SPARSESAM: STRUCTURED SPARSIFICATION OF ACTIVATIONS IN SEGMENT ANYTHING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The Segment Anything Model (SAM) achieves strong promptable segmentation, but its ViT-based image encoders dominate inference latency and memory footprint. Existing activation-compression methods, such as token merging, shorten the processed sequence but introduce non-trivial runtime overhead and can suffer severe quality degradation under high compression. Sparse-attention methods typically leave the MLP dense, limiting end-to-end speedup. We propose *SparseSAM*, a training-free structured-sparsification framework that jointly accelerates attention and MLP layers while preserving token identity. SparseSAM introduces *Stripe-Sort Attention*, which forms fixed spatial groups using Z-order, ranks those groups with an image-dependent saliency permutation computed once per image, and applies a fixed A-shaped sparse pattern compiled into the attention kernel. SparseSAM further introduces a *Residual-Consistency MLP* that routes only informative tokens through the MLP while propagating the remaining tokens through the residual pathway. On SAM-L , attention-only SparseSAM reaches $2.15\times$ latency speedup at 25% density with an absolute mAP decrease of 0.8 mAP points, while joint attention–MLP compression reaches $2.95\times$ latency speedup with an absolute decrease of 2.7 mAP points.

1 INTRODUCTION

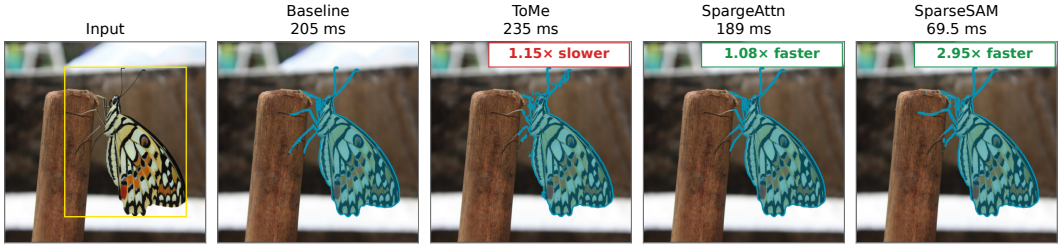

Figure 1: End-to-end latency and qualitative masks on an NVIDIA L4 GPU.

SAMs have demonstrated strong performance across a wide range of image segmentation tasks, enabled in large part by their strong zero-shot generalization capabilities (Kirillov et al., 2023; Ravi et al., 2025; Lu et al., 2023). Their modular architecture consists of an image encoder, a prompt encoder, and a lightweight mask decoder, with the image encoder accounting for over 99% of the model parameters and dominating the overall computation. Serving SAMs either at scale, where high-throughput inference is required, or under constrained compute and memory budgets, such as on mobile and edge devices, remains prohibitively expensive (Wen et al., 2026; Zhang et al., 2026).

Recent work has developed lightweight SAM variants and alternative efficient segmentation architectures (Zhang et al., 2023; Zhou et al., 2025; Xiong et al., 2024; Zhang et al., 2024; Zhao et al., 2023). While effective, they typically require training a new model, incurring substantial data and computational costs while often sacrificing accuracy on tasks that demand high-quality segmentation. A complementary line of work accelerates attention through token permutation (Wang et al., 2025; Pengyu et al., 2024), modality-aware or input-dependent sparsity (Li et al., 2025), dynamic query/key dropping and hashing (Pagliardini et al., 2023), and per-head sparse-pattern selection (Jiang et al., 2024). Spatially structured vision architectures (Yu et al., 2021; Tu et al., 2022) distribute global

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.

interactions across local partitions, but incorporate these patterns during training. Most sparse-attention methods target long-sequence workloads (Wang et al., 2025; Li et al., 2026; Zhang et al., 2025a; Li et al., 2025), such as LLMs and video-generation models, and require runtime token selection or sparse-index construction. In SAM's multi-scale encoder, attention is repeatedly applied to many short local sequences, so this overhead can offset the theoretical computational savings. This motivates a different form of adaptive sparsity: one that adapts token priority to each image while retaining a regular attention structure that does not require dynamically constructed sparse indices.

Moreover, sparsifying attention alone leaves a substantial fraction of SAM's computation untouched, as the MLP layers contribute latency comparable to attention (Fig. 2). Existing advances to reducing MLP latency face trade-offs in accuracy, computational overhead, or training requirements. Post-training quantization (Lv et al., 2024; Ranjan and Savakis, 2025; Zhang et al., 2025b; 2026; Duan et al., 2026) reduces numerical precision and memory footprint. The practical GPU speedups from quantization depend on the available low-precision kernels and operation sizes. Token merging (Bolya et al., 2022; Tran et al., 2024; 2025; Zeng et al., 2025) shortens the token sequence but introduces merging and unmerging overhead to restore the full spatial resolution required for segmentation. Structured pruning (Chen et al., 2024) directly reduces MLP computation but requires retraining or distillation. Existing selective-computation approaches require additional training (Raposo et al., 2024; Zeng et al., 2025), rely on calibration and online similarity checks during autoregressive decoding (Jaiswal et al., 2024), or merge and remove visual tokens (Zeng et al., 2025). These designs limit direct application to pretrained SAM, which is very sensitive to runtime overhead.

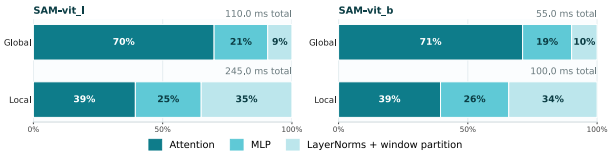

Figure 2: Latency contributions of attention and MLP layers in SAM.

These limitations highlight a key gap: *efficient, training-free methods that jointly reduce attention and MLP computation while preserving the token identities and spatial alignment required for segmentation*. To address this gap, we introduce *SparseSAM*, a framework with two complementary techniques that take advantage of SAM's sparsity pattern:

- **Stripe-Sort Attention.** We rank spatial token groups by image-dependent saliency and interleave their tokens to preserve broad spatial coverage under sparse attention. We share a fixed A-shaped attention pattern across heads and compile its tile traversal directly into the CUDA kernel, eliminating runtime mask construction and dynamic block selection. The kernel fuses token reordering and restoration of the original spatial order to minimize per-layer overhead (Fig. 4).
- **Residual-Consistency MLP.** Our analysis further shows that SAM's MLP updates concentrate on distinctive spatial regions. Therefore, we apply the MLP only to the top-K tokens in the saliency-ranked, non-interleaved Morton-group ordering, while the remaining tokens propagate through the residual pathway. This fine-grained routing reduces MLP computation while preserving every spatial token and requires no learned routers, calibration, or retraining (Fig. 6).

Both techniques share a *one-time, image-dependent saliency ranking*, reusing the attention permutation and MLP keep set across encoder blocks to amortize selection overhead while preserving token identity. On the four HQ-44K benchmarks at 50% density, attention-only compression yields 1.56–$1.67\times$ speedup with absolute mIoU decreases of 0.08–0.25 points, whereas joint attention–MLP compression yields 1.88–$2.53\times$ speedup with decreases of 0.51–1.74 points (Table 7).

2 Background and Motivation

SAM Architecture. SAM's ViT encoder combines local attention within non-overlapping windows with global attention over the full token sequence. As shown in Fig. 2, attention accounts for up to 70% of runtime in global blocks, whereas local blocks exhibit a dual bottleneck in attention and MLP computation. This distinction is important because SAM architectures contain substantially more local blocks than global blocks. For example, SAM-L contains 20 local blocks and only 4 global blocks. Thus, practical end-to-end acceleration requires reducing both attention and MLP costs

while minimizing per-layer overhead and preserving feature quality. This motivates an alternative to dynamically identifying sparse regions: *can we restructure the token layout so that efficient sparsity patterns emerge deterministically?* We next present the key insights behind our approach.

Observation 1: SAM exhibits substantial representational redundancy, stemming from how the mask decoder utilizes encoder outputs. To investigate this, at each block we perform k-means clustering on the attention outputs and replace the encoder's 4096 tokens after the MLP update with their nearest cluster centroids. Even with $k = 128$ (approximately 3% of the original token count), the decoder retains 93% of the baseline mIoU on COIFT (Mansilla et al., 2016) (a relative retention ratio, not a 93-point absolute score; Fig. 3), with performance saturating near the baseline at $k = 256$. These results suggest that tokens within the same cluster are largely interchangeable from the decoder's perspective. What matters is the contrast between clusters, which enables the model to distinguish different objects, rather than the precise representation of each patch.

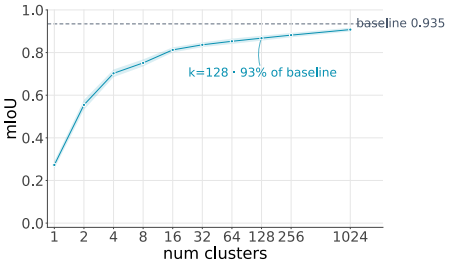

Figure 3: Effect of replacing encoder tokens with their nearest k-means centroids.

	Layer 1	Layer 6	Layer 16	Layer 22
ρ	0.64	0.62	0.75	0.78

Table 1: Layer-wise correlation between token dissimilarity and the MLP update magnitude $\|\Delta_{\text{MLP}}\|_2$.

Observation 2: Segmentation models require broad spatial coverage to capture objects across the image and preserve the local details needed for accurate boundaries. Prior activation-compression studies (Tran et al., 2025; Liu et al., 2024; Endo et al., 2025; Zhu et al., 2026) likewise show that preserving spatial diversity is more important than retaining only contiguous foreground regions. Together with Observation 1, this motivates our *Stripe-Sort Attention* design, which exploits representational redundancy while maintaining broad spatial coverage (Section 3).

Observation 3: MLP update magnitudes are highly non-uniform across tokens. We analyze how strongly MLP layers transform token representations: in the illustrated examples, we notice that background tokens often receive smaller updates than foreground tokens (Fig. 6(b)). To quantify this behavior, Table 1 measures the correlation between the MLP update magnitude, $\|\Delta_{\text{MLP}}\|_2$, and token dissimilarity measured by cosine distance. More distinctive tokens tend to receive larger updates, with a strong positive correlation ($\rho = 0.62$–0.78 across the reported layers). This association motivates the *Residual-Consistency MLP*, which uses spatial saliency as an inexpensive routing proxy.

3 SparseSAM Design

SparseSAM reduces computation in redundant regions while preserving the spatial token structure required for dense prediction. It uses a shared image-dependent ranking across the encoder: *Stripe-Sort Attention* prioritizes informative regions while maintaining spatial coverage (§3.1), and *Residual-Consistency MLP* selectively preserves token-wise MLP updates (§3.2). Both mechanisms are training-free, preserve all spatial tokens, and reuse the ranking across encoder blocks.

3.1 Stripe-Sort Attention

We first introduce *Stripe-Sort Attention*, a structured sparse attention mechanism that reorders spatial tokens to direct more computation toward distinctive regions while maintaining broad spatial coverage through a fixed static sparse attention pattern (Fig. 4).

Saliency ranking. Given a feature map $\mathbf{X} \in \mathbb{R}^{H \times W \times D}$, we construct a one-dimensional token ordering based on token saliency. We use gradient magnitude as a lightweight proxy for saliency: large gradients tend to highlight object boundaries, textures, and other distinctive structures, while small gradients indicate smooth or homogeneous regions. This provides a simple, image-dependent criterion for prioritizing regions with richer details. To preserve spatial locality during grouping, we traverse the $N = HW$ tokens in Morton (Z-) order, which tends to keep nearby spatial locations close in the resulting sequence. We then partition this sequence into groups of G tokens and compute saliency using the Sobel kernels $\mathbf{S}_x$ and $\mathbf{S}_y$:

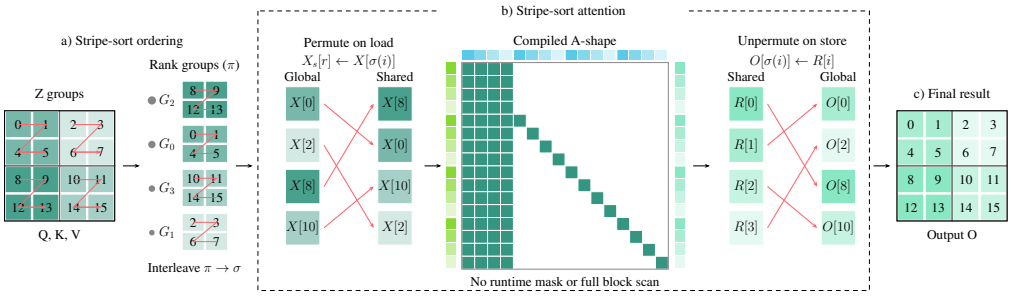

Figure 4: **(a) Stripe-Sort ordering.** Tokens are partitioned into fixed spatial groups along Z-order, ranked by saliency to obtain π, and interleaved to form the stripe-sort order σ. Both π and σ are image-dependent and computed once per image. **(b) Stripe-Sort Attention.** The fixed A-shaped tile pattern is compiled into the kernel, with permutation and inverse permutation fused into indexed loads and output stores using σ. **(c) Output restoration.** Attention outputs are written directly to their original spatial positions using σ, eliminating a separate reordering pass.

$$\mathbf{M}[i,j] = \sqrt{\|\mathbf{G}_x[i,j]\|_2^2 + \|\mathbf{G}_y[i,j]\|_2^2}, \quad \mathbf{G}_x = \mathbf{S}_x * \mathbf{X}, \quad \mathbf{G}_y = \mathbf{S}_y * \mathbf{X}, \tag{1}$$

where $*$ denotes channel-wise spatial convolution and $\|\cdot\|_2$ is the Euclidean norm across channels. We score each Z-order group $\mathcal{G}_t$ by its mean saliency, $e_t = \frac{1}{G}\sum_{(i,j)\in\mathcal{G}_t} \mathbf{M}[i,j]$. Sorting groups by descending e_t while preserving their internal Z-order yields the saliency rank π, computed once per image and reused across encoder layers with the same spatial layout and sparsity configuration.

Stripe-Sort Permutation. We next interleave the ranked groups to distribute representatives from all spatial regions throughout the reordered sequence. Reshaping the ranked token ordering into $\mathbf{T} \in \mathbb{N}^{(N/G)\times G}$ such that $\mathbf{T}[t,g] = \pi[tG+g]$, and setting $\sigma = \mathrm{flatten}(\mathbf{T}^\top)$. As shown in Fig. 5, the resulting sequence consists of G *stripes*, each containing one token from every spatial group. Consequently, retaining a prefix of r complete stripes preserves exactly r representatives from every group, yielding uniform spatial coverage. When the sparsity budget includes only part of the next stripe, the additional representatives are assigned first to groups with higher saliency under ρ.

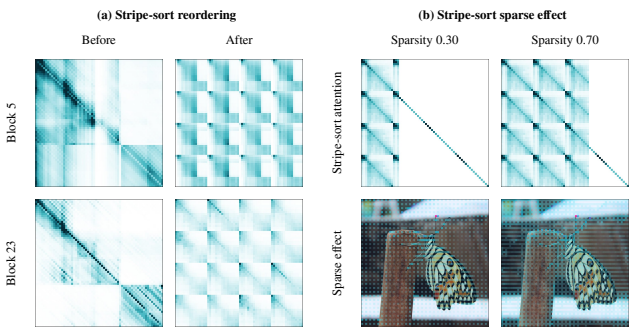

Figure 5: Stripe-sort attention kernel effect.

Why does Stripe-Sort preserve information? We compare dense and sparse attention at the same input. Fix a query i and an attention head. In a group $\mathcal{G}_g$ of G keys, we retain $S_g \subseteq \mathcal{G}_g$ with $1 \le q_g = |S_g| \le G$. Let $\boldsymbol{\mu}_g$ and $\widetilde{\boldsymbol{\mu}}_g$ be the value averages over all and retained keys, respectively, weighted by $e^{z_{ij}}$ and normalized within each set. Here z_{ij} is the logit including positional terms, α_g is its range within the group, and d_g is the largest distance between two value vectors in the group.

Lemma 1 (Approximation within one group)**.** *With retained logits and values unchanged,*

$$\|\boldsymbol{\mu}_g - \widetilde{\boldsymbol{\mu}}_g\|_2 \le \frac{(G-q_g)e^{\alpha_g}}{q_g + (G-q_g)e^{\alpha_g}} d_g. \tag{2}$$

Lemma 1 shows that a few retained keys can approximate a group's average when its value vectors are similar (small d_g), motivating representatives from every group. At equal retained-key counts and logit ranges, an extra key reduces this bound more for groups with greater value dissimilarity (larger d_g). This motivates using Sobel saliency as a proxy for value dissimilarity and giving higher-ranked groups extra keys in a partial stripe. Combining this bound with changes in the attention weight

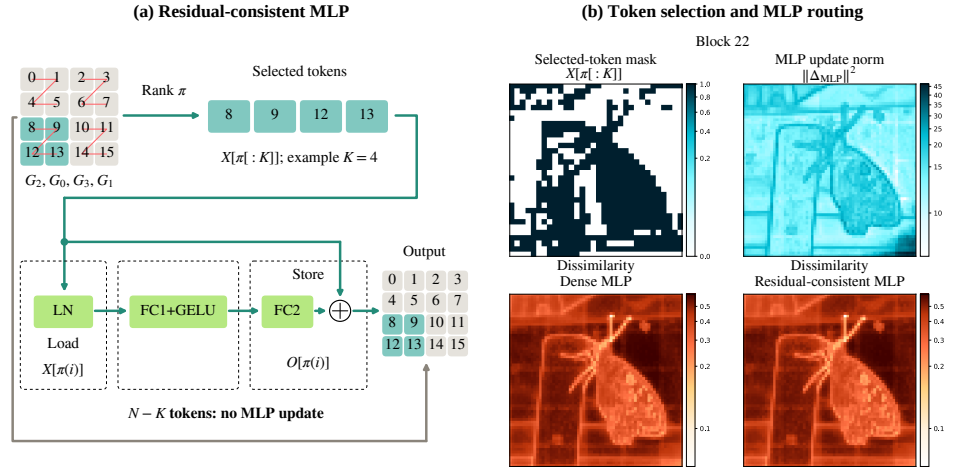

Figure 6: Visualization of the Residual-Consistency MLP. **(a)** The image-dependent ranked token order π is reused to partition tokens into two sets; the top-K highest-saliency tokens receive MLP updates. **(b)** Dense SAM focuses MLP updates on a small set of representative, highly dissimilar tokens. After routing, the inter-token similarity structure remains close to the dense baseline.

assigned to each group gives the following result. Let $\mathbf{o}_i$ and $\widetilde{\mathbf{o}}_i$ be the dense and sparse head outputs, $\alpha = \max_g \alpha_g$, $d_v = \max_g d_g$, and $V = \max_j \|\mathbf{v}_j\|_2$.

Proposition 1 (Attention approximation under Stripe-Sort)**.** *Consider a prefix of r complete stripes, $1 \le r \le G$, so that exactly r keys are retained from every equal-size spatial group. With the retained logits unchanged and excluding additional diagonal keys, the dense and sparse attention outputs satisfy*

$$\|\mathbf{o}_i - \widetilde{\mathbf{o}}_i\|_2 \le \left(1 - \frac{r}{G}\right)\left[e^{\alpha} d_v + V\left(e^{2\alpha} - 1\right)\right]. \tag{3}$$

Proposition 1 gives a small error bound when logits and values are similar within each group, and decreases as more complete stripes are retained (r increases). This directly validates our design. Attention is exact when all keys are retained, or when one key per group is retained, where logits and values are identical within each group. Proofs of Lemma 1 and Proposition 1 are available in Appendix A.1. Empirical evidence for prioritizing high saliency groups is available in Appendix A.3.

Efficient execution. For a given input shape, density, and tile size, we compile the sparse tile traversal directly into the kernel. The image-dependent permutation σ is computed once per image and cached for reuse across all layers. We fuse permutation into query/key/value loads and restore spatial order in output stores, using the original coordinates for relative-position bias. The kernel visits only retained tiles and applies online softmax, avoiding separate reordering passes and runtime mask construction. Appendix A.5 provides kernel details and pseudocode.

3.2 Residual-Consistency MLP

Our second contribution is motivated by the non-uniform structure of MLP updates (**O3** in Section 2). Across spatial tokens, a subset receives substantially larger MLP updates, while many others remain close to their residual representations. These larger updates tend to occur at spatially distinctive regions, such as boundaries and textured structures, and are positively correlated with feature dissimilarity (Table 1). This suggests that MLP computation need not be allocated uniformly across all tokens.

Selective MLP Updates. Given input tokens $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N \times d}$, a standard residual MLP produces $\mathbf{y}_i = \mathbf{x}_i + \Delta_i$, where $\Delta_i = \mathrm{MLP}(\mathrm{LN}(\mathbf{x}_i))$. We reuse the image-dependent, non-interleaved token ordering π introduced in Section 3.1. For an MLP density $r_{\mathrm{mlp}} \in (0, 1)$, we retain the first $K = \lfloor r_{\mathrm{mlp}} N \rfloor$ tokens in this ordering and define $\mathcal{K} = \{\pi_1, \ldots, \pi_K\}$, with corresponding keep and

residual sets $\mathbf{X}_k = \{\mathbf{x}_i \mid i \in \mathcal{K}\}$, $\mathbf{X}_r = \{\mathbf{x}_i \mid i \notin \mathcal{K}\}$. As shown in Fig. 6, the MLP is then executed only on tokens in $\mathbf{X}_k$, while tokens in $\mathbf{X}_r$ bypass the MLP and propagate unchanged through the residual pathway. Specifically, only K of the N tokens incur MLP computation:

$$\mathbf{y}_i = \begin{cases} \mathbf{x}_i + \mathrm{MLP}(\mathrm{LN}(\mathbf{x}_i)), & i \in \mathcal{K}, \\ \mathbf{x}_i, & i \notin \mathcal{K}. \end{cases} \tag{4}$$

Unlike token dropping or merging, this operation reduces the computation while preserving the complete token set, output shape, and spatial correspondence.

Why prioritize high-ranked tokens? Let $\mathbf{Y}$ be the dense residual MLP output and $\widetilde{\mathbf{Y}}^{(\mathcal{K})}$ the routed output for keep set $\mathcal{K}$, where $\Delta_i = \mathrm{MLP}(\mathrm{LN}(\mathbf{x}_i))$ is the MLP update for token i.

Proposition 2 (Optimal residual routing)**.** *For token-wise LayerNorm and MLP operations at a shared input, the squared representation error for any size-K keep set equals the total energy of the omitted updates:*

$$\left\|\mathbf{Y} - \widetilde{\mathbf{Y}}^{(\mathcal{K})}\right\|_F^2 = \sum_{i \notin \mathcal{K}} \|\Delta_i\|_2^2. \tag{5}$$

Consequently, retaining the K tokens with the largest update magnitudes minimizes this error among all size-K keep sets; retaining the K smallest magnitudes maximizes it.

Selecting the largest updates by their exact magnitudes would require running the MLP on every token, defeating the purpose of skipping computation. Based on results from observation 3 and 1, we instead reuse the saliency ranking as a proxy to select tokens cheaply. Table 4 empirically validates this choice through segmentation accuracy with both attention and MLP compression; Appendix A.4 further justifies the effectiveness of cached initial Sobel (saliency) ranking.

Efficient execution. As illustrated in figure 6 **(a)**, gathering is fused into LayerNorm to load selected tokens. The output kernel fuses the second linear projection, residual addition, and scattering to original positions, while copying unselected residual features unchanged. This preserves the full spatial grid without computing all MLP updates; Appendix A.6 provides pseudocode.

4 Experiments

Experimental Setup. Unless otherwise specified, all experiments and latency measurements are conducted on a single NVIDIA L4 edge server GPU. For system efficiency, we measure end-to-end latency, peak GPU memory usage, and throughput under varying density (i.e., $100\% - sparsity$) ratios. Unless otherwise specified, "speedup" denotes latency speedup (dense latency divided by compressed latency); Table 9 explicitly reports throughput speedup instead. All latency results are averaged over multiple runs with synchronized CUDA timing. MS-COCO (Lin et al., 2014) and HQ-44K (Ke et al., 2023) are used to evaluate zero-shot generalization on common object categories and high-quality fine-grained segmentation. For MS-COCO, we utilize DINO (Zhang et al., 2022), H-DETR (Jia et al., 2023), and YOLOX (Ge et al., 2021) as the primary object detectors.

We compare SparseSAM against three baseline categories: efficient SAM architectures, including FastSAM (Zhao et al., 2023), MobileSAM (Zhang et al., 2023), EfficientSAM (Xiong et al., 2024), and EfficientViT-SAM (Zhang et al., 2024); attention-sparsification methods, including SpargeAttn (Zhang et al., 2025a), MMInference (Li et al., 2025), and PISA (Li et al., 2026); and joint attention–MLP compression methods, including ToMe (Bolya et al., 2022) and StructSAM (Nguyen et al., 2026). For a fair comparison, we extend all attention-sparsification baselines with fused relative positional encoding support for SAM's ViT encoder. More results, attributing speedup to kernel optimization and sparsification, are also available in Appendix A.9

4.1 Segmentation Results

4.1.1 Common-Object Segmentation

Table 2 reports zero-shot segmentation results on MS-COCO (Lin et al., 2014) using the DINO detector (Zhang et al., 2022). We report attention-only and joint attention–MLP results separately because joint compression provides higher speedup at a larger accuracy cost.

Table 2: Zero-shot segmentation on MS-COCO using DINO as a bounding-box detector to generate box prompts. Markers indicate compression type: ○ attention-only and • joint attention–MLP.

Method	SAM-B mAP ↑	SAM-B Density ↓	SAM-B Speedup ↑	SAM-L mAP ↑	SAM-L Density ↓	SAM-L Speedup ↑	SAM-H mAP ↑	SAM-H Density ↓	SAM-H Speedup ↑
Base	46.8	100%	×1.00	49.5	100%	×1.00	49.9	100%	×1.00
○ SpargeAttn	44.7	25%	×1.25	46.1	25%	×1.23	–	25%	–
	46.3	50%	×1.24	49.1	50%	×1.21	–	50%	–
○ PISA	45.5	25%	×0.73	48.3	25%	×0.68	48.6	25%	×0.69
	46.5	50%	×0.72	49.4	50%	×0.67	49.8	50%	×0.65
○ MMInference	41.8	25%	×0.43	35.5	25%	×0.62	43.5	25%	×0.70
	46.3	50%	×0.42	46.5	50%	×0.59	49.4	50%	×0.63
○ GridAttention	46.8	25%	×1.27	49.3	25%	×1.11	49.8	25%	× 1.09
• ToMe	40.3	50%	×0.83	42.5	50%	×0.86	43.0	50%	×0.80
	46.7	75%	×0.74	49.4	75%	×0.75	49.8	75%	×0.64
• StructSAM	30.6	25%	×0.86	31.8	25%	×0.89	31.8	25%	×0.94
	42.2	50%	×0.71	44.5	50%	×0.74	45.1	50%	×0.78
○ SparseSAM (Ours)	**45.9**	25%	**×2.43**	**48.7**	25%	**×2.15**	**49.4**	25%	**×1.89**
	46.7	50%	**×2.26**	**49.4**	50%	**×1.81**	**49.9**	50%	**×1.80**
• SparseSAM (Ours)	**45.1**	25%	**×2.89**	**46.8**	25%	**×2.95**	**47.2**	25%	**×2.50**
	45.9	50%	**×2.62**	**48.2**	50%	**×2.53**	**48.7**	50%	**×2.24**

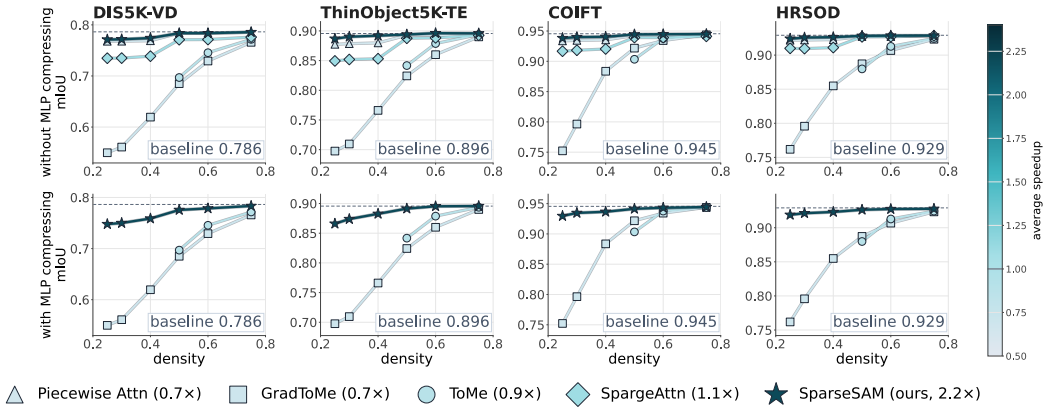

Figure 7: mIoU across density levels on four HQ-44K benchmarks. The top row reports attention-only compression and the bottom row reports joint attention–MLP compression. The plot's legacy labels "Piecewise Attn" and "GradToMe" correspond to PISA and StructSAM, respectively. Color indicates average speedup.

Attention Comparison. Table 2 shows that reducing attention computation does not guarantee lower latency. PISA (Li et al., 2026) retains near-dense accuracy at 50% density but runs slower than dense attention, suggesting that selection and execution overhead outweigh computational savings in SAM's small local windows. The two strongest baselines, SpargeAttn (Zhang et al., 2025a) and GridAttention (Pengyu et al., 2024), offer only modest speedups. At 25% density, attention-only SparseSAM achieves $2.43\times$, $2.15\times$, and $1.89\times$ speedups on SAM-B/L/H, with mAP drops of 0.9, 0.8, and 0.5 points, respectively. These gains come from reusing saliency rankings across blocks and compiling sparse traversal, which together reduce overhead while preserving broad image coverage.

Joint Attention and MLP Compression. At the same density, SparseSAM jointly compresses attention and MLP layers to achieve speedups of $2.89\times$, $2.95\times$, and $2.50\times$ on SAM-B/L/H, with mAP drops of 1.7, 2.7, and 2.7 points, respectively. Compared with token-merging baselines ToMe (Bolya et al., 2022) and StructSAM (Nguyen et al., 2026), SparseSAM delivers substantially higher speedups while preserving better quality. These gains stem from avoiding merge-unmerge overhead and preserving contrast between regions that token merging can otherwise blur.

4.1.2 High-Quality Segmentation

To stress-test compression robustness, we evaluate SparseSAM on HQ-44K (Ke et al., 2023), which includes DIS5K-VD (Qin et al., 2022), ThinObject5K-TE (Liew et al., 2021), COIFT (Mansilla et al., 2016), and HRSOD (Zeng et al., 2019). Fig. 7 shows that SparseSAM stays closest to dense

Table 3: Comparison with efficient SAM baselines. SparseSAM uses joint attention–MLP compression. Bold and underlined values indicate the best and second-best results per column, respectively.

Method	HQ-44K								MS-COCO (avg.)	
	DIS5K-VD		COIFT		HRSOD		ThinObject5K-TE		DINO / H-DETR / YOLOX	
	mIoU ↑	Speedup ↑	mIoU ↑	Speedup ↑	mIoU ↑	Speedup ↑	mIoU ↑	Speedup ↑	mAP ↑	Speedup ↑
SAM-B (baseline)	76.0	1.00×	93.0	1.00×	91.0	1.00×	84.0	1.00×	42.0	1.00×
FastSAM	25.0	2.37×	60.0	2.39×	66.0	2.39×	52.0	2.32×	32.3	**3.46**×
MobileSAM	61.0	<u>2.71×</u>	<u>91.0</u>	**3.15**×	86.0	**3.26**×	56.0	**3.12**×	39.0	<u>2.95×</u>
EfficientSAM	60.0	1.91×	90.0	1.91×	<u>87.0</u>	1.92×	54.0	1.92×	39.7	2.52×
EfficientViT-SAM (XL1)	<u>63.0</u>	1.68×	**93.0**	1.68×	**90.0**	1.70×	<u>78.0</u>	1.64×	**44.3**	1.83×
SparseSAM (ours, joint, 25%)	**72.0**	**2.78**×	<u>91.0</u>	<u>2.86×</u>	**90.0**	<u>2.80×</u>	**81.0**	<u>2.82×</u>	<u>40.7</u>	2.88×

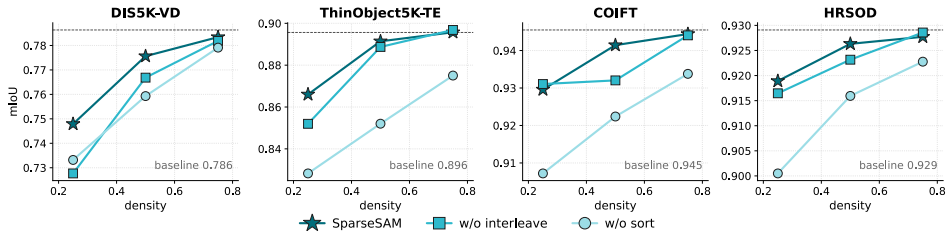

Figure 8: Performance breakdown with different permutation orders.

accuracy across density settings under both attention-only and joint compression, indicating greater robustness as compression increases. Table 7 further shows that joint attention-MLP compression delivers substantial acceleration with limited quality loss, whereas token mergering baselines incurs a much larger accuracy drop and runs slower than the dense baseline at the same density. These results highlight the value of preserving spatial token identities while reducing computation, enabling training-free acceleration without sacrificing the distinctions needed for detailed segmentation. Although SparseSAM requires no retraining, we also include recovery results after 15 min fine-tuning in Appendix A.8.

4.1.3 Comparison with Efficient SAM Baselines

Table 3 shows that SparseSAM offers competitive speed-accuracy trade-offs with lightweight SAM backbones while remaining entirely training-free. On ThinObject5K-TE, SparseSAM preserves substantially higher segmentation quality than MobileSAM (Zhang et al., 2023), FastSAM (Zhao et al., 2023), and EfficientSAM (Xiong et al., 2024) at a comparable speedup. On MS-COCO, it achieves greater acceleration than EfficientViT-SAM (Zhang et al., 2024), with some loss in accuracy. These results demonstrate that substantial acceleration can be achieved directly on pretrained SAM models, without the training or distillation needed to develop a lightweight replacement.

4.2 Ablation Studies

Table 4: Saliency selectors's average performance on HQ-44K under joint attention-MLP compression at density 30%.

Selector type	Spatial derivatives						Feature statistics		Attention	Random
Estimator	Sobel-3	Sobel-5	Sobel-7	Sobel-9	Scharr-3	Laplacian-3	Variance dissimilarity	Feature norm	Attention score	Uniform sampling
HQ-44K mIoU	88.32	88.19	88.06	87.96	88.39	87.95	88.32	86.67	84.80	85.25

Saliency operators. Table 4 compares selectors under joint compression on HQ-44K. Sobel-k uses a $k \times k$ kernel; its variants differ by only 0.36 mIoU points. Sobel-3 and variance dissimilarity achieve 88.32, close to Scharr-3 (88.39) and above feature norm (86.67) and attention score (84.80). These results support spatial variation as a practical approach motivated by Lemma 1 and Proposition 2, without establishing optimal routing.

Permutation ablations. Our token permutation combines (i) *saliency ordering*, which prioritizes high-importance Morton groups, and (ii) *Stripe-Sort interleaving*, which distributes tokens from

ranked groups across the reordered sequence. We evaluate two ablations on SAM-HQ ViT-L over HQ-44K: *w/o interleave*, which removes interleaving while preserving the saliency ranking, and *w/o sort*, which preserves interleaving but removes content-aware ranking. As shown in Fig. 8, all variants perform similarly at mild compression ($r = 0.75$), but differences become significant at lower densities. At $r = 0.25$, removing interleaving causes an absolute mIoU decrease of up to 2.0 points on DIS5K-VD. Removing ranking causes larger absolute decreases, including 3.8 points on ThinObject5K-TE and 2.2 points on COIFT, indicating that content-aware prioritization is the primary factor under aggressive compression.

Generalization to other segmentation backbones. We evaluate the Residual-Consistency MLP on SAM 3 (Carion et al., 2026) and Segmenter (Strudel et al., 2021) by applying the same token-routing mechanism within their MLP layers. Table 5 supports its applicability across these segmentation backbones. We further examine a contrastively trained backbone on ImageNet (Deng et al., 2009); see Appendix A.12 and Appendix A.13.

Model	Density	Latency (ms)	Speedup	mIoU
Segmenter ViT-L	1.0	35.09	1.00×	50.81
	0.7	23.16	1.52×	51.61
	0.5	20.47	1.72×	52.14
	0.3	17.66	1.99×	45.38
SAM 3	1.0	52.15	1.00×	70.36
	0.7	40.85	1.28×	71.17
	0.5	36.46	1.43×	67.58
	0.3	32.67	1.60×	62.51

Table 5: Residual-Consistency MLP on Segmenter and SAM 3 encoders on ADE20K.

5 Related Work

Spatially Structured Attention. GG-Transformer (Yu et al., 2021) and MaxViT (Tu et al., 2022) capture long-range interactions through spatially distributed views. Grid-Attention (Pengyu et al., 2024) applies uniformly subsampled views to pretrained models without fine-tuning but leaves MLPs unchanged. SparseSAM interleaves saliency-ranked Morton groups and shares their ranking between attention and MLP sparsification.

Sparse-Attention Approaches. SpargeAttn (Zhang et al., 2025a), PISA (Li et al., 2026), Sparse VideoGen (Xi et al., 2025), Sparse VideoGen2 (Yang et al., 2025), SCFA (Pagliardini et al., 2023), PBS-Attn (Wang et al., 2025), and MMInference (Li et al., 2025) reduce interactions through filtering, approximation, or reordering. Runtime selection and indexing can offset savings in SAM's short attention windows. MInference (Jiang et al., 2024) selects head patterns offline, combining static A-shaped masks with dynamically indexed patterns. SparseSAM instead shares an A-shaped pattern across heads, with active tiles fixed for each input shape and density and traversal compiled into the kernel. Adaptivity comes from a per-image saliency permutation reused across blocks, eliminating runtime mask construction and dynamic block selection.

Activation Compression Approaches. Weight-activation quantization (Lv et al., 2024; Wang et al., 2024; Ranjan and Savakis, 2025; Zhang et al., 2025b) reduces precision and memory use, but conversion and kernel overhead can limit speedups for small operations. Token merging (Bolya et al., 2022; Tran et al., 2024; 2025) shortens sequences at the cost of matching, merging, and spatial restoration (Nguyen et al., 2026); aggressive merging can also erase distinctions needed for fine boundaries. SparseSAM cheaply reduces attention and MLP computation while preserving token identities and spatial resolution with minimal runtime overhead.

Selective MLP Computation. Mixture-of-Depths (Raposo et al., 2024) learns per-layer token routers, whereas FFN-SkipLLM (Jaiswal et al., 2024) skips feed-forward computation during autoregressive decoding without retraining. Skip-Vision (Zeng et al., 2025) incorporates visual-token FFN skipping into vision-language training and removes selected tokens from the decoding KV cache. SparseSAM reuses its attention saliency ranking to select MLP updates, passing unselected tokens through the residual pathway. It requires no learned routers or retraining and retains tokens for dense prediction.

6 Conclusion

We introduce SparseSAM, built on the insight that efficient SAM inference does not require discarding tokens, but rather allocating computation selectively while preserving spatial structure. A shared image-dependent ranking coordinates two complementary mechanisms: Stripe-Sort Attention maintains broad spatial interactions under structured sparsity, while Residual-Consistency MLP updates only informative tokens and carries the remainder through the residual path. This coordinated design enables efficient computation while retaining the dense representations essential for segmentation.

AI USE STATEMENT

Generative AI tools were used for language editing, proofreading, drafting proof sketches, and assisting with implementation. The original research ideas, observations, method, empirical and theoretical verification/derivation were developed by the authors. All AI-assisted contributions were reviewed and verified by the authors, who take responsibility for the final content of this work.

ETHICS STATEMENT

This work develops a training-free method for improving the inference efficiency of pretrained segmentation models and does not involve human subjects or new sensitive data. We use existing public datasets and pretrained models under their respective licenses. While greater efficiency may broaden access to segmentation models, it may also facilitate downstream uses such as large-scale visual monitoring. Users should therefore consider applicable privacy, legal, licensing, and ethical requirements when deploying the method.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3 describes our token ordering, sparse attention, and MLP routing. Pseudocodes for all proposed algorithms are provided in Appendices A.5 and A.6, and the implementation is available in our anonymous repository. The theoretical assumptions are stated alongside the results, with complete proofs and evidence in Appendices A.1, A.2, and A.3. Section 4 documents the experimental datasets, baselines, hardware, and evaluation procedures.

REFERENCES

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.

Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Bangalath, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Shang-Wen Li, Piotr Dollar, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 60884–60937. Curran Associates, Inc., 2025. doi: 10.52202/085713-2036. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/57bc0a850255e2041341bf74c7e2b9fa-Paper-Conference.pdf.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. In *International conference on learning representations*, volume 2026, pages 138846–138923, 2026.

Zigeng Chen, Gongfan Fang, Xinyin Ma, and Xinchao Wang. Slimsam: 0.1% data makes segment anything slim. *Advances in Neural Information Processing Systems*, 37:39434–39461, 2024.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pages 35549–35562, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Yuzhe Duan, Xuanxuan Ren, Guizhe Dong, Xu Yang, and Yanhua Yang. Mix-qsam2: Mixed-precision quantization for high fidelity segmentation in resource constrained scenarios. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 3741–3749, 2026.

Mark Endo, Xiaohan Wang, and Serena Yeung-Levy. Feather the throttle: Revisiting visual token pruning for vision-language model acceleration. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22826–22835. IEEE, 2025.

Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

Ajay Kumar Jaiswal, Bodun Hu, Lu Yin, Yeonju Ro, Tianlong Chen, Shiwei Liu, and Aditya Akella. Ffn-skipllm: A hidden gem for autoregressive decoding with adaptive feed forward skipping. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16943–16956, 2024.

Ding Jia, Yuhui Yuan, Haodi He, Xiaopei Wu, Haojun Yu, Weihong Lin, Lei Sun, Chao Zhang, and Han Hu. Detrs with hybrid matching. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19702–19712. IEEE, 2023.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

Lei Ke, Mingqiao Ye, Martin Danelljan, Yifan liu, Yu-Wing Tai, Chi-Keung Tang, and Fisher Yu. Segment anything in high quality. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 29914–29934. Curran Associates, Inc., 2023. doi: 10.52202/075280-1303. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5f828e38160f31935cfe9f67503ad17c-Paper-Conference.pdf.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *2023 IEEE/CVF international conference on computer vision (ICCV)*, pages 3992–4003. IEEE, 2023.

Haopeng Li, Shitong Shao, Wenliang Zhong, Zikai Zhou, Lichen Bai, Hui Xiong, and Zeke Xie. Pisa: Piecewise sparse attention is wiser for efficient diffusion transformers. *arXiv preprint arXiv:2602.01077*, 2026.

Yucheng Li, Huiqiang Jiang, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Amir H Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, et al. Mminference: Accelerating pre-filling for long-context vlms via modality-aware permutation sparse attention. *arXiv preprint arXiv:2504.16083*, 2025.

Jun Hao Liew, Scott Cohen, Brian Price, Long Mai, and Jiashi Feng. Deep interactive thin object selection. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 305–314. IEEE, 2021.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Yifei Liu, Mathias Gehrig, Nico Messikommer, Marco Cannici, and Davide Scaramuzza. Revisiting token pruning for object detection and instance segmentation. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2646–2656. IEEE, 2024.

Zhihe Lu, Zeyu Xiao, Jiawang Bai, Zhiwei Xiong, and Xinchao Wang. Can sam boost video super-resolution? *arXiv preprint arXiv:2305.06524*, 2023.

Chengtao Lv, Hong Chen, Jinyang Guo, Yifu Ding, and Xianglong Liu. Ptq4sam: Post-training quantization for segment anything. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15941–15951. IEEE, 2024.

Lucy AC Mansilla, Paulo AV Miranda, and Fábio AM Cappabianco. Oriented image foresting transform segmentation with connectivity constraints. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2554–2558. IEEE, 2016.

Duy MH Nguyen, Tuan A Tran, Duong Nguyen, Siwei Xie, Trung Q Nguyen, Mai TN Truong, Daniel Palenicek, An T Le, Michael Barz, TrungTin Nguyen, et al. Structsam: Structure-and spectrum-preserving token merging for segment anything models. *arXiv preprint arXiv:2603.07307*, 2026.

Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. Fast attention over long sequences with dynamic sparse flash attention. *Advances in Neural Information Processing Systems*, 36:59808–59831, 2023.

Li Pengyu, Guo Tianchu, Wang Biao, and Hua Xian-Sheng. Grid-attention: Enhancing computational efficiency of large vision models without fine-tuning. In *European Conference on Computer Vision*, pages 54–70. Springer, 2024.

Xuebin Qin, Hang Dai, Xiaobin Hu, Deng-Ping Fan, Ling Shao, and Luc Van Gool. Highly accurate dichotomous image segmentation. In *European conference on computer vision*, pages 38–56. Springer, 2022.

Navin Ranjan and Andreas Savakis. Mix-qsam: Mixed-precision quantization of the segment anything model. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 3271–3281. IEEE, 2025.

David Raposo, Sam Ritter, Blake Richards, Timothy Lillicrap, Peter Conway Humphreys, and Adam Santoro. Mixture-of-depths: Dynamically allocating compute in transformer-based language models. *arXiv preprint arXiv:2404.02258*, 2024.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, volume 2025, pages 28085–28128, 2025.

Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7242–7252. IEEE, 2021.

Hoai-Chau Tran, Duy M. H. Nguyen, Duy M Nguyen, TrungTin Nguyen, Ngan Le, Pengtao Xie, Daniel Sonntag, James Zou, Binh T Nguyen, and Mathias Niepert. Accelerating transformers with spectrum-preserving token merging. *Advances in Neural Information Processing Systems*, 37: 30772–30810, 2024.

Tuan Anh Tran, Duy M. H. Nguyen, Hoai-Chau Tran, Michael Barz, Khoa D. Doan, Roger Wattenhofer, Vien Anh Ngo, Mathias Niepert, Daniel Sonntag, and Paul Swoboda. How many tokens do 3d point cloud transformer architectures really need? In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 13464–13496. Curran Associates, Inc., 2025. doi: 10.52202/085713-0454. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/1418afb35f5fa69b4f7af27b6480101b-Paper-Conference.pdf.

Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *European conference on computer vision*, pages 459–479. Springer, 2022.

Changyuan Wang, Ziwei Wang, Xiuwei Xu, Yansong Tang, Jie Zhou, and Jiwen Lu. Q-vlm: Post-training quantization for large vision-language models. *Advances in Neural Information Processing Systems*, 37:114553–114573, 2024.

Xinghao Wang, Pengyu Wang, Dong Zhang, Chenkun Tan, Shaojun Zhou, Zhaoxiang Liu, Shiguo Lian, Fangxu Liu, Kai Song, and Xipeng Qiu. Sparser block-sparse attention via token permutation. *arXiv preprint arXiv:2510.21270*, 2025.

Houji Wen, Jiangyong Yu, Dawei Yang, and Jun Li. Car-sam: Cross-attention reconstruction for post-training quantization of the segment anything model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 33632–33641, 2026.

Haocheng Xi, Shuo Yang, Yilong Zhao, Chenfeng Xu, Muyang Li, Xiuyu Li, Yujun Lin, Han Cai, Jintao Zhang, Dacheng Li, et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

Yunyang Xiong, Bala Varadarajan, Lemeng Wu, Xiaoyu Xiang, Fanyi Xiao, Chenchen Zhu, Xiaoliang Dai, Dilin Wang, Fei Sun, Forrest Iandola, et al. Efficientsam: Leveraged masked image pretraining for efficient segment anything. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16111–16121. IEEE, 2024.

Shuo Yang, Haocheng Xi, Yilong Zhao, Muyang Li, Jintao Zhang, Han Cai, Yujun Lin, Xiuyu Li, Chenfeng Xu, Kelly Peng, Jianfei Chen, Song Han, Kurt Keutzer, and Ion Stoica. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 96965–96991. Curran Associates, Inc., 2025. doi: 10.52202/085713-3243. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/8bd5148caced2d73cea7b6961a874a49-Paper-Conference.pdf.

Qihang Yu, Yingda Xia, Yutong Bai, Yongyi Lu, Alan L Yuille, and Wei Shen. Glance-and-gaze vision transformer. *Advances in Neural Information Processing Systems*, 34:12992–13003, 2021.

Weili Zeng, Ziyuan Huang, Kaixiang Ji, and Yichao Yan. Skip-vision: Efficient and scalable acceleration of vision-language models via adaptive token skipping. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21384–21397. IEEE, 2025.

Yi Zeng, Pingping Zhang, Jianming Zhang, Zhe Lin, and Huchuan Lu. Towards high-resolution salient object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7234–7243, 2019.

Chaoning Zhang, Dongshen Han, Yu Qiao, Jung Uk Kim, Sung-Ho Bae, Seungkyu Lee, and Choong Seon Hong. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.

Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

Jing Zhang, Zhikai Li, Chengzhi Hu, Xuewen Liu, and Qingyi Gu. Saq-sam: Semantically-aligned quantization for segment anything model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 12547–12555, 2026.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025a.

Wenlun Zhang, Yunshan Zhong, Shimpei Ando, and Kentaro Yoshioka. Ahcptq: Accurate and hardware-compatible post-training quantization for segment anything model. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22383–22392. IEEE, 2025b.

Zhuoyang Zhang, Han Cai, and Song Han. Efficientvit-sam: Accelerated segment anything model without performance loss. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 7859–7863. IEEE, 2024.

Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023.

Chong Zhou, Xiangtai Li, Chen Change Loy, and Bo Dai. Edgesam: Prompt-in-the-loop distillation for sam. *International Journal of Computer Vision*, 133(12):8452–8468, 2025.

Wenhui Zhu, Xiwen Chen, Zhipeng Wang, Shao Tang, Sayan Ghosh, Xuanzhao Dong, Rajat Koner, and Yalin Wang. Evtp-ivs: Effective visual token pruning for unifying instruction visual segmentation in multi-modal large language models. In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 7158–7167. IEEE, 2026.

A Appendix

A.1 Proof of Proposition 1: Attention Approximation

We compare dense and compressed sublayers at the same input in exact arithmetic, with randomness disabled. Attention retains the original logits and restores original token positions. We first prove Lemma 1, which supplies the group-average bound used in Proposition 1.

Proof of Lemma 1. The key idea is that removing keys changes the group average according to two quantities: how much attention weight those keys carry, and how different their values are from the retained values. We first bound the fraction of attention weight assigned to omitted keys. We then show that the error is at most this fraction multiplied by d_g, the largest distance between two value vectors in the group.

Fix one query i and one attention head. The group $\mathcal{G}_g$ contains G keys, of which q_g belong to the retained set S_g. If $q_g = G$, no keys are omitted and the two averages coincide. Below, assume $1 \le q_g < G$.

Each key j has unnormalized attention weight $e^{z_{ij}}$. Define the two weight sums

$$R_g = \sum_{j \in S_g} e^{z_{ij}}, \qquad O_g = \sum_{j \in \mathcal{G}_g \setminus S_g} e^{z_{ij}}.$$

Let $z_g^- = \min_{j \in \mathcal{G}_g} z_{ij}$ and $z_g^+ = \max_{j \in \mathcal{G}_g} z_{ij}$, so $\alpha_g = z_g^+ - z_g^-$. Every key's weight lies between $e^{z_g^-}$ and $e^{z_g^+}$. There are q_g retained keys and $G - q_g$ omitted keys, hence

$$q_g e^{z_g^-} \le R_g \le q_g e^{z_g^+}, \qquad (G - q_g) e^{z_g^-} \le O_g \le (G - q_g) e^{z_g^+}.$$

Dividing the lower bound on O_g by the upper bound on R_g, and vice versa, gives

$$\frac{G - q_g}{q_g} e^{-\alpha_g} \le \frac{O_g}{R_g} \le \frac{G - q_g}{q_g} e^{\alpha_g}.$$

To express this ratio as a fraction of the full group's attention weight, normalize the omitted weight by the total weight:

$$\eta_g = \frac{O_g}{R_g + O_g} = \frac{O_g / R_g}{1 + O_g / R_g}.$$

Since $x/(1+x)$ increases with $x \ge 0$, the upper bound on O_g/R_g implies

$$\eta_g \le \frac{\frac{G - q_g}{q_g} e^{\alpha_g}}{1 + \frac{G - q_g}{q_g} e^{\alpha_g}} = \frac{(G - q_g) e^{\alpha_g}}{q_g + (G - q_g) e^{\alpha_g}}.$$

To relate this omitted mass to the approximation error, define the attention-weighted average of the omitted values

$$\boldsymbol{\nu}_g = \frac{\sum_{j \in \mathcal{G}_g \setminus S_g} e^{z_{ij}} \mathbf{v}_j}{O_g}.$$

The dense group average combines the retained and omitted averages with masses $1 - \eta_g$ and η_g:

$$\boldsymbol{\mu}_g = (1 - \eta_g) \widetilde{\boldsymbol{\mu}}_g + \eta_g \boldsymbol{\nu}_g, \qquad \boldsymbol{\mu}_g - \widetilde{\boldsymbol{\mu}}_g = \eta_g (\boldsymbol{\nu}_g - \widetilde{\boldsymbol{\mu}}_g).$$

Both $\boldsymbol{\nu}_g$ and $\widetilde{\boldsymbol{\mu}}_g$ are convex combinations of values in the same group. Their difference is a convex combination of pairwise value differences, whose norms are at most d_g. Therefore

$$\|\boldsymbol{\mu}_g - \widetilde{\boldsymbol{\mu}}_g\|_2 \le \eta_g d_g \le \frac{(G - q_g) e^{\alpha_g}}{q_g + (G - q_g) e^{\alpha_g}} d_g.$$

Thus, the error bound is the omitted attention fraction multiplied by the largest within-group value distance; reducing either quantity tightens the bound when the other is held fixed. □

The partial-stripe observation follows directly from this bound. Write its right-hand side as $B_g(q)$ and set $a_g = e^{\alpha_g}$. For $1 \le q < G$, subtracting the bounds for q and $q+1$ retained tokens gives

$$B_g(q) - B_g(q+1) = \frac{G a_g d_g}{[q + (G-q)a_g][q + 1 + (G-q-1)a_g]}.$$

For groups of the same size G, this reduction is proportional to d_g when both q and α_g are the same. Here, equal coverage means the groups currently retain the same number q of keys; equal logit variation means their maximum-minus-minimum logits are equal for the query and head being considered. Thus, under these conditions, a group with twice the value diameter receives twice the reduction in its bound from one extra key. After $1 \le r < G$ complete stripes, every group has $q = r$, so the coverage condition holds before allocating the partial stripe. Logit ranges can still differ across groups; in that case, the reduction depends on both d_g and α_g, as the expression above shows.

Proof of Proposition 1. Each complete stripe retains one key from every group, so r stripes give the common fraction $f = r/G$. Let $Z_g = R_g + O_g$, $\widetilde{Z}_g = R_g$, and define the dense and sparse group weights

$$p_g = \frac{Z_g}{\sum_h Z_h}, \qquad \widetilde{p}_g = \frac{\widetilde{Z}_g}{\sum_h \widetilde{Z}_h}.$$

The outputs are $\mathbf{o} = \sum_g p_g \boldsymbol{\mu}_g$ and $\widetilde{\mathbf{o}} = \sum_g \widetilde{p}_g \widetilde{\boldsymbol{\mu}}_g$. Adding and subtracting $\sum_g p_g \widetilde{\boldsymbol{\mu}}_g$, then using $\|\widetilde{\boldsymbol{\mu}}_g\|_2 \le V$, yields

$$\|\mathbf{o} - \widetilde{\mathbf{o}}\|_2 \le \sum_g p_g \|\boldsymbol{\mu}_g - \widetilde{\boldsymbol{\mu}}_g\|_2 + V \sum_g |p_g - \widetilde{p}_g|. \tag{6}$$

For the first term, Lemma 1 with $q_g = r$, $\alpha_g \le \alpha$, and $d_g \le d_v$ gives

$$\sum_g p_g \|\boldsymbol{\mu}_g - \widetilde{\boldsymbol{\mu}}_g\|_2 \le \frac{(1-f)e^{\alpha}}{f + (1-f)e^{\alpha}} d_v \le (1-f)e^{\alpha} d_v,$$

where the last denominator is at least one. For the second term, the weight-ratio bounds in the lemma's proof imply

$$a := \frac{f}{f + (1-f)e^{\alpha}} \le t_g := \frac{\widetilde{Z}_g}{Z_g} \le b := \frac{f}{f + (1-f)e^{-\alpha}}.$$

Since $\widetilde{p}_g / p_g = t_g / \bar{t}$ with $\bar{t} = \sum_h p_h t_h \in [a, b]$, this ratio lies in $[a/b, b/a]$. Its distance from one is at most $b/a - 1$, so

$$\begin{aligned} \sum_g |p_g - \widetilde{p}_g| &= \sum_g p_g \left| 1 - \frac{\widetilde{p}_g}{p_g} \right| \\ &\le \frac{b}{a} - 1 = \frac{(1-f)(e^{\alpha} - e^{-\alpha})}{f + (1-f)e^{-\alpha}} \le (1-f)(e^{2\alpha} - 1). \end{aligned}$$

The final inequality uses $f + (1-f)e^{-\alpha} \ge e^{-\alpha}$. Substituting both terms into Eq. 6 gives

$$\|\mathbf{o} - \widetilde{\mathbf{o}}\|_2 \le (1-f)\left[e^{\alpha} d_v + V(e^{2\alpha} - 1)\right],$$

as required. At $f = 1$, all keys are retained and the error is zero. □

What the attention bound guarantees. Proposition 1 compares dense attention with attention that retains r complete stripes and no extra diagonal keys. It gives three conclusions.

- First, the approximation error is small when keys within each group have similar logits and similar value vectors, provided the value norms are bounded. Groups may still differ from one another; the bound depends on variation within each group.
- Second, keeping more complete stripes lowers the error bound. For the same input, the quantity $C = e^{\alpha} d_v + V(e^{2\alpha} - 1)$ is fixed, so the bound $(1 - r/G)C$ decreases as r increases. This lowers the maximum error allowed by the bound, although the actual error need not decrease after every addition.

- Third, the sparse and dense outputs are exactly equal in two cases: when all keys are retained ($r = G$), or when every group contains identical logits and identical value vectors ($\alpha = d_v = 0$). In the second case, one complete stripe is enough. Keeping one key per group preserves each group's value average and retains the same fraction $1/G$ of every group's total attention weight. Normalization therefore preserves the relative weights of the groups, giving the same final output as dense attention.

Implications for partial stripes and Sobel ranking. After r complete stripes, every group has r retained keys. If the remaining budget cannot fit another complete stripe, only some groups receive an extra key. Our ordering gives these extra keys to the groups with the highest Sobel scores. Lemma 1 motivates this choice. Consider two groups that currently retain the same number of keys and have the same logit range for a given query and head. Adding one key to each group reduces the group-average error bound more for the group whose value vectors differ more from one another. Sobel scores measure spatial feature changes, so we use them to identify groups that may contain greater value variation. This is the reason for placing higher-scoring groups earlier within each stripe. The result concerns the error bound within a group. The final attention output also depends on how much weight each group receives, and retaining an extra key can change those weights. Thus, the lemma motivates the ranking but does not prove that adding a high-saliency key always reduces the final attention error. We evaluate the ranking empirically in Table 4.

A.2 Proof of Proposition 2: Optimal Residual Routing

We compare dense and routed residual MLP outputs at the same input in exact arithmetic, with randomness disabled and token-wise LayerNorm and MLP operations.

Index the updates in descending order of magnitude, $\|\Delta_{(1)}\|_2 \geq \cdots \geq \|\Delta_{(N)}\|_2$. Then Proposition 2 also gives

$$\min_{|\mathcal{K}|=K} \left\|\mathbf{Y} - \widetilde{\mathbf{Y}}^{(\mathcal{K})}\right\|_F^2 = \sum_{j=K+1}^{N} \|\Delta_{(j)}\|_2^2. \tag{7}$$

Conversely, retaining the K smallest magnitudes maximizes the error.

Proof of Proposition 2. Let $m_i = \mathbf{1}\{i \in \mathcal{K}\}$. Token-wise computation leaves retained updates unchanged, so

$$\mathbf{y}_i - \widetilde{\mathbf{y}}_i^{(\mathcal{K})} = (\mathbf{x}_i + \Delta_i) - (\mathbf{x}_i + m_i\Delta_i) = (1 - m_i)\Delta_i.$$

Summing squared row errors gives

$$\left\|\mathbf{Y} - \widetilde{\mathbf{Y}}^{(\mathcal{K})}\right\|_F^2 = \sum_{i \notin \mathcal{K}} \|\Delta_i\|_2^2 = \sum_{i=1}^{N} \|\Delta_i\|_2^2 - \sum_{i \in \mathcal{K}} \|\Delta_i\|_2^2,$$

which proves Eq. 5. The total energy is fixed, so minimizing error is equivalent to maximizing retained energy. If a retained token a has $\|\Delta_a\|_2 < \|\Delta_b\|_2$ for an omitted token b, swapping them decreases squared error by $\|\Delta_b\|_2^2 - \|\Delta_a\|_2^2$. Repeating such exchanges yields a keep set of the K largest magnitudes, leaving the tail sum in Eq. 7. Reversing the exchanges shows that retaining the K smallest magnitudes maximizes error. Ties can give multiple optimal sets; if all updates vanish, every set has zero error. The optimum concerns a shared-input sublayer error; the cached group ranking is a proxy and need not match the per-layer, individual-token optimum.

□

A.3 Empirical Evidence: High Saliency Prioritization

Figure 9 evaluates the within-group approximation in Lemma 1 at 30% attention density using two oracle orderings: descending and ascending value dissimilarity d_g. For each query, we average the measured approximation errors and their bounds across the M groups:

$$\overline{E} = \frac{1}{M}\sum_{g=1}^{M} \|\boldsymbol{\mu}_g - \widetilde{\boldsymbol{\mu}}_g\|_2, \qquad \overline{B} = \frac{1}{M}\sum_{g=1}^{M} B_g(q_g),$$

where $B_g(q_g)$ denotes the right-hand side of Eq. 2. Averaging the lemma's groupwise inequalities yields $\overline{E} \le \overline{B}$.

The measured errors remain below their corresponding bounds, consistent with the lemma. Prioritizing groups with larger d_g generally lowers the errors and tightens the bound. The inset further shows that $\overline{B}_{\text{high}}/\overline{B}_{\text{low}}$ ranges from approximately 0.76 to 0.89, indicating a tighter bound when high-dissimilarity groups are prioritized. By using value dissimilarity directly, this comparison supports the lemma's rationale for allocating more representatives to groups with greater value variation.

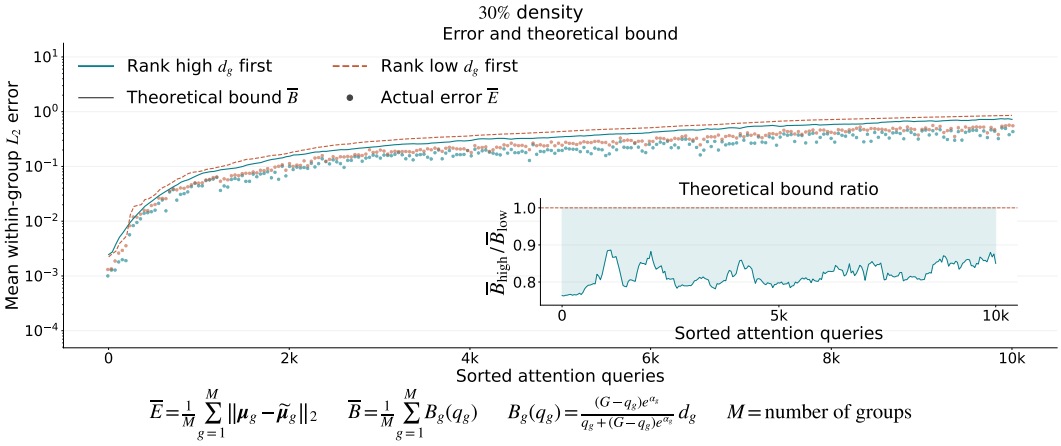

Figure 9: Attention error comparison at 30% density. The horizontal axis represents the attention queries after sorting based on its attention error magnitude, covering about 10,000 different queries on the SAM-L model during inference. Each query has a mean within-group error $\overline{E}$ and bound $\overline{B}$, averaged across its groups. Curves show the mean theoretical bound from Lemma 1, and points show the mean measured group error, for high-dissimilarity-first (teal) and low-dissimilarity-first (orange) orderings. The inset compares their theoretical bounds.

A.4 Empirical Analysis of Proposition 2

Proposition 2 shows that, for a fixed number of retained tokens, selecting the largest MLP updates minimizes the local representation error. This provides a criterion for evaluating the cached Sobel ranking: does the initial ranking continue to select tokens whose MLP updates matter in later blocks? We examine this question by comparing selection strategies at several network depths.

Selection strategies. Figure 10 compares four strategies at 30% token coverage. Cached Sobel reuses the initial Sobel group ranking, while refreshed Sobel recomputes the ranking at the evaluated block. Random selection provides an expected-error reference at the same token budget. The oracle retains tokens with the largest MLP update norms, giving the minimum error characterized by Proposition 2. This oracle is a diagnostic reference, since obtaining the exact update norms requires evaluating the dense MLP.

Representation error. At block ℓ, let $\mathbf{Y}_\ell$ be the dense residual MLP output and $\widetilde{\mathbf{Y}}_\ell^{(\mathcal{K})}$ the routed output for keep set $\mathcal{K}$, evaluated at the same input. Proposition 2 gives

$$\mathcal{E}_\ell(\mathcal{K}) = \left\|\mathbf{Y}_\ell - \widetilde{\mathbf{Y}}_\ell^{(\mathcal{K})}\right\|_F^2 = \sum_{i \notin \mathcal{K}} \|\Delta_{\ell,i}\|_2^2, \tag{8}$$

where $\Delta_{\ell,i}$ is the dense MLP update of token i. We report the gap to the oracle,

$$G_\ell(\mathcal{K}) = \mathcal{E}_\ell(\mathcal{K}) - \mathcal{E}_{\ell,\mathrm{opt}}, \qquad \mathcal{E}_{\ell,\mathrm{opt}} = \min_{|\mathcal{S}|=K} \mathcal{E}_\ell(\mathcal{S}). \tag{9}$$

A smaller gap indicates that the selected tokens better approximate the optimal routing decision.

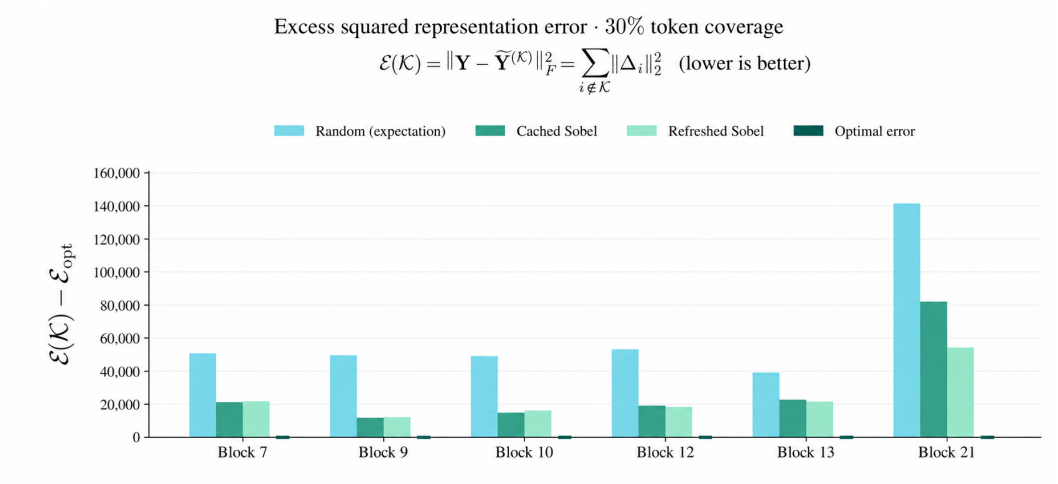

Figure 10: Empirical analysis of Proposition 2 at 30% token coverage, cached Sobel outperforms random selection at every displayed block and performs similarly to refreshed Sobel at the displayed blocks from 7 to 13. Refreshing gives a larger improvement at block 21. The oracle has zero excess error by definition.

Results. Cached Sobel yields lower excess representation error than random selection at all six evaluated blocks. At different blocks, its error is close to that of refreshed Sobel, suggesting that the initial ranking remains useful for identifying consequential MLP updates at these depths. At block 21, refreshing the ranking gives a clearer improvement, although cached selection still performs better than random. These observations connect the routing criterion in Proposition 2 to the empirical effectiveness of cached selection. The proposition characterizes error at a shared block input.

A.5 ATTENTION KERNEL PSEUDOCODE

Algorithm 1 Fused Stripe-Sort Attention with Decomposed 2D Rel-Pos Bias

Input: $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d}$; $\mathbf{B}^H, \mathbf{B}^W \in \mathbb{R}^{N \times w}$
// $N = w^2$: tokens; d: head dimension; $\mathbf{B}^H, \mathbf{B}^W$: precomputed spatial biases
σ: Stripe-Sort token order shared by queries, keys, and values
// $\sigma(t)$ is the original token index at reordered position t
Configuration: B: tokens per tile; $T = \lceil N/B \rceil$; $\tau = 1/\sqrt{d}$
$1 \le P \le T$: compiled prefix tile count; $\delta \in \{0, 1\}$: enable diagonal attention
Output: $\mathbf{O} \in \mathbb{R}^{N \times d}$ in original spatial order
for all query tiles $i \in \{0, \ldots, T-1\}$ **in parallel do**
  // Fuse Stripe-Sort permutation into query loads
  $\mathcal{I}_i \leftarrow \{iB, \ldots, \min((i+1)B, N) - 1\}$
  $\mathbf{q} \leftarrow \sigma[\mathcal{I}_i]$; $\mathbf{Q}_i \leftarrow \mathbf{Q}[\mathbf{q}, :]$
  // $\mathcal{I}_i$: reordered indices; $\mathbf{q}$: original query positions
  // $\mathcal{J}_i$: directly visited key tiles; no runtime mask buffer
  $\mathcal{J}_i \leftarrow \{0, \ldots, P-1\}$
  if $\delta = 1$ **and** $i \ge P$ **then**
    $\mathcal{J}_i \leftarrow \mathcal{J}_i \cup \{i\}$
  end if
  // Bias lookup preserves original spatial coordinates
  $\mathbf{B}_i^H \leftarrow \mathbf{B}^H[\mathbf{q}, :]$; $\mathbf{B}_i^W \leftarrow \mathbf{B}^W[\mathbf{q}, :]$
  // Row maxima, normalization sums, and output accumulator
  $\mathbf{m} \leftarrow -\boldsymbol{\infty}$; $\boldsymbol{\ell} \leftarrow \mathbf{0}$; $\mathbf{O}_{\text{acc}} \leftarrow \mathbf{0}$
  for $j \in \mathcal{J}_i$ **do**
    // Load keys and values using the same Stripe-Sort order
    $\mathcal{I}_j \leftarrow \{jB, \ldots, \min((j+1)B, N) - 1\}$
    $\mathbf{k} \leftarrow \sigma[\mathcal{I}_j]$; $\mathbf{K}_j \leftarrow \mathbf{K}[\mathbf{k}, :]$; $\mathbf{V}_j \leftarrow \mathbf{V}[\mathbf{k}, :]$
    // Indexed loads go directly into shared memory; no reordered global buffers
    $\mathbf{S} \leftarrow \mathbf{Q}_i \mathbf{K}_j^\top$
    // a, b: tile-local indices; k_b: original key position
    $\mathbf{S}[a, b] \leftarrow \mathbf{S}[a, b] + \big(\mathbf{B}_i^H[a, \lfloor k_b / w \rfloor] + \mathbf{B}_i^W[a, k_b \bmod w]\big)/\tau$
    // Online softmax; $\widetilde{\mathbf{P}}$: weights; $\boldsymbol{\alpha}$: rescaling factors
    $\mathbf{m}' \leftarrow \max(\mathbf{m}, \text{rowmax}(\mathbf{S}))$
    $\widetilde{\mathbf{P}} \leftarrow \exp\big(\tau(\mathbf{S} - \mathbf{m}')\big)$; $\boldsymbol{\alpha} \leftarrow \exp\big(\tau(\mathbf{m} - \mathbf{m}')\big)$
    $\boldsymbol{\ell} \leftarrow \boldsymbol{\alpha} \odot \boldsymbol{\ell} + \text{rowsum}(\widetilde{\mathbf{P}})$
    $\mathbf{O}_{\text{acc}} \leftarrow \text{diag}(\boldsymbol{\alpha})\mathbf{O}_{\text{acc}} + \widetilde{\mathbf{P}}\mathbf{V}_j$; $\mathbf{m} \leftarrow \mathbf{m}'$
  end for
  // Fuse unpermutation into stores using the same order σ
  $\mathbf{O}[\mathbf{q}, :] \leftarrow \text{diag}(\boldsymbol{\ell})^{-1}\mathbf{O}_{\text{acc}}$
end for

In SAM-based models, relative positional bias is added directly to the attention logits. We fuse this addition into a FlashAttention-2-style sparse attention kernel using the decomposed 2D bias $\mathbf{B}[q, k] = \mathbf{B}^H[q, k_{\text{row}}] + \mathbf{B}^W[q, k_{\text{col}}]$ on a $w \times w$ grid, where $w = \sqrt{S_k}$. Given $\mathbf{Q} \in \mathbb{R}^{S_q \times d}$ and $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{S_k \times d}$, the image-dependent permutations σ_Q and σ_K are fused into global-to-shared memory loads: query and key/value tiles are loaded directly from the original tensors without materializing reordered tensors. The precomputed bias tensors are indexed using the corresponding original spatial coordinates, preserving positional relationships after reordering.

Following FlashAttention-2 (Dao, 2024), the kernel uses online softmax to accumulate attention outputs across active tiles without materializing the full attention matrix. For query tile i, the A-shaped pattern restricts computation to key/value tiles in $\mathcal{J}_i = \{0, \ldots, P-1\} \cup \{i\}$, where P is the prefix tile count determined by the density setting. The prefix length and diagonal mode specialize the compiled kernel, allowing it to traverse the prefix tiles and, when not already included, the diagonal tile directly. This eliminates runtime mask-buffer construction and scanning of inactive

attention blocks while retaining boundary checks for partial tiles. For prefix-only local attention, the diagonal tile is omitted. The implementation uses aligned 128-bit asynchronous loads and `mma.sync.aligned.m16n8k16` Tensor Core operations with FP16 operands and FP32 accumulation. Prefix length, diagonal mode, and tile dimensions specialize the cached kernel; boundary checks and initialization predicates remain.

Permutation and inverse permutation are fused into the kernel's memory operations: $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ are loaded directly into shared memory in the order specified by σ, and each output row is written back to its original position. This eliminates intermediate reordered tensors and separate permutation kernels while preserving the original spatial layout of the output.

A.6 Residual-Consistency MLP Pseudocode

Algorithm 2 Fused Residual-Consistent Partial MLP

Input: $\mathbf{X} \in \mathbb{R}^{N\times d}$; ranked token order π; keep count K
// N: token count; d: feature dimension; K: determined by sparsity
$\mathbf{W}_1 \in \mathbb{R}^{d\times h}$, $\mathbf{W}_2 \in \mathbb{R}^{h\times d}$; biases $\mathbf{b}_1, \mathbf{b}_2$
$\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^d$; LayerNorm constant ϵ
// h: hidden dimension; π is the ranked order, not interleaved σ
Output: $\mathbf{O} \in \mathbb{R}^{N\times d}$ in original token order
// Kernel 1: fuse selected-token gather with LayerNorm
for all $i \in \{0, \ldots, K-1\}$ **in parallel do**
  $p \leftarrow \pi(i)$; $\mathbf{x} \leftarrow \mathbf{X}[p, :]$
  $\mu \leftarrow \mathrm{mean}(\mathbf{x})$; $v \leftarrow \mathrm{mean}((\mathbf{x}-\mu)^2)$
  $\mathbf{Z}[i, :] \leftarrow \boldsymbol{\gamma} \odot (\mathbf{x}-\mu)/\sqrt{v+\epsilon} + \boldsymbol{\beta}$
end for
// Kernel 2: fuse FC1 and GELU for the selected tokens
$\mathbf{H} \leftarrow \mathrm{GELU}(\mathbf{Z}\mathbf{W}_1 + \mathbf{b}_1)$
// $\mathbf{Z} \in \mathbb{R}^{K\times d}$ and $\mathbf{H} \in \mathbb{R}^{K\times h}$ are compact intermediate buffers
// Kernel 3: fuse FC2, residual addition, and scatter stores
for all selected output tiles **in parallel do**
  $\mathcal{I} \subseteq \{0, \ldots, K-1\}$: selected-row indices in the tile
  $\mathcal{C} \subseteq \{0, \ldots, d-1\}$: output-channel indices in the tile
  $\mathbf{p} \leftarrow \pi[\mathcal{I}]$
  $\boldsymbol{\Delta} \leftarrow \mathbf{H}[\mathcal{I}, :]\mathbf{W}_2[:, \mathcal{C}] + \mathbf{b}_2[\mathcal{C}]$
  $\mathbf{O}[\mathbf{p}, \mathcal{C}] \leftarrow \mathbf{X}[\mathbf{p}, \mathcal{C}] + \boldsymbol{\Delta}$
end for
// Copy in the same launch preserve unselected tokens
for all $i \in \{K, \ldots, N-1\}$ **in parallel do**
  $\mathbf{O}[\pi(i), :] \leftarrow \mathbf{X}[\pi(i), :]$
end for

We implement the Residual-Consistency MLP using three fused kernels. The ranked token order π is reused to select the first K tokens, where K is determined by the density setting. The first kernel fuses gathering with LayerNorm, loading each selected token directly from $\mathbf{X}[\pi(i), :]$ and writing its normalized features to a compact buffer. The second kernel fuses the first linear projection with GELU, processing only the selected tokens. The third kernel fuses the second linear projection, residual addition, and scattering. Each computed update is added to the original feature $\mathbf{X}[\pi(i), :]$ and stored directly at $\mathbf{O}[\pi(i), :]$, restoring the original token layout without a separate scatter pass.

An additional copy operation in the same launch preserves the remaining $N - K$ tokens unchanged. This design retains compact intermediate buffers between kernels while eliminating separate gather, activation, residual-addition, and scatter kernels.

A.7 FULL MS-COCO RESULTS

Table 6: MS-COCO results. Markers indicate compression type: ○ attention-only and ● joint attention–MLP. SparseSAM DINO speedups are end-to-end measurements relative to the original baseline, excluding detector time; H-DETR and YOLOX speedups are estimates obtained by scaling the corresponding DINO measurements.

Detector	Method	SAM-B			SAM-L			SAM-H		
		mAP	Density	Speedup	mAP	Density	Speedup	mAP	Density	Speedup
DINO	Base	46.8	100%	×1.00	49.5	100%	×1.00	49.9	100%	×1.00
	Attention-only compression									
	○ SpargeAttn	44.7	25%	×1.25	46.1	25%	×1.23	–	25%	–
		46.3	50%	×1.24	49.1	50%	×1.21	–	50%	–
	○ PISA	45.5	25%	×0.73	48.3	25%	×0.68	48.6	25%	×0.69
		46.5	50%	×0.72	49.4	50%	×0.67	49.8	50%	×0.65
	○ SparseSAM (Ours)	**45.9**	25%	×**2.43**	**48.7**	25%	×**2.15**	**49.4**	25%	×**1.89**
		46.7	50%	×**2.26**	**49.4**	50%	×**1.81**	**49.9**	50%	×**1.80**
	Joint attention–MLP compression									
	● ToMe	40.3	50%	×0.83	42.5	50%	×0.86	43.0	50%	×0.80
	● StructSAM	30.6	25%	×0.86	31.8	25%	×0.89	31.8	25%	×0.94
		42.2	50%	×0.71	44.5	50%	×0.74	45.1	50%	×0.79
	● SparseSAM (Ours)	**45.1**	25%	×**2.89**	**46.8**	25%	×**2.95**	**47.2**	25%	×**2.50**
		45.9	50%	×**2.62**	**48.2**	50%	×**2.53**	**48.7**	50%	×**2.24**
H-DETR	Base	40.2	100%	×1.00	42.4	100%	×1.00	42.7	100%	×1.00
	Attention-only compression									
	○ SpargeAttn	38.6	25%	×1.25	39.8	25%	×1.20	–	25%	–
		39.8	50%	×1.24	42.1	50%	×1.18	–	50%	–
	○ PISA	39.1	25%	×0.72	41.5	25%	×0.65	41.8	25%	×0.68
		40.1	50%	×0.70	42.2	50%	×0.63	42.6	50%	×0.64
	○ SparseSAM (Ours)	**39.6**	25%	×**2.58**	**41.8**	25%	×**2.14**	**42.3**	25%	×**1.88**
		40.1	50%	×**2.42**	**42.3**	50%	×**1.80**	**42.7**	50%	×**1.77**
	Joint attention–MLP compression									
	● ToMe	35.1	50%	×0.83	36.9	50%	×0.86	37.3	50%	×0.78
	● StructSAM	27.1	25%	×0.86	28.0	25%	×0.88	28.0	25%	×0.93
		36.6	50%	×0.70	38.6	50%	×0.72	39.0	50%	×0.78
	● SparseSAM (Ours)	**38.9**	25%	×**2.90**	**40.3**	25%	×**2.95**	**40.6**	25%	×**2.48**
		39.4	50%	×**2.65**	**41.5**	50%	×**2.53**	**41.7**	50%	×**2.19**
YOLOX	Base	39.0	100%	×1.00	41.2	100%	×1.00	41.6	100%	×1.00
	Attention-only compression									
	○ SpargeAttn	37.4	25%	×1.25	38.6	25%	×1.20	–	25%	–
		38.6	50%	×1.24	40.9	50%	×1.18	–	50%	–
	○ PISA	37.9	25%	×0.71	40.3	25%	×0.65	40.6	25%	×0.67
		38.8	50%	×0.69	41.0	50%	×0.63	41.4	50%	×0.63
	○ SparseSAM (Ours)	**38.4**	25%	×**2.53**	**40.6**	25%	×**2.14**	**41.1**	25%	×**1.90**
		39.0	50%	×**2.38**	**41.0**	50%	×**1.80**	**41.5**	50%	×**1.81**
	Joint attention–MLP compression									
	● ToMe	34.3	50%	×0.83	36.1	50%	×0.86	36.5	50%	×0.79
	● StructSAM	26.7	25%	×0.85	27.6	25%	×0.88	27.6	25%	×0.92
		35.6	50%	×0.69	37.7	50%	×0.72	38.1	50%	×0.77
	● SparseSAM (Ours)	**37.7**	25%	×**2.89**	**39.1**	25%	×**2.95**	**39.5**	25%	×**2.48**
		38.4	50%	×**2.66**	**40.3**	50%	×**2.53**	**40.5**	50%	×**2.24**

Table 6 presents a detailed performance comparison of SparseSAM across the MS-COCO (Lin et al., 2014) dataset, utilizing SAM-B, SAM-L, and SAM-H backbones under three distinct detectors:

DINO (Zhang et al., 2022), H-DETR (Jia et al., 2023), and YOLOX (Ge et al., 2021). At 25% density, the attention-only configuration spans 1.88–2.58× speedup with absolute mAP decreases of 0.4–0.9 points across the reported model–detector pairs. Joint attention–MLP compression spans 2.48–2.95× speedup with larger decreases of 1.3–2.7 mAP points. These operating points should therefore be compared within the same compression configuration and hardware protocol.

A.8 Full HQ-44K Results

Table 7: HQ-44K segmentation results. Markers indicate compression type: ○ attention-only, ● joint attention–MLP, and ● fine-tuned joint compression.

Method	Density	DIS5K-VD		COIFT		ThinObject5K-TE		HRSOD	
		mIoU	Speedup	mIoU	Speedup	mIoU	Speedup	mIoU	Speedup
Base Model	100%	78.63	×1.00	94.55	×1.00	89.56	×1.00	92.91	×1.00
Attention-only compression									
○ SpargeAttn	25%	73.48	×1.26	91.67	×1.25	84.94	×1.24	90.99	×1.25
	50%	77.09	×1.24	93.92	×1.23	88.76	×1.25	92.68	×1.24
	75%	77.61	×1.16	94.12	×1.14	89.26	×1.15	92.84	×1.15
○ PISA	25%	76.79	×0.71	93.37	×0.70	87.74	×0.72	92.18	×0.70
	50%	78.38	×0.69	94.25	×0.71	89.26	×0.70	92.73	×0.71
	75%	78.54	×0.68	94.51	×0.67	89.49	×0.67	92.87	×0.66
○ SparseSAM (Ours)	25%	**77.13**	**×1.75**	**93.83**	**×1.63**	**88.68**	**×1.68**	**92.41**	**×1.71**
	50%	**78.38**	**×1.67**	**94.44**	**×1.56**	**89.40**	**×1.61**	**92.83**	**×1.63**
	75%	**78.58**	**×1.61**	**94.49**	**×1.50**	**89.58**	**×1.55**	**92.86**	**×1.58**
Joint attention–MLP compression									
● ToMe	50%	69.69	×0.81	90.33	×0.81	84.17	×0.81	87.98	×0.84
	75%	77.16	×0.52	94.36	×0.51	89.38	×0.52	92.43	×0.51
● StructSAM	25%	54.97	×0.74	75.22	×0.75	69.76	×0.75	76.19	×0.77
	50%	68.50	×0.63	92.17	×0.64	82.42	×0.64	88.73	×0.63
	75%	76.58	×0.53	94.37	×0.54	89.00	×0.55	92.32	×0.54
● SparseSAM (Ours)	25%	**74.79**	**×2.95**	**91.79**	**×2.81**	**85.95**	**×2.08**	**91.29**	**×2.85**
	50%	**77.56**	**×2.53**	**93.12**	**×2.42**	**87.82**	**×1.88**	**92.40**	**×2.50**
	75%	**78.43**	**×1.73**	**94.12**	**×1.64**	**89.21**	**×1.70**	**92.62**	**×1.71**
Fine-tuning after joint compression									
● SparseSAM (Ours)	25%	**76.92**	**×2.95**	**93.01**	**×2.81**	**88.05**	**×2.08**	**91.96**	**×2.85**
● SparseSAM (Ours)	50%	**78.28**	**×2.53**	**93.97**	**×2.42**	**89.41**	**×1.88**	**92.71**	**×2.50**

Table 7 presents zero-shot segmentation results on DIS5K-VD, COIFT, ThinObject5K-TE, and HRSOD from HQ-44K (Ke et al., 2023). At 25% density, attention-only SparseSAM achieves 1.63–1.75× speedup with absolute mIoU decreases of 0.50–1.50 points across the four datasets. Joint attention–MLP compression increases speedup to 2.08–2.95×, with larger decreases of 1.62–3.84 points. Thus, the attention-only rows represent the lower-loss operating points, whereas the joint rows represent the higher-speed operating points.

The final tier reports fine-tuning of the joint-compression configuration without changing its speedup. At 25% density, fine-tuning reduces the remaining absolute gap to the dense baseline to 0.95-1.71 mIoU points across the four datasets. At 50% density, the remaining gap is 0.15–0.58 points, while speedup remains 1.88–2.53×.

A.9 Attributing Speedup to Kernel Optimization and Sparsification

Kernel engineering and structured sparsification both contribute to name's runtime improvements. To distinguish these contributions, Table 8 compares the original dense baseline with the fused-dense attention kernel and MLP, sparse attention with dense MLP, and joint attention-MLP sparsification on

Table 8: Speedup attribution on NVIDIA L4. All speedup ratios are calculated from the listed latencies using the SAM-L backbone. The fused-dense control retains full attention density and the full MLP. Attention-only and joint configurations both use 30% density.

Configuration	Latency (ms)	Peak memory (MB)	vs. original dense	vs. fused dense
A: Original dense baseline	191.51	2221.5	1.00×	0.66×
B: Fused - dense attention + dense MLP	127.05	765.3	1.51×	1.00×
C: Fused Sparse attention ($d = 0.30$) + dense MLP	96.79	765.3	1.98×	1.31×
D: SparseSAM ($d = 0.30$)	69.45	765.3	2.76×	1.83×

NVIDIA L4. We report latency speedups relative to both dense implementations, computed directly as reference latency divided by configuration latency.

Contribution of kernel optimization. Moving from A to B reduces latency from 191.51 to 127.05 ms, yielding a $1.51\times$ speedup without sparsification. Peak memory decreases from 2221.5 to 765.3 MB, a 65.6% reduction, consistent with fused attention avoiding materialization of the dense attention matrix. Both sparse configurations retain the same reported peak memory as B, indicating that the fused implementation accounts for the observed peak-memory savings in this comparison.

Additional gains from sparsification. Relative to B, attention-only sparsification at $d = 0.30$ reduces latency by 30.26 ms, from 127.05 to 96.79 ms, yielding a $1.31\times$ speedup. Joint sparsification at the same density reduces latency by 57.60 ms, to 69.45 ms, yielding a $1.83\times$ speedup over B. Adding MLP routing to sparse attention (C to D) saves a further 27.34 ms and provides a $1.39\times$ speedup. The overall $2.76\times$ gain from A to D therefore combines a $1.51\times$ gain from dense-kernel optimization with a $1.83\times$ additional gain from sparse execution.

A.10 Timing Protocol

For the main experiments on COCO and HQ-44K, inputs are preprocessed to 1024×1024 resolution, and inference uses FP16 precision with batch size $B = 1$ on an NVIDIA L4 GPUS (SM80). The software environment comprises Python 3.12, PyTorch 2.5.1, and CUDA 12.1. We perform 10 warm-up iterations before collecting 20 measurements per configuration. We call `torch.cuda.synchronize()` after warm-up and immediately before and after each timed forward pass. Latency is measured and reported in milliseconds.

The main latency measurements include the image encoder, prompt encoder, and mask decoder; object-detector inference used to generate COCO prompts is excluded. Image resizing, normalization, padding, and mask post-processing are also excluded. SparseSAM computes Sobel saliency, Morton-group scores, group sorting, the attention permutation, and the MLP keep set once per image and reuses them across encoder blocks. These image-dependent preparation operations are included in the reported latency and are charged to the timing stage. Static attention-mask construction and compilation are not counted as they are done once and reuse for later inferences. Per-block permutation, inverse permutation, and MLP-routing overheads are included.

A.11 HARDWARE-AWARE PERFORMANCE ANALYSIS

Table 9: Throughput speedup on different hardware and density levels. These ratios are distinct from the latency speedups in the segmentation tables. We evaluate attention-only compression and joint attention–MLP compression.

Batch size	GPU	25% density		50% density		75% density	
		Joint	Attn. only	Joint	Attn. only	Joint	Attn. only
1	A100	×1.91	×1.65	×1.75	×1.56	×1.33	×1.23
	RTX 3090	×1.42	×1.32	×1.35	×1.25	×1.17	×1.15
2	A100	×1.95	×1.62	×1.75	×1.55	×1.33	×1.22
	RTX 3090	×1.45	×1.33	×1.36	×1.25	×1.16	×1.16
4	A100	×2.00	×1.62	×1.80	×1.55	×1.46	×1.42
	RTX 3090	×1.53	×1.33	×1.39	×1.26	×1.23	×1.27
8	A100	×2.05	×1.64	×1.83	×1.57	×1.66	×1.50
	RTX 3090	×1.57	×1.34	×1.45	×1.27	×1.28	×1.28

Table 9 provides a throughput speedup analysis of SparseSAM, comparing attention-only compression against joint attention–MLP compression across A100 and RTX 3090 GPUs. These throughput ratios are distinct from the latency speedups reported in the segmentation tables. On A100 at batch size 8 and 25% density, joint compression reaches $2.05\times$ throughput versus $1.64\times$ for attention-only compression. On RTX 3090 under the same setting, the corresponding values are $1.57\times$ and $1.34\times$. Because Table 9 does not report accuracy, its values characterize hardware scaling rather than a standalone accuracy–latency operating point.

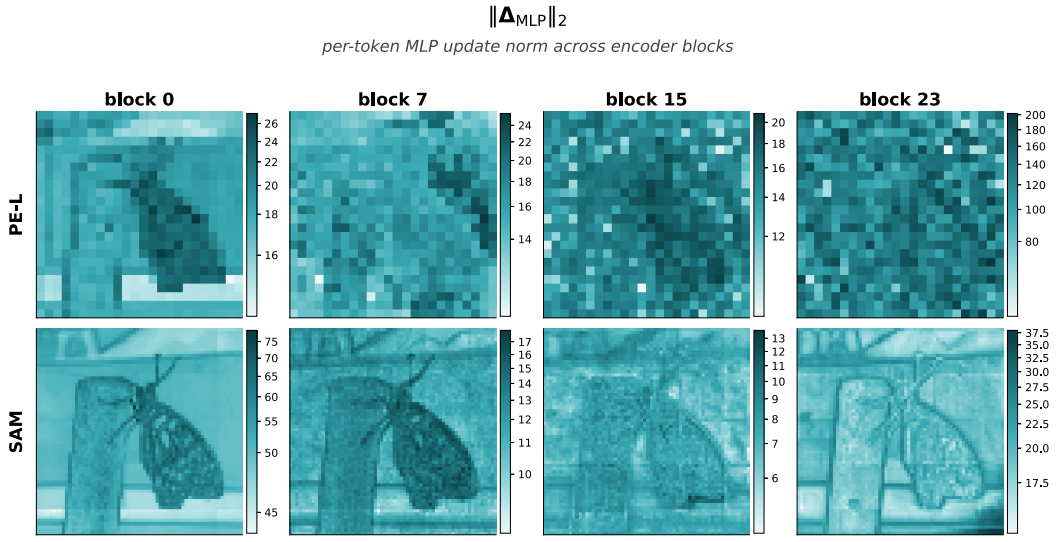

Figure 11: Per-token MLP update norm $\|\Delta_{\text{MLP}}\|_2$ at blocks 0, 7, 15, and 23 of PE-Core-L14-336 (top) and SAM-HQ ViT-L (bottom) on the same input. Both models show object-aligned structure in the first block; later blocks retain clearer object structure in the segmentation backbone. Panels use separate color scales, and the global backbone's class-token update is omitted.

A.12 MLP UPDATE BEHAVIOR IN A GLOBAL-REPRESENTATION BACKBONE

We visualize the per-token MLP update magnitude $\|\Delta_i\|_2 = \|\text{MLP}(\text{LN}(\mathbf{x}_i))\|_2$ at four blocks of Perception Encoder (Bolya et al., 2025) (PE-Core-L14-336) and SAM-HQ (Ke et al., 2023) (ViT-L)

on the same image. Both backbones show object-aligned updates at block 0 (Fig. 11). PE (Bolya et al., 2025), trained with an image-level contrastive objective, gradually loses object-aligned spatial selectivity: by the middle and later blocks, updates are distributed more diffusely across the spatial grid, consistent with learning a global image representation. SAM-HQ (Ke et al., 2023), trained for dense prediction, retains spatially distinctive updates, with the butterfly outline still visible at block 23. In our experiments, Residual-Consistency MLP causes severe accuracy degradation on the global-representation backbone, while remaining effective on the evaluated segmentation models. The routing proposition provides a way to interpret this empirical contrast.

Proposition 2 makes this connection precise: at a shared input, squared representation error equals the update energy omitted by the keep set. For nonzero total update energy, define the retained fraction

$$\rho(\mathcal{K}) = \frac{\sum_{i\in\mathcal{K}} \|\Delta_i\|_2^2}{\sum_{i=1}^{N} \|\Delta_i\|_2^2}, \qquad \frac{\|\mathbf{Y} - \widetilde{\mathbf{Y}}^{(\mathcal{K})}\|_F^2}{\sum_{i=1}^{N} \|\Delta_i\|_2^2} = 1 - \rho(\mathcal{K}).$$

If a small subset carries most of the update energy, retaining it gives a small error relative to the total update energy. Conversely, if all update magnitudes $\|\Delta_i\|_2$ are equal and nonzero, every size-K keep set omits the fraction $1 - K/N$; even the optimal selector cannot improve this fraction. Equation 7 gives the corresponding optimum for arbitrary magnitudes. Thus, routing depends on both energy concentration and the selector's ability to identify high-energy tokens.

This suggests a mechanism for the observed contrast: spatial routing can preserve more update energy when saliency identifies the tokens carrying large updates, but may omit substantial energy when updates are broadly distributed or poorly aligned with the ranking.

A.13 Stripe-sort Attention's performance on ImageNet

To examine attention sparsification in the global-representation backbone, we evaluate Stripe-Sort Attention on ImageNet (Deng et al., 2009), both alone and combined with token merging (Bolya et al., 2022). The combined configuration uses merging in place of Residual-Consistency MLP, so it does not directly test the residual-routing error characterized by Proposition 2.

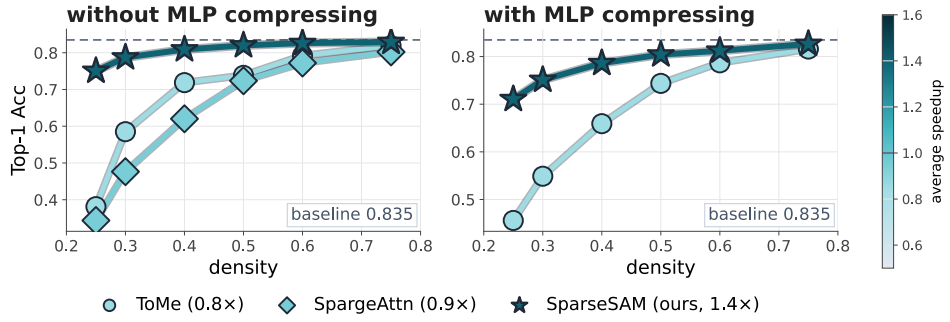

Figure 12: Top-1 accuracy on ImageNet (Deng et al., 2009) under attention-only compression (left) and attention combined with token merging (right).

Figure 12 shows higher Top-1 accuracy for SparseSAM than the plotted ToMe (Bolya et al., 2022) and SpargeAttn (Zhang et al., 2025a) baselines at the reported densities, with the largest gaps under stronger compression. Accuracy approaches the dense reference of 0.835 as density increases; low-density configurations remain below it. The combined configuration follows a similar trend, and the figure reports an average $1.4\times$ speedup for SparseSAM.

These results support the usefulness of spatially distributed key selection beyond segmentation. Proposition 1 provides a local explanation for complete-stripe prefixes: greater coverage tightens the attention bound when within-group logit and value variation are controlled. It does not directly bound Top-1 accuracy or cover added diagonal keys and token merging. Together with the update-energy analysis above, the experiment motivates evaluating attention sparsification and MLP routing separately when transferring to another backbone.

A.14 Additional Segmentation Outputs

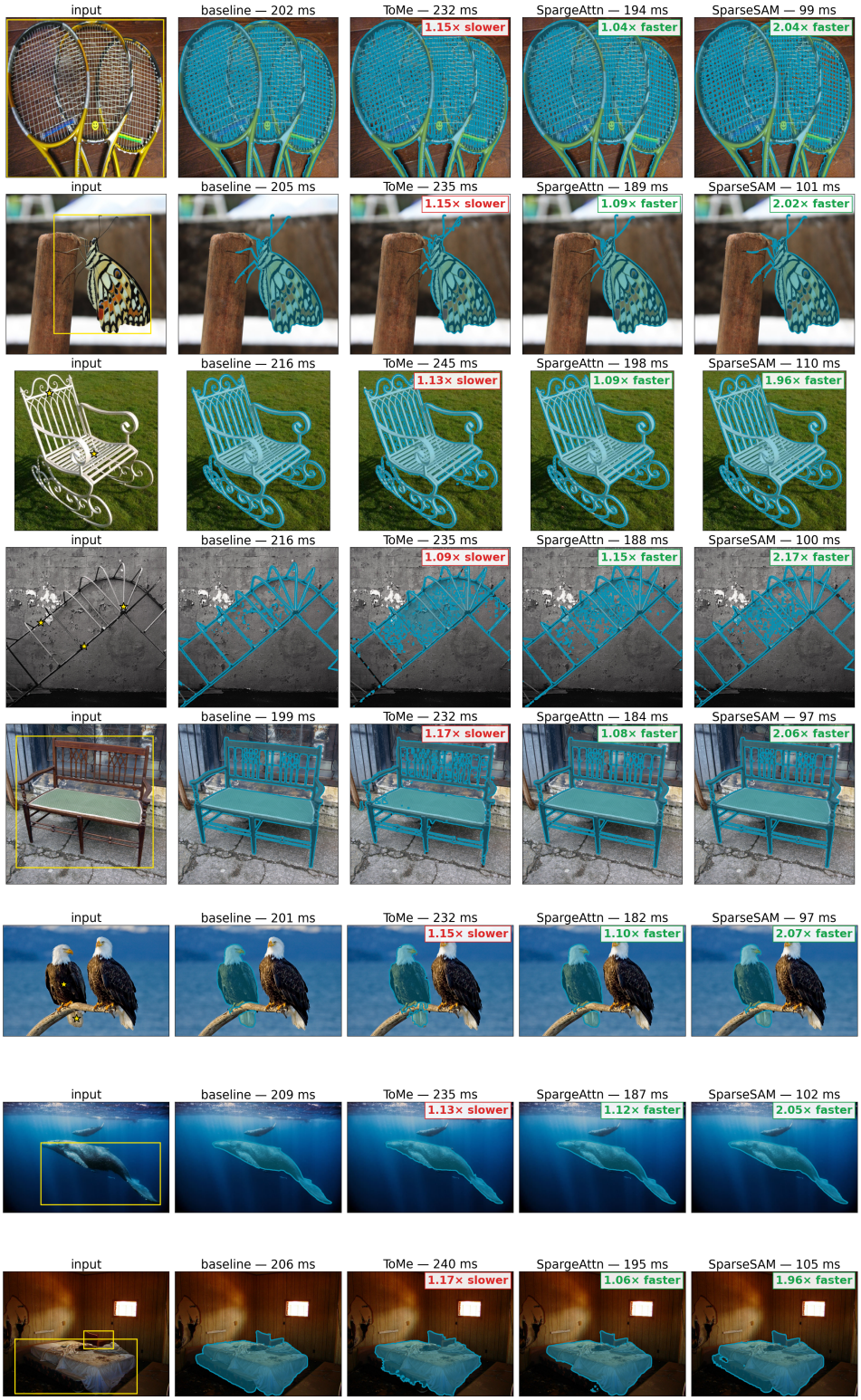

Figure 13: Additional qualitative segmentation outputs from SparseSAM on A100 .