REDUNDANCY OR RELEVANCE: WHAT LANGUAGE-GUIDED TOKEN PRUNING SELECTS IN VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) model token pruning methods rank tokens typically by computing an importance score based on attention weights, spatial redundancy via inter-token similarity, or task relevance via language conditioning. In this work, we analyze pre-decoder language-vision representations of VLA models. We start this paper by proposing a simple, yet effective, language-vision cosine-similarity based pruning method. Surprisingly, the simple pruning method outperformed more complex state-of-the-art pruning methods. So to better understand how pruning works, we study in more details how cross-modal similarities drive pruning. We find that in many cases pruning is effective in VLAs, not because of semantic grounding, but rather due to the criterion's projection geometry acting as a redundancy suppressor along the visual typicality axis. Since each backbone produces a uniquely structured embedding space, we characterize its geometry to determine what the score actually ranks. We show that when the geometry allows, this signal could be sufficient to determine which tokens to drop, despite the lack of strong semantic alignment. We test over four different VLA models with three different robotic datasets, showing how our language pruning method and the simpler method achieve high accuracy with the lowest latency. For example, on OpenVLA-OFT, we achieve up to $1.68\times$ latency reduction compared to vanilla OpenVLA-OFT, with only 1% drop in success rate.

1 INTRODUCTION

Vision-Language-Action (VLA) models have emerged as a foundational approach for general-purpose robotic control O'Neill et al. (2024); Kim et al. (2024); Intelligence et al. (2025), mapping visual observations and language instructions into physical actions. These models are typically built upon pretrained Vision-Language Models (VLMs) by adding action-generation capabilities and fine-tuning on robotic demonstration data. However, as each camera observation contributes hundreds of patch tokens, the computational cost of processing these dense visual inputs remains a primary bottleneck when deploying these models in high-frequency, closed-loop settings.

To reduce the computational overhead of visual inputs, several token pruning strategies have been proposed Chen et al. (2024); Xu et al. (2026); Liu et al. (2025). These strategies exploit the fact that robotic visual streams exhibit significant spatial and temporal redundancy. Since manipulation tasks are often spatially localized, most image patches capture static or less informative backgrounds. Furthermore, the high sampling frequencies required for continuous control ensure that consecutive frames remain largely unchanged. By discarding a substantial fraction of visual tokens with minimal impact on task performance, these techniques aim to reduce inference latency.

Some pruning strategies start by assigning an importance score to rank the visual tokens, typically defining task relevance through a token's alignment with the provided language instruction. A natural proxy for this relevance is the attention score that vision tokens receive from the language tokens Chen et al. (2024); Zhang et al. (2024). However, one drawback of attention-based methods is that they are not compatible with fused attention operators such as FlashAttention Dao et al. (2022). Alternative methods rank patches either by vision-to-vision intra-modal similarity to minimize redundancy, or by their similarity to the text instructions, where both modalities first share

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.

a joint embedding space. Similarity is then reduced over the instruction sequence via mean, maximum, or softmax pooling Li et al. (2026b); Cai et al. (2025). The specific text tokens selected for this reduction directly impact the ranking: because VLA prompts rely on fixed templates (e.g., `What action should the robot take to [instruction]?`), a large chunk of the sequence consists of boilerplate, i.e., "What action should the robot take to", rather than task-specific content, i.e., *[instruction]?*, requiring these approaches to filter instruction tokens prior to pooling Li et al. (2026b;a); Ye et al. (2026); Cheng et al. (2026). This approach rests on an assumption that a cross-modal alignment is already present at this early stage. VLA training objectives do not explicitly enforce this alignment, and contrastively trained encoders are known to hold the two modalities in separate regions of the latent space Liang et al. (2022). However, we observe that mean-pooled similarity-based criterion appears to select relevant patches correctly. Seeing which patches the criterion keeps throughout a rollout, the retained patches stay on the gripper as arm moves and cover the target region when the arm arrives there (Fig 1).

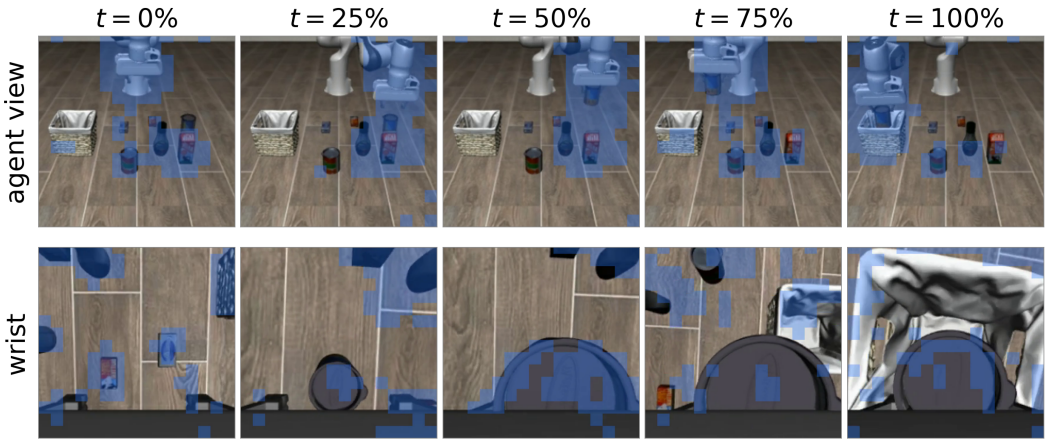

Figure 1: One OpenVLA-OFT episode at 75% compression. Tinted patches are the 64 of 256 per camera that are kept. The instruction is "pick up the alphabet soup and place it in the basket". The retained set moves with the scene.

In this work, we start by proposing a simple, yet effective, language cosine-similarity based VLA pruner that is based on scoring each patch using cosine similarity, ranking them, then pruning based on a maximum predefined token budget threshold. Despite being much simpler compared to many state-of-the-art pruning methods, as we show in the evaluation, this very simple pruner performs better compared to four state-of-the-art more complex pruning mechanisms.

This led us to analyzing what pre-decoder similarity score actually ranks. The score separates into two parts: one that can be computed from the image alone, and one that depends on the instruction. Across the policies we study, the image-only part accounts for nearly the whole ranking. What the score measures is "how ordinary a patch looks beside the rest of the frame?" so the criterion keeps whatever stands out from the current scene and discards what blends into it. The outcome is what vision-only pruners aim to achieve by discarding patches the rest of the frame already accounts for Bolya et al. (2022); Jeddi et al. (2025); Alvar et al. (2025); Wen et al. (2025). The instruction is not idle, but what it mainly decides is which end of that ordering to keep.

We analyse the geometry of this criterion across four VLAs, namely; $\pi_{0.5}$ Intelligence et al. (2025) with Paligemma backbone Beyer et al. (2024), OpenVLA-OFT Kim et al. (2025) and CogAct Li et al. (2024a), sharing the Prismatic backbone Karamcheti et al. (2024), and LLava-VLA Song et al. (2026) built on LLaVA-1.5 Liu et al. (2024). We run our analysis with three datasets; LIBERO Liu et al. (2023), SIMPLER Li et al. (2024b), and CALVIN Mees et al. (2022) datasets.

2 Related Work

Vision-Language-Action Models: Vision-Language-Action (VLA) models typically inherit their multimodal backbones from web-scale pretrained Vision-Language Models (VLMs). Architectural

innovations have mainly focused on action generation, and can be split into two main paradigms: (1) the discrete-token approach which treats control as autoregressive text generation (Brohan et al. (2022; 2023); O'Neill et al. (2024); Kim et al. (2024); Song et al. (2026)); (2) the continuous-decoding approach which predicts chunks of future actions in a single pass via parallel decoding (Kim et al. (2025)), diffusion (Li et al. (2024a)), or flow-matching (Intelligence et al. (2025)). For all architectures, the dense visual prefill imposes a heavy computational overhead making it the primary target for token reduction strategies.

Token Pruning in VLMs: Token reduction strategies can be broadly divided into attention-based and similarity-based methods. Attention-based methods use the language models' intermediate attention scores as an importance metric. FastV (Chen et al. (2024)) discards visual tokens receiving the lowest attention in early decoder layers, while SparseVLM (Zhang et al. (2024)) uses relevant text tokens to guide visual pruning via decoder attention maps, sparsifying progressively at successive layers. PyramidDrop Xing et al. (2024) varies the pruning budget with depth, demonstrating that shallow layers require the full visual sequence while redundancy increases deeper in the network. Similarity-based methods instead compare visual tokens against one another. Because no attention matrix is materialized, these approaches remain compatible with fused attention kernels (e.g. FlashAttention Dao et al. (2022)). For instance, ToMe Bolya et al. (2022) merges tokens based on similarity of tokens' *Key* values. DivPrune and DART(Alvar et al. (2025); Wen et al. (2025)) retain visual tokens that maximize feature diversity by minimizing inter-token similarity within the pre-decoder representations.

Token Pruning in VLAs: VLA pruning adapts attention- and similarity-based methods for VLMs. However, closed-loop control in robotics provides two temporal signals unavailable in single-image settings: end-effector velocity and inter-frame redundancy. ADP, SP-VLA, and SpecPrune-VLA (Pei et al. (2026); Li et al. (2025); Wang et al. (2025)) increase the aggressiveness of pruning as the speed of the end-effector increases. VLA-Cache (Xu et al. (2026)) exploits the second signal, reusing the key-value cache for patches that exhibit high pixel-level similarity with the previous frame. EfficientVLA (Yang et al. (2026)) augments this importance scoring with a coverage term to keep the retained set diverse. VLA-Pruner (Liu et al. (2025)) observes that the tokens deemed salient during the initial vision-language encoding and those attended during action decoding are not the same; it therefore combines language-to-vision attention with action-to-vision attention estimated over recent decoding steps, dropping low-scoring tokens at an early layer similarly to FastV.

Some methods compute importance scores prior to the language model by measuring the similarity between projected patch embeddings and instruction embeddings. Notably, none of these approaches rely on this similarity metric in isolation. VLA-IAP Cheng et al. (2026) combines language similarity with a motion prior that measures trajectory deviations over recent frames, and with geometric edge features extracted from wrist-mounted cameras and reprojected into the main view. VLA-IAP observes that similarity to the instruction does not reliably localize the manipulation target and favor visually distinctive regions, so the method combines them with motion and geometric cues. TEAM-VLA Ye et al. (2026) computes the similarity between every instruction token and patch embedding to retain the single best-matching patch per word as an anchor, and spatially expands these anchors across their neighborhoods to preserve complete object structures rather than isolated patches, deferring the final token merging to an intermediate decoder layer. TEAM-VLA observes that per-word similarity matches are too sparse to cover whole objects, attributing this to poor alignment between text and image embeddings Ye et al. (2026). However, what the similarity itself measures is not studied in any previous work.

3 Understanding pruning in VLAs

3.1 A simple, yet effective, token scoring mechanism

In VLAs, a vision encoder processes each camera image into a grid of patch features; a projector then maps these features into the language model's embedding space, where they are concatenated with the text instruction. Let C be the number of camera views, N the number of visual patches per view, and M the number of instruction tokens. The decoder is prefilled with

$$[\underbrace{v_1 \dots v_{CN}}_{\text{visual}}, \underbrace{l_1 \dots l_M}_{\text{instruction}}], \qquad v_i, l_j \in \mathbb{R}^d. \tag{1}$$

The visual tokens dominate the sequence length ($CN \gg M$), as multiple high-resolution cameras could yield hundreds of patches. Pruning deletes visual tokens from the sequence; the decoder processes a shortened sequence and incurs no computational cost for the discarded tokens. Let $\mathcal{K} \subseteq \{1, \ldots, N\}$ be the retained indices, with $|\mathcal{K}| = K = \lceil \rho N \rceil$ at keep-ratio $\rho \in (0, 1]$, and let $\pi_{\mathcal{K}}$ be the policy run on the shortened sequence $[\, \{v_i\}_{i \in \mathcal{K}}, l_1 \ldots l_M \,]$. The kept tokens stay in their original relative order and keep their original position ids, ensuring that the Rotary Position Embedding (RoPE) distance between any two patches remains unchanged. Writing $J(\cdot)$ for task success, pruning seeks $\arg\max_{|\mathcal{K}|=K} \mathbb{E}[J(\pi_{\mathcal{K}})]$. Evaluating J requires executing a full episode, so the objective cannot be evaluated at the moment $\mathcal{K}$ must be committed. Training-free methods instead rank tokens by a heuristic importance score:

$$\mathcal{K} \;=\; \underset{i \in \{1 \ldots N\}}{\text{top-}K} \;\; \text{Score}(v_i \mid V, L), \qquad V = \{v_i\}_{i=1}^{N},\; L = \{l_j\}_{j=1}^{M}, \tag{2}$$

Our simple ranking method scores each patch by its mean cosine similarity to the instruction tokens, computed where the two modalities first share a space — after the projector, before the first decoder layer — and keeps the highest-scoring patches within each camera block:

$$s_i = \frac{1}{M} \sum_{j=1}^{M} \frac{\langle v_i, l_j \rangle}{\|v_i\| \|l_j\|}. \tag{3}$$

The score lightweight to calculate, and it is also *ostensibly* instruction-conditioned, since l_j are the embeddings of the task string. Our simple pruner uses this score to rank all patches, and then using a fixed token budget, keeps only the tokens within that budget.

Surprisingly, this simple pruning method surpasses state-of-the-art VLA pruning methods we test with. In addition, it is both generalizable and lightweight, i.e., can be easily used with almost any VLA across any architecture (as can be seen in the results section). Hence, we started asking the question of what does language pruning selects in VLA models.

3.2 What the Criterion Ranks

The Language Side is One Direction. Since $\hat{v}_i = v_i / \|v_i\|$ and $\hat{l}_j = l_j / \|l_j\|$, therefore Eq. 3 is an average of dot products. As the dot product is linear in each of its arguments, the average can move inside:

$$s_i = \frac{1}{M} \sum_j \langle \hat{v}_i, \hat{l}_j \rangle = \Big\langle \hat{v}_i,\; \underbrace{\tfrac{1}{M} \textstyle\sum_j \hat{l}_j}_{\bar{q}} \Big\rangle. \tag{4}$$

Splitting $\bar{q}$ into length and direction, $s_i = \|\bar{q}\| \, \langle \hat{v}_i, \hat{q} \rangle$ with $\hat{q} = \bar{q} / \|\bar{q}\|$. The factor $\|\bar{q}\|$ is one number per frame, identical for every patch, and scaling cannot reorder a list. For the purpose of deciding which patches are dropped, the instruction reduces to one unit direction $\hat{q}$.

Visual Hub. We define the visual *hub*, $\hat{\mu}_v$, as the normalized mean direction of a frame's patch embeddings,

$$\mu_v = \frac{1}{N} \sum_i \hat{v}_i, \qquad \hat{\mu}_v = \frac{\mu_v}{\|\mu_v\|}, \tag{5}$$

and each patch's alignment with it as:

$$x_i \;=\; \langle \hat{v}_i, \hat{\mu}_v \rangle. \tag{6}$$

The hub represents the mean visual direction of the current frame and is computed solely from its patches. Thus, x_i measures **how typical patch i is of this frame**: high for patches resembling the bulk of the scene, low for those that stand out. The average alignment is positive by construction, since μ_v is the mean of the very unit vectors being projected onto it.

On OpenVLA-OFT, CogACT, and $\pi_{0.5}$, the score s_i is strongly *anti*-correlated with *typicality* x_i (Table 1). Patches that look like the rest of the frame receive the lowest scores and are dropped first. The average $x_{drop} > x_{kept}$ (Table 1). The language-free counterpart is therefore $-x_i$: ranking by $-x_i$ retains on average 56 of the same 64 patches on OpenVLA-OFT. This can be viewed as frame-adaptive redundancy suppression, i.e., what counts as removable is decided by the current frame's

own statistics so when the arm moves or the scene changes, the definition of "typical" moves with it, which is why the selection tracks the arm. Figure 1 illustrates this on OpenVLA-OFT. The basket is named in the instruction, yet at $t = 0$ the budget goes to the arm and the small objects, the most atypical content of the frame. Patches over the basket are kept only at $t = 75$–100%, when the arm carrying the *can* moves over it. The arm now overlaps those patches, and because the vision encoder contextualises every patch through self-attention over the whole frame, a basket patch's embedding also shifts with the arm's position. Those patches become atypical relative to the frame's mean.

However, $-x_i$ is not similar to a temporal background subtraction, as there is no memory, and is also not similar to an image-space edge detector—a Sobel gradient-energy ranking retains only 0.32 of the same patches, because texture and typicality differ, i.e., wood grain of a table is a strong edge while remaining the most typical content of a frame. In addition, $-x_i$ is not semantic, as the x_i has no notion of which object the task concerns, and coincides with foreground selection only because the background is the majority of a tabletop scene.

Table 1: **The criterion ranks typicality. corr**(s, x): correlation between the score and typicality. α: the query's component along the hub; $\alpha < 0$ is what makes corr(s, x) negative. $x_{dropped/kept}$: mean typicality of each set, so suppression shows as dropped $>$ kept. **IoU** columns compare the criterion's retained set against a language-free $-x_i$ ranking and against Sobel gradient energy.

policy	α	corr(s, x)	$x_{dropped}$	x_{kept}	IoU s vs $-x$ chance 0.14	IoU s vs Sobel
OpenVLA-OFT	-0.29	-0.93	$+0.72$	$+0.42$	0.77	0.32
CogACT	-0.27	-0.91	$+0.66$	$+0.33$	0.76	0.23
$\pi_{0.5}$ + ABTT	-0.87	-0.72	$+0.34$	-0.15	0.63	0.37
$\pi_{0.5}$ *raw*	$+0.53$	$+0.16$	$+0.97$	$+0.98$	0.13	0.22

The Language Contributes the Sign. We show that for many VLA architectures, language mainly supplies the *direction* of the ranking. Decomposing the query into its vision-hub-aligned and vision-hub-orthogonal parts,

$$\hat{q} \;=\; q_{\parallel} \;+\; q_{\perp}, \qquad q_{\parallel} = \langle \hat{q}, \hat{\mu}_v \rangle \, \hat{\mu}_v, \qquad q_{\perp} = \hat{q} - q_{\parallel}. \tag{7}$$

$q_{\parallel}$ is the projection of the query direction $\hat{q}$ onto the hub axis and $q_{\perp}$ is everything left over (and $\langle q_{\perp}, \hat{\mu}_v \rangle = 0$).

Naming the two lengths of $q_{\parallel}$ and $q_{\perp}$: along the hub,

$$\alpha \;=\; \langle \hat{q}, \hat{\mu}_v \rangle, \qquad \text{so} \qquad q_{\parallel} = \alpha \, \hat{\mu}_v, \tag{8}$$

This is a signed quantity, negative when the query points away from the hub. Across it, $\|q_{\perp}\| = \beta$. Together, $\hat{q} = \alpha \, \hat{\mu}_v + \beta \, \hat{q}_{\perp}$. Substituting this into s_i:

$$\begin{aligned} s_i &= \|\bar{q}\| \cdot \langle \hat{v}_i, \hat{q} \rangle \\ &= \|\bar{q}\| \cdot \langle \hat{v}_i, \alpha \cdot \hat{\mu}_v + \beta \cdot \hat{q}_{\perp} \rangle \\ &= \|\bar{q}\| \cdot (\alpha \langle \hat{v}_i, \hat{\mu}_v \rangle + \beta \langle \hat{v}_i, \hat{q}_{\perp} \rangle) \\ &= \|\bar{q}\| \cdot (\alpha \cdot x_i + \beta \cdot y_i) && y_i = \langle \hat{v}_i, \hat{q}_{\perp} \rangle \\ s_i &\propto \alpha \cdot x_i + \beta \cdot y_i && \text{drop } \|\bar{q}\| > 0 \end{aligned} \tag{9}$$

The sign of α decides which end of the typicality axis is discarded. With $\alpha < 0$ the product αx_i falls as typicality rises, so the most typical patches score lowest and go first. With $\alpha > 0$ the same rule keeps them.

What the query-orthogonal term contributes. Because the split in 7 is orthogonal and $\hat{q}$ is a unit vector, $\alpha^2 + \beta^2 = 1$, so on OpenVLA-OFT $\beta = 0.96$ is more than three times $|\alpha| = 0.29$ and βy_i looks like the larger contributor. If β is zero, ranking by s_i with $\alpha < 0$ would be ranking by $-x_i$; every patch on which the two rankings disagree is therefore the work of βy_i. On OpenVLA-OFT that is, on average, 8 of the 64 retained patches.

However, these few patches that βy_i selects could be important for task success. Therefore, to test it, we decouple the text used for *scoring* from the text the policy *reads*. The policy receives

Table 2: Instruction substitution effect on success rate. 500 episodes per LIBERO cell, 240/216 on CogACT. **n/a:** LIBERO-Long's ten tasks do not share a scene, so there's no other task whose instruction names a different target in the same scene. The condition cannot be constructed there, unlike on Object where all ten tasks share one scene. We

	OpenVLA-OFT, 12.5%		$\pi_{0.5}$, 12.5%		CogACT, 25%	
Text pooled into $\bar{q}$	Object	Long	Object	Long	MoveNear	Drawer
True instruction	96.2	88.0	92.4	72.6	79.2	69.9
same scene, named different target	−0.2	n/a	+0.2	n/a	+2.1	+2.3
named objects not in scene	−0.4	−2.0	−1.6	+1.8	+0.4	+0.9
one sentence for every task	−2.0	−1.0	−0.4	−3.2	+2.9	−1.9
random tokens, length-matched	+0.6	−0.8	+1.0	+5.4	+1.7	+0.9
template alone, instruction removed	−2.6	−1.6	0.6	+6.9	+2.9	−0.5
no text: hub rule $-x$	92.2	70.0	95.2	63.2	80.4	64.8

the true instruction, and $\bar{q}$ is rebuilt from a substitute string with the policy's own tokenizer and embedding table. Any change in success is therefore caused by which patches were kept. Table 2 shows how success changes when the text used for scoring is replaced. On OpenVLA-OFT and CogACT, replacing the instruction with another task's, with random tokens or with nothing changes success by at most 2.9 points (within the noise floor on CogACT). On the long-horizon LIBERO-Long suite, the criterion outperforms the language-free $-x_i$ ranking by 18 points. However, this gap is not explained by the instruction, as scoring with the template only result in 1.6% success rate drop only. The template rows, which dominate $\bar{q}$, therefore carry both parts of the score. Their mostly negative α_j set the negative weight on the typicality x_i, and their residual direction contributes the βy_i term that $-x_i$ discards and that accounts for the gap.

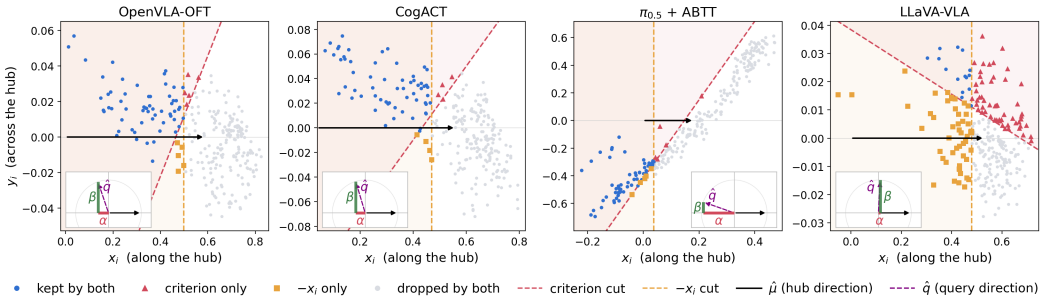

Figure 2: One representative frame per policy in the plane of Eq. (9): x_i (alignment with the hub, horizontal) against y_i (alignment with the query's hub-orthogonal direction, vertical). Colors mark patches kept by both the criterion and the hub rule (blue), by the criterion only (red), by the hub rule only (orange), or by neither (grey) at 75%. Dashed lines are the two rules' decision boundaries; insets show $\hat{q}$ decomposed into α and β. $\pi_{0.5}$ is shown after common-direction removal.

A language-free control. Motivated by Eq. (9), we use the rule $r_i = -x_i$, which keeps the patches least aligned with the frame's own mean direction. We define this language-free formulation as the ***hub rule***, utilizing it as a diagnostic control to evaluate how much of the language criterion's behaviour the image alone can explain. Since the squared distance $\|\hat{v}_i - \mu_v\|^2 = 1 - 2\|\mu_v\|x_i + \|\mu_v\|^2$, and $\|\mu_v\| > 0$ is the same for all patches of a frame, ranking by $-x_i$ is ranking by distance from the frame centroid. We then study if this is a generalizable rule on two models,

3.3 A CASE OF PATHOLOGICAL ANISOTROPY

On $\pi_{0.5}$ dropped and kept patches have almost identical typicality (0.97 against 0.98, (Table 1). The pruning mechanism fails to suppress redundancy. This collapse stems from extreme anisotropy. On this backbone, anisotropy — defined as the mean cosine similarity between any two distinct patches

Table 3: Success rate (%) on LIBERO for OpenVLA-OFT with 25% tokens kept. **Language:** (equation 3) .**Hub** $-x_i$ (sec. 3.2).

	LIBERO suite					Cost	
Method	***Spatial***	***Object***	***Goal***	***Long***	**Avg**	**TFLOPs**	**Latency (ms)**
OpenVLA-OFT	98.8	97.4	97.8	94.8	97.2	4.05	163.56
DivPrune	98.2	91.2	94.8	92.2	94.1	1.67	97.6
VLA-Cache	97.0	97.6	95.4	89.4	94.9	1.94	119.7
VLA-Pruner	98.2	**98.4**	96.4	93.0	96.5	1.69	107.5
FastV	98.6	78.6	94.8	91.8	91.0	1.68	103.7
Language	98.2	97.4	**97.0**	**92.4**	**96.2**	1.67	97.1
Hub $-x_i$	**99.4**	96.6	96.4	91.2	95.9	1.67	**96.9**

within a frame — reaches $A = 0.95$. every patch resembles every other patch, x_i sits near 1, and almost no variation is left to rank on.

Restoring the spread. To restore some variance, we construct a joint embedding matrix $[V; L]$ of $N + M$ rows by stacking a camera's patch embeddings V with the instruction embeddings L. We extract the top right-singular vector u_1 of this matrix and subtract it out of both modalities:

$$\tilde{v} = v - \langle v, u_1 \rangle u_1, \qquad \tilde{l} = l - \langle l, u_1 \rangle u_1, \tag{10}$$

recomputing $\hat{\mu}_v$, $\bar{q}$, α, and x_i on $\tilde{v}$ and $\tilde{l}$. This is All-but-the-Top (ABTT) (Mu et al. (2017)) applied at the co-embedding point. Measured on $\pi_{0.5}$: A collapses $0.95 \to 0.01$, the spread of x_i is restored, and α flips $+0.53 \to -0.87$, moving the policy from redundancy selection to redundancy suppression (Table 5).

3.4 A CASE OF LITTLE ALIGNMENT

On OpenVLA-OFT, αx_i supplies the dominant hub-aligned ordering, while βy_i adjusts it. On LLaVA-VLA, by contrast, α is near zero and the score is dominated by βy_i. The resulting selection is genuinely instruction-sensitive: swapping the instruction changes about 35% of the retained set, compared with 6% on OpenVLA-OFT. This is the backbone on which the criterion behaves most like an instruction-guided selector, yet it is also where that selector fails operationally: at the same budget, the criterion underperforms a baseline that ignores the text instruction entirely (Sec. 4).

Replacing LLaVA-VLA's $\alpha = +0.04$ with OpenVLA-OFT's $\alpha = -0.29$, while leaving the other score components fixed, raises the overlap between the retained set and the hub rule's retained set from 0.14 to 0.73. By comparison, flipping only the sign to $\alpha = -0.04$ raises the overlap only to 0.42. These interventions show that x_i has sufficient spread to carry the ranking once x_i is assigned enough negative weight. Fig. 2 shows this plane for each policy.

4 EXPERIMENTS

We evaluate four pruning methods, namely, VLACache Xu et al. (2026), VLA-Pruner Liu et al. (2025), FastVChen et al. (2024), and DivPrune Alvar et al. (2025). Our evaluation considers four state-of-the-art VLAs; OpenVLA-OFT, CogACT, $\pi_{0.5}$ and LLaVA-VLA. We note that we are unaware of any previously published work on pruning LLAVA-VLA. In our evaluation, we run on three benchmarks; LIBERO contributes four suites, Spatial, Object, Goal and Long, 500 episodes per suite; SIMPLER visual matching contributes four tasks, 'pick coke can', 'move near', 'open drawer', and 'place in drawer', at 300, 240, 216 and 108 episodes, respectively; CALVIN tasks are `task_ABC_D` split at 500 sequences of five chained subtasks. For comparisons with the four VLA pruning methods, we run them with OpenVLA-OFT only because their implementations are open source. However, for the other VLA models, we report, when relevant, the results directly from the papers as the implementations are not available. We calculate the score from projector embeddings before the decoder runs, and drop all selected tokens in one go at decoder layer 2, similar to FastV Chen et al. (2024). We have run our experiments on two different GPU models, H100 and A10G. However, we report on the latency results measured on NVIDIA A10G GPU.

Table 4: CogACT success rates on four SIMPLER (visual matching) tasks with 25% tokens kept. **Language:** (equation 3) .**Hub** $-x_i$ (3.2). **Avg:** Weighted Average by the number of episodes per task. VLA-Cache results are taken from their paper for the same tasks with keep ratio of $\approx 70\%$

Method	***PickCan***	***MoveNear***	***Drawer***	***PlaceIn***	**Avg**	**TFLOPs**	**Latency (ms)**
CogACT	92.3	86.7	72.7	43.5	79.7	1.84	123.6
VLA-Cache	92.0	**83.3**	70.5	51.6	–	1.49	–
Language	**93.3**	79.2	**71.8**	**53.7**	**79.1**	0.68	49.6
Hub $-x_i$	90.0	80.4	64.8	43.5	75.2	0.68	49.5

Table 5: $\pi_{0.5}$ success rates on LIBERO with 25% keep ratio. The ranking is done after applying ABTT, the decoder receive the original embeddings.

Method	***Spatial***	***Object***	***Goal***	***Long***	**Avg**	**TFLOPS**	**Latency (ms)**
$\pi_{0.5}$	98.6	98.0	97.6	93.0	96.8	2.02	97.0
Language	**95.6**	**97.8**	90.2	**85.2**	92.2	0.82	39.2
Hub $-x_i$	89.8	95.8	88.4	74.2	87.0	0.82	39.1

OpenVLA-OFT and CogACT: On OpenVLA-OFT (Table 3), the hub rule achieves a 95.9% success rate compared to the language criterion's 96.2%, demonstrating that the contribution of language to overall accuracy is on average 0.3%. The two rules keep substantially the same patches and reach substantially the same success rate. CogACT shares OpenVLA-OFT's Prismatic backbone and sits in the same geometric regime ($\alpha = -0.27$). CogACT on tasks *Drawer* and *PlaceIn* (Table 4) has a wider gap between the criterion and hub-only pruning. Since the hub rule is the criterion's αx_i term alone, any real gap must come from the orthogonal term βy_i, i.e., from the language contribution. Table. 2 shows it is not from the instruction's content but rather from the template tokens. For reference, we also add the numbers reported for VLA-Cache in Xu et al. (2026) when they drop about 30% (according to the FLOPS mentioned in their paper).

$\pi_{0.5}$. Our evaluation on $\pi_{0.5}$ (Table 5) shows that Eq. 10 restores the spread, resulting in both our simple methods to work. None of the considered VLA pruning methods has an evaluation on $\pi_{0.5}$. Our results show the generalizability of our approaches.

LLaVA-VLA. The criterion reduces the cosine matrix $c_{ij} = \langle \hat{v}_i, \hat{l}_j \rangle$ over prompt rows j by taking the mean (Eq. equation 3). On LLaVA-VLA we also test max and softmax pooling:

$$s_i^{\max} = \max_j c_{ij}, \qquad \text{and} \qquad s_i^{\tau} = \sum_j w_{ij}\, c_{ij}, \quad w_{ij} = \frac{\exp(c_{ij}/\tau)}{\sum_k \exp(c_{ik}/\tau)}. \tag{11}$$

Max pooling scores each patch by its best-matching row. Softmax pooling interpolates between the mean ($\tau \to \infty$) and the max ($\tau \to 0$); we use $\tau = 0.05$, following FlashVLM (Cai et al., 2025).

On LLaVA-VLA evaluated on CALVIN, the pooled query has almost no component along the visual hub. We measure $\alpha = +0.04$, which gives $\beta = \sqrt{1-\alpha^2} = 0.999$ and places the query approximately $88°$ from the hub. For comparison, two random unit vectors in 4096 dimensions are concentrated around $90° \pm 0.9°$. LLaVA-VLA is therefore fewer than three standard deviations from this isotropic reference. Most of LLaVA-VLA's query norm consequently lies in the hub-orthogonal component, and its selection is considerably more sensitive to the instruction than OpenVLA-OFT's (Fig. 2). Unlike raw $\pi_{0.5}$, LLaVA-VLA does not exhibit a collapsed patch cloud. We measure $A = 0.29$ and $\|\mu_v\| = 0.54$, while x_i spans $[0.03, 0.75]$. This is comparable to OpenVLA-OFT's range of $[0.17, 0.80]$ and substantially broader than the $[0.93, 0.99]$ range observed for $\pi_{0.5}$. For LLaVA-VLA representations evaluated on CALVIN, 19.5% of patches have a near-duplicate elsewhere in the same frame, defined by cosine similarity above 0.999. Thus, the failure to suppress typical patches cannot be explained by an absence of redundancy in these representations.

Table 6 reports the mean chain lengths on the CALVIN benchmark. At a 75% retention budget, the language criterion degrades the unpruned policy's mean chain length from 3.61 to 1.85, whereas the

language-free hub rule maintains a length of 2.99. This failure occurs because the language query $\bar{q}$ is tilted toward the visual mean, resulting in a positive α. Consequently, the language criterion fails to implicitly suppress redundancy, while the genuine cross-modal semantic alignment remains too weak to guide valid token selection. The hub rule is beneficial here because it explicitly enforces a negative sign on typicality ($r_i = -x_i$), directly pruning redundant tokens even in the presence of weak semantic alignment.

Table 6: LLaVA-VLA, CALVIN, 500 sequences. Columns are the percentage of chains completing at least 1 to 5 subtasks. **Avg. len.:** mean chain length out of 5.

	subtasks completed						Cost	
Method	**1**	**2**	**3**	**4**	**5**	**Avg. len.**	**TFLOPs**	**Latency (ms)**
LLaVA-VLA	94.6	85.2	72.2	59.8	49.2	3.61	4.48	179.90
Language,	72.5	47.7	32.5	19.4	12.2	1.85	1.86	86.1
Softmax τ=0.05,	79.6	56.3	39.7	28.5	19.0	2.24	1.86	86.1
Max pooling,	87.4	66.9	49.7	36.3	28.9	2.70	1.86	86.1
Hub $-x_i$	**89.0**	**73.5**	**58.7**	**43.5**	**33.5**	**2.99**	1.86	85.9

Max pooling scores each patch by the single row that matches it best, so a row that matches specific image content is not averaged away. Closed-loop results follow this order, with mean task length rising from 1.85 (mean) to 2.24 (softmax) and 2.70 (max; Table 6). Pooling reduces the harm but does not close the gap, as max pooling still trails the language-free hub rule (2.99). On OpenVLA-OFT, CogACT and $\pi_{0.5}$ with ABTT, the useful ranking comes from the pooled prompt's shared negative alignment with the visual mean. We expect that max and softmax pooling would largely discard it by concentrating each patch's score on its best-matching rows, leaving the ranking to word-level residual directions that, as the substitution results show, carry little task-relevant signal.

5 Conclusion

We showed that simple cosine-similarity based ranking for VLAs can be used to prune vision tokens effectively, surpassing state-of-the-art complex pruning methods. We then analysed the pre-decoder language–vision similarity in VLA models. For many VLAs, decomposing the criterion shows that the score it produces is largely carried by the patch cloud's own mean, and that how much of it the instruction contributes depends on the geometry the backbone produces. That component measures how typical each patch is of the frame the camera currently sees, so what the criterion performs is frame-adaptive redundancy suppression rather than selection of the content the instruction names. Where that geometry leaves the instruction with little to contribute beyond a sign, the criterion can be replaced by a rule that fixes the sign in advance. For other VLAs where this does not hold, we showed that simple fixes to the criterion can still yield competitive pruning. Ranking patches by their distance from the frame's own mean direction also helps when the vision–language alignment is too weak for the criterion to rank at all.

AI use statement

We used large language model assistants for writing and editing parts of this manuscript (drafting and revising text). We used AI tools for initial literature search and for help in coding analysis and plotting scripts. We reviewed all AI-assisted text and code. We take full responsibility for the content of the paper.

Reproducibility statement

We will release the code required for reproducing our results.

REFERENCES

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9392–9401. IEEE, 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.

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

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.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Kaitong Cai, Jusheng Zhang, Jing Yang, Yijia Fan, Pengtao Xie, Jian Wang, and Keze Wang. Flashvlm: Text-guided visual token selection for large multimodal models. *arXiv preprint arXiv:2512.20561*, 2025.

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

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.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

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

Ahmadreza Jeddi, Negin Baghbanzadeh, Elham Dolatabadi, and Babak Taati. Similarity-aware token pruning: Your vlm but faster. *arXiv preprint arXiv:2503.11549*, 2025.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*, 2024.

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

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

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

Wei Li, Renshan Zhang, Rui Shao, Zhijian Fang, Kaiwen Zhou, Zhuotao Tian, and Liqiang Nie. Semanticvla: Semantic-aligned sparsification and enhancement for efficient robotic manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18397–18405, 2026a.

Xu Li, Yi Zheng, Yuxuan Liang, Zhe Liu, Xiaolei Chen, Haotian Chen, Rui Zhu, and Xiangyang Xue. Resprune: Text-conditioned subspace reconstruction for visual token pruning in large vision-language models. *arXiv preprint arXiv:2603.21105*, 2026b.

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

Ye Li, Yuan Meng, Zewen Sun, Kangye Ji, Chen Tang, Jiajun Fan, Xinzhu Ma, Shu-Tao Xia, Zhi Wang, and Wenwu Zhu. Sp-vla: A joint model scheduling and token pruning approach for vla model acceleration. In *The Fourteenth International Conference on Learning Representations*, 2025.

Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *Advances in neural information processing systems*, 35:17612–17625, 2022.

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

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26286–26296. IEEE, 2024.

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Bridging the semantic-action gap in visual token pruning for efficient vla inference. *arXiv preprint arXiv:2511.16449*, 2025.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*, 2017.

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

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. In *International Conference on Learning Representations*, volume 2026, pp. 10832–10851, 2026.

Wenxuan Song, Jiayi Chen, Xiaoquan Sun, Huashuo Lei, Yikai Qin, Wei Zhao, Pengxiang Ding, Han Zhao, Tongxin Wang, Pengxu Hou, et al. Rethinking the practicality of vision-language-action model: A comprehensive benchmark and an improved baseline. *arXiv preprint arXiv:2602.22663*, 2026.

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

Zichen Wen, Yifeng Gao, Shaobo Wang, Junyuan Zhang, Qintong Zhang, Weijia Li, Conghui He, and Linfeng Zhang. Stop looking for "important tokens" in multimodal language models: Duplication matters more. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9972–9991, 2025.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. *Advances in Neural Information Processing Systems*, 38:164448–164473, 2026.

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. *Advances in Neural Information Processing Systems*, 38:40891–40914, 2026.

Yifan Ye, Jiaqi Ma, Jun Cen, and Zhihe Lu. Token expand-merge: Training-free token compression for vision-language-action models. *IEEE Robotics and Automation Letters*, 2026.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024.

A Appendix

A.1 Complexity Analysis

At one control step the decoder prefill processes n_0 tokens. Pruning removes Δn of them, leaving $n_1 = n_0 - \Delta n$. At the decoder layer, For hidden width D, key/value width D_{kv} and gated-MLP width D_{ff}, a layer processing n tokens costs

$$m(n) = n\,(2D^2 + 2DD_{\text{kv}} + 3DD_{\text{ff}}) \tag{12}$$

For prefilling, An L-layer decoder processing n tokens costs

$$T(n) = L\,m(n) + T_{\text{head}}(n), \tag{13}$$

where T_{head} is the output head, pruning saves the fraction

$$\frac{T(n_0) - T(n_1)}{T(n_0)} \approx \frac{n_0 - n_1}{n_0} \tag{14}$$

of the prefill compute: the saving is essentially the fraction of tokens removed.