Sharing and Action-Head Adaptation. Each action head has one predictor shared by TQA, FastV, and SparseVLM. All predictor training and fine-tuning labels come from TQA rollbacks. The StarVLA-OFT predictor is trained on the main set; the StarVLA-π and StarVLA-GR00T predictors are initialized from it and are each fine-tuned once using their own TQA minibatch. The 118 episodes in each target-head TQA batch are split into 92 for fine-tuning and 26 for validation; the π fine-tuning split contains 1,220 control steps with valid DI labels. FastV and SparseVLM batches do not train separate predictors. Predictor scores provide a shared ordering signal, while the calibrator determines the pruner-specific absolute retention ratio.

Calibration by Head and Pruner. StarVLA has nine head–pruner configurations, each with its own calibrator at the chosen tolerance. On LIBERO, each pair uses a small counterfactual batch from initial-state indices 0–2 of each task, with 118 recorded episodes. Reference trajectories are shared across pruners for a given head, but each pair has its own pruning rollbacks and quality curves. Thus, sharing the predictor does not assume that the pruners have identical DI labels or critical ratios. OpenVLA-OFT uses only TQA, with one predictor and one calibrator; its calibrator is fitted directly on the predictor training set.

LIBERO-PLUS Adaptation. The StarVLA-OFT predictor is transferred unchanged. The π and GR00T predictors are further fine-tuned using their respective TQA minibatches on LIBERO-PLUS and then shared across the three pruners. All nine calibrators are re-fit using the corresponding head–pruner minibatches. The available label records cover 86–93 episodes per pair, as reported in Table B.5. This protocol includes target-benchmark adaptation for the latter two predictors as well as for the calibrators.

Reuse across Tolerances. We retain the full quality curves rather than only their DI integrals. For a fixed head, pruner, and benchmark, the same stored curves produce $r_t^\star(\tau)$ at different tolerances, allowing g_τ to be re-fit without repeating counterfactual collection solely to change τ. The main TQA collection is performed once for each backbone family; additional head–pruner configurations use the smaller adaptation batches rather than repeating the main collection.

Rollback and Environment-Step Accounting. Table B.5 counts the suffix executions actually recorded, including full-token references. A failed full-token reference contributes one rollback even though its DI label is undefined; probe results reused during an upgraded search are counted only once. Adaptive search can skip candidate ratios, so the six-ratio grid does not imply a fixed number of executions per selected control step; the StarVLA main set averages about 5.8 recorded rollbacks per selected control-step record. Environment steps count only execution from the restored state until success or the step cap, without repeating the prefix. These totals exclude generating the original reference trajectories and training the predictor.

Hardware-Independent Workload. We normalize suffix environment steps by one complete 2,000-episode LIBERO evaluation: 327,764 environment steps for StarVLA-OFT and 312,808 for OpenVLA-OFT. All StarVLA rows use the former reference, including the other action heads and

Table B.5: **Offline counterfactual collection workload.** Counts are taken from the label records. Selected control-step records include undefined (NaN) DI values; rollbacks include full-token references. Environment steps count executed suffixes only. *Eval. equiv.* uses the 2,000-episode LIBERO reference described below, including for LIBERO-PLUS batches. The last column is an estimate converted using full-token inference latency, not measured total GPU time.

Collection	Episodes	Selected steps	Rollbacks	Env. steps (M)	Eval. equiv.	Forward equiv. (h)
StarVLA main (OFT, TQA)	2,151	32,250	187,064	30.82	94.0	68.2
OpenVLA-OFT main (TQA)	962	14,354	66,550	8.16	26.1	30.9
LIBERO batch per head–pruner pair	118	1,767–1,778	8,463–9,719	1.13–1.64	3.4–5.0	2.5–17.5
LIBERO batches nine pairs combined	1,062	15,942	81,857	12.44	38.0	84.4
LIBERO-PLUS batch per head–pruner pair	86–93	1,327–1,454	6,325–6,948	0.89–1.18	2.7–3.6	2.0–12.6
LIBERO-PLUS batches nine pairs combined	804	12,435	59,959	9.18	28.0	62.0

The two main sets contain 3,620 and 2,095 undefined DI records, respectively. Each LIBERO batch contains 166–230 such records, and each LIBERO-PLUS batch contains 234–324. Combined episode counts sum configuration-specific records; reference trajectories shared across pruners are not unique new trajectories in these totals. Ranges summarize individual configurations.

LIBERO-PLUS batches. Thus the two main sets cost 94.0 and 26.1 LIBERO reference equivalents, while a LIBERO adaptation batch costs 3.4–5.0 equivalents per head–pruner pair. These are environment-step equivalents, not matched wall-clock runs or full LIBERO-PLUS evaluations.

Inference-Time Conversion. For an approximate conversion to full-token-equivalent inference hours, let E_h be the suffix environment-step count for head h and ℓ_h^{full} its full-token forward latency in milliseconds. Using eight environment steps per action chunk, we report

$$\widehat{C}_{\text{fwd}} = \sum_h \frac{E_h}{8} \frac{\ell_h^{\text{full}}}{3.6 \times 10^6}. \tag{B.4}$$

The audit uses 63.7 ms for StarVLA-OFT, 355.9 ms for StarVLA-π, 167.6 ms for StarVLA-GR00T, and 108.9 ms for OpenVLA-OFT, from the full-token timing records accompanying this cost audit. These conversion factors are separate from the module timings in Table D.2. The estimate excludes simulation, resets, data handling, predictor training, and scheduling, and does not account for partially executed final chunks or the change in forward cost under pruning. It is a normalized inference-work estimate, not measured total GPU usage or wall-clock time. Predictor training uses one GPU for 15,000 optimization steps (Table C.7); its runtime is not included in Table B.5.

B.6 Label Distribution

Fig. B.3 shows the collected labels. The raw DI distribution is multimodal. Averaging within each suite gives four curves that stay separated over most of the episode, with Long highest and Object lowest, as the fixed-ratio results of Table 1 of the main paper would suggest. The per-suite distributions differ markedly, with LIBERO Long shifted highest. Because each label evaluates the remaining rollout, its temporal pattern can reflect changes in future failure and delay as well as the conditions at the current control step. The curves therefore describe suffix sensitivity, not a direct measurement of instantaneous visual demand.

C Predictor and Calibrator

This appendix specifies the two online components of DVA: the decision-importance predictor f_θ (Eq. (5)) and the score-to-ratio calibrator g_τ (Eq. (8)). Sec. C.1 gives the architecture, parameter count

and FLOPs of f_θ; Sec. C.2 describes the pooling operators used to reduce the variable-length token sequences to fixed-size summaries; Sec. C.3 writes out every term of the training objective in Eq. (6) together with its hyperparameters; and Sec. C.4 details the isotonic fit of g_τ, the non-decreasing envelope and interpolation used to obtain the critical ratio $r_t^\star(\tau)$ in Eq. (7), and the clipping applied to the deployed ratio r_t.

C.1 Architecture of f_θ

Overview. f_θ is a two-branch encoder followed by a fusion MLP and a scalar prediction head. It consumes only tensors the VLA has *already* materialised during its own forward pass, so no extra vision or language computation is introduced:

- $\mathbf{v}_t \in \mathbb{R}^{N_v \times D_v}$ — the projected visual tokens, i.e. the output of the projector $\mathcal{P}(\cdot)$ that is fed to the LLM. D_v is the LLM hidden width and N_v denotes the number of visual tokens.
- $\boldsymbol{\ell} \in \mathbb{R}^{N_l \times D_l}$ — the instruction token embeddings. $D_l = D_v$, and $N_l \approx 32$ for LIBERO prompts. $\boldsymbol{\ell}$ is constant within an episode.
- $\mathbf{p}_t \in \mathbb{R}^{D_p}$ — *optional* raw proprioceptive state ($D_p = 8$ on LIBERO: end-effector pose + gripper). The branch is instantiated only when the supervision set carries proprioception, which is the case on OpenVLA-OFT; the StarVLA runs omit it.

Each modality is reduced to a single D-dimensional summary, the summaries are concatenated and fused, and a two-layer head emits one logit that is squashed by a sigmoid:

$$\widehat{\mathrm{DI}}_t \;=\; \sigma\Big(W_2\,\mathrm{GELU}\big(W_1\,[\,\mathbf{u}_t^v;\mathbf{u}^\ell;\mathbf{u}_t^p\,]\big)\Big) \;\in\; (0,1), \tag{C.1}$$

where $\mathbf{u}_t^v, \mathbf{u}^\ell, \mathbf{u}_t^p \in \mathbb{R}^D$ are the pooled summaries.

Branch encoders. The two branches are pooled differently. The *visual* branch uses **channel max-pooling**: a bias-free projection to the working width followed by an element-wise maximum over the token axis and a LayerNorm,

$$\mathbf{u}_t^v \;=\; \mathrm{LN}\Big(\max_{i \le N_v}\big(\mathbf{v}_{t,i}\,W_v\big)\Big), \qquad W_v \in \mathbb{R}^{D_v \times D}, \tag{C.2}$$

where the maximum is taken independently per channel. This carries no learnable query and no attention: the summary is *the strongest response anywhere in the visual field*. Sec. C.2 reports the comparison that motivated it.

The *language* branch uses **single-query cross-attention**: one learnable query $\mathbf{q} \in \mathbb{R}^{1\times D}$ attends over the instruction tokens,

$$[\mathbf{K};\mathbf{V}] = \boldsymbol{\ell}\,W_{kv}, \quad W_{kv} \in \mathbb{R}^{D_l \times 2D} \text{ (no bias)}, \tag{C.3}$$

$$\mathbf{u}^\ell = \mathrm{LN}\big(\mathrm{MHA}_H(\mathbf{q},\mathbf{K},\mathbf{V})\big) \;\in \mathbb{R}^D. \tag{C.4}$$

W_{kv} is a bias-free map that first projects the wide backbone features down to the working width, so the attention itself runs at $D = 1024$ rather than $D_l = 4096$; MHA_H is a standard multi-head attention block. A single query suffices because the target is a scalar. Table C.1 gives the exact configuration.

Parameter count. Table C.2 gives the closed-form parameter count of each module and its value at the OpenVLA-OFT configuration (D_v=D_l=4096, D=1024, H=4, D_p=8). The total is **22.0 M** parameters, about 0.31% of the 7B policy it steers.

Table C.3 instantiates the same architecture and the same $(D, H, \text{pooling})$ on both backbone families. The only differences are backbone-imposed: the LLM width D_v, the visual-token count N_v, and whether proprioception is available. StarVLA is the cheaper instantiation on both axes: a narrower backbone (2560 vs 4096) and $4\times$ fewer visual tokens (128 post-PatchMerger vs 512 patches).

Table C.1: Configuration of the language branch's cross-attention pool. The visual branch is max-pooled (Eq. (C.2)).

Setting	Value
Number of learnable queries	1
Embedding width D	1024
Heads H	4
Pre-projection W_{kv}	$D_l \to 2D$, no bias
Output normalisation	LayerNorm(D)
Fusion / head dropout	0.1
Activation	GELU
Weight init.	trunc. normal σ=0.02; biases 0
Visual pool (Eq. (C.2))	$D_v \to D$ no bias, ch. max, LN

Table C.2: Parameter count of f_θ as deployed. Values are for OpenVLA-OFT (D_v=D_l=4096, D=1024, H=4, D_p=8). MHA$(D) = 4D^2 + 4D$, LN$(D) = 2D$.

Module	Closed form	Params
Visual pool (max)	$DD_v + 2D$	4,196,352
Language pool	$2DD_l + 4D^2 + 7D$	12,590,080
Proprio encoder	$DD_p + D^2 + 4D$	1,060,864
ConcatFusion	$3D^2 + D$	3,146,752
Prediction head	$D^2 + 2D + 1$	1,050,625
Total		**22,044,673**

FLOPs and latency. Table C.4 reports the per-control-step cost of one f_θ forward pass at batch size 1, counting one multiply–accumulate as 2 FLOPs. The total is 4.98 GFLOPs, of which 86% is the visual branch: the $N_v \times D_v \to D$ projection over all 512 patch tokens is the only term that scales with the token count, and it is what makes the visual side dominate the *compute* even though it is the smaller side in *parameters*. The language branch scales linearly in N_l but contributes under 14%, so the exact prompt length is immaterial. For reference, a single OpenVLA-OFT prefill over the same sequence (552 tokens through 32 LLaMA-2-7B layers) costs ≈ 7.3 TFLOPs, so f_θ **adds** 0.068% **of the backbone FLOPs**.

C.2 Pooling Operators

The pooled summary $\mathbf{u}_t^v$ must compress N_v patch tokens into one vector. Because this choice determines both the inductive bias and the parameter budget of f_θ, we implemented and compared three operators.

1. **Channel max-pooling** (MaxPool; *the one we deploy*). A bias-free projection $\mathbf{X}W$ ($D_{\text{in}} \to D$) followed by an element-wise maximum over the token axis and a LayerNorm, i.e. Eq. (C.2).
2. **Single-query cross-attention** (LearnablePool). One learnable query attends over all tokens. The attention weights are content-dependent, so the pool can concentrate on task-relevant regions. This is the operator the language branch retains.
3. **Spatial pyramid pooling** (PyramidPool). The flat token sequence is reshaped back to its per-view $H \times W$ grid (16×16 on OpenVLA-OFT, 8×8 on StarVLA), projected by a 1×1 convolution, and average-pooled in parallel at bin resolutions $\{1, 2, 4\}$. The 21 resulting cells per view carry additive learnable scale and camera-view embeddings and serve as keys/values for a single-query cross-attention module. Unlike the other two, this operator *retains spatial structure* (where in the frame the evidence lies) while attending over only $2 \times 21 = 42$ keys instead of 512.

Table C.5 gives the cost of each operator, and Table C.6 the accuracy comparison.

Table C.3: f_θ instantiated on the two backbone families. The predictor configuration is identical (D=1024, H=4, MAXPOOL visual branch); only the backbone-imposed quantities differ. D_v is the *LLM* hidden width, since f_θ reads post-projector features. FLOPs are per control step at batch 1.

Backbone (LLM)	$D_v{=}D_l$	N_v	D_p	Params (M)	FLOPs (G)
OpenVLA-OFT (LLaMA-2-7B)	4096	512	8	22.04	4.98
StarVLA (Qwen3-VL-4B)	2560	128	–	15.22	1.15

Table C.4: Per-step cost of f_θ on OpenVLA-OFT (N_v=512, N_l=32, D=1024, batch 1; 1 MAC = 2 FLOPs).

Module	FLOPs	Share
Visual pool (max)	4.295 G	86.2%
Language pool	0.675 G	13.6%
Proprio + fusion + head	10.5 M	0.2%
f_θ total	**4.981 G**	**0.068**% of VLA
g_τ (isotonic + EMA)	$< 10^3$	≈ 0
OpenVLA-OFT prefill	7.31 T	–

Discussion. We deploy MAXPOOL: it is both the cheapest variant and the most accurate on every metric, cutting MAE from 0.136–0.138 to 0.119 and roughly doubling R^2 (0.416 versus $R^2{\approx}0.21$ for both alternatives).

C.3 TRAINING OBJECTIVE

The predictor is trained with $\mathcal{L} = \mathcal{L}_{\text{reg}} + \lambda_{\text{rank}}\mathcal{L}_{\text{rank}} + \lambda_{\text{var}}\mathcal{L}_{\text{var}}$, with $\lambda_{\text{rank}} = 1.0$ and $\lambda_{\text{var}} = 0.3$ throughout. Write $\hat{y}_i \equiv \widehat{\text{DI}}_{t_i}$ and $y_i \equiv \text{DI}_{t_i}$ for the i-th element of a minibatch of size B.

$\mathcal{L}_{\text{reg}}$: Huber (smooth-L_1). The regression term anchors predictions to the measured DI labels using the smooth-L_1 form:

$$\mathcal{L}_{\text{reg}} = \frac{1}{B}\sum_{i=1}^{B} w_i\, h_\beta(\hat{y}_i - y_i),$$
$$h_\beta(x) = \begin{cases} \frac{1}{2\beta}x^2, & |x| < \beta, \\ |x| - \frac{\beta}{2}, & \text{otherwise}, \end{cases} \tag{C.5}$$

with $\beta = 1$ and $w_i \equiv 1$ in all reported runs. Under these settings the robustness is nominal: since $\hat{y}_i, y_i \in [0,1]$, the residual never exceeds 1, so the loss stays in its quadratic branch and is *exactly* $\frac{1}{2}$ MSE on our data. We report it as implemented so that the λ values in Eq. (6) remain reproducible; the Huber form only matters if β is lowered.

$\mathcal{L}_{\text{rank}}$: margin ranking with an adaptive margin. The ordering of predicted scores is important for the monotone calibrator. We therefore supplement regression with a hinge penalty over a set $\mathcal{P}$ of sampled index pairs,

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{P}|}\sum_{(i,j)\in\mathcal{P}} \Big[\underbrace{m_0 + |y_i - y_j|}_{\text{margin } m_{ij}} - \text{sgn}(y_i - y_j)\,(\hat{y}_i - \hat{y}_j) \Big]_+, \tag{C.6}$$

Table C.5: Cost of the three pooling operators, at the OpenVLA-OFT configuration (D=1024). "Visual params" isolates the pooled branch; "Total" includes the shared language branch, proprioception, fusion and head.

Visual pool	Visual params	Total params	FLOPs
MAXPOOL (deployed)	4.20 M	22.04 M	4.98 G
LEARNABLEPOOL	12.59 M	30.44 M	11.43 G
PYRAMIDPOOL	8.42 M	26.27 M	5.16 G

Table C.6: Pooling ablation at D=1024. Predictor quality is evaluated using validation R^2, MAE, and Pearson r between predicted and ground-truth DI.

Visual pool	val R^2 ↑	val MAE↓	Pearson r ↑
MAXPOOL (deployed)	**0.416**	**0.119**	**0.715**
LEARNABLEPOOL	0.206	0.138	0.656
PYRAMIDPOOL	0.208	0.136	0.672

where $[\cdot]_+ = \max(\cdot, 0)$ and $m_0 = 0.02$ is the base margin. For a label gap Δ, the pair incurs no hinge penalty only when its signed prediction gap reaches $m_0 + \Delta$. Larger label differences thus encourage greater score separation. This is a soft objective rather than a hard constraint. In particular, exact label predictions still incur a penalty of m_0 for each non-tied pair, so the positive base margin introduces a trade-off between numerical fit and separation, controlled by λ_{rank}.

$\mathcal{P}$ is resampled at every step: we draw $P = \min(B, 64) = 64$ index pairs uniformly with replacement from the batch and discard self-pairs and pairs whose labels are equal to within $\epsilon = 10^{-6}$ (self-pairs and exact ties carry no ordering signal, while near-ties provide only weak distinctions); the term costs $\mathcal{O}(P)$.

$\mathcal{L}_{\text{var}}$: variance matching. Mean-seeking predictions can compress score variation, reducing the distinctions available to the calibrator. To discourage this behavior, we penalise the minibatch dispersion gap,

$$\mathcal{L}_{\text{var}} = \big(\text{std}(\hat{y}) - \text{std}(y)\big)^2, \tag{C.7}$$

using the sample standard deviation computed with denominator $B - 1$, evaluated per batch and returning 0 when $B < 2$. The two auxiliary terms act at different levels: $\mathcal{L}_{\text{var}}$ penalizes a difference in batch-wide dispersion but does not determine *which* control steps receive high scores or guarantee correct ordering, whereas $\mathcal{L}_{\text{rank}}$ acts only on the sampled pairs and, being a hinge, contributes no gradient once a pair's margin is satisfied.

Training Configuration. Table C.7 gives the predictor configuration. One predictor per action head is shared across pruners and tolerances, with all training and fine-tuning labels obtained using TQA. StarVLA-π and GR00T initialize from StarVLA-OFT and receive head-specific fine-tuning. On LIBERO-PLUS, OFT remains unchanged, π/GR00T are fine-tuned again, and all nine calibrators are re-fit. Appendix B.5 specifies the $92/26$ episode split, calibration data, OpenVLA-OFT training-set calibration, and collection costs.

C.4 CALIBRATOR DETAILS

This appendix specifies the two stages of the calibration pipeline in the main paper: how the critical ratio $r_t^\star(\tau)$ of Eq. (7) is extracted from the counterfactual grid, and how the isotonic map g_τ of Eq. (8) is fitted and evaluated.

Eq. (7) defines $r_t^\star(\tau) = \min\{r : \varphi_t(r) \geq 1 - \tau\}$, but φ_t is observed only on the finite grid $\mathcal{R} \cup \{1\}$ with $\mathcal{R} = \{0.05, 0.1, 0.2, 0.4, 0.6, 0.8\}$ and $\varphi_t(1) \equiv 1$. Taking the minimum literally would quantise $r_t^\star$ to seven values and discard most of the resolution the rollbacks provide, so we evaluate the crossing in two steps.

Table C.7: Training configuration of f_θ. Shared by OpenVLA-OFT and the three StarVLA action-head variants; differences between backbone families are the input widths and the presence of the proprioceptive branch (Table C.3).

Setting	Value
Optimisation	
Optimiser	AdamW, β=$(0.9, 0.999)$, eps 10^{-8}
Weight decay	10^{-5} (all parameters)
Peak LR / schedule	2×10^{-4}, warm-up $\rightarrow$ cosine to 0
Steps (warm-up/total)	1,000 / 15,000
Batch size	128 frames (no accumulation)
Gradient clipping	global ℓ_2 norm 1.0
Precision	fp32 (features cast from fp16)
Objective (Sec. C.3)	
λ_{rank}, λ_{var}	1.0, 0.3
Margin m_0, tie thr. ϵ	0.02, 10^{-6}
Pairs per step P	$\min(B, 64) = 64$
Sample re-weighting	off
Architecture (Sec. C.1)	
Visual / language pool	MAXPOOL / LEARNABLEPOOL (1 query)
D, H	1024, 4
Proprioceptive branch	OpenVLA-OFT: on (D_p=8); StarVLA: off
Dropout	0.1
Output range	$(0, 1)$, i.e. plain sigmoid
Data and evaluation	
Supervision set	all 4 LIBERO suites, pooled
Train/val split	per-task episode holdout
Validation	every 500 steps (+ final), batch 256
Reproducibility	
Seeds	42 for training and (independently) split
Checkpointing	every 5,000 steps

Non-decreasing envelope. For calibration, we impose a non-decreasing relationship between retention and suffix quality as a modeling assumption. Each $\varphi_t(r)$ comes from a *single* counterfactual rollout, and the observed grid can contain local inversions. We replace φ_t by its running maximum over ascending r,

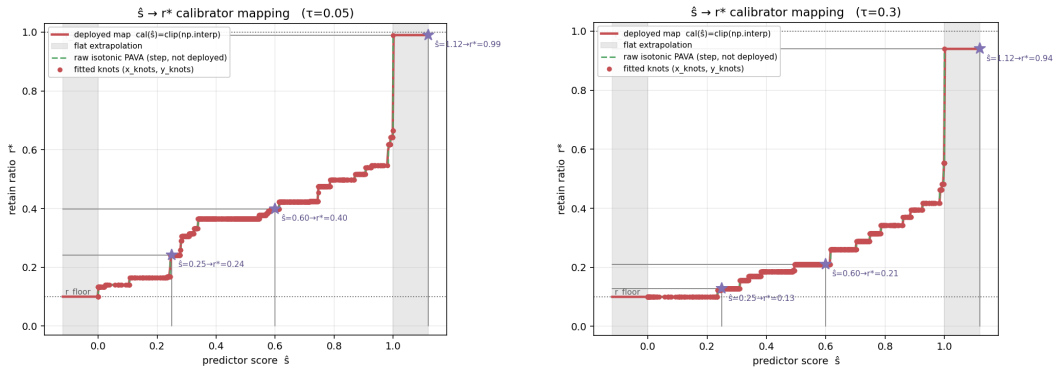
$$\bar{\varphi}_t(r_k) = \max_{m \le k} \varphi_t(r_m), \qquad r_1 < \cdots < r_K, \tag{C.8}$$

the smallest non-decreasing function that upper-bounds φ_t at every grid point. This is a cumulative maximum rather than an isotonic fit: we want an envelope, not a least-squares projection, so that a local downward fluctuation does not lower the fitted quality at larger ratios. The resulting threshold crossing is an estimate under this imposed monotonicity. The envelope is enabled throughout; it can be disabled via a flag in the released code.

Linear interpolation of the crossing. Let k be the smallest index with $\bar{\varphi}_t(r_k) \ge 1 - \tau$. Then

$$r_t^\star(\tau) = \begin{cases} r_1, & k = 1, \\ r_{k-1} + \lambda_k \, (r_k - r_{k-1}), & k > 1, \\ r_K, & \text{otherwise}, \end{cases} \tag{C.9}$$

where $\lambda_k = \big[(1-\tau) - \bar{\varphi}_{k-1}\big] \big/ \big[\bar{\varphi}_k - \bar{\varphi}_{k-1}\big]$ and $\bar{\varphi}_k \equiv \bar{\varphi}_t(r_k)$. The middle case interpolates the crossing between the two bracketing grid points, giving a continuous sub-grid estimate; minimality of k gives $\bar{\varphi}_t(r_{k-1}) < 1 - \tau \le \bar{\varphi}_t(r_k)$, so the denominator is strictly positive. The boundary



(a) $\tau = 0.05$ (b) $\tau = 0.30$

Figure C.1: The calibrator g_τ fitted at two tolerances. Red: the deployed piecewise-linear map and its fitted knots; green dashed: the raw PAVA step function it interpolates; shaded: the flat-extrapolation regions outside $[x_1, x_K]$; dotted: $r_{\text{floor}} = 0.1$. Vertical markers give g_τ at three reference scores. Raising τ lowers the budget across the mid-range while the high-$\widehat{\text{DI}}$ end stays near full tokens.

cases are saturations: $k = 1$ means the most aggressive ratio already meets the bar, while the third branch is defensive and unreachable as long as $\varphi_t(1) \equiv 1$. Hence $r_t^\star(\tau) \in [\min \mathcal{R}, 1] = [0.05, 1]$ by construction.

Isotonic fit of g_τ. Given the paired samples $\{(\widehat{\text{DI}}_t,\, r_t^\star(\tau))\}_{t=1}^n$, g_τ is the weighted least-squares projection of $r^\star$ onto the cone of non-decreasing functions of $\widehat{\text{DI}}$. The fit uses the *predictor's own outputs*. The fitted values at the calibration samples are invariant to strictly increasing score transformations, which preserve order and ties. This invariance does not in general extend to the piecewise-linear deployment map between knots. After non-finite pairs are dropped, the scores are reduced to their K unique ascending values $x_1 < \cdots < x_K$, each knot carrying the mean $r^\star$ of the pairs at that score and a weight w_k equal to their count; this makes the fit a weighted problem on K knots with strictly increasing abscissae. PAVA then solves $\min_{y_1 \le \cdots \le y_K} \sum_k w_k (y_k - \bar{r}_k^\star)^2$ in $\mathcal{O}(K)$ by scanning left to right and merging any block whose mean violates monotonicity into its predecessor, and the fitted values are clipped to $[r_{\text{floor}}, r_{\text{ceil}}] = [0.1, 1.0]$.

Evaluation at deployment. g_τ is stored as the knot pairs $\{(x_k, y_k)\}_{k \le K}$ and evaluated by piecewise-linear interpolation, with *flat* extrapolation outside $[x_1, x_K]$, followed by the same clip:

$$g_\tau(s) = \text{clip}\Big(\text{interp}\big(s; \{(x_k, y_k)\}\big),\; r_{\text{floor}},\; r_{\text{ceil}}\Big). \tag{C.10}$$

Flat extrapolation is deliberate: a score outside the calibration range carries no information about how far outside it lies, so the safest action is to hold the boundary budget.

Fig. C.1 shows g_τ fitted at two tolerances on the same calibration set. Raising τ from 0.05 to 0.3 shifts the map downward across the mid-range while leaving the top end nearly fixed.

D Experimental Implementation Details

This appendix covers the remaining ingredients needed to reproduce Sec. 4: the closed-loop evaluation protocol of the two benchmarks (Sec. D.1), the four VLA configurations and where the compression is inserted in each (Sec. D.2), the hyperparameters of every pruning baseline (Sec. D.3), the definition of the Task-Query Attention (TQA) pruner we introduce (Sec. D.4), and the environment, seeds and measurement protocol (Sec. D.5).

Table D.1: Backbone configurations. All four take 224×224 inputs from two camera views (third-person and wrist) and prune at $L^\star=1$. A cell spanning several columns means those variants share the value. The three StarVLA variants share the vision encoder, the LLM and the pruning mechanism, and differ only in the action decoder and in how that decoder consumes the now variable-length VLM output. d is the LLM hidden width, which is also the input width of the importance predictor (Table C.3).

	OpenVLA-OFT	**StarVLA** (one shared Qwen3-VL-4B-Instruct backbone)		
		OFT	π	**GR00T**
Vision-language backbone				
Vision encoder	SigLIP + DINOv2 (Prismatic)	Qwen3-VL ViT + PatchMerger (2×2)		
LLM, hidden width d	Llama-2 7B, 4096	Qwen3-VL-4B-Instruct, 2560		
Visual tokens N_v	$2 \times 256 = 512$	$2 \times 64 = 128$		
Non-visual prefix	1 proprio + text	chat-template text + action anchors		
Action decoder				
Type	L1-regression head	Parallel decoding	Flow matching	Dual system
Proprioception	8-D		-	
Interface to the compressed prefix				
Head reads	Full-length hidden states	Full-length, index-aware	Variable-length context	
Compression handling	Positions preserved	Scatter-back to S	Native	

D.1 Benchmarks

LIBERO. We use the four standard evaluation suites of LIBERO (Liu et al., 2023), SPATIAL, OBJECT, GOAL and LONG, each containing 10 tasks, and follow the per-suite protocol of OpenVLA-OFT: 50 trials per task, i.e. **500** episodes per suite and $\mathbf{2{,}000}$ per full evaluation. Each episode is preceded by $n_{\text{wait}} = 10$ no-op actions so that objects settle; these are excluded from the reported episode length.

LIBERO-PLUS. LIBERO-PLUS (Fei et al., 2025) keeps the four LIBERO suite names but replaces each task with a large family of perturbed variants, for a total of $\mathbf{10{,}030}$ episodes spanning *seven* robustness dimensions: camera pose of the third-person view, robot initial configuration, paraphrased language instruction, scene illumination, background and surface texture, observation sensor noise, and object/distractor layout. Because the perturbations themselves provide the variation, each variant is evaluated for a *single* trial, so a suite's episode budget is its number of variants.

D.2 Backbone Configurations

The main comparison (Table 2) is run on **OpenVLA-OFT**. The generalisation study (Table 3) additionally uses three variants from the StarVLA codebase (StarVLA Community, 2026) that share one vision-language backbone, **Qwen3-VL-4B-Instruct**, and differ only in the action decoder: **StarVLA-OFT** (parallel decoding), **StarVLA-π** (flow matching) and **StarVLA-GR00T** (dual-system). Table D.1 summarises them.

OpenVLA-OFT. We use the released 7B OFT checkpoint (LoRA rank 32) with the parallel-decoding, continuous-action configuration: L1-regression action head, action chunk of 8, proprioceptive input enabled, two camera views, centre-cropped 224×224 inputs. The Prismatic vision backbone produces $16 \times 16 = 256$ patch tokens per view, so the LLM prefix is [BOS] $[v_1 \ldots v_{512}]$ [proprio] [text] with $N_v = 512$.

StarVLA variants. StarVLA-OFT pairs the Qwen3-VL backbone with an OFT-style parallel-decoding head that is *index-aware*. StarVLA-π (a π_0-style flow-matching action expert) and StarVLA-GR00T (a GR00T-style dual-system diffusion head) are instead *context consumers*: their action queries cross-attend to the prefix as a variable-length key/value sequence, so both accept the compressed output natively and need no scatter-back. Both decode action chunks with 10 DDIM steps.

Table D.2: Per-module latency of one full-token control step, in ms, averaged over the four LIBERO suites. *Vision* is the visual encoder, *VLM* the LLM backbone, *Head* the action decoder together with the proprioceptive projector where present, and *Other* the visual projector, input assembly and host-side overhead.

Model	Vision (ms)	VLM (ms)	Head (ms)	Other (ms)	Total (ms)	VLM (%)
OpenVLA-OFT	33.2	58.8	0.7	14.3	107.0	54.9
StarVLA-OFT	22.0	45.4	1.3	11.9	80.6	56.3
StarVLA-GR00T	24.4	47.8	105.7	12.8	190.7	25.1
StarVLA-π	22.5	51.4	286.1	15.6	375.6	13.7

Where pruning is inserted. For all four VLA configurations the mechanism is identical and touches neither the policy weights nor the token-scoring rule of the pruner: decoder layers $0 \dots L^\star$ run on the full prefix, the selector produces the retained index set at $L^\star$, and layers $L^\star{+}1 \dots N$ run on the compressed sequence. Surviving tokens keep their *original absolute positions*, so the index set has gaps and the RoPE phase relations established in the early layers are preserved. Compression is applied to the prefill only, and we use $L^\star = 1$ everywhere (Table 5). The per-step ratio fed to the selector is $r_t = g_\tau(\widehat{\mathrm{DI}}_t)$ from Appendix C.4, smoothed by the EMA of Sec. 3.2 with $\alpha = 0.5$ and reset at every episode boundary.

Per-module latency. Table D.2 decomposes one full-token control step ($r{=}1$) into the stages of Table D.1, measured under the protocol of Appendix D.5. The three StarVLA variants share a vision encoder and an LLM, and the corresponding two columns are stable across them; the action decoder is the only stage that differs by design, and it differs by more than two orders of magnitude. It therefore sets both the absolute control-step latency and the share of that step spent in the backbone.

D.3 Baseline Configurations and Reproduction

Common harness. FastV, SparseVLM and TQA are re-implemented inside one harness and share *everything* except the rule that decides which visual tokens survive: the same checkpoints, the same episodes and initial states, the same prefill-only compression, the same position handling, and the same latency instrumentation. This isolates the token-selection rule as the only free variable. The single-shot pruners, namely FastV, TQA, and DVA on top of either, additionally share the pruning layer $L^\star = 1$; SparseVLM retains its published pruning-layer schedule, which is part of its mechanism. Table D.3 lists all settings, and the mechanisms themselves are described in Appendix A.

FastV. Each visual token is scored by the attention it receives from the *last* prefill query row, averaged over heads, and the highest-scoring fraction r of visual tokens is retained in a single shot. We prune at $L^\star = 1$ rather than the paper's layer 2, so that FastV, TQA and DVA are compared at the same pruning layer; at a fixed retention ratio a shallower pruning layer is strictly *more* favourable to the baseline in speed, and Table 5 shows the accuracy cost of the choice is small. We use neither token recycling nor a per-layer schedule.

SparseVLM. We follow the released inference recipe for the rater selection, the token recycling and the head averaging (Table D.3). The one necessary change is the budget schedule, which the released code hard-codes as *absolute* keep counts $[300, 200, 110]$ for 576 visual tokens. We instead anchor it to the retention ratio at the last pruning layer and preserve its shape earlier, with $k_{15} = \lfloor r\, N_v \rceil$, $k_6 = \lfloor 200\, k_{15}/110 \rceil$ and $k_2 = \lfloor 300\, k_{15}/110 \rceil$, each capped by the previous layer's count and by $N_v - 1$ as in the official implementation, so that the ratio knob remains meaningful across backbones with different N_v.

VLA-ADP and EfficientVLA. These two are evaluated with the authors' released implementations rather than re-implemented, since their mechanisms are not expressible in our selector interface. We run them with the same OpenVLA-OFT checkpoint under the same 2,000-episode LIBERO protocol, using their default configurations; VLA-ADP is reported at its two operating points Ratio $\in \{0.50, 0.30\}$.

Table D.3: Baseline hyperparameters.

Method	Layers	Scoring	Budget
FastV	1	last prefill row	top-k, one shot
TQA (ours)	1	all task queries	top-k, one shot
SparseVLM	$2, 6, 15$	rater rows	$[300, 200, 110]$
VLA-Cache	$2, 6, 9, 11$	cosine 0.996	$(1-r)N_v/rN_v$

Retention-ratio accounting. For every method we log the *realised* number of visual tokens entering the post-$L^\star$ layers at every control step, and $\bar{r}$ is their step- and token-weighted mean. For the comparison in Table 3, static retention targets are set separately for each suite using the mean ratio from a reference DVA run. Token rounding and evaluation variability can produce small differences in realized average retention. The FLOPs accounting is described in Appendix D.6.

D.4 Task-Query Attention (TQA)

TQA is the simple VLA-tailored pruner we introduce as a strong reference point. It is deliberately minimal and matches the scoring rule used in our data collection and in the motivation analysis of Sec. 2.2.

Definition. Let the prefill sequence at layer $L^\star$ have length S, let $\mathcal{V} \subset \{1, \dots, S\}$ be the visual positions ($|\mathcal{V}| = N_v$) and let $\mathcal{Q}$ be the *task-query* rows, i.e. all non-visual rows of the prefill: the tokenized instruction together with the chat-template text and, on OFT-style backbones, the action-anchor rows that the head later reads. With $A^{(L^\star)} \in \mathbb{R}^{H \times S \times S}$ the causally masked, post-softmax self-attention of layer $L^\star$, the importance of visual token j is the mean attention it receives from the task queries, averaged over heads:

$$s_j \;=\; \frac{1}{H\,|\mathcal{Q}|} \sum_{h=1}^{H} \sum_{q \in \mathcal{Q}} A^{(L^\star)}_{h,\,q,\,j}, \qquad j \in \mathcal{V}. \tag{D.1}$$

Given a retention ratio r, TQA keeps the top

$$K \;=\; \max\bigl(1,\; \min(\lfloor r\,N_v \rfloor,\; N_v)\bigr) \tag{D.2}$$

visual tokens by s, discards the rest, and leaves every non-visual token untouched. The retained indices are sorted in ascending order so the compressed sequence preserves the original token order, and each surviving token keeps its original absolute position id, so the compressed sequence has *gaps* rather than a renumbered position axis. Layers $L^\star{+}1 \dots N$ then run on it. Under dynamic allocation, the only change is that r in Eq. (D.2) is replaced by the per-step $r_t = g_\tau(\widehat{\mathrm{DI}}_t)$.

Cost. TQA reads one $|\mathcal{Q}| \times N_v$ attention block per head at a single layer, i.e. $O(H|\mathcal{Q}|N_v)$ additions plus one top-k over N_v scores. With $|\mathcal{Q}|$ on the order of tens of tokens this is negligible against the S^2 attention cost of the subsequent layers, and it adds no parameters and no additional forward pass.

D.5 Hardware, Software and Seeds

Hardware and Software. All experiments run on a node of eight NVIDIA A800-SXM4 (80 GB) GPUs. Success-rate evaluation, counterfactual collection and the tolerance sweeps are run in parallel. Two stages are exempt: predictor training uses one GPU, and latency is measured in a dedicated pass with a single process alone on a GPU, so the timings of Table 2 are free of the contention present in the parallel runs.

We use Python 3.10, PyTorch 2.6.0 (CUDA 11.8, cuDNN 9.1.0) and NumPy 1.26.4. Modelling uses `transformers` 4.40.1 for OpenVLA-OFT and 4.57.0 for the StarVLA variants, each the version its backbone requires; simulation uses robosuite 1.4.1 and MuJoCo 3.3.2.

Seeds and determinism. Evaluation uses a single global seed of 7; predictor training uses the seeds listed in Table C.7. One seed suffices because the evaluation is deterministic by construction: initial

Algorithm 1 Task-Query Attention (TQA) pruning at layer $L^\star$

Require: prefill embeddings $X \in \mathbb{R}^{S\times d}$; visual positions $\mathcal{V}$; decoder layers $\{\mathrm{Layer}_\ell\}_{\ell=0}^{N}$; pruning layer $L^\star$; ratio r (static, or $r_t = g_\tau(\widehat{\mathrm{DI}}_t)$)
Ensure: action-decoder context Z

$\mathcal{Q} \leftarrow \{1,\dots,S\} \setminus \mathcal{V}$; $P \leftarrow (1,\dots,S)$; $h \leftarrow X$
for $\ell \leftarrow 0$ **to** $L^\star$ **do**
  if $\ell = L^\star$ **then**
    $A \leftarrow \mathrm{Attn}(\mathrm{Layer}_\ell, h, P)$ ▷ causally masked, post-softmax
  end if
  $h \leftarrow \mathrm{Layer}_\ell(h, P)$
end for
$s_j \leftarrow \frac{1}{H|\mathcal{Q}|}\sum_{h'}\sum_{q\in\mathcal{Q}} A_{h',q,j}$ for all $j \in \mathcal{V}$ ▷ Eq. (D.1)
$K \leftarrow \max\big(1, \min(\lfloor rN_v \rfloor, N_v)\big)$
$\mathcal{I} \leftarrow \mathrm{sort}\big(\mathrm{TopK}(s,K) \cup \mathcal{Q}\big)$ ▷ drop only visual tokens
$h \leftarrow h[\mathcal{I}]$; $P \leftarrow P[\mathcal{I}]$ ▷ positions keep gaps, RoPE phases unchanged
for $\ell \leftarrow L^\star{+}1$ **to** N **do**
  $h \leftarrow \mathrm{Layer}_\ell(h, P)$
end for
if the action head is index-aware (OFT) **then**
  $Z \leftarrow \mathrm{ScatterBack}(h, \mathcal{I}, S)$ ▷ restore length S
else
  $Z \leftarrow h$ ▷ context consumers: π, GR00T
end if
return Z

Table D.4: **Analytical FLOPs audit for the 18 StarVLA comparisons in Table 3.** Static and DVA costs are percentages of the corresponding full-token reference under the accounting in Appendix D.6. Δ denotes DVA minus static, in percentage points (pp). Differences retain finer precision than the displayed costs.

		LIBERO			LIBERO-PLUS		
Action head	Pruner	Static (%)	DVA (%)	Δ (pp)	Static (%)	DVA (%)	Δ (pp)
OFT	FastV	64.6	64.9	+0.30	73.5	73.7	+0.14
	TQA	63.2	63.6	+0.36	72.4	72.6	+0.16
	SparseVLM	61.2	61.2	+0.03	76.5	76.3	−0.19
GR00T	FastV	73.5	73.7	+0.22	76.0	76.2	+0.28
	TQA	74.0	74.2	+0.19	67.3	67.6	+0.28
	SparseVLM	59.4	59.5	+0.08	73.5	73.1	−0.41
π	FastV	64.0	64.3	+0.36	70.5	70.7	+0.16
	TQA	69.8	70.1	+0.30	69.9	70.2	+0.37
	SparseVLM	63.1	62.9	−0.22	72.3	72.0	−0.29

The LIBERO results for StarVLA-OFT with FastV and TQA are averages over three repeated runs.

states are read from LIBERO's stored per-task arrays rather than sampled, action decoding is deterministic, the counterfactual rollbacks restore simulator states, and the calibrator fit is deterministic given its input pairs.

D.6 FLOPs Accounting and Retention Variability

We estimate the computational cost of the StarVLA comparisons in Table 3 from the evaluation logs, counting one multiply–accumulate as two FLOPs. For a decoder layer ℓ with n input tokens, the dominant linear transformations and attention products give

$$\widehat{F}_\ell(n) = 2n \sum_{W\in\mathcal{W}_\ell} d_{\mathrm{in}}(W) d_{\mathrm{out}}(W) + 4n^2 H_\ell d_{h,\ell} = a_\ell n + b_\ell n^2, \tag{D.3}$$

where $\mathcal{W}_\ell$ contains the attention and feed-forward projection matrices, H_ℓ is the number of query heads, and $d_{h,\ell}$ is the head dimension. Both variants use the same dense-attention counting convention. We sum layer costs using the token lengths at each pruning stage, include the modeled action-head attention over all denoising steps where applicable, and aggregate costs over policy-inference steps relative to the corresponding full-token reference.

This estimate covers the decoder and modeled action-head attention; it excludes the vision encoder, action-head MLPs, and the DVA predictor, whose cost is reported separately in Appendix C.1. For SparseVLM, the calculation follows the planned layerwise token budgets and does not fully account for recycled merged tokens.

To characterize the effect of retention variability, fix a layer and let $n_t = n_0 + N_v r_t$, where n_0 is a fixed nonvisual length and N_v is the original visual-token count. With $\bar{n} = \mathbb{E}[n_t]$,

$$\mathbb{E}[\widehat{F}_\ell(n_t)] - \widehat{F}_\ell(\bar{n}) = b_\ell \operatorname{Var}(n_t) = b_\ell N_v^2 \operatorname{Var}(r_t). \tag{D.4}$$

Thus, at equal mean length, only the quadratic attention term contributes additional FLOPs; the coefficients are architecture dependent and the identity applies separately to each layer. The observed retention variability gives a small additional cost under this model. Across the 18 StarVLA comparisons in Table D.4, the dynamic and static variants have comparable estimated FLOPs: the mean absolute difference is 0.24 percentage points and the maximum absolute difference is 0.41 percentage points of the full-token baseline under this accounting scope. Small differences in realized mean retention from token rounding and evaluation variability can also affect the cost, so this comparison does not assume exact FLOPs matching.