AC-SPARSE: ACTION-CENTRIC DYNAMIC ACTIVATION SPARSITY FOR EFFICIENT VISION-LANGUAGE-ACTION MODEL INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models enable robotic manipulation but require substantial computation during inference. Existing methods for efficient VLA inference often rely on coarse-grained layer skipping or token optimization, while computation reuse depends on temporal similarity between observations. Activation sparsification offers a fine-grained alternative by reducing linear computation while preserving the network structure and token sequence. However, existing sparsification criteria often overlook how perturbations at different tokens affect action generation, potentially degrading task performance at high sparsity levels. We propose **AC-Sparse**, a training-free dynamic activation sparsification method for VLA inference. Specifically, AC-Sparse combines attention weights from action representations with activation magnitudes and weight-column norms to estimate the cost of removing each token–channel activation. We formulate an attention-weighted expected squared local output perturbation objective and derive a separable upper bound that AC-Sparse minimizes under a fixed sparsity budget for each linear projection. We further prove that the derived sparsification criterion exactly minimizes the original objective when the weight columns are mutually orthogonal. Experiments across different VLA architectures and manipulation benchmarks demonstrate that AC-Sparse better preserves task success than the evaluated sparsification baselines at high sparsity levels, achieving an average success rate only 0.7 percentage points below dense inference on LIBERO at 65% activation sparsity.

1 INTRODUCTION

Vision-Language-Action (VLA) models have emerged as a powerful paradigm for general-purpose robotic manipulation by integrating visual perception, language understanding, and action generation within a unified policy (Kim et al., 2024; Intelligence et al., 2025; Black et al., 2026). Despite their strong capabilities, modern VLA models typically rely on computationally intensive vision encoders, large language backbones, and action-prediction modules, leading to substantial inference computation. This growing computational burden has motivated increasing efforts toward efficient VLA inference (Yue et al., 2024; Yang et al., 2025b; Zhang et al., 2025a; Xu et al., 2025; Pei et al., 2026; Wang et al., 2026a). In practical deployment, an appealing solution should incur minimal additional training cost, be readily applicable to pretrained VLA models, and support flexible adjustment of computation according to different resource budgets. Developing such an efficient and adaptable inference mechanism while preserving manipulation performance therefore remains an important issue.

Existing approaches to efficient VLA inference mainly exploit redundancy through adaptive network execution, visual-token reduction, and computation reuse. Layer-skipping methods dynamically shorten the executed network depth (Yue et al., 2024; Zhang et al., 2025a; Yang et al., 2025b), while token-pruning and compression methods reduce the number of visual tokens processed by subsequent layers (Pei et al., 2026; Wang et al., 2026a; Liu et al., 2026; Ye et al., 2025). Computation-reuse methods instead cache or reuse intermediate representations across observations or action-generation steps (Xu et al., 2025; Tan et al., 2025). Although effective, these approaches either operate at relatively coarse granularity or rely on temporal redundancy. Training-free activation

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.

sparsification (Liu et al., 2025; Chen et al., 2026; Huang et al., 2026) provides a more fine-grained alternative by directly removing less important activation channels without modifying pretrained model parameters. However, existing criteria mainly rely on activation magnitude or local output perturbation, without considering whether perturbations at different tokens have different effects on action generation, which becomes particularly limiting under aggressive sparsity (e.g., $\geq 65\%$).

The key limitation is that VLA action generation does not depend uniformly on all tokens. Even when sparsifying two tokens induces comparable local output perturbations, their effects on subsequent action representations can differ substantially. It depends on how strongly those tokens contribute to action generation. Attention from task-related representations has been used as a signal of token importance in vision Transformers (Liang et al., 2022; Long et al., 2023), suggesting a natural way to characterize this difference in VLA models. In particular, tokens receiving greater attention from action representations are expected to be more sensitive to local perturbations. This becomes increasingly important under aggressive sparsity, where a large number of activations must be removed and the sparsity budget needs to be allocated carefully across tokens. Therefore, effective high-sparsity inference requires considering both the local cost of removing an activation, and the action relevance of the token to which it belongs.

Motivated by these observations, we propose AC-Sparse, a training-free and off-the-shelf activation sparsification method for VLA inference. AC-Sparse can be directly applied to pretrained VLA models without additional training or parameter modification, while allowing the sparsity level to be flexibly adjusted according to the available computation budget. The key idea is to make sparsity allocation action-centric, i.e., AC-Sparse uses attention from action representations to estimate the relative importance of individual tokens, and combines it with activation magnitudes and weight-column norms to measure the cost of removing each token–channel activation. Under a shared sparsity budget for each linear projection, the method dynamically allocates sparsity across tokens and channels, preserving more computation for tokens receiving higher attention from action representations while aggressively sparsifying lower ones. Our main contributions are summarized as follows.

- We propose AC-Sparse, a training-free and off-the-shelf activation sparsification method for efficient VLA inference, which can be directly applied to pretrained VLA models without additional training or parameter modification, while allowing the sparsity level to be flexibly adjusted according to the desired computation budget.
- AC-Sparse is designed to preserve task performance under aggressive sparsity by introducing action-centric sparsity allocation. It uses attention from action representations to distinguish the importance of different tokens and dynamically allocates a shared sparsity budget across token–channel positions.
- We provide a theoretical formulation for AC-Sparse by introducing an attention-weighted local output perturbation objective and deriving the corresponding sparsification criterion. The resulting criterion minimizes an upper bound for general weight matrices and becomes exactly optimal when the weight columns are orthogonal.
- Extensive experiments across different VLA architectures and manipulation benchmarks demonstrate that AC-Sparse achieves stronger performance preservation than existing activation sparsification methods at high sparsity levels.

2 Related Work

2.1 Efficient Inference for VLA Models

Existing methods for efficient VLA inference reduce computation through adaptive network execution, visual-token optimization, and computation reuse. DeeR-VLA (Yue et al., 2024) dynamically adjusts inference depth through early exiting, while MoLe-VLA (Zhang et al., 2025a) and DySL-VLA (Yang et al., 2025b) adapt layer execution to reduce redundant computation. Visual-token optimization methods reduce the number of tokens processed by subsequent layers through token selection, pruning, or compression. VLA-ADP (Pei et al., 2026) combines text-guided token selection with action-dependent retention ratios, while SpecPrune-VLA (Wang et al., 2026a) exploits historical actions and layer-wise token importance to guide pruning. VLA-Pruner (Liu et al., 2026) incorporates attention during action decoding into visual-token selection, whereas TEAM-VLA (Ye et al., 2025) and Compressor-VLA (Gao et al., 2025) reduce visual-token redundancy through to-

ken merging and instruction-guided semantic and spatial compression, respectively. Computation reuse methods exploit redundancy across observations or action-generation steps. VLA-Cache (Xu et al., 2025) reuses cached representations of visual tokens that exhibit limited changes between consecutive observations, while FlashVLA (Tan et al., 2025) combines action reuse with visual-token selection. Other methods integrate multiple optimization strategies. EfficientVLA (Yang et al., 2025a) combines language-layer pruning, visual-token selection, and intermediate-feature caching, while SP-VLA (Li et al., 2026) combines model scheduling with visual-token pruning. Unlike these approaches, AC-Sparse preserves the network structure and full token sequence while dynamically sparsifying activations within individual tokens.

2.2 Activation Sparsification

Activation sparsification reduces linear computation by omitting operations associated with selected input activations and has been extensively studied in LLMs. Early approaches often introduce sparsity through model adaptation. Deja Vu (Liu et al., 2023b) predicts input-dependent execution paths for attention heads and MLPs, while CATS (Lee et al., 2024), Turbo Sparse (Song et al., 2024), and Q-Sparse (Wang et al., 2024) promote sparse activations through activation-function design or sparsity-aware training. More recent training-free methods exploit redundancy in pretrained models without modifying their parameters. TDA (Ma et al., 2024) and TEAL (Liu et al., 2025) rely on activation statistics, while R-Sparse (Zhang et al., 2025b), WINA (Chen et al., 2026), and WAS (Wang et al., 2025) further incorporate weight information to estimate removal effects or allocate sparsity.

However, these criteria are largely based on local activation and weight statistics. In VLA models, this can be limiting because tokens from different modalities and spatial regions may contribute differently to action generation. EcoVLA (Huang et al., 2026) extends channel pruning to VLA inference using current and historical features, but does not explicitly model how individual tokens relate to action representations. Incorporating such action representation attention into fine-grained activation sparsification remains less explored. AC-Sparse addresses this gap by incorporating action representation attention into the sparsification criterion and allocating sparsity across tokens under a shared budget without retraining.

3 Methodology

Figure 1 provides an overview of AC-Sparse. During inference, AC-Sparse combines action representation attention with local activation and weight information to rank token–channel activations, and removes them under a shared sparsity budget for each linear projection.

3.1 Problem Statement

Ideally, activation sparsification should preserve the predicted action under a fixed sparsity budget. Directly optimizing the final action discrepancy, however, requires tracing each perturbation through the remaining network and is impractical during training-free inference. Since sparsification directly perturbs the output of each linear projection before propagating downstream, we use this local perturbation as a tractable surrogate.

Let $\mathcal{P}_l$ denote the set of linear projections selected for sparsification in Transformer block l. For projection $P \in \mathcal{P}_l$, let $x_{l,t}^P \in \mathbb{R}^{d_{\text{in}}^P}$ denote the input activation of token t, $W_l^P \in \mathbb{R}^{d_{\text{out}}^P \times d_{\text{in}}^P}$ its weight matrix, and $g_{l,t}^P \in \{0,1\}^{d_{\text{in}}^P}$ its channel-wise sparsity mask, where zero indicates a sparsified activation. The local output perturbation of projection P is defined as the difference between its original and sparsified outputs.

$$\Delta y_{l,t}^P = W_l^P \left[x_{l,t}^P - g_{l,t}^P \odot x_{l,t}^P \right]. \tag{1}$$

Let $D_l^P = \{(t,c) \mid g_{l,t,c}^P = 0\}$ denote the set of sparsified token–channel positions.

As discussed in Sec. 1, local perturbations of comparable magnitude at different tokens may have different effects on downstream action representations. We therefore use attention from action representations to weight these perturbations across tokens. Specifically, let $p_{l,P}(t)$ denote a normalized token distribution constructed from these attention weights, as detailed in Sec. 3.2.1. We define

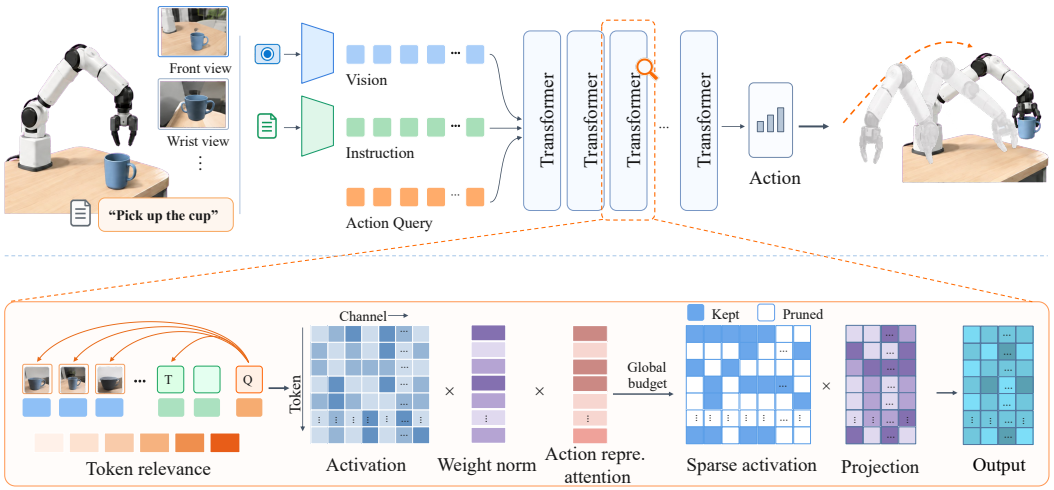


Figure 1: **Overview of AC-Sparse.** The VLA inference process is shown at the top, with the token–channel sparsification process illustrated below. AC-Sparse dynamically selects activations for removal across tokens under a shared sparsity budget for each linear projection.

the attention-weighted local objective as the expected squared local output perturbation under this distribution:

$$\mathcal{J}_l^P(D_l^P) = \mathbb{E}_{t\sim p_{l,P}}\left[\left\|\Delta y_{l,t}^P\right\|_2^2\right] = \sum_{t=1}^{T} p_{l,P}(t)\left\|\Delta y_{l,t}^P\right\|_2^2. \tag{2}$$

For each projection, we minimize $\mathcal{J}_l^P(D_l^P)$ under a fixed sparsity budget $|D_l^P| = K_l^P$, while keeping the token distribution fixed during selection.

3.2 AC-Sparse

3.2.1 Attention-Based Token Distribution

AC-Sparse constructs the token distribution in Eq. 2 from attention associated with action representations, i.e., hidden states that integrate multimodal information for action prediction. For projection P in layer l, let $\mathcal{Q}_{l,P}$ denote the corresponding action-representation positions, and let $A_{l,P}^h[q,t]$ denote the attention from position q to token t in head h. We average over action-representation positions and attention heads.

$$r_{l,P}(t) = \frac{1}{|\mathcal{Q}_{l,P}|H}\sum_{q\in\mathcal{Q}_{l,P}}\sum_{h=1}^{H} A_{l,P}^h[q,t]. \tag{3}$$

We then convert these scores into a normalized token distribution.

$$p_{l,P}(t) = \frac{r_{l,P}(t)^\beta}{\sum_{j=1}^{T} r_{l,P}(j)^\beta}, \qquad \beta > 0, \tag{4}$$

where β controls the concentration of the distribution. We use $\beta = 0.5$ to smooth differences in attention across tokens and analyze its sensitivity in Sec. 4.4. The attention source depends on the projection. Q, K, and V projections use attention from the preceding layer, whereas O and FFN projections use attention from the current layer. Architecture-specific definitions of action representations and attention assignment are provided in Appendix A.1.

3.2.2 Dynamic Activation Sparsification Across Tokens

Given the attention-weighted local objective in Eq. 2, we derive a criterion for selecting token–channel activations to sparsify under a shared projection-level budget. Let $D_{l,t}^P = \{c \mid (t,c) \in D_l^P\}$

denote the sparsified channels of token t. The local output perturbation can be written as

$$\Delta y_{l,t}^P = \sum_{c \in D_{l,t}^P} x_{l,t,c}^P W_{l,:,c}^P, \tag{5}$$

where $W_{l,:,c}^P$ denotes the weight column corresponding to input channel c.

Directly minimizing Eq. 2 requires accounting for interactions among perturbations from different sparsified channels. Under the fixed sparsity budget $|D_l^P| = K_l^P$, applying the Cauchy–Schwarz inequality gives the separable upper bound

$$\mathcal{J}_l^P(D_l^P) \le K_l^P \sum_{(t,c) \in D_l^P} p_{l,P}(t) \left| x_{l,t,c}^P \right|^2 \left\| W_{l,:,c}^P \right\|_2^2 . \tag{6}$$

The complete derivation is provided in Appendix B.1. Substituting Eq. 4 and removing factors shared by all candidates within a projection yields the AC-Sparse sparsification cost

$$C_{l,t,c}^P = r_{l,P}(t)^{\beta/2} \left| x_{l,t,c}^P \right| \left\| W_{l,:,c}^P \right\|_2 . \tag{7}$$

This cost combines action-conditioned token relevance with activation magnitude and weight-column norm. The weight-column norms are input-independent and can be precomputed when the model is loaded.

The theoretical derivation yields the unnormalized selection criterion above. In practice, we use modality-wise normalization as an additional calibration step before sparsity selection to reduce scale differences across modalities, as detailed in Appendix A.2. Let Ω_l^P denote the sparsifiable token–channel positions, excluding protected special tokens such as BOS and STOP. AC-Sparse jointly ranks these positions and selects the K_l^P smallest costs.

$$D_l^{P*} = \mathrm{BottomK}_{(t,c) \in \Omega_l^P} C_{l,t,c}^P. \tag{8}$$

This joint selection dynamically distributes the projection-level sparsity budget across tokens instead of assigning a predefined budget to each token. Positions outside Ω_l^P are retained.

3.3 Theoretical Analysis

We provide theoretical support for the AC-Sparse selection rule under the local objective in Eq. 2. Because perturbations from multiple sparsified channels interact in the projection output, the objective is generally non-separable. Eq. 6 bounds these interactions with a separable form whose terms are proportional to $\left(C_{l,t,c}^P\right)^2$, making Bottom-$K$ selection directly applicable.

Theorem 1 (Local Optimality of AC-Sparse)**.** *For fixed token weights $p_{l,P}(t)$, eligible set Ω_l^P, and sparsity budget K_l^P, the Bottom-K rule in Eq. 8 minimizes the separable upper bound in Eq. 6. If the columns of W_l^P are mutually orthogonal, the cross-channel terms vanish and the same rule exactly minimizes the local objective in Eq. 2.*

Thus, the proposed cost provides a principled selection criterion in the general case and becomes exact when cross-channel interactions vanish. Proofs are provided in Appendix B.1 and Appendix B.2.

We further analyze the computational cost of AC-Sparse. At activation sparsity s, it saves $2sTd_{\mathrm{in}}d_{\mathrm{out}}$ ideal linear-computation FLOPs with only small additional overhead; detailed analysis is provided in Appendix C.

4 Experiments

4.1 Experimental Setup

Models and benchmarks. We evaluate AC-Sparse on two representative VLA architectures: action-query-based VLA-Adapter (Wang et al., 2026b), which uses a Prismatic VLM with a Qwen2.5-0.5B backbone (Yang et al., 2024), and diffusion-based $\pi_{0.5}$ (Intelligence et al., 2025).

AC-Sparse is applied to all Transformer linear projections in the Qwen2.5-0.5B backbone of VLA-Adapter and to both the PaliGemma backbone and action expert in $\pi_{0.5}$. Architecture-specific attention extraction is detailed in Appendix A.1.

We evaluate on the standard LIBERO benchmark (Liu et al., 2023a), which contains four 10-task suites (Spatial, Object, Goal, and Long), and the more challenging LIBERO-Plus (Fei et al., 2026). LIBERO-Plus introduces seven perturbation types covering visual observations, language instructions, and initial conditions, with more than 2,500 task variants per suite.

Baselines. We compare AC-Sparse with activation sparsification methods TEAL (Liu et al., 2025) and WINA (Chen et al., 2026), and visual-token pruning methods VLA-ADP (Pei et al., 2026) and SpecPrune-VLA (Wang et al., 2026a). TEAL and WINA are evaluated at the same activation sparsity as AC-Sparse, while VLA-ADP and SpecPrune-VLA are configured to match its theoretical FLOPs, with an absolute difference below 0.005 in normalized FLOPs ratio.

Evaluation details. Activation sparsity denotes the fraction of eligible input token–channel activations removed within each linear projection under a budget shared across tokens. We evaluate sparsity levels of 25%, 50%, 65%, and 75%, using $\beta = 0.5$ and modality-wise normalization unless otherwise specified. Each LIBERO task is evaluated over 50 trials, while each LIBERO-Plus variant is evaluated once; all evaluations use random seed 7. We report suite-level and average success rates and the relative drop from the corresponding full model (Rel. drop).

4.2 Main Results

We compare AC-Sparse with the evaluated baselines across different VLA architectures, benchmarks, and sparsity levels. As shown in Tables 1–3, AC-Sparse maintains success rates close to the full model at moderate sparsity levels and exhibits smaller performance reductions than TEAL and WINA at higher sparsity levels.

Table 1: Performance of AC-Sparse on VLA-Adapter in LIBERO.

Sparsity	Method	LIBERO suite				Avg.	Rel. drop (%) ↓
		Long	Goal	Object	Spatial		
0%	Baseline (full model)	86.00	93.60	99.60	97.20	94.10	–
25%	VLA-ADP	59.00	71.40	90.20	82.80	75.85	19.39
	SpecPrune-VLA	70.60	86.00	99.40	81.60	84.40	10.31
	TEAL	85.00	92.60	99.40	96.40	93.35	0.80
	WINA	84.60	92.60	98.80	96.00	93.00	1.17
	AC-Sparse	**86.00**	92.20	98.60	**97.20**	**93.50**	**0.64**
50%	VLA-ADP	30.60	52.00	62.00	62.00	51.65	45.11
	SpecPrune-VLA	68.00	70.80	96.80	74.60	77.55	17.59
	TEAL	86.60	87.80	98.80	96.20	92.35	1.86
	WINA	89.20	89.20	98.80	96.00	93.30	0.85
	AC-Sparse	**89.80**	**91.40**	99.60	95.80	**94.15**	**-0.05**
65%	SpecPrune-VLA	70.20	66.80	97.40	77.80	78.05	17.06
	TEAL	81.80	46.60	96.40	68.00	73.20	22.21
	WINA	81.00	50.80	97.00	69.00	74.45	20.88
	AC-Sparse	**89.40**	**89.00**	**99.40**	**95.80**	**93.40**	**0.74**
75%	SpecPrune-VLA	72.20	64.20	98.40	81.40	79.05	15.99
	TEAL	52.40	15.60	74.80	51.80	48.65	48.30
	WINA	48.00	16.00	72.00	52.20	47.05	50.00
	AC-Sparse	**79.00**	**73.00**	**98.40**	**83.60**	**83.50**	**11.26**

VLA-Adapter on LIBERO. As shown in Table 1, AC-Sparse preserves performance well at moderate sparsity levels, with average success rates at 25% and 50% sparsity remaining within 0.60 per-

centage points of the full model. The differences between methods become much larger as sparsity increases. At 65%, AC-Sparse achieves 93.40% average success, only 0.70 percentage points below the full model, while outperforming WINA and SpecPrune-VLA by 18.95 and 15.35 points, respectively. At 75%, it still maintains 83.50% success, compared with 48.65% for TEAL and 79.05% for SpecPrune-VLA. This trend indicates that the benefit of AC-Sparse becomes more evident when a larger fraction of activations must be removed, while the performance gap to the full model remains relatively small. Overall, AC-Sparse provides stronger performance preservation than both activation sparsification and visual-token pruning methods across the evaluated sparsity range.

$\pi_{0.5}$ on LIBERO. As shown in Table 2, all three activation sparsification methods remain close to the full model at 50% sparsity, indicating that moderate sparsification has little effect on task performance. At 65%, however, their behavior differs substantially. AC-Sparse maintains 93.40% average success, only 3.55 percentage points below the full model, whereas TEAL and WINA decrease to 78.95% and 34.40%, respectively. AC-Sparse also retains high success rates across all four LIBERO suites, including 99.20% on Object and 97.60% on Spatial, rather than preserving performance only on a subset of tasks. These results show that the effectiveness of AC-Sparse is not specific to the action-query-based setting and extends to the diffusion-based $\pi_{0.5}$ architecture under high sparsity.

Table 2: Performance of AC-Sparse on $\pi_{0.5}$ in LIBERO.

Sparsity	Method	LIBERO suite				Avg.	Rel. drop (%) ↓
		Long	Goal	Object	Spatial		
0%	Baseline (full model)	92.80	97.80	98.60	98.60	96.95	–
50%	TEAL	95.60	97.40	99.40	98.00	97.60	-0.67
	WINA	94.80	98.00	98.40	98.40	97.40	-0.46
	AC-Sparse	94.60	96.20	98.40	**98.40**	96.90	0.05
65%	TEAL	54.20	79.40	98.20	84.00	78.95	18.57
	WINA	23.00	37.40	56.40	20.80	34.40	64.52
	AC-Sparse	**84.00**	**92.80**	**99.20**	**97.60**	**93.40**	**3.66**

VLA-Adapter on LIBERO-Plus. As shown in Table 3, AC-Sparse remains close to the full model at 50% sparsity under the more challenging LIBERO-Plus evaluation. The advantage becomes clearer as sparsity increases. At 65%, AC-Sparse achieves 67.73% average success, only 2.37 percentage points below the full model, while TEAL and WINA decrease to 52.93% and 52.53%, respectively. At 75%, AC-Sparse retains 57.78% success, compared with 32.55% for TEAL and 31.80% for WINA. The consistent gap at both high sparsity levels indicates that the performance preservation observed on standard LIBERO is maintained when the evaluation includes changes in visual observations, language instructions, and initial conditions. These results support the robustness of AC-Sparse under more challenging evaluation conditions.

4.3 Analysis of Action-Centric Sparsification

We further examine how action representation attention relates to token-dependent perturbation sensitivity and sparsity allocation. We first test whether attention distinguishes downstream sensitivity under controlled local perturbations, and then analyze how it changes the shared sparsity allocation across tokens.

Controlled perturbation analysis. For each observation, we select high- and low-attention tokens within the same layer and projection and sparsify their activations to induce matched ℓ_2 perturbations at the projection output. We measure the resulting perturbations at the action representation positions in the next Transformer block, denoted by D_{H} and D_{L}, while holding its self-attention weights fixed to avoid changes in attention routing. We report $P_{\mathrm{H}} = D_{\mathrm{H}}/(D_{\mathrm{H}} + D_{\mathrm{L}})$, where $P_{\mathrm{H}} > 0.5$ indicates a larger downstream effect from the high-attention token.

Table 3: Performance of AC-Sparse on VLA-Adapter in LIBERO-Plus.

Sparsity	Method	LIBERO-Plus suite				Avg.	Rel. drop (%) ↓
		Long	Goal	Object	Spatial		
0%	Baseline (full model)	60.60	56.80	79.20	83.80	70.10	–
50%	TEAL	62.40	50.50	76.10	81.60	67.65	3.50
	WINA	61.70	49.40	76.80	80.10	67.00	4.42
	AC-Sparse	61.90	**56.30**	**79.80**	**84.10**	**70.53**	**-0.61**
65%	TEAL	53.80	28.60	67.70	61.60	52.93	24.50
	WINA	52.50	28.60	67.60	61.40	52.53	25.07
	AC-Sparse	**60.20**	**52.00**	**78.30**	**80.40**	**67.73**	**3.39**
75%	TEAL	32.70	9.10	44.30	44.10	32.55	53.57
	WINA	33.60	9.80	41.60	42.20	31.80	54.64
	AC-Sparse	**50.50**	**39.70**	**74.10**	**66.80**	**57.78**	**17.58**

As shown in Figure 2, P_{H} remains above 0.5 across all evaluated Transformer layers, indicating that high-attention tokens tend to induce larger downstream perturbations under matched local perturbations. This supports action representation attention as a useful signal for token-dependent downstream sensitivity. Similar trends for the gate, up, and down projections are provided in Appendix D.

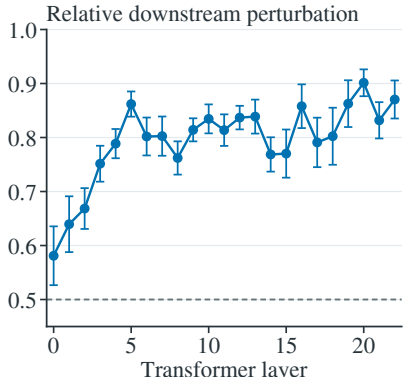


Figure 2: **Controlled perturbation analysis.** Downstream perturbation proportion for the O projection.

Effect on token-wise sparsity allocation. Figure 3 further shows how action representation attention changes the shared sparsity allocation. At the modality level, AC-Sparse assigns a larger share of sparsity to visual tokens and less to language and action tokens. Within the visual modality, removing action representation attention produces a more uniform allocation, whereas AC-Sparse assigns less sparsity to highly attended interaction regions, such as those around the gripper and bowl, and more sparsity elsewhere. This shows that the attention signal redistributes sparsity at both the modality and visual-token levels.

4.4 Ablation Studies

We study the contributions of the main components of AC-Sparse and the effect of the attention exponent β on LIBERO at 65% and 75% activation sparsity.

Component ablation. Table 4 shows that a shared budget alone does not improve WINA. Adding action representation attention produces the dominant gain, increasing average success to 92.00% and 78.00% at 65% and 75% sparsity, respectively, while modality-wise normalization further improves performance to 93.40% and 83.50%. These results indicate that attention provides the key token-dependent guidance for shared-budget allocation, while normalization mitigates cross-modal scale differences in the activation–weight scores.

Table 4: **Component ablation.** Average success rates (%) on LIBERO.

Variant	Shared	Attn.	Norm.	65%	75%
WINA (token-wise)	–	–	–	74.50	47.10
WINA + shared budget	✓	–	–	68.20	46.90
AC-Sparse w/o norm.	✓	✓	–	92.00	78.00
AC-Sparse	✓	✓	✓	**93.40**	**83.50**

Sensitivity to β. Figure 4 evaluates the attention exponent β, which controls the influence of action representation attention in the sparsification score. AC-Sparse maintains strong performance across

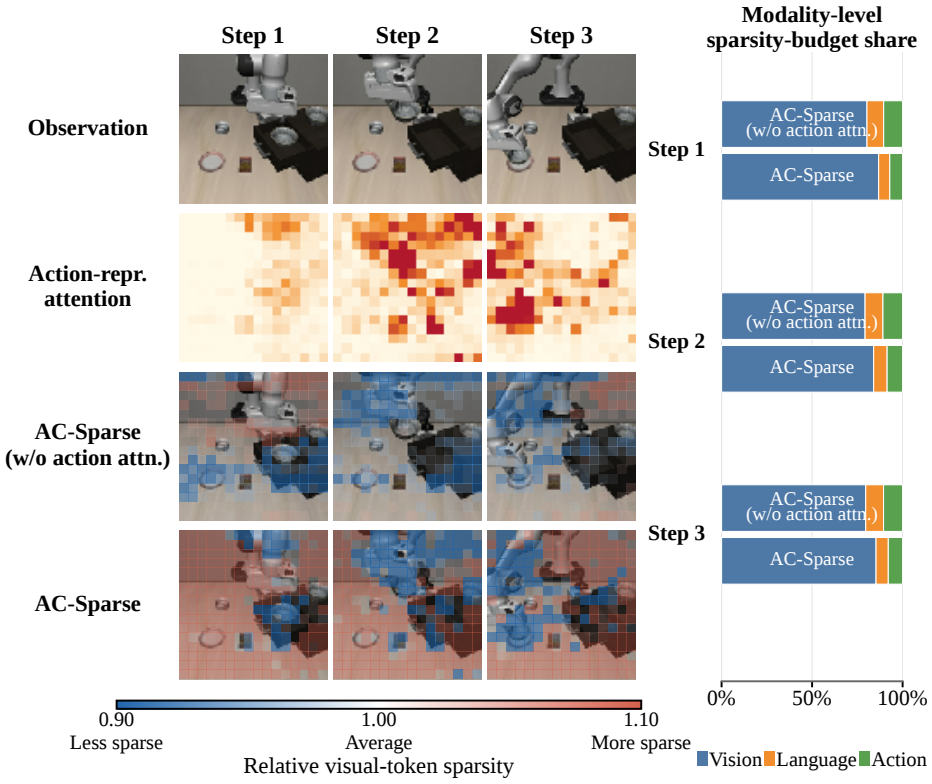


Figure 3: **Effect of action representation attention on sparsity allocation.** Relative visual-token sparsity and modality-level sparsity-budget shares across three representative manipulation steps. Visual-token sparsity is normalized by its mean within each inference run, with values below and above 1 indicating lower and higher sparsity than the visual-token average.

the tested values, indicating limited sensitivity to the exact choice of β. Adjusting the attention strength provides an additional performance gain at both sparsity levels, showing that β offers a simple way to further refine attention-guided sparsity allocation without requiring careful tuning.

5 Conclusion

We presented AC-Sparse, a training-free activation sparsification method for reducing computation in VLA inference. AC-Sparse uses action representation attention to guide sparsity allocation across token–channel activations under a shared budget. We further provide theoretical support for the local sparsification objective, showing that the derived selection criterion minimizes a separable upper bound and becomes exact when the weight columns are mutually orthogonal. Across different VLA architectures and manipulation benchmarks, AC-Sparse preserves task performance more effectively than existing activation sparsification methods at high sparsity.

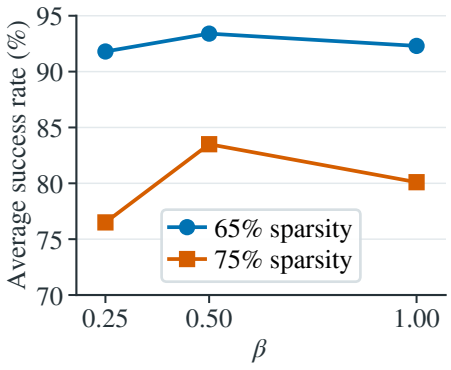


Figure 4: **Sensitivity to the attention exponent β.** Average success rates on LIBERO at 65% and 75% activation sparsity.

6 Limitations

Our evaluation focuses on FLOPs reduction rather than wall-clock acceleration. Realizing speedups from fine-grained, token-dependent sparsity requires specialized sparse kernels, which we leave for future work.

AI USE STATEMENT

Generative AI tools were used in this work for five purposes. First, they were used to aid and polish the writing of the manuscript, including improving grammar, clarity, conciseness, and academic presentation. Second, they were used for retrieval and discovery of relevant literature and for summarizing related work; all cited papers and technical claims were checked against the original sources by the authors. Third, they were used to support research ideation and execution, including brainstorming research directions, refining hypotheses, discussing methodological choices, assisting with experimental design, and analyzing experimental results. Fourth, they were used to assist in drafting portions of the manuscript, which were subsequently reviewed, revised, and integrated by the authors. Finally, generative AI tools were used to assist with mathematical reasoning and proofs, including suggesting derivation steps, checking proof structure, and identifying potential gaps. All mathematical claims and proofs were independently examined and verified by the authors.

All AI-assisted content was reviewed by the authors. Literature-related claims were verified against the original publications, experimental interpretations were checked against the underlying results, and mathematical arguments were independently validated. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Sihan Chen, Dan Zhao, Jongwoo Ko, Colby Banbury, Huiping Zhuang, Luming Liang, Pashmina Cameron, and Tianyi Chen. WINA: Weight informed neuron activation for accelerating large language model inference. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=l7Vb3yxmuz.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-plus: A progressive robustness benchmark for visual-language-action models. In *Conference on Computer Vision and Pattern Recognition 2026*, 2026. URL https://openreview.net/forum?id=Mi1AYOZ9uE.

Juntao Gao, Feiyang Ye, Jing Zhang, and Wenjing Qian. Compressor-vla: Instruction-guided visual token compression for efficient robotic manipulation, 2025. URL https://arxiv.org/abs/2511.18950.

Yuting Huang, Leilei Ding, Zhipeng Tang, Zenghuan Zhu, Jiajun Deng, Xinrui Lin, Shuo Liu, Haojie Ren, Jianmin Ji, and Yanyong Zhang. EcoVLA: Environment-aware adaptive pruning with interleaved inference orchestration for vision-language-action models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=7kH5uphAes.

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.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn.

Openvla: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

Donghyun Lee, Jaeyong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. CATS: Context-aware thresholding for sparsity in large language models. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=v3w2a7EInO.

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*, 2026. URL https://openreview.net/forum?id=RwdGIIjPlC.

Youwei Liang, Chongjian GE, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. EVit: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=BjyvwnXXVn_.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023a. URL http://papers.nips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=dGVZwyq5tV.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. Deja vu: Contextual sparsity for efficient llms at inference time. In *ICML*, pp. 22137–22176, 2023b. URL https://proceedings.mlr.press/v202/liu23am.html.

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Bridging the semantic-action gap in visual token pruning for efficient vla inference, 2026. URL https://arxiv.org/abs/2511.16449.

Sifan Long, Zhen Zhao, Jimin Pi, Shengsheng Wang, and Jingdong Wang. Beyond attentive tokens: Incorporating token importance and diversity for efficient vision transformers. In *CVPR*, pp. 10334–10343, 2023. URL https://doi.org/10.1109/CVPR52729.2023.00996.

Chi Ma, Mincong Huang, Ying Zhang, Chao Wang, Yujie Wang, Lei Yu, Chuan Liu, and Wei Lin. First activations matter: Training-free methods for dynamic activation in large language models, 2024. URL https://arxiv.org/abs/2408.11393.

Xiaohuan Pei, Yuxing Chen, Siyu Xu, Yunke Wang, Yuheng Shi, and Chang Xu. Action-aware dynamic pruning for efficient vision-language-action manipulation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ea6j8k8Rnw.

Yixin Song, Haotong Xie, Zhengyan Zhang, Bo Wen, Li Ma, Zeyu Mi, and Haibo Chen. Turbo sparse: Achieving llm sota performance with minimal activated parameters. *CoRR*, abs/2406.05955, 2024. URL https://doi.org/10.48550/arXiv.2406.05955.

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, 2025. URL https://arxiv.org/abs/2505.21200.

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. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=MjE62ZnpRH.

Hongyu Wang, Shuming Ma, Ruiping Wang, and Furu Wei. Q-sparse: All large language models can be fully sparsely-activated, 2024. URL https://arxiv.org/abs/2407.10969.

Ming Wang, Miao Zhang, Xuebo Liu, and Liqiang Nie. Weight-aware activation sparsity with constrained Bayesian optimization scheduling for large language models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1086–1098, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.57. URL https://aclanthology.org/2025.emnlp-main.57/.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, Siteng Huang, Yifan Tang, Wenhui Wang, Ru Zhang, Jianyi Liu, and Donglin Wang. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *AAAI*, pp. 18638–18646, 2026b. URL https://doi.org/10.1609/aaai.v40i22.38931.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. VLA-cache: Efficient vision-language-action manipulation via adaptive token caching. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=QZYZ0Xm58q.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. EfficientVLA: Training-free acceleration and compression for vision-language-action models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a. URL https://openreview.net/forum?id=SELYlDHZk2.

Zebin Yang, Yijiahao Qi, Tong Xie, Bo Yu, Shaoshan Liu, and Meng Li. DySL-VLA: Efficient vision-language-action model inference via dynamic-static layer-skipping for robot manipulation, 2025b. URL https://openreview.net/forum?id=sZrjfrFPdj.

Yifan Ye, Jiaqi Ma, Jun Cen, and Zhihe Lu. Token expand-merge: Training-free token compression for vision-language-action models, 2025. URL https://arxiv.org/abs/2512.09927.

Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. In *NeurIPS*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/67b0e7c7c2a5780aeefe3b79caac106e-Abstract-Conference.html.

Rongyu Zhang, Menghang Dong, Yuan Zhang, Liang Heng, Xiaowei Chi, Gaole Dai, Li Du, Yuan Du, and Shanghang Zhang. Mole-vla: Dynamic layer-skipping vision language action model via mixture-of-layers for efficient robot manipulation, 2025a. URL https://arxiv.org/abs/2503.20384.

Zhenyu Zhang, Zechun Liu, Yuandong Tian, Harshit Khaitan, Zhangyang Wang, and Steven Li. R-sparse: Rank-aware activation sparsity for efficient LLM inference. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=9VMW4iXfKt.

A Implementation Details of AC-Sparse

A.1 Action Representation Selection and Attention Routing

Action representations and attention extraction. AC-Sparse extracts token-wise attention scores from the action representations used by each VLA architecture. In VLA-Adapter, the selected action queries correspond to the token positions initialized by learnable action queries. These positions contain updated hidden states in subsequent Transformer layers, allowing AC-Sparse to extract attention from the evolving action representations throughout the network.

For VLA models based on iterative action denoising, such as $\pi_{0.5}$, action representations are obtained from the action-side hidden states involved in refining the noisy action sequence. In the initial pass, action tokens attend to both the visual-language prefix and the action tokens. AC-Sparse uses the corresponding attention scores to sparsify the prefix and action tokens once, after which the prefix key-value states are stored in the KV cache. During subsequent denoising steps, the cached prefix states are reused, while AC-Sparse is applied to the action-side suffix tokens at each step using attention among the suffix tokens.

For projection P in layer l, the selected action queries form the set $\mathcal{Q}_{l,P}$. AC-Sparse averages their attention weights across attention heads, as defined in Eq. 3, to obtain the token-wise score $r_{l,P}(t)$. The attention scores are aligned with the input tokens of the projection being sparsified.

Projection-wise attention routing. AC-Sparse selects the attention source for each projection according to the execution order within the Transformer block. Q, K, and V projections use attention from the preceding layer, whereas O and FFN projections use attention from the current layer. Table 5 summarizes the routing used in our implementation.

Table 5: Attention sources used by different linear projections.

Projection	Attention source
Q, K, V	Preceding layer
O	Current layer
Gate, Up, Down	Current layer

In the first Transformer layer, AC-Sparse sets $r_{l,P}(t) = 1$ for all tokens and projections, yielding an activation–weight score without attention weighting. From the second layer onward, each projection uses the attention source specified in Table 5. For architectures with separate visual-language and action-processing pathways, attention scores are assigned to the corresponding projection inputs according to the token interactions available at each inference stage.

Normalization of token attention. AC-Sparse constructs the token distribution $p_{l,P}(t)$ from the selected attention scores, as defined in Eq. 4. The normalization factor is

$$Z_{l,P} = \sum_{j=1}^{T} r_{l,P}(j)^{\beta}. \tag{9}$$

Since $Z_{l,P}$ is constant across all token–channel candidates within a projection, it does not affect their relative ranking. The Bottom-K selection in Eq. 8 can therefore be performed directly using $r_{l,P}(t)^{\beta/2}|x^P_{l,t,c}|\|W^P_{l,:,c}\|_2$. With $r_{l,P}(t) = 1$ in the first layer, this score reduces to $|x^P_{l,t,c}|\|W^P_{l,:,c}\|_2$.

A.2 Modality-wise Normalization

Visual, language, and action tokens may exhibit different scales in their local channel costs, which can bias sparsity allocation across modalities. AC-Sparse therefore normalizes the activation–weight term within each modality before shared-budget selection.

Let $\mathcal{V}$ denote the set of input modalities and let $\Omega^P_{l,v} \subseteq \Omega^P_l$ denote the eligible token–channel positions belonging to modality $v \in \mathcal{V}$. For each modality with a nonempty eligible set, we compute

$$\mu^P_{l,v} = \frac{1}{|\Omega^P_{l,v}|} \sum_{(t,c)\in\Omega^P_{l,v}} \left|x^P_{l,t,c}\right| \left\|W^P_{l,:,c}\right\|_2 . \tag{10}$$

For an eligible position (t, c) belonging to modality $v(t)$, the normalized AC-Sparse cost is

$$\widetilde{C}^P_{l,t,c} = r_{l,P}(t)^{\beta/2} \frac{\left|x^P_{l,t,c}\right| \left\|W^P_{l,:,c}\right\|_2}{\mu^P_{l,v(t)} + \varepsilon}, \tag{11}$$

where $\varepsilon > 0$ is a small constant for numerical stability. AC-Sparse applies the shared-budget Bottom-K rule to $\widetilde{C}^P_{l,t,c}$ to select the sparsified positions.

Modality-wise normalization reduces scale differences across modalities before joint sparsity allocation. Its effect is evaluated in Sec. 4.4. The theoretical results in Sec. 3.3 are derived for the unnormalized cost in Eq. 7.

B Proof of the Theoretical Results

B.1 Proof of the Separable Upper Bound

We provide the proof of Theorem 1. For readability, we omit the layer and projection indices in this subsection and write $r(t) = r_{l,P}(t)$, $p(t) = p_{l,P}(t)$, and $C_{t,c} = C^P_{l,t,c}$.

Let $D \subseteq \Omega$ be a set of eligible token–channel positions selected for sparsification, with $|D| = K$, and let $D_t = \{c \mid (t, c) \in D\}$. The local output perturbation of token t is

$$\Delta y_t = \sum_{c\in D_t} x_{t,c} W_{:,c}. \tag{12}$$

Applying the Cauchy–Schwarz inequality gives

$$\begin{aligned} \|\Delta y_t\|_2^2 &= \left\| \sum_{c\in D_t} x_{t,c} W_{:,c} \right\|_2^2 \\ &\le |D_t| \sum_{c\in D_t} |x_{t,c}|^2 \|W_{:,c}\|_2^2 . \end{aligned} \tag{13}$$

Since $|D_t| \le K$ for every token, the attention-weighted local objective satisfies

$$\begin{aligned} \mathcal{J}(D) &= \sum_{t=1}^{T} p(t) \|\Delta y_t\|_2^2 \\ &\le \sum_{t=1}^{T} p(t)|D_t| \sum_{c\in D_t} |x_{t,c}|^2 \|W_{:,c}\|_2^2 \\ &\le K \sum_{(t,c)\in D} p(t) |x_{t,c}|^2 \|W_{:,c}\|_2^2 . \end{aligned} \tag{14}$$

Substituting $p(t) = r(t)^\beta / Z$, where $Z = \sum_{j=1}^{T} r(j)^\beta$, yields

$$\begin{aligned} \mathcal{J}(D) &\le \frac{K}{Z} \sum_{(t,c)\in D} r(t)^\beta |x_{t,c}|^2 \|W_{:,c}\|_2^2 \\ &= \frac{K}{Z} \sum_{(t,c)\in D} C_{t,c}^2 . \end{aligned} \tag{15}$$

For a fixed sparsity budget $K > 0$, the factor K/Z is independent of the selected positions, and each remaining term depends only on its corresponding token–channel position. The upper bound is therefore minimized by selecting the K eligible positions with the smallest costs $C_{t,c}$, as specified by the Bottom-K rule in Eq. 8. When $K = 0$, no positions are sparsified and the objective is zero. This completes the proof of separable upper-bound optimality.

B.2 Exact Optimality under Orthogonal Weight Columns

Suppose that the weight columns of projection P are mutually orthogonal.

$$\left(W^P_{l,:,c}\right)^\top W^P_{l,:,c'} = 0, \qquad c \neq c'. \tag{16}$$

Under this condition, the cross terms in the squared local output perturbation vanish. For token t,

$$\begin{aligned} \left\|\Delta y^P_{l,t}\right\|_2^2 &= \left\| \sum_{c \in D^P_{l,t}} x^P_{l,t,c} W^P_{l,:,c} \right\|_2^2 \\ &= \sum_{c \in D^P_{l,t}} \left|x^P_{l,t,c}\right|^2 \left\|W^P_{l,:,c}\right\|_2^2 . \end{aligned} \tag{17}$$

Substituting this expression into Eq. 2, we obtain

$$\begin{aligned} \mathcal{J}^P_l(D^P_l) &= \sum_{t=1}^{T} p_{l,P}(t) \sum_{c \in D^P_{l,t}} \left|x^P_{l,t,c}\right|^2 \left\|W^P_{l,:,c}\right\|_2^2 \\ &= \frac{1}{Z_{l,P}} \sum_{(t,c) \in D^P_l} \left(C^P_{l,t,c}\right)^2 , \end{aligned} \tag{18}$$

where $Z_{l,P} = \sum_{j=1}^{T} r_{l,P}(j)^\beta$.

The local objective is therefore separable across token–channel positions. Selecting the K^P_l eligible positions with the smallest costs exactly minimizes the attention-weighted local output perturbation objective under the fixed sparsity budget.

B.3 Relation to the WINA Criterion

The core AC-Sparse cost is

$$C^P_{l,t,c} = r_{l,P}(t)^{\beta/2} \left|x^P_{l,t,c}\right| \left\|W^P_{l,:,c}\right\|_2 . \tag{19}$$

When token attention scores are uniform, the attention-dependent factor is constant across all token–channel candidates. The AC-Sparse ranking then reduces to

$$\left|x^P_{l,t,c}\right| \left\|W^P_{l,:,c}\right\|_2 , \tag{20}$$

which recovers the WINA scoring form. The resulting sparsity patterns may still differ when the two methods use different budget-allocation rules.

The shared projection-level budget allows token attention to influence sparsity allocation across tokens. Under a fixed per-token sparsity budget, multiplying channel costs by the same token-dependent factor does not change the channel ranking within that token. In contrast, joint Bottom-K selection compares candidates across tokens, allowing attention scores to influence the number of activations sparsified in each token.

C Detailed FLOPs Analysis

We analyze ideal sparse linear computation and the arithmetic operations used to construct the AC-Sparse scores. The resulting FLOPs estimates concern the specified computations and do not include all runtime overheads.

C.1 Dense and Sparse Linear Computation

Consider a linear projection P with T input tokens, input dimension d_{in}^P, and output dimension d_{out}^P. Counting each multiply–accumulate as two FLOPs, the dense projection requires

$$F_{\text{dense}}^P = 2Td_{\text{in}}^P d_{\text{out}}^P. \tag{21}$$

Let $k_{l,t}^P$ be the number of retained input channels for token t. Under ideal sparse computation, only retained channels participate in the linear operation, giving

$$F_{\text{sparse}}^P = 2d_{\text{out}}^P \sum_{t=1}^{T} k_{l,t}^P. \tag{22}$$

For overall activation sparsity s_l^P, the number of retained token–channel positions is

$$\sum_{t=1}^{T} k_{l,t}^P = (1 - s_l^P)Td_{\text{in}}^P. \tag{23}$$

Therefore,

$$F_{\text{sparse}}^P = (1 - s_l^P)F_{\text{dense}}^P. \tag{24}$$

This reduction counts the multiply–accumulate operations associated with sparsified input channels. It does not include operations in model components outside the sparsified projections.

C.2 Attention Aggregation and Score Construction

AC-Sparse constructs channel sparsification costs from token attention scores, activation magnitudes, and weight-column norms. The weight-column norms are input-independent and can be precomputed when the model is loaded.

For a projection using attention scores obtained by averaging $|\mathcal{Q}_l|$ action-representation positions over H heads, computing all T token scores requires

$$T(|\mathcal{Q}_l|H - 1) \tag{25}$$

additions, followed by T multiplications by the constant $1/(|\mathcal{Q}_l|H)$. Counting these additions and multiplications gives

$$F_{\text{rel}} = T|\mathcal{Q}_l|H. \tag{26}$$

This count assumes that the attention weights have already been computed during the forward pass and does not include recomputing query–key products or attention softmax operations.

Once $r(t)^{\beta/2}$ is available, the core sparsification cost

$$C_{l,t,c}^P = r(t)^{\beta/2} \left|x_{l,t,c}^P\right| \left\|W_{l,:,c}^P\right\|_2 \tag{27}$$

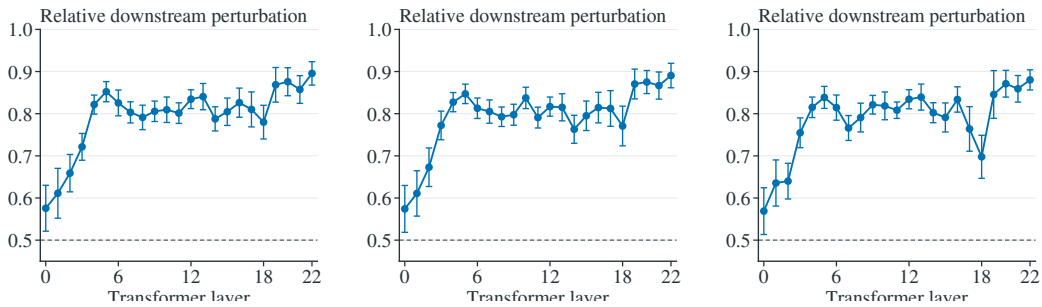
requires two multiplications per token–channel candidate. The corresponding multiplication count is

$$F_{\text{cost}}^P = 2Td_{\text{in}}^P. \tag{28}$$

Assigning one complete attention-aggregation operation to projection P, the counted arithmetic cost of score construction is

$$F_{\text{score}}^P = F_{\text{rel}} + F_{\text{cost}}^P = T|\mathcal{Q}_l|H + 2Td_{\text{in}}^P. \tag{29}$$

In practice, token attention scores can be reused across projections that share the same attention source. When computing the total cost across such projections, the aggregation term should be counted once rather than assigned repeatedly to every projection.



(a) Gate projection (b) Up projection (c) Down projection

Figure 5: **Additional controlled perturbation results.** Downstream perturbation proportion of high-attention tokens for the gate, up, and down projections. The dashed line at 0.5 indicates equal downstream effects.

C.3 Combined Arithmetic Cost

Under the accounting above, the sparse linear computation and the counted score-construction operations for projection P require

$$F^P_{\text{counted}} = F^P_{\text{sparse}} + F^P_{\text{score}}. \tag{30}$$

Relative to dense linear computation, this gives

$$\frac{F^P_{\text{counted}}}{F^P_{\text{dense}}} = 1 - s^P_l + \frac{1}{d^P_{\text{out}}} + \frac{|\mathcal{Q}_l|H}{2d^P_{\text{in}}d^P_{\text{out}}}. \tag{31}$$

The last term is reduced when attention aggregation is shared across multiple projections. The expression also excludes the computation of token-attention powers, absolute values, modality-wise normalization, Bottom-K selection, indexing, memory access, and the cost of materializing attention weights when they are not otherwise available. It therefore represents a partial arithmetic-cost breakdown rather than the complete end-to-end inference cost.

The theoretical reduction in sparse linear FLOPs is realized only when the corresponding multiply–accumulate operations are skipped. Multiplying dense activations by a sparsity mask before executing an ordinary dense linear operator does not produce the same reduction in executed arithmetic operations.

D Additional Controlled Perturbation Results

We provide additional results of the controlled perturbation analysis in Section 4.3. Figure 5 reports the downstream perturbation proportion for the gate, up, and down projections. We use the same experimental protocol as in the main text: the high- and low-attention tokens are selected within the same layer and projection, their induced local output perturbations are matched in ℓ_2 magnitude, and the subsequent attention weights are kept fixed.

Across the three projections, the downstream perturbation proportion of the high-attention token remains consistently above the equal-effect reference of 0.5 over the evaluated Transformer layers. This result is consistent with the O projection reported in Figure 2, showing that the observed difference between high- and low-attention tokens is not specific to the attention output projection.