SparseSAM: Structured Sparsification of Activations in Segment Anything Models

Hoai-Chau Tran [1 2] **Chi H. Nguyen** [3 2] **Duy M. H. Nguyen** [4 5 6] **Mathias Niepert** [5 6] **Fan Lai** [1] **Khoa D. Doan** [3 2]

Abstract

The Segment Anything Model (SAM) achieves strong open-vocabulary segmentation, but its ViT-based image encoders dominate inference latency and memory. Existing activation compression methods, such as token merging, reduce the token length to process, yet introduce non-trivial runtime overhead and encounter catastrophic quality drop under high compression. Other methods applying Sparse Attention focus on attention alone, leaving the MLP fully dense and capping achievable speedup. We propose *SparseSAM*, a *(i) training-free structured sparsification* framework that jointly accelerates attention and MLP layers while preserving token identity. SparseSAM introduces *(ii) Stripe-Sort Attention*, which uses a deterministic Z-order permutation to transform dense attention into static hardware-friendly sparse patterns, eliminating dynamic masking overhead. SparseSAM further introduces a *(iii) Residual-Consistency MLP* that routes only informative tokens through the MLP while propagating remaining tokens through the residual pathway. Across four segmentation benchmarks, SparseSAM loses only 0.004 mIoU at a 0.4 density and 0.021 mIoU at 0.3, a 2.10× reduction in accuracy loss versus token merging advances, while achieving 2x faster inference and 2.8× memory reduction. Code is available at: `https://github.com/hchautran/SparseSAM.git`.

1 Introduction

SAMs have shown strong zero-shot segmentation performance and are widely used in downstream vision tasks (Liang et al., 2022; Liu et al., 2017; 2019; 2021). However, their modular design places almost all computation and parameters in the ViT image encoder, which accounts for over 99% of the model cost. This makes SAM expensive to deploy in latency-sensitive settings such as large-scale serving and edge devices.

Recent efficient SAM variants replace the image encoder with compact architectures (Zhang et al., 2023; Zhou et al., 2023; Xiong et al., 2023; Zhao et al., 2023), but typically require training a new model and often lose accuracy. Training-free sparse attention methods (Jiang et al., 2024; Xiao et al., 2024; Xi et al., 2025) avoid retraining, yet they are poorly matched to SAM's local-window structure: many attention heads contain fewer than 200 tokens, so dynamic mask construction and importance estimation can outweigh the skipped computation.

Moreover, attention-only sparsification leaves the MLP blocks dense, limiting end-to-end speedup. Existing alternatives are also insufficient: MLP pruning requires retraining or distillation (Chen et al., 2024); post-training quantization rarely yields practical GPU speedups for SAM's small matrix sizes (Lv et al., 2024; Wang et al., 2024; Ranjan & Savakis, 2025; Zhang et al., 2025b); and token merging must repeatedly merge and unmerge tokens to preserve the full spatial output, introducing overhead and degrading mask quality (Bolya et al., 2022; Tran et al., 2024; 2025). These limitations motivate a training-free method that accelerates both attention and MLP layers while preserving token identity for segmentation.

These limitations point to a key gap: *efficient, training-free methods that jointly reduce computation in both attention and MLP layers, while preserving token-level fidelity required for segmentation.* We introduce *SparseSAM*, a training-free approach designed to leverage the inherent sparsity exposed by the activations in both the attention and MLP layers of SAM. We propose two complementary techniques that improve encoder throughput while preserving segmentation quality.

- **Stripe-Sort Attention.** We introduce a new structured sparse attention mechanism based on a deterministic Z-order (Morton) permutation (Morton, 1966; Erkan &

[1]University of Illinois at Urbana-Champaign [2]VinUni-Illinois Smart Health Center, VinUniversity [3]College of Engineering & Computer Science, VinUniversity [4]DFKI [5]Max Planck Research School for Intelligent Systems (IMPRS-IS) [6]University of Stuttgart. Correspondence to: Hoai-Chau Tran <chauht2@illinois.edu>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

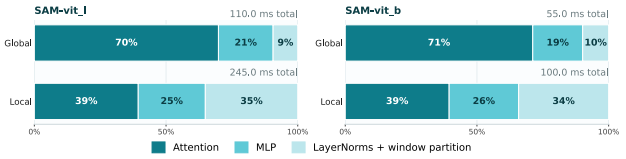


Figure 1. SAM latency profiling.

Aksoy, 2023; Stücker et al., 2026). This reordering induces spatial locality in token indices, enabling a static, block-structured sparsity pattern implemented via a custom CUDA kernel. The resulting pattern consists of a banded diagonal capturing local interactions, augmented with a small global keep set for long-range dependencies. Notably, this design avoids dynamic mask construction and preserves the full key/value set, ensuring both efficiency and quality (Fig. 4 and Fig. 5).

- **Residual-Consistency MLP.** We propose a token routing mechanism that applies the MLP only to a top-K subset of tokens selected based on their importance (i.e., the high-rank prefix of the sorted z-group tokens), while allowing the remaining tokens to bypass the MLP through the residual connection. This design retains information flow without incurring the full cost of dense MLP computation (Fig. 6a).

Both techniques utilize a **static, one-shot permutation set** that is computed once and reused across all model layers, ensuring that per-layer overhead remains negligible. Our evaluations across five distinct SAM checkpoints and five benchmarks demonstrate that [Model Name] achieves an average 2.0× inference speedup. Even at high-density compression rates, the method maintains 50% sparsity with a minimal segmentation accuracy drop (1% IoU loss), as illustrated in Figure 7.

2 Background and Related Work

SAM Architecture. SAM uses a ViT image encoder with local and global attention blocks. Local blocks apply self-attention within non-overlapping windows, while global blocks attend over the full token sequence. As shown in Fig. 1, global blocks are dominated by attention, which can account for up to 70% of runtime. In contrast, local blocks have a dual bottleneck: both attention and MLP layers contribute substantially to latency. Since SAM-L contains many more local than global blocks, e.g., 20 local versus 4 global blocks, attention-only acceleration gives limited end-to-end speedup. Efficient SAM inference, therefore, requires reducing both attention and MLP computation.

Sparse Attention Approaches. Recent sparse attention methods reduce Transformer attention cost through online filtering, block-wise approximation, or semantic token permutation (Zhang et al., 2025a; Li et al., 2026; Xi et al., 2025; Yang et al., 2025). These methods are effective for long-context LLMs and diffusion models, but are less suitable for SAM. Most SAM layers use local-window attention with fewer than 200 tokens per head, where dynamic mask construction and importance estimation can cost more than the skipped attention computation. Moreover, sparse attention leaves MLP layers dense, limiting end-to-end acceleration. In contrast, our method uses a deterministic Z-order permutation to create a static, hardware-efficient sparsity pattern with no runtime search overhead.

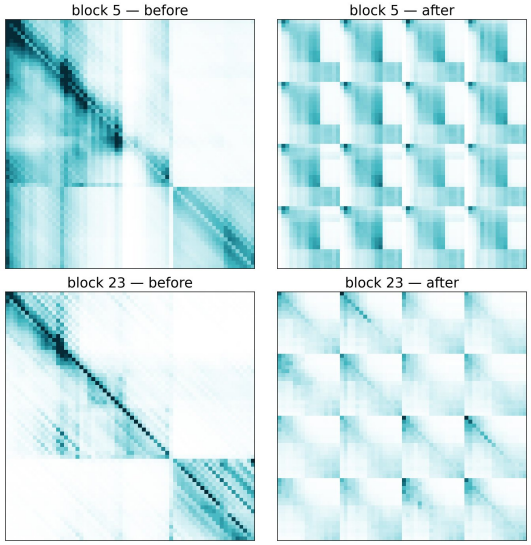


Figure 2. **Permuted attention visualization**. Comparison between the original SAM attention pattern (left) and our Z-order permuted pattern (right). The permutation induces a banded diagonal structure, allowing for significant hardware acceleration with minimal loss in spatial information.

Activation Compression Approaches. Activation compression aims to accelerate both attention and MLP layers. Quantization reduces theoretical FLOPs and memory (Lv et al., 2024; Wang et al., 2024; Ranjan & Savakis, 2025; Zhang et al., 2025b), but SAM's relatively small matrix tiles limit practical GPU speedups because quantization overheads can dominate (Kim et al., 2023). Token merging reduces sequence length (Bolya et al., 2022; Tran et al., 2024; 2025), but segmentation requires full spatial resolution, forcing repeated merge-unmerge operations that add scatter-gather overhead and degrade mask quality (Nguyen et al., 2026). SparseSAM preserves the token grid with structured Z-order sparsity, avoiding quantization overhead and merging-induced information loss.

3 Observations and Motivation

Because SAM is highly overhead-sensitive, we need an approach with minimal per-layer cost and high-fidelity features. Rather than dynamically identifying sparse regions, we explore an alternative perspective: *can we restructure*

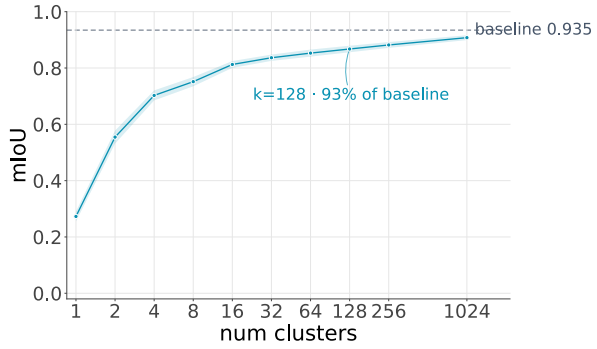


Figure 3. K-means clustering replacement.

Table 1. Layer-wise correlation between token dissimilarity and the norm of $\|\Delta_{\text{MLP}}\|_2^2$ for each token.

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

the token layout itself such that efficient sparsity patterns emerge deterministically? We next introduce the key insights that motivate our work.

Observation 1: Structured token permutation induces reusable sparse attention patterns. Prior activation compression studies show that preserving spatial diversity is more important for segmentation than keeping contiguous foreground regions (Tran et al., 2024; 2025). Motivated by this, we permute the $\mathbf{Q}$ and $\mathbf{K}$ matrices so that sparse attention behaves like spatial downsampling while maintaining global token coverage.

We implement this with a parameter-free **stripe-sort attention kernel**. As shown in Fig. 2, the deterministic permutation decomposes the original attention map into smaller phase-shifted sub-maps with similar structure, enabling reusable sparse patterns with minimal overhead. Details are given in Section 4.

Observation 2: SAM's decoder depends more on inter-region contrast than precise per-token representations. We further observe that 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 $k = 128$ ($\sim 3\%$ of tokens), the decoder retains *93% of baseline mIoU* on COIFT, saturating near baseline at $k = 256$ (Fig. 3). This suggests same-cluster tokens are largely interchangeable, and the decoder relies more on *cluster-level contrast* than exact per-patch representations.

Observation 3: Which tokens actually require expensive MLP updates? We analyze how MLP layers modify token representations and find that SAM's MLP exhibits a *strong inductive bias toward separating distinct image regions*: a large fraction of background tokens remain close to the residual branch, while semantically rich foreground tokens undergo significantly larger updates (Fig. 6b). Table 1 quantifies this via the correlation between MLP update magnitude, $||\Delta_{\text{MLP}}||_2$, and cosine token dissimilarity. We observe that tokens that are more distinct in feature space tend to receive larger updates, with a strong positive correlation ($\rho \approx 0.6$–0.7).

This redundancy motivates our second contribution, *residual-consistency MLP*, which prioritizes MLP updates for semantically important tokens while maintaining lightweight residual propagation for the remaining tokens.

4 Methods

4.1 Stripe-Sort Attention

We first introduce *Stripe-Sort Attention*, a structured sparse attention mechanism that reorganizes token layouts to expose deterministic sparsity patterns amenable to efficient GPU execution. Specifically, we use a deterministic, parameter-free permutation that decomposes the $N \times N$ attention matrix into a 4×4 mosaic of $(N/4) \times (N/4)$ sub-maps. Since the permutation depends only on spatial position, it can be precomputed once and statically compiled into the attention kernel with no runtime overhead. Importantly, the four resulting token subsets are phase-shifted, half-resolution views of the original image, where each subset remains spatially distributed across the entire input rather than localized to a single region. Consequently, each sub-map preserves global context while operating on only a quarter of the tokens.

Gradient-Based Ordering. Given an input feature map $\mathbf{X} \in \mathbb{R}^{H \times W \times D}$, our goal is to construct a one-dimensional token ordering that places high-information regions early in the sequence while spreading them evenly across spatial positions. The first property ensures that when the sequence is later truncated or sparsified, the most informative tokens are retained, while the second property ensures that the retained tokens collectively cover the full spatial extent of the image rather than concentrating in a single region. To achieve both properties, we use local image gradients as a lightweight proxy for information density. Intuitively, regions with large gradient magnitude correspond to object boundaries, texture, and visually salient structures, while regions with low gradient magnitude correspond to smooth or homogeneous areas that contribute relatively little to dense prediction.

We measure local gradient magnitude using the Sobel operator (Gonzalez, 2009), a classical edge-detection filter that approximates the spatial derivatives of an image through a

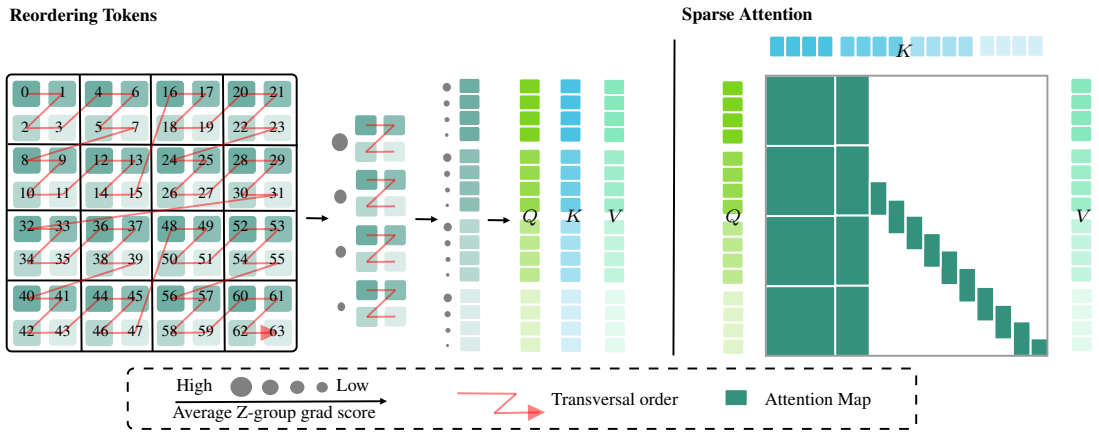


Figure 4. We employ a deterministic **Z-order (Morton) traversal** to linearize 2D spatial tokens into a 1D sequence. This traversal naturally preserves spatial locality by grouping neighboring pixels into localized "Z-groups." Such grouping enables the efficient computation of gradient-based importance scores within local windows (Eq.1), which are then used to rank and route tokens. The resulting permuted sequence transforms the attention map into a hardware-friendly block-banded structure, consisting of a dense global prefix for high-saliency tokens and a banded diagonal for local spatial interactions.

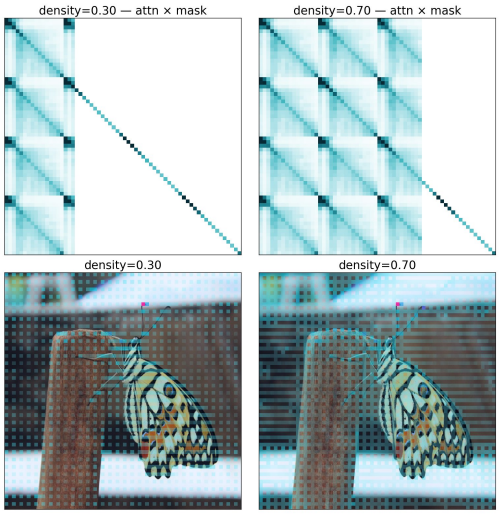


Figure 5. Token keep-set and MLP routing visualization.

pair of fixed 3×3 convolutional kernels. Specifically, the gradient magnitude at each spatial position is

$$\mathbf{M}[i,j] = \sqrt{(\mathbf{S}_x * \mathbf{X})^2[i,j] + (\mathbf{S}_y * \mathbf{X})^2[i,j]}, \quad (1)$$

where $\mathbf{S}_x$ and $\mathbf{S}_y$ denote the horizontal and vertical Sobel kernels, and $*$ represents channel-wise 2D convolution followed by summation across the D channels. The resulting map $\mathbf{M} \in \mathbb{R}^{H \times W}$ is a single-channel saliency map, where high $\mathbf{M}$ values correspond to object boundaries and low values correspond to smooth regions. This computation is fully deterministic, parameter-free, and only performed once in the first SAM layer, introducing negligible overhead. Sorting the $N = HW$ spatial positions in descending order of $\mathbf{M}$ produces a permutation π, placing high-gradient regions at the front and low-gradient regions at the back.

Stripe-Sort Permutation. To prevent foreground regions from gathering at the start of the token set, we view π as a matrix $\mathbf{T} \in \mathbb{N}^{(N/G) \times G}$ with $\mathbf{T}[t,g] = \pi[t \cdot G + g]$ and define the final scan order as $\sigma = \mathrm{flatten}(\mathbf{T}^\top)$.

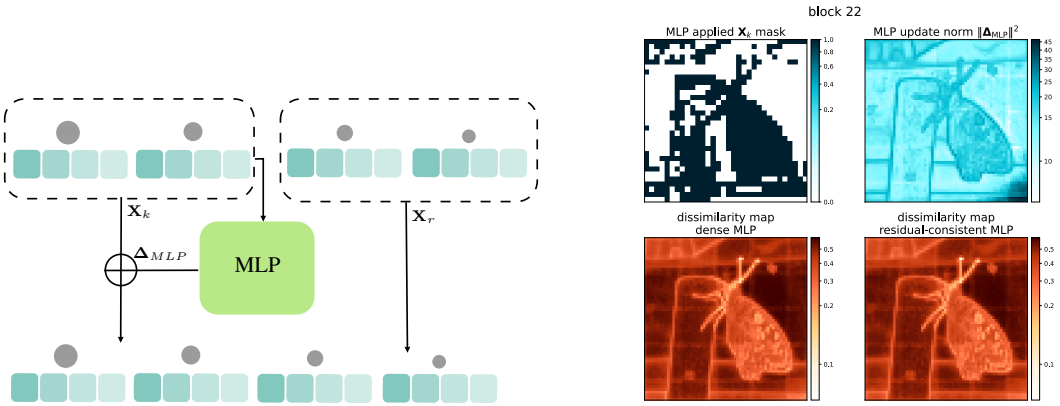
This visits every G-th element of π before returning to the next offset, so each of the G resulting blocks of σ contains a uniformly subsampled mixture of tokens drawn from across the entire image. As demonstrated in Figure 5, applying this reordering to both queries and keys transforms global attention into a deterministic block-diagonal pattern that can be efficiently executed using static block-sparse kernels like FlashAttention. Our method avoids runtime mask generation, semantic clustering, and online scheduling overhead.

We implement Stripe-Sort Attention as a fused CUDA kernel to avoid intermediate scattering and extra kernel launches. Since local layers contain at most 196 tokens, standard FlashAttention (Dao, 2023a;b) and FlashInfer (Ye et al., 2025) are inefficient with fixed tile sizes. Our kernel uses flexible tiling, with 32×32 tiles for local layers and 128×128 tiles for global layers, while preserving efficient `mma.sync.aligned.m16n16k16` tensor-core execution. Pseudocode is given in Appendix A.2.

4.2 Residual-Consistency MLP

Our second contribution is motivated by the structure of MLP updates: the decoder depends more on inter-region contrast than precise per-token representations. Given input tokens $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N \times d}$, the MLP update for token i is $\Delta_i = \mathrm{MLP}(\mathrm{LN}(\mathbf{x}_i))$, where $\mathrm{LN}(\cdot)$ denotes LayerNorm. We use the update magnitude $u_i = \|\Delta_i\|_2$ to quantify how strongly each token is modified.

As shown in Fig. 6b, MLP update magnitudes are highly



(a) Residual-consistency MLP. *(b)* Token keep-set and MLP routing.

Figure 6. Visualization of the residual-consistency MLP. (a) The deterministic token order π is reused to partition tokens into a keep-set $\mathbf{X}_k$, updated by the MLP, and a residual set $\mathbf{X}_r$, which bypasses the MLP unchanged. (b) Dense SAM implicitly focuses MLP updates on a small set of representative, highly dissimilar tokens. After residual-consistency MLP, inter-token similarity stays close to the dense baseline, preserving decoder geometry.

skewed: only a small subset of tokens receives large updates, while most stay near the residual path. These high-update tokens typically correspond to distinctive regions such as boundaries and salient textures, whereas smooth regions change little. Table 1 confirms this trend, showing a strong correlation between token uniqueness and update magnitude u_i.

$$\mathbf{y}_i = \begin{cases} \mathbf{x}_i + \Delta_i, & i \in \mathcal{K}, \\ \mathbf{x}_i + \epsilon_i, & i \notin \mathcal{K}, \end{cases}$$

where $\mathcal{K}$ is the representative token subset and $\epsilon_i \ll \Delta_i$ denotes a small residual update.

Formulation. Motivated by this observation, we make the routing behavior explicit. Given we reuse the deterministic permutation order to reorder the tokens $\pi(\mathbf{X})$ and partition the sequence into a keep-set $\mathbf{X}_k = \{\mathbf{x}_i \mid i \in \mathcal{K}\}$ and a residual set $\mathbf{X}_r = \{\mathbf{x}_i \mid i \notin \mathcal{K}\}$. As shown in Fig. 6b(a), the MLP is applied only to the keep-set, while residual tokens bypass the MLP unchanged.

$$\mathbf{Y}_k = \mathbf{X}_k + \mathrm{MLP}(\mathrm{LN}(\mathbf{X}_k)), \quad \mathbf{Y}_r = \mathrm{LN}(\mathbf{X}_r).$$

By routing only representative tokens through the MLP, our method preserves the feature geometry of the dense baseline while substantially reducing MLP computation. Despite its simplicity, this approach proves highly effective. As shown in Fig. 6b, residual-consistency MLP closely matches the dense MLP's inter-token dissimilarity, preserving the structure needed by the decoder.

4.3 Experimental Setup

Models, Datasets, and Baselines. We apply SparseSAM to the official SAM-B, SAM-L, and SAM-H checkpoints, with all experiments conducted on a single NVIDIA A100 GPU unless otherwise noted. We evaluate fine-grained segmentation on HQ-44K (Ke et al., 2023) and zero-shot common-object segmentation on MS-COCO (Lin et al., 2014), using DINO (Zhang et al., 2022) as box-prompt detectors. We compare against two classes of training-free compression baselines: sparse attention methods, including SpargeAttention (Zhang et al., 2025a) and PISA (Li et al., 2026), and activation compression methods, including ToMe (Bolya et al., 2022) and StructSAM (Nguyen et al., 2026). For fair comparison, we add fused relative positional encoding support to sparse attention baselines for compatibility with SAM's ViT encoder.

Metrics. We report segmentation quality using mIoU and Boundary IoU (BIoU), and system efficiency using end-to-end latency, peak GPU memory, and throughput across different density levels. Latency is averaged over multiple synchronized CUDA runs.

4.4 Segmentation Results

Table 2 reports zero-shot segmentation results on MS-COCO (Lin et al., 2014) using the DINO detector (Zhang et al., 2022).Across all settings, SparseSAM achieves the best accuracy–efficiency trade-off, preserving quality while delivering the highest speedup.

Efficiency Comparison. SpargeAttention (Zhang et al., 2025a) and PISA (Li et al., 2026) are constrained by SAM's local attention design, where attention operates on relatively small token maps (196×196), limiting the benefits of FlashAttention-style kernels (Dao, 2023b). Their dynamic sparsification overhead further reduces efficiency, with PISA even falling below the dense baseline in some settings ($0.73\times$ on SAM-B at 25% density). In contrast, SparseSAM reuses a fixed Z-curve permutation across encoder layers, avoiding costly runtime operations and achieving the best trade-off between speed and accuracy. On

Table 2. Zero-shot segmentation on MS-COCO datasets using DINO as a bounding box detector generates a box prompt for the SAM model. Markers indicate compression type: ○ attention-only and ● 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	0.468	100%	× 1.00	0.495	100%	× 1.00	0.499	100%	× 1.00
○ Sparge Attention	0.447	25%	× 1.25	0.461	25%	× 1.23	–	25%	–
	0.463	50%	× 1.24	0.491	50%	× 1.21	–	50%	–
○ PieceWise Attention	0.455	25%	× 0.73	0.483	25%	× 0.68	0.486	25%	× 0.69
	0.465	50%	× 0.72	0.494	50%	× 0.67	0.498	50%	× 0.65
● ToMe	0.403	25%	× 0.83	0.425	25%	× 0.86	0.430	25%	× 0.80
	0.467	50%	× 0.74	0.494	50%	× 0.75	0.498	50%	× 0.64
● StructSAM	0.306	25%	× 0.86	0.318	25%	× 0.89	0.318	25%	× 0.94
	0.422	50%	× 0.71	0.445	50%	× 0.74	0.451	50%	× 0.79
○ SparseSAM (Ours)	**0.459**	25%	× **1.63**	**0.487**	25%	× **1.64**	**0.494**	25%	× **1.28**
	0.467	50%	× **1.55**	**0.494**	50%	× **1.57**	**0.499**	50%	× **1.22**
● SparseSAM (Ours)	**0.451**	25%	× **2.15**	**0.468**	25%	× **2.05**	**0.472**	25%	× **1.59**
	0.459	50%	× **1.89**	**0.482**	50%	× **1.83**	**0.487**	50%	× **1.40**

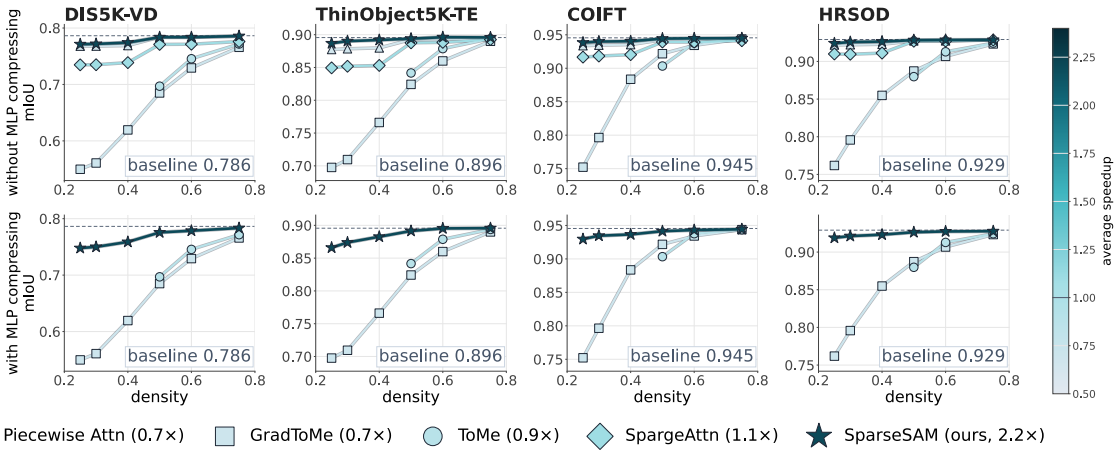


Figure 7. **Robustness Under Extreme Compression**. SparseSAM sets a new standard for efficient segmentation, consistently outperforming existing dynamic and merging-based approaches across density levels. Notably, at high compression rates (density < 0.4), where competing methods suffer catastrophic quality degradation, SparseSAM preserves near-baseline fidelity. This stability allows for a 2.2× end-to-end acceleration without the typical trade-off between inference speed and mask quality.

SAM-B at 25% density, it reaches 1.63× speedup versus 1.25× for SpargeAttention, while also outperforming token-merging methods such as ToMe (Bolya et al., 2022), which suffer from inaccurate merge-and-unmerge operations that degrade segmentation fidelity. For example, on SAM-L at 25% density, ToMe achieves 0.425 mAP, whereas SparseSAM preserves 0.468 mAP with up to 2.05× speedup.

4.5 High Quality Segmentation

To further stress-test compression robustness, we evaluate SparseSAM on HQ-44K (Ke et al., 2023), which focuses on thin and small-object segmentation. As shown in Figure 7, SparseSAM consistently preserves higher segmentation fidelity than both dynamic masking and token-reduction baselines under heavy compression. Thanks to deterministic Z-order interleaving, static A-shape masking, and residual-consistent MLP compression, SparseSAM maintains performance close to the dense model even at 0.25 density. For example, on COIFT at 0.3 density, SparseSAM achieves 0.945 mIoU, outperforming SpargeAttention (Zhang et al., 2025a) (0.919) and PISA (Li et al., 2026) (0.937), while on ThinObject5K-TE it reaches 0.892 mIoU versus 0.878 for PISA and 0.850 for SpargeAttention. Compared to token-merging approaches such as ToMe (Bolya et al., 2022), SparseSAM also provides a stronger accuracy–efficiency trade-off by preserving information flow through residual propagation; at 0.5 density on DIS5K-VD, it achieves 0.782 mIoU, substantially outperforming ToMe (0.692).

5 Conclusion

We introduced SparseSAM, a training-free framework for accelerating SAM and related segmentation models. By combining Stripe-Sort Attention with Residual-Consistency MLP, SparseSAM jointly reduces attention and MLP computation while preserving token identity. Across five datasets, it delivers up to 2× speedup and 2.8× memory reduction with minimal fidelity loss. For future work, there are several promising avenues. First, adaptive or hybrid permutations instead of fixing z-order as the current could better capture long-range dependencies in complex scenes while retaining

the efficiency of static sparsity. Second, layer-adaptive MLP routing may better handle changing token importance across the encoder. Finally, porting SparseSAM to mobile and edge backends, such as DSP (Sanaei et al., 2013) accelerators, would broaden its deployment.

References

Bolya, D., Fu, C.-Y., Dai, X., Zhang, P., Feichtenhofer, C., and Hoffman, J. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022.

Bolya, D., Huang, P.-Y., Sun, P., Cho, J. H., Madotto, A., Wei, C., Ma, T., Zhi, J., Rajasegaran, J., Rasheed, H., et al. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv preprint arXiv:2504.13181*, 2025.

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

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023a.

Dao, T. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023b.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Erkan, C. and Aksoy, S. Space-filling curves for modeling spatial context in transformer-based whole slide image classification. In *Medical Imaging 2023: Digital and Computational Pathology*, volume 12471, pp. 416–423. SPIE, 2023.

Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

Gonzalez, R. C. *Digital image processing*. Pearson education india, 2009.

Jia, D., Yuan, Y., He, H., Wu, X., Yu, H., Lin, W., Sun, L., Zhang, C., and Hu, H. Detrs with hybrid matching. *arXiv preprint arXiv:2207.13080*, 2022.

Jiang, H., Li, Y., Zhang, C., Wu, Q., Luo, X., Ahn, S., Han, Z., Abdi, A. H., Li, D., Lin, C.-Y., 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.

Ke, L., Ye, M., Danelljan, M., Liu, Y., Tai, Y.-W., Tang, C.-K., and Yu, F. Segment anything in high quality. *arXiv preprint arXiv:2306.01567*, 2023.

Kim, J., Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Marlin: FP16xINT4 LLM inference kernel that can achieve near-ideal 4x speedups up to medium batchsizes. https://github.com/IST-DASLab/marlin, 2023. Accessed: 2026-05-07.

Li, H., Shao, S., Zhong, W., Zhou, Z., Bai, L., Xiong, H., and Xie, Z. Pisa: Piecewise sparse attention is wiser for efficient diffusion transformers. *arXiv preprint arXiv:2602.01077*, 2026.

Liang, W., Yuan, Y., Ding, H., Luo, X., Lin, W., Jia, D., Zhang, Z., Zhang, C., and Hu, H. Expediting large-scale vision transformer for dense prediction without fine-tuning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35462–35477, 2022.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. *arXiv preprint arXiv:1405.0312*, 2014.

Liu, L., Zhang, S., Kuang, Z., Zhou, A., Xue, J.-H., Wang, X., Chen, Y., Yang, W., Liao, Q., and Zhang, W. Group fisher pruning for practical network compression. In *International Conference on Machine Learning*, pp. 7021–7032. PMLR, 2021.

Liu, Y., Chen, K., Liu, C., Qin, Z., Luo, Z., and Wang, J. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2604–2613, 2019.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., and Zhang, C. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017.

Lv, C., Chen, H., Guo, J., Ding, Y., and Liu, X. Ptq4sam: Post-training quantization for segment anything. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 15941–15951, 2024.

Morton, G. M. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company, 1966.

Nguyen, D. M., Tran, T. A., Nguyen, D., Xie, S., Nguyen, T. Q., Truong, M. T., Palenicek, D., Le, A. T., Barz, M., Nguyen, T., et al. Structsam: Structure-and spectrum-preserving token merging for segment anything models. *arXiv preprint arXiv:2603.07307*, 2026.

Ranjan, N. and Savakis, A. Mix-qsam: Mixed-precision quantization of the segment anything model. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 3280–3290, 2025.

Sanaei, Z., Abolfazli, S., Gani, A., and Buyya, R. Heterogeneity in mobile cloud computing: taxonomy and open challenges. *IEEE Communications Surveys & Tutorials*, 16(1):369–392, 2013.

Stücker, J., Hahn, O., Winkler, L., Adame, A. G., and Flöss, T. Jz-tree: Gpu friendly neighbour search and friends-of-friends with dual tree walks in jax plus cuda. *arXiv preprint arXiv:2604.05885*, 2026.

Tran, C., MH Nguyen, D., Nguyen, M.-D., Nguyen, T., Le, N., Xie, P., Sonntag, D., Zou, J. Y., Nguyen, B., and Niepert, M. Accelerating transformers with spectrum-preserving token merging. *Advances in Neural Information Processing Systems*, 37:30772–30810, 2024.

Tran, T. A., Nguyen, D. M., Tran, H.-C., Barz, M., Doan, K. D., Wattenhofer, R., Vien, N. A., Niepert, M., Sonntag, D., and Swoboda, P. How many tokens do 3d point cloud transformer architectures really need? *arXiv preprint arXiv:2511.05449*, 2025.

Wang, C., Wang, Z., Xu, X., Tang, Y., Zhou, J., and Lu, J. Q-vlm: Post-training quantization for large vision-language models. *Advances in Neural Information Processing Systems*, 37:114553–114573, 2024.

Xi, H., Yang, S., Zhao, Y., Xu, C., Li, M., Li, X., Lin, Y., Cai, H., Zhang, J., Li, D., et al. Sparse videogen: Accelerating video diffusion transformers with spatial-temporal sparsity. *arXiv preprint arXiv:2502.01776*, 2025.

Xiao, G., Tang, J., Zuo, J., Guo, J., Yang, S., Tang, H., Fu, Y., and Han, S. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*, 2024.

Xiong, Y., Varadarajan, B., Wu, L., Xiang, X., Xiao, F., Zhu, C., Dai, X., Wang, D., Sun, F., Iandola, F., et al. Efficientsam: Leveraged masked image pretraining for efficient segment anything. *arXiv preprint arXiv:2312.00863*, 2023.

Yang, S., Xi, H., Zhao, Y., Li, M., Zhang, J., Cai, H., Lin, Y., Li, X., Xu, C., Peng, K., et al. Sparse videogen2: Accelerate video generation with sparse attention via semantic-aware permutation. *arXiv preprint arXiv:2505.18875*, 2025.

Ye, Z., Chen, L., Lai, R., Lin, W., Zhang, Y., Wang, S., Chen, T., Kasikci, B., Grover, V., Krishnamurthy, A., et al. Flashinfer: Efficient and customizable attention engine for llm inference serving. *arXiv preprint arXiv:2501.01005*, 2025.

Zhang, C., Han, D., Qiao, Y., Kim, J. U., Bae, S.-H., Lee, S., and Hong, C. S. Faster segment anything: Towards lightweight sam for mobile applications. *arXiv preprint arXiv:2306.14289*, 2023.

Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L. M., and Shum, H.-Y. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. *arXiv preprint arXiv:2203.03605*, 2022.

Zhang, J., Xiang, C., Huang, H., Wei, J., Xi, H., Zhu, J., and Chen, J. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025a.

Zhang, W., Zhong, Y., Ando, S., and Yoshioka, K. Ahcptq: Accurate and hardware-compatible post-training quantization for segment anything model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22383–22392, 2025b.

Zhao, X., Ding, W., An, Y., Du, Y., Yu, T., Li, M., Tang, M., and Wang, J. Fast segment anything. *arXiv preprint arXiv:2306.12156*, 2023.

Zhou, C., Li, X., Loy, C. C., and Dai, B. Edgesam: Prompt-driven edge-aware segmentation with segment anything. *arXiv preprint*, 2023.

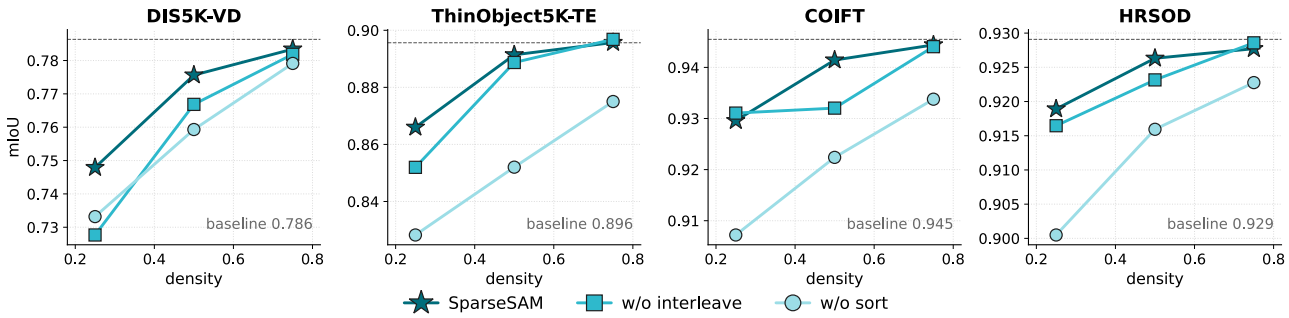


Figure 8. Performance breakdown with different permutation orders.

A Appendix

A.1 Finetune recovery

Despite being a training-free method by design, we also attempt to recover the performance of the model by fine-tuning only the MLP layers of SparseSAM for 15 minutes under low-density settings. As shown in Table 3, this simple adaptation consistently improves mIoU across datasets while keeping inference latency unchanged. At 25% density, mIoU increases from 74.79 to 76.92 on DIS5K-VD and from 91.79 to 93.01 on COIFT, with similar gains at 50% density. This demonstrates that minimal MLP fine-tuning effectively recovers accuracy under aggressive token reduction without sacrificing speed.

Table 3. **Performance Recovery via Fine-tuning.** Fine-tuning SparseSAM at low densities (25% and 50%) significantly closes the mIoU gap while maintaining the same accelerated inference latency.

Method	Density	DIS5K-VD		COIFT		HRSOD	
		mIoU	Time (ms)	mIoU	Time (ms)	mIoU	Time (ms)
Base Model	100%	78.63	55.02	94.55	50.23	92.91	54.67
• SparseSAM	25%	74.79	25.97	91.79	24.90	91.29	26.69
• SparseSAM (+FT)	25%	**76.92**	25.97	**93.01**	24.90	**91.96**	26.69
• SparseSAM (finetuned)	50%	77.56	29.16	93.12	27.79	92.40	29.31
• SparseSAM (+FT)	50%	**78.28**	29.16	**93.97**	27.79	**92.71**	29.31

Different permutation orders in Sparse Attention kernel. Our token permutation combines (i) *Gradient-Based Ordering*, which prioritizes high-importance Z-curve groups, and (ii) *Stripe-Sort Permutation*, which interleaves groups to maintain spatially uniform kept tokens. We evaluate two ablations on SAM-HQ ViT-L over HQ44K: *w/o interleave*, which removes interleaving while preserving ranking, and *w/o sort*, which preserves interleaving but removes content-aware ranking. As shown in Figure 8, all variants perform similarly at mild compression ($r = 0.75$), but differences become significant at lower densities. At $r = 0.25$, removing interleaving reduces performance by up to -0.020 mIoU on DIS5K-VD, particularly for thin structures. Removing ranking causes larger drops, especially on ThinObject5K (-0.038) and COIFT (-0.022), indicating that content-aware prioritization is the primary factor under aggressive compression.

Can residual-consistency MLP be used on other segmentation backbones? To further confirm the generality of our residual-consistency MLP, we apply SparseSAM to SAM2, where the tokens passing through each MLP layer are routed by the same residual-consistency mechanism. As shown in Table 4, our observation transfers cleanly to the Hiera encoder of SAM2, whose backbone likewise exhibits distinct token semantics across different regions of the image. We also attempted to adapt SparseSAM to other backbones that was train on a

Table 4. SparseSAM applied to SAM2 with the HQ-Hiera-L encoder.

	COIFT		DIS5K-VD		HRSOD		ThinObject5K	
r	mIoU	bIoU	mIoU	bIoU	mIoU	bIoU	mIoU	bIoU
1.00	0.951	0.908	0.753	0.670	0.909	0.854	0.903	0.819
0.70	0.951	0.907	0.754	0.674	0.912	0.858	0.906	0.825
0.50	0.951	0.906	0.761	0.677	0.916	0.861	0.913	0.834
0.30	0.946	0.901	0.767	0.678	0.925	0.867	0.912	0.824

contrastive learning objective and evaluated on Imagenet (Deng et al., 2009). For more information, please refer to A.6 and A.7.

A.2 Attention Kernel Pseudo Code

Algorithm 1 A-Shape Windowed FlashAttention-2 with Decomposed 2D Rel-Pos Bias

// Attention input with decomposed 2D bias $\mathbf{B}[q,k]=\mathbf{B}^H[q,k_{row}]+\mathbf{B}^W[q,k_{col}]$ *on a* $w\times w$ *grid*
Input: $\mathbf{Q}\in\mathbb{R}^{S_q\times d}$, $\mathbf{K},\mathbf{V}\in\mathbb{R}^{S_k\times d}$; $\mathbf{B}^H,\mathbf{B}^W\in\mathbb{R}^{S_q\times w}$, $w=\sqrt{S_k}$
// Window-major → spatial perms (for bias lookup); A-mask density ratio; softmax scale
$\sigma_Q:[S_q]\to[S_q]$, $\sigma_K:[S_k]\to[S_k]$; $r\in[0,1]$; $\tau=1/\sqrt{d}$
// Tile sizes, counts, slices, tile-local indices
Tiling: $B_{\text{row}},B_{\text{col}}$; $T_{\text{row}}=\lceil S_q/B_{\text{row}}\rceil$, $T_{\text{col}}=\lceil S_k/B_{\text{col}}\rceil$; $\mathbf{Q}_i\triangleq\mathbf{Q}[iB_{\text{row}}{:}(i{+}1)B_{\text{row}}]$, $\mathbf{K}_j,\mathbf{V}_j$; row $\in[0,B_{\text{row}})$, col $\in[0,B_{\text{col}})$
Output: $\mathbf{O}\in^{S_q\times d}$
// Active K-tiles: first $\lfloor rT_{col}\rfloor$ *columns plus the diagonal* $j=i$
$\mathcal{J}_i\leftarrow\{0,\dots,\lfloor rT_{\text{col}}\rfloor-1\}\cup\{i\}$
// Gather bias rows: query at tile-local row row has spatial position $\sigma_Q(iB_{row}{+}row)$
$\mathbf{B}_i^H[\text{row},p]\leftarrow\mathbf{B}^H[\sigma_Q(iB_{\text{row}}{+}\text{row}),p]$; $\mathbf{B}_i^W[\text{row},p]\leftarrow\mathbf{B}^W[\sigma_Q(iB_{\text{row}}{+}\text{row}),p]$
// Online-softmax state: per-row max, denom, 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**
  $\mathbf{S}\leftarrow\mathbf{Q}_i\,\mathbf{K}_j^\top$
  // Add bias; $\div\tau$ *cancels the* τ *in the softmax exponent below*
  $\mathbf{S}[\text{row},\text{col}]\mathrel{+}=\big(\mathbf{B}_i^H[\text{row},\,\sigma_K(\text{col})\div w]+\mathbf{B}_i^W[\text{row},\,\sigma_K(\text{col})\bmod w]\big)/\tau$
  $\mathbf{S}[\text{row},\text{col}]\leftarrow-\infty$ **where** $jB_{\text{col}}+\text{col}\geq S_k$
  // FA online-softmax update
  $\mathbf{m}'\leftarrow\max(\mathbf{m},\text{rowmax}\,\mathbf{S})$; $\widetilde{\mathbf{P}}\leftarrow\exp(\tau(\mathbf{S}-\mathbf{m}'))$; $\boldsymbol{\alpha}\leftarrow\exp(\tau(\mathbf{m}-\mathbf{m}'))$
  $\boldsymbol{\ell}\leftarrow\boldsymbol{\alpha}\odot\boldsymbol{\ell}+\text{rowsum}\,\widetilde{\mathbf{P}}$; $\mathbf{O}_{\text{acc}}\leftarrow\text{diag}(\boldsymbol{\alpha})\mathbf{O}_{\text{acc}}+\widetilde{\mathbf{P}}\mathbf{V}_j$; $\mathbf{m}\leftarrow\mathbf{m}'$
end for
$\mathbf{O}_i\leftarrow\text{diag}(\boldsymbol{\ell})^{-1}\mathbf{O}_{\text{acc}}$ =0

In SAM-based models, the attention operator includes a non-trivial relative positional bias that is directly added to the attention logits. To efficiently integrate a customized sparse attention kernel, we fuse this operation into a FlashAttention-2-style computation with decomposed 2D positional encoding, where the bias is factorized as $\mathbf{B}[q,k]=\mathbf{B}^H[q,k_{\text{row}}]+\mathbf{B}^W[q,k_{\text{col}}]$ on a $w\times w$ grid with $w=\sqrt{S_k}$.

The input features $\mathbf{Q}\in\mathbb{R}^{S_q\times d}$, $\mathbf{K},\mathbf{V}\in\mathbb{R}^{S_k\times d}$ are first reordered using spatial permutations σ_Q and σ_K, which improve memory locality under a window-major tiling scheme. Attention is computed in a block-wise manner over tiles $\mathbf{Q}_i$ and $\mathbf{K}_j$, with tile sizes B_{row} and B_{col}, and only a subset of key blocks $\mathcal{J}_i$ is evaluated according to a structured A-shaped sparsity mask (controlled by density ratio r), which is statically compiled and incurs no runtime overhead.

For each tile interaction, attention scores are computed as $\mathbf{S}=\mathbf{Q}_i\mathbf{K}_j^\top$, and the decomposed positional bias is added via efficient index-based lookup using $\mathbf{B}^H$ and $\mathbf{B}^W$ after applying σ_Q and σ_K. Entries outside valid key boundaries are masked to $-\infty$.

The kernel follows the FlashAttention-2 online softmax formulation, maintaining running statistics $\mathbf{m}$ (row-wise maxima), $\boldsymbol{\ell}$ (normalization terms), and $\mathbf{O}_{\text{acc}}$ (output accumulator). These are updated incrementally across key tiles to ensure numerical stability while streaming. The final output $\mathbf{O}_i$ is obtained by normalizing $\mathbf{O}_{\text{acc}}$ with $\boldsymbol{\ell}$ after processing all active blocks in $\mathcal{J}_i$.

A.3 Full MS-COCO Results

Table 5 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., 2022), and YOLOX (Ge et al., 2021). The results indicate that SparseSAM consistently outperforms existing attention-only

and token-reduction baselines in both segmentation accuracy (mAP) and inference efficiency. Notably, under joint attention and MLP compression, our method achieves state-of-the-art speedups—reaching up to 2.16× at 25% density—while maintaining mAP scores remarkably close to the dense base models. This demonstrates the robustness of SparseSAM's structured sparsity across various model scales and detection frameworks, significantly mitigating the drastic accuracy drops observed in merging-based approaches like ToMe (Bolya et al., 2022).

A.4 Full HQ44k Results

Table 6 presents an extensive evaluation of SparseSAM's performance on zero-shot object segmentation across four challenging datasets, including DIS5K-VD, COIFT, ThinObject5K-TE, and HRSOD from HQ-44K (Ke et al., 2023). The results demonstrate that SparseSAM maintains high segmentation accuracy while achieving superior speedup compared to existing training-free compression methods like SpargeAttention (Zhang et al., 2025a) and PISA (Li et al., 2026). Under joint attention and MLP compression at 25% density, the model reaches peak speedup factors exceeding 2.0× while suffering significantly less accuracy degradation than competitive methods such as ToMe (Bolya et al., 2022). The performance on ThinObject5K-TE particularly underscores the method's ability to preserve intricate spatial details even at low density levels.

The final tier of the evaluation (blue markers) demonstrates that a lightweight finetuning stage successfully bridges the performance gap inherent in high-compression regimes. By allowing the model to adapt to the structured sparsity patterns in both the attention and MLP layers, SparseSAM recovers mIoU to near-base levels while fully preserving its throughput advantages. Specifically, the finetuned configurations at 25% and 50% density show negligible performance degradation compared to the 100% density baseline. This underscores the efficacy of our approach in delivering a robust, hardware-efficient solution that maintains high-fidelity segmentation without sacrificing state-of-the-art inference acceleration.

A.5 Hardware-Aware Performance Analysis

Table 7 provides a detailed throughput speedup analysis of SparseSAM, comparing the performance of attention-only compression against joint attention and MLP compression across A100 and RTX 3090 GPUs. The results reveal that incorporating our Residual-Preserving MLP technique consistently enhances efficiency, yielding superior throughput compared to attention-only masking across all tested density ratios. Notably, the A100 architecture achieves a peak speedup of 2.05× at 25% sparsity, demonstrating higher hardware utilization than the RTX 3090. Furthermore, the speedup gains for the joint compression configuration scale positively with larger batch sizes, validating that our structured sparsity approach effectively maximizes computational throughput while maintaining the benefits of training-free acceleration.

A.6 MLP update behavior in other global task backbones

To understand how MLPs distribute work across an encoder, we visualize the per-token MLP update norm $\|\mathbf{\Delta}_{\mathrm{MLP}}\|_2 = \|\mathrm{MLP}(\mathrm{LN}(x))\|_2$ at four representative blocks of Perception Encoder (Bolya et al., 2025) (PE-Core-L14-336) and SAM-L on the same image. From Figure 9, it can be seen that the two backbones use their MLPs in markedly different ways. At block 0, behavior is broadly aligned: both models drive larger updates on the butterfly than on the background, indicating early object-aware encoding. Beyond the first block, the patterns diverge. PE, trained with an image-level contrastive objective, gradually loses spatial selectivity — by mid- and late-encoder, MLP updates are roughly uniform across all tokens, consistent with the network preparing a globally pooled embedding. SAM, trained for dense prediction, retains spatial selectivity throughout: foreground tokens consistently receive distinctive MLP updates, with the butterfly outline still clearly visible at block 23.

As expected from this observation, residual-consistency MLP fails catastrophically when applied to PE backbones, but performs surprisingly well on segmentation models.

A.7 Imagenet Results

Based on the observation in the previous section, we retain the Stripe-Sort Attention mechanism while replacing the MLP compression approach with a more traditional method.token merging used in (Bolya et al., 2022). Figure 10 compares Top-1 accuracy across different sparsity densities under two settings: without MLP compression (left) and with MLP compression (right). In both cases, SparseSAM consistently outperforms token-merging baselines (ToMe) and sparse attention (SpargeAttn) across all density levels, maintaining performance close to or above the dense baseline (0.835). Without MLP compression, SparseSAM remains highly stable even at low densities (0.25–0.4), where competing methods show noticeable degradation. As density increases, all methods converge toward the baseline, but SparseSAM consistently

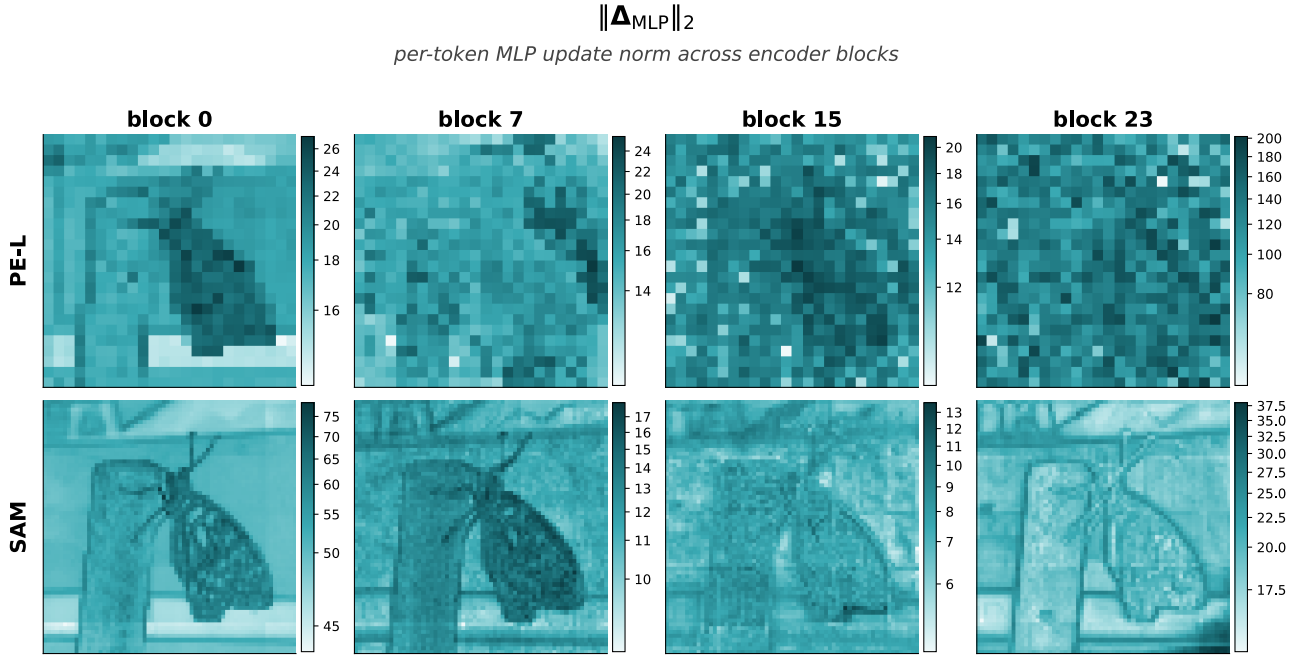


Figure 9. Per-token MLP update norm $||\Delta_{\text{MLP}}||_2$ at blocks 0/7/15/23 of PE-Core-L14-336 (top) and SAM-HQ ViT-L (bottom) on the same input. Reflecting their different pretraining inductive biases, the two backbones distribute MLP work in opposite ways. The first block looks similar across models — both focus updates on the butterfly — but in later layers, PE updates tokens uniformly across the image, while SAM stays spatially selective and keeps its MLP work concentrated on the foreground.

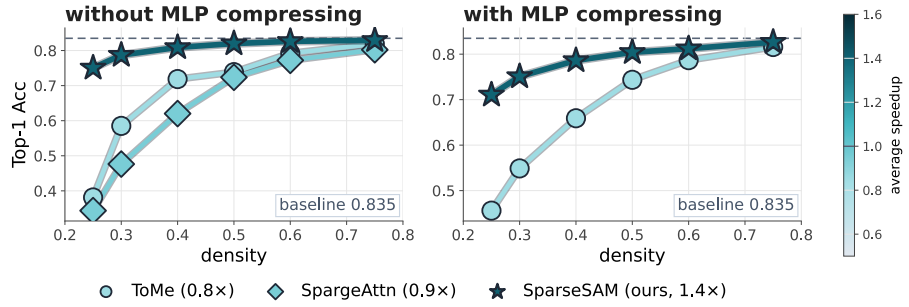


Figure 10. Results on ImageNet(Deng et al., 2009)

stays ahead. When MLP compression is enabled, SparseSAM retains nearly identical accuracy trends while achieving higher efficiency (1.4× speedup). Notably, even under stronger compression, it preserves robustness across all density regimes, whereas baselines remain significantly below the dense reference, especially at low densities.

Overall, the results highlight that SparseSAM 's stripe-sort attention mechanism maintains a better accuracy-efficiency trade-off, with minimal degradation under joint attention and MLP compression.

A.8 More segment output comparisons

Table 5. MS-COCO results. Markers indicate compression type: ○ attention-only and ● attention+MLP.

Detector	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
DINO	Base	0.468	100%	×1.00	0.495	100%	×1.00	0.499	100%	×1.00
	Attention-only compression									
	○ Sparge Attn	0.447	25%	×1.25	0.461	25%	×1.23	–	25%	–
		0.463	50%	×1.24	0.491	50%	×1.21	–	50%	–
	○ PieceWise Attn	0.455	25%	×0.73	0.483	25%	×0.68	0.486	25%	×0.69
		0.465	50%	×0.72	0.494	50%	×0.67	0.498	50%	×0.65
	○ SparseSAM (Ours)	**0.459**	25%	×**1.63**	**0.487**	25%	×**1.64**	**0.494**	25%	×**1.28**
		0.467	50%	×**1.55**	**0.494**	50%	×**1.57**	**0.499**	50%	×**1.22**
	Attention + MLP compression									
	● ToMe	0.403	50%	×0.83	0.425	50%	×0.86	0.430	50%	×0.80
	● GradToMe	0.306	25%	×0.86	0.318	25%	×0.89	0.318	25%	×0.94
		0.422	50%	×0.71	0.445	50%	×0.74	0.451	50%	×0.79
	● SparseSAM (Ours)	**0.451**	25%	×**2.15**	**0.468**	25%	×**2.05**	**0.472**	25%	×**1.59**
		0.459	50%	×**1.89**	**0.482**	50%	×**1.83**	**0.487**	50%	×**1.40**
HDETR	Base	0.402	100%	×1.00	0.424	100%	×1.00	0.427	100%	×1.00
	Attention-only compression									
	○ Sparge Attn	0.386	25%	×1.25	0.398	25%	×1.20	–	25%	–
		0.398	50%	×1.24	0.421	50%	×1.18	–	50%	–
	○ PieceWise Attn	0.391	25%	×0.72	0.415	25%	×0.65	0.418	25%	×0.68
		0.401	50%	×0.70	0.422	50%	×0.63	0.426	50%	×0.64
	○ SparseSAM (Ours)	**0.396**	25%	×**1.73**	**0.418**	25%	×**1.63**	**0.423**	25%	×**1.27**
		0.401	50%	×**1.66**	**0.423**	50%	×**1.56**	**0.427**	50%	×**1.20**
	Attention + MLP compression									
	● ToMe	0.351	50%	×0.83	0.369	50%	×0.86	0.373	50%	×0.78
	● GradToMe	0.271	25%	×0.86	0.280	25%	×0.88	0.280	25%	×0.93
		0.366	50%	×0.70	0.386	50%	×0.72	0.390	50%	×0.78
	● SparseSAM (Ours)	**0.389**	25%	×**2.16**	**0.403**	25%	×**2.05**	**0.406**	25%	×**1.58**
		0.394	50%	×**1.91**	**0.415**	50%	×**1.83**	**0.417**	50%	×**1.37**
YOLOX	Base	0.390	100%	×1.00	0.412	100%	×1.00	0.416	100%	×1.00
	Attention-only compression									
	○ Sparge Attn	0.374	25%	×1.25	0.386	25%	×1.20	–	25%	–
		0.386	50%	×1.24	0.409	50%	×1.18	–	50%	–
	○ PieceWise Attn	0.379	25%	×0.71	0.403	25%	×0.65	0.406	25%	×0.67
		0.388	50%	×0.69	0.410	50%	×0.63	0.414	50%	×0.63
	○ SparseSAM (Ours)	**0.384**	25%	×**1.70**	**0.406**	25%	×**1.63**	**0.411**	25%	×**1.29**
		0.390	50%	×**1.63**	**0.410**	50%	×**1.56**	**0.415**	50%	×**1.23**
	Attention + MLP compression									
	● ToMe	0.343	50%	×0.83	0.361	50%	×0.86	0.365	50%	×0.79
	● GradToMe	0.267	25%	×0.85	0.276	25%	×0.88	0.276	25%	×0.92
		0.356	50%	×0.69	0.377	50%	×0.72	0.381	50%	×0.77
	● SparseSAM (Ours)	**0.377**	25%	×**2.15**	**0.391**	25%	×**2.05**	**0.395**	25%	×**1.58**
		0.384	50%	×**1.92**	**0.403**	50%	×**1.83**	**0.405**	50%	×**1.40**

Table 6. Segmentation results. Markers indicate compression type: ○ attention-only, ● attention+MLP, and ● denotes finetuned models.

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 Sparsify									
○ 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
○ PieceWise Attention	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**
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
● GradToMe	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**	**×2.12**	**91.79**	**×2.02**	**85.95**	**×2.08**	**91.29**	**×2.05**
	50%	**77.56**	**×1.89**	**93.12**	**×1.81**	**87.82**	**×1.88**	**92.40**	**×1.87**
	75%	**78.43**	**×1.73**	**94.12**	**×1.64**	**89.21**	**×1.70**	**92.62**	**×1.71**
Finetuning after Joint Compression									
● SparseSAM (Ours)	25%	**76.92**	**×2.12**	**93.01**	**×2.02**	**88.05**	**×2.08**	**91.96**	**×2.05**
● SparseSAM (Ours)	50%	**78.28**	**×1.89**	**93.97**	**×1.81**	**89.41**	**×1.88**	**92.71**	**×1.87**

Table 7. Throughput speedup comparison on different hardware and sparsity levels. We evaluate SparseSAM under two configurations: attention-only compression and joint attention + MLP compression.

Batchsize	GPU	25% Sparsity		50% Sparsity		75% Sparsity	
		Attn. + MLP	Attn. Only	Attn. + MLP	Attn. Only	Attn. + MLP	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

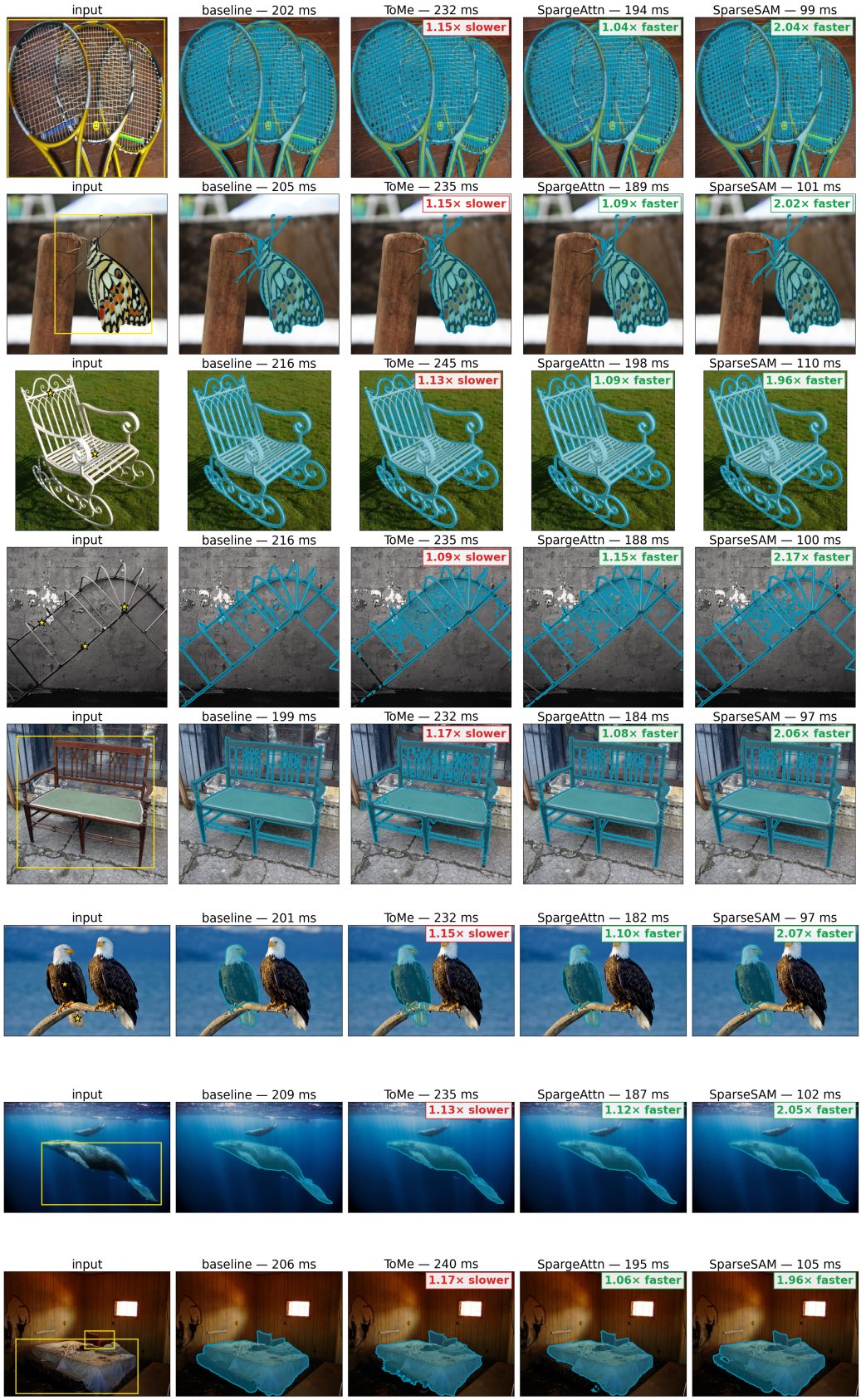


Figure 11. More output visualization