Leveraging Sparsity for Efficient Inference of High-Resolution Vision Foundation Models

Xin Xu[1]
xinx8@illinois.com
Jason Kuen[2]
kuen@adobe.com
Brian L. Price[2]
bprice@adobe.com
Kangning Liu[2]
kangningl@adobe.com
Zijun Wei[2]
zwei@adobe.com
Yu-Xiong Wang[1]
yxw@illinois.edu

[1] University of Illinois Urbana-Champaign, USA
[2] Adobe, USA

Abstract

Resolution scaling enhances the performance of Vision Transformers (ViTs) but incurs substantial computational costs due to the quadratic time complexity of self-attention. Our study reveals that sparsity naturally emerges in the attention maps of pre-trained vision encoders. Building on this observation, we introduce Sparse Vision Encoder (SVE), a post-training optimization framework that exploits sparsity to accelerate inference of vision encoders. SVE selectively applies sparsity in key layers, performs sparsity distillation to adapt models to sparse attention, and incorporates a lightweight predictor to eliminate redundant computations. Furthermore, we leverage cross-layer consistency in sparsity patterns, enabling efficient reuse of sparsity structures. Experiments on DINOv2, CLIP, and SAM2 show that SVE scales effectively for high-resolution encoding, delivering up to 80% speedup while preserving model performance. This demonstrates that SVE is a scalable and cost-effective solution for high-resolution visual representation. Our code is available at https://github.com/xuxalan/SVE.

1 Introduction

Recent advances in Vision Transformers (ViTs) [14] demonstrate the benefits of resolution scaling [31, 40], which enhances feature granularity and is essential for tasks such as semantic segmentation [41, 46], object detection [5, 50], and Vision-Language Model (VLM) encoders [3, 9]. In contrast to Convolutional Neural Networks (CNNs) [25], which face scalability limits due to their restricted receptive fields, ViTs offer a global receptive field, enabling them to effectively capture both long-range and local dependencies.

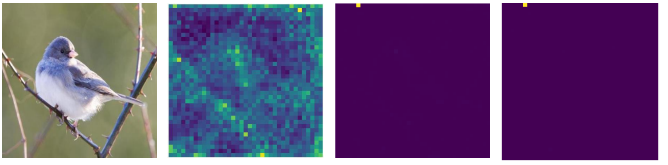

Figure 1: **Visualization of Attention Maps.** We plot the average attention maps in DINOv2 layers and have the following observations: **(a)** Sparsity emerges in deep layers but not in shallow layers. **(b)** The sparsity patterns are consistent across layers (*e.g.*, layers 18 and 21).

However, resolution scaling introduces significant computational challenges. The self-attention module has quadratic complexity, leading to a quartic increase in time cost as resolution grows. Even with optimizations such as FlashAttention [10, 11], attention remains the primary bottleneck due to the large number of tokens, limiting real-time applicability.

While prior methods aim to improve attention efficiency [28, 38], they require costly re-training and rely on fixed, heuristic sparsity patterns that lack adaptability. A more practical approach is to leverage pre-trained vision foundation models [32, 35, 36], apply lightweight high-resolution fine-tuning through parameter-efficient methods [19] or adapters [15], and then perform post-training efficiency optimization. This strategy not only adapts to dynamic attention patterns but also generalizes across diverse models, offering a flexible and cost-effective solution. Given the strong representational power of vision foundation models, improving their inference efficiency remains a critical yet underexplored research direction.

Our pilot study reveals that *sparsity naturally emerges* in the attention of vision encoders. In deeper layers, attention maps concentrate on a few tokens, while most show near-zero activations (Fig. 1). These "attention sinks" dominate feature representation, suggesting that computation can be selectively reduced without affecting performance. Despite the opportunities for sparsity optimization, efficiently extracting sparsity is challenging, as vision encoders process all tokens simultaneously, making direct attention computation costly. We show that lightweight modules can *accurately predict attention maps* without expensive matrix multiplications. Additionally, the observed *cross-layer consistency* in sparsity patterns allows reuse across layers, reducing redundancy and further improving efficiency.

Building on this observation, we propose ***S**parse **V**ision **E**ncoder (**SVE**)*, a framework that accelerates inference of pre-trained vision encoders by exploiting attention sparsity. SVE is guided by two objectives: preserving model performance and maximizing inference efficiency. To maintain accuracy, it begins with *layer selection*, identifying layers where sparsity can be introduced without degrading feature representations. Next, *sparsity distillation* adapts the model from dense to sparse attention, avoiding the need for full re-training. To reduce overhead, SVE incorporates a lightweight *sparsity projector* that directly estimates attention weights from input tokens, eliminating costly matrix multiplications for sparsity extraction. Finally, SVE leverages *cross-layer consistency* by reusing sparsity patterns across adjacent layers to further accelerate inference. Extensive experiments validate the effectiveness of SVE, showing that it achieves substantial inference speedups while preserving strong model performance. Our contributions can be summarized as follows:

- We present key insights into optimizing the inference efficiency of pre-trained vision encoders by exploiting attention sparsity, identifying both bottlenecks and strategies for acceleration.

- We propose SVE, a novel framework that accelerates inference of diverse pre-trained vision encoders while preserving accuracy.
- We validate our approach through extensive experiments on DINOv2 [32], CLIP [35], and SAM2 [36] across classification and segmentation tasks, demonstrating strong scalability to high-resolution encoding and achieving up to 80% speedup with minimal performance loss.

2 Related Work

Vision Foundation Model. Vision Foundation Models are large-scale, pre-trained models with general feature representations. Training techniques include contrastive learning in multi-modal models [20, 26, 35, 44] for tasks like zero-shot learning, and self-supervised learning [6, 7, 32] for learning without labels. Models like [24, 36] employ semi-supervised learning for promptable segmentation. Most of these models use the Vision Transformer (ViT) [14], built on the Attention module [39]. In this work, we focus on optimizing the backbone, the core component of vision foundation models.

Resolution Scaling. Scaling laws [22] are applied to model parameters to enhance performance [18, 45]. In the vision domain, researchers have also identified scaling laws in image resolution [31, 40], where increasing the number of tokens per image leads to performance improvements. However, a key challenge is that computational costs grow significantly with higher resolution, necessitating efficient inference techniques to maintain feasibility.

Sparse Attention. Sparse attention has gained growing interest [17, 29, 43] for its ability to reduce computational costs without compromising performance. In the computer vision domain, where the number of image tokens can scale significantly, various strategies have been proposed to enhance attention efficiency by introducing heuristic sparsity [27, 28, 37, 38]. However, reliance on pre-determined sparsity may limit adaptability and hinder optimal performance. [8] also exploits attention sparsity in the post-training stage but is restricted to window-based attention. In contrast, we target general-purpose ViTs.

LLM Inference. To accelerate inference in Large Language Models (LLMs), particularly with long-context inputs, researchers have explored sparse attention as a way to reduce computation and memory usage without compromising accuracy [1, 4, 16, 21, 42, 47]. However, applying these techniques to vision models remains challenging, as vision encoders process all tokens simultaneously, other than autoregressively. This key difference underscores the need for optimization strategies tailored to the unique characteristics of visual tasks.

3 Observations and Key Insights

We present several key empirical insights from pre-trained vision encoders that motivate the design of SVE. In this section, we use DINOv2 [32] as a representative example, with supporting evidence from other models provided in Sec. C.

3.1 Attention Sparsity

Consider the attention computation:

$$\text{Attention}(Q,K,V) = \text{softmax}(QK^T)\cdot V, \tag{1}$$

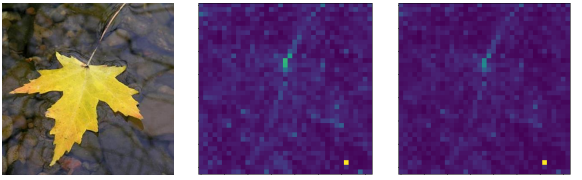

Figure 2: **Linear Projector Prediction.** We compare the average attention map m from layer 17 of DINOv2 with the predictions given by a linear projector applied to the input tokens of the same layer. The results show that the projector effectively identifies tokens with high activation.

where $Q, K, V \in \mathbb{R}^{L\times d}$, with L denoting the number of tokens and d the feature dimension.[1] The full attention map is given by $M = \text{softmax}(QK^T) \in \mathbb{R}^{L\times L}$, and the average attention map $m \in \mathbb{R}^L$ is defined as $m_i = \frac{1}{L}\sum_{j=1}^{L} M_{ji}$. If some values in m are close to zero, the corresponding Key-Value (KV) pairs contribute less to the output, which presents an opportunity to skip computation. When most values are near zero, the attention is considered sparse.

Sparsity in Deeper Layers. Fig. 1 shows average attention maps m in DINOv2, revealing that sparsity naturally emerges in certain layers (*e.g.*, layers 18 and 21). This supports the findings of [12] that Vision Transformers (ViTs) often rely on a small subset of tokens for representation. In contrast, the shallower layers (*e.g.*, layer 9) show more diffuse attention. We hypothesize that sparsity arises as the model shifts from local to global feature extraction, making deeper layers ideal for exploiting sparsity. Pruning less-contributive KV pairs in these layers could reduce redundancy and speed up inference without harming performance.

3.2 Sparsity Extraction

Sparsity patterns in vision encoders are highly dynamic and data-dependent, requiring online extraction during inference. The most direct way to obtain the average attention map m is to compute the full attention map M, which requires expensive matrix multiplications and softmax operations. As noted in [11], computing M is the main computational bottleneck and can incur even greater latency than full attention with FlashAttention [10, 11]. This underscores the need for a more efficient method of extracting sparsity patterns.

Sparsity Prediction. Since sparsity is encoded in the average attention map m, rather than the full attention map M, we can bypass the costly computation of M by introducing a lightweight projector. Our experiments show that a simple linear projector can accurately predict m from input tokens. Specifically, each token $x_i \in \mathbb{R}^d$ predicts its corresponding attention weight in m, *i.e.*, $m_i \in \mathbb{R}$, through the projector. As demonstrated in Fig. 2, predicting tokens with large values in m achieves a high recovery rate, enabling efficient sparsity prediction.

Layer Consistency. Another key observation is the consistency of the sparsity patterns in consecutive layers. Specifically, average attention maps persist over layers (*e.g.*, layers 18 and 21 in Fig. 1). This cross-layer similarity can reduce the cost of sparsity extraction (*e.g.*, computing QK^T or applying a linear projector) by allowing patterns to propagate from earlier layers, improving efficiency with minimal impact on performance.

[1] The scaling factor $1/\sqrt{d}$ is omitted for simplicity.

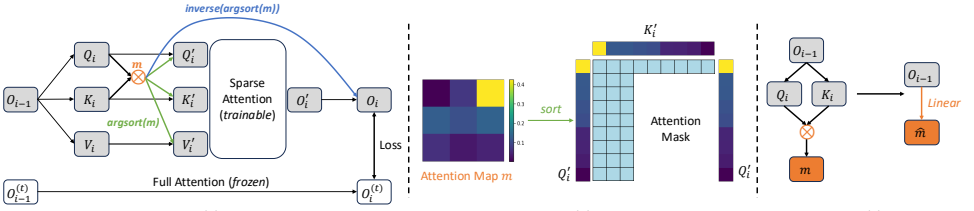

Figure 3: **SVE Overview. (a)** We use a pre-trained vision encoder as the teacher and apply layer-wise distillation to a student model. The student model employs sparse attention for adaptation and extracts sparsity patterns from the average attention map m. To enable efficient computation, we sort tokens based on m, compute sparse attention, and then restore the original order. In the illustration, green arrows denote sorting indices, and the blue arrow shows inverse indices. **(b)** We visualize the efficient sparse attention computation. **(c)** After distillation, we replace m with a lightweight linear projector to reduce overhead by predicting attention maps without heavy matrix multiplication.

4 Method

We propose SVE to accelerate inference in pre-trained vision encoders by efficiently incorporating sparse attention. Our framework begins with **layer selection** (Sec. 4.1) to identify layers suitable for sparsity. Then, **sparsity distillation** (Sec. 4.2) adapts the model to sparse attention while maintaining feature quality. Next, we train linear projections for fast **sparsity prediction** (Sec. 4.3). Finally, we reuse sparsity patterns across layers to further reduce computation, achieving a balance between efficiency and performance.

4.1 Sparse Layer Selection

As discussed in Sec. 3.1, sparsity appears only in a subset of layers in pre-trained vision encoders. To avoid large output deviations, we apply sparsity optimization selectively to layers that exhibit clear sparsity patterns. Specifically, we define a sparsity indicator $s^{(l)}$ for layer l based on its average attention map $m^{(l)}$, *i.e.*,

$$s^{(l)} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[m_i^{(l)} < m_{\max}^{(l)}/c], \tag{2}$$

where N is the number of tokens, $m_{\max}^{(l)}$ is the maximum value in $m^{(l)}$, and c is a threshold ($c = 10$ in our experiments). We compute these statistics on a calibration set and select layers where the sparsity indicator $s^{(l)}$ exceeds the threshold. The curves are shown in Fig. 4.

Sparse Attention Computation. A common sparse attention strategy is to compute attention using only a subset of Key-Value (KV) pairs. However, if the same subset is applied uniformly across all queries, tokens excluded from it lose the ability to interact with each other, thereby reducing the model's expressive capacity. To address this, we introduce a small subset of tokens with a global receptive field, enabling interaction with all KV pairs. These global tokens are selected according to the highest values in the average attention map. As shown in the cross-layer consistency analysis (Sec. 3.2), this subset remains stable across layers, ensuring that any two tokens can exchange information within two consecutive layers through global tokens.

To further enhance efficiency, we transform the token layout before performing attention computation. Specifically, we sort tokens based on their values in the average attention map

m, enabling efficient block-sparse attention [33]. This process is formulated as follows:

$$\text{index} = \text{argsort}(m), \tag{3}$$
$$Q' = \text{permute}(Q, \text{index}), \tag{4}$$
$$K' = \text{permute}(K, \text{index}), \tag{5}$$
$$V' = \text{permute}(V, \text{index}), \tag{6}$$
$$O' = \text{Attention}(Q', K', V', mask), \tag{7}$$
$$O = \text{permute}(O', \text{inverse}(\text{index})), \tag{8}$$

where the attention mask is defined as

$$mask(i,j) = \begin{cases} 0, & \text{if } i < n_q \text{ or } j < n_k \\ -\infty, & \text{otherwise} \end{cases}. \tag{9}$$

Here, n_q denotes the number of global tokens, and n_k represents the size of the compressed KV pairs. An illustration of this process is provided in Fig. 3.

4.2 Sparsity Distillation

Although sparsity naturally emerges in deeper layers of vision encoders, directly applying sparse attention causes performance degradation because the visual encoder is not trained with sparse computation. To address this issue, we employ a layer-wise distillation strategy on a calibration set. By minimizing deviations introduced by sparse computation, we aim to maintain the feature trajectory of the pre-trained vision encoder while enabling adaptation to sparsity. Specifically, we first collect the layer-wise outputs of the pre-trained vision encoder, denoted as $O_i^{(t)}$ for layer i. The student model follows the sparse computation process:

$$Q_i, K_i, V_i = f_{Q,i}(O_{i-1}), f_{K,i}(O_{i-1}), f_{V,i}(O_{i-1}) \tag{10}$$
$$m_i = \text{mean}\left(\text{softmax}(Q_i K_i^T)\right) \tag{11}$$
$$O_i = \text{SparseAttention}(Q_i, K_i, V_i, m_i) \tag{12}$$
$$O_i = f_{O,i}(O_i) \tag{13}$$

where $f_{Q,i}, f_{K,i}, f_{V,i}, f_{O,i}$ represents the projections for queries, keys, values, and outputs in layer i, respectively.[2] The loss is defined as

$$\mathcal{L}_{\text{distill}} = \sum_{i \in \mathcal{S}} \|O_i - O_i^{(t)}\|^2, \tag{14}$$

where $\mathcal{S}$ denotes the set of selected layers (Sec. 4.1). In the student model, the output from sparse attention at each layer is used as input for the subsequent layer, explicitly addressing error accumulation resulting from sparsity. By applying knowledge distillation, the student model effectively adapts to sparse computation while preserving the original feature trajectory of the pre-trained vision encoder.

[2] O_i is intentionally abused for simplicity, with normalization and MLP omitted for clarity.

4.3 Sparsity Prediction

As discussed in Sec. 3.2, a key challenge is efficiently extracting sparsity patterns. The standard approach requires computing the full attention map M, which incurs quadratic memory cost and can even be slower than full attention with FlashAttention [10, 11] due to memory overhead. To address this, we introduce a lightweight linear projection that predicts the average attention map m. Each token $x_i \in \mathbb{R}^d$ is projected to $\hat{m}_i \in \mathbb{R}$ and trained with a cross-entropy loss:

$$\mathcal{L}_{\text{proj}} = \text{CE}(\hat{m}, m). \tag{15}$$

This is applied per layer after sparsity distillation (Sec. 4.2). While less expressive than dot-product attention, the projection only needs to identify high-value tokens in m, enabling efficient sparsity prediction. See Sec. 5.3 for further discussion.

Layer Consistency. As discussed in Sec. 3.2, cross-layer consistency enables further latency reduction by reusing sparsity patterns predicted in earlier layers. This eliminates redundant computations and preserves a static token layout across layers, minimizing the cost of layout transformations. To optimize performance, we explore different configurations to determine which layers should predict sparsity patterns and which can simply reuse them.

5 Experiments

We demonstrate the effectiveness of SVE through a series of experiments. The experimental setup is introduced in Sec. 5.1, with implementation details provided in the supplementary material. The main results are presented in Sec. 5.2, followed by ablation studies in Sec. 5.3. Additional discussions are also included in the supplementary material.

5.1 Experiment Setup

We conduct experiments on three vision foundation models: DINOv2 [32], CLIP [35], and SAM2 [36]. For the experiments in Table 1, we largely follow the evaluation protocols from the original papers. The models are evaluated on both image classification and image segmentation tasks. For classification, we use the ImageNet-1K [13] dataset, and for segmentation, we use ADE20K [48, 49] and Pascal VOC [30]. DINOv2, as a general-purpose vision encoder, is evaluated on both tasks. CLIP, designed for image-level representations, lacks support for dense prediction and is therefore evaluated only on classification. SAM2, trained for promptable segmentation, is evaluated solely on segmentation tasks, with five prompt points provided per input. For the experiments in Table 2, we perform resolution scaling studies on the salient object segmentation task using HQSeg-44K [23]. HQSeg-44K contains high-resolution images and masks (exceeding 1K resolution), making it well-suited for high-resolution evaluation. Additional details are provided in Sec. A and Sec. B.

5.2 Main Result

Table 1 reports the results of SVE on CLIP, DINOv2, and SAM2 across image classification and segmentation tasks, using a default resolution of 1K (1024x1024). SVE delivers *substantial efficiency gains*, achieving a 15%–31% speedup that closely aligns with the reduction in FLOP. Despite these improvements, it *maintains model performance* with only minimal degradation. In some cases, such as SAM2 on Pascal VOC, it even outperforms

Models		ImageNet-1K [13]	ADE20K [48, 49]	Pascal VOC [30]	TFLOP	Speedup
DINOv2 [32] (1K)	Pre-trained	84.47	45.54	75.37	6.2	1.76
	SVE (Ours)	84.17	44.70	75.15	**5.5** (-11%)	**2.03** (+15%)
CLIP [35] (1K)	Pre-trained	73.99	-	-	5.8	1.97
	SVE (Ours)	72.76	-	-	**4.6** (-21%)	**2.58** (+31%)
SAM2 [36] (1K)	Pre-trained	-	61.22	74.22	0.6	14.62
	SVE (Ours)	-	60.81	74.50	**0.5** (-17%)	**16.80** (+15%)

Table 1: **Main Results.** We evaluate three models using only pre-trained weights and a calibration set for distillation. The default input resolution is 1K (1024x1024). The table reports *Acc@1* on ImageNet-1K and *IoU* on ADE20K and Pascal VOC. *Speedup* is measured in throughput with batch size 1 under fp32 precision. Our results show that SVE consistently improves efficiency across all three vision foundation models while preserving performance.

Models	Resolution	HQSeg-44K [34]	Speedup
DINOv2 [32]	518x518*	55.95	1.1x
	1036x1036	**58.24**	**1.2x**
SAM2 [36]	1024x1024*	86.96	1.2x
	2048x2048	**87.63**	**1.8x**

Table 2: **Resolution Scaling.** We report the *IoU* and *Speedup* (relative to baselines) of DINOv2 and SAM2 fine-tuned on HQSeg-44K dataset at both native resolution (∗) and higher resolutions. The results demonstrate the advantages of high-resolution inputs and highlight the greater efficiency gains achieved by SVE when scaling to higher resolutions.

Sparsity	Distillation	ADE20K [48, 49]
-	-	45.25
✓	-	38.41
-	✓	**45.47**
✓	✓	45.45

Table 3: **Ablations on Sparsity Distillation.** The table shows the result of SVE on DINOv2 with ADE20K dataset. *Sparsity* represents whether sparsity optimization is applied, and *Distillation* denotes whether knowledge distillation is applied.

the pre-trained baseline, indicating redundancy in the original computation. Overall, these results show that SVE effectively reduces inference cost while preserving feature quality.

Table 2 highlights the effect of resolution scaling. Models are evaluated at their native resolutions (518x518 for DINOv2 and 1024x1024 for SAM2) as well as at higher resolutions. As expected, increasing input resolution improves performance but also increases latency significantly. In these high-resolution settings, SVE proves particularly effective: as attention computation becomes the dominant bottleneck, it delivers *greater efficiency gains*. Notably, SVE achieves up to a 1.8x speedup with SAM2 at 2K resolution. This is impressive given that SAM2 uses the highly optimized Hiera [37] backbone for efficient image and video processing. These results demonstrate the *scalability* of SVE for high-resolution encoding tasks.

Overall, our results validate the importance of resolution scaling, that higher resolutions yield better performance. We also show that SVE provides a scalable and effective solution to reduce latency while preserving accuracy in such scenarios.

5.3 Ablation Study

The ablation studies follow the setup of Table 1, unless stated otherwise.

Sparse Layer Selection. We analyze how the sparsity indicator $s^{(l)}$ evolves with increasing layer depth. As shown in Fig. 4, across all three models, deeper layers generally exhibit clear sparsity patterns, although the final layers may deviate from this trend. The curves also suggest a rapid transition from diffuse to sparse attention maps, indicating that different layers may operate at distinct stages. At intermediate depths, the model appears to emphasize

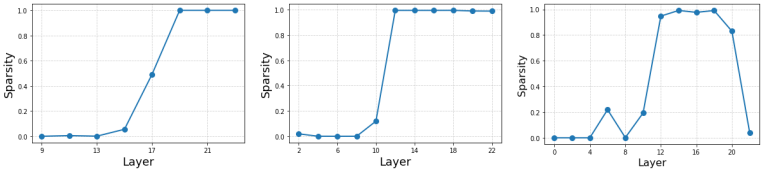

Figure 4: **Sparsity Indicator.** We show sparsity indicator $s^{(l)}$ for DINOv2, CLIP, and SAM2. The curve indicates that deeper layers often present sparsity.

Methods	Latency (ms)	Speedup
Full Attention	17.66	1.0x
Sparse Attention w/ QK^T	22.83	0.8x
Sparse Attention w/ predictor	7.10	2.5x
Sparse Attention w/ speedup	**6.21**	**2.8x**

Table 4: **Necessity of Efficient Sparsity Prediction.** The table shows the profiling results of a single DINOv2 attention module under fp32 precision. *Sparse Attention w/ QK^T* refers to sparse attention where sparsity patterns are derived from dot-product. *Sparse Attention w/ Predictor* denotes sparse attention where a linear projector predicts sparsity patterns. *Sparse Attention w/ Speedup* leverages sparsity patterns from previous layers to improve computational efficiency.

global features over regional ones. Furthermore, visualizations of individual samples show that this transition occurs consistently across all cases. Additional discussions on attention sparsity and sparse layer selection are provided in Sec. C.

Sparsity Distillation. We first compare the performance of the model under full-attention computation. As shown in Table 3, the distilled model with full attention achieves an IoU of 45.47, closely matching the pre-trained model's IoU of 45.25. This confirms that distillation effectively preserves learned representations. We then evaluate the pre-trained model with sparse attention (IoU of 38.41), where each selected layer is independently compressed using the "ground-truth" average attention map to define the sparsity pattern. Although sparse patterns naturally emerge, directly applying sparsity optimization to a pre-trained vision encoder causes a significant performance drop due to changes in attention computation. By contrast, incorporating distillation mitigates this issue, enabling the model to recover its performance. These results indicate that the model can successfully adapt to sparse computation, improving efficiency without compromising accuracy.

Sparsity Prediction. We first emphasize the necessity of efficient sparsity prediction. As shown in Table 4, computing sparsity directly with dot-product attention introduces substantial computational overhead, whereas a lightweight linear projector provides a far more efficient alternative. To evaluate its effectiveness, Fig. 5 presents the recovery curve of a linear projector trained on a single DINOv2 layer. The projector is tasked with predicting the top-1/8 tokens with the highest values in the average attention map, which correspond to the compressed KV pairs. Our results show that selecting the top-1/8 tokens achieves a recall of 86%, while expanding to the top-1/4 tokens raises recall above 99%. These findings demonstrate that the linear projector can accurately capture the most important tokens, enabling efficient and reliable sparsity prediction.

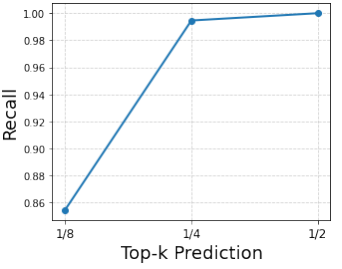

Figure 5: **Recall Curve of a Linear Projector.** We plot the recall of the linear projector trained on DINOv2 layer 17. In this task, the projector predicts the top-1/8 tokens with the highest values in the average attention map. The results indicate that it can reliably recover these tokens.

6 Conclusions

We present Sparse Vision Encoder (SVE), a post-training optimization framework that exploits the inherent sparsity of pre-trained vision foundation models to accelerate inference efficiently. SVE integrates sparse layer selection, sparsity distillation, and lightweight sparsity prediction to reduce computational overhead while maintaining accuracy. Across diverse encoders, including DINOv2, CLIP, and SAM2, our experiments show that SVE consistently achieves substantial efficiency gains, delivering up to 80% speedup in high-resolution scenarios with negligible accuracy loss.

These results not only validate sparsity as a scalable principle for efficient visual computation but also demonstrate that post-training optimization can extend the utility of powerful pre-trained models without costly re-training. By enabling practical high-resolution inference, SVE broadens the applicability of vision foundation models to real-world scenarios such as dense perception, large-scale image retrieval, and interactive systems. We believe this work opens promising directions for leveraging sparsity to make foundation models more efficient, adaptable, and widely deployable.

Acknowledgment

This work was supported in part by NSF Grant 2106825 and NIFA Award 2020-67021-32799. This work used computational resources, including the NCSA Delta and DeltaAI supercomputers through allocations CIS230012, CIS240133, and CIS240387 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, as well as the TACC Frontera supercomputer and Amazon Web Services (AWS) through the National Artificial Intelligence Research Resource (NAIRR) Pilot.

References

[1] Shantanu Acharya, Fei Jia, and Boris Ginsburg. Star Attention: Efficient llm inference over long sequences. *arXiv preprint arXiv:2411.17116*, 2024.

[2] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. PyTorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *ASPLOS*, 2024.

[3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[4] William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. Reducing transformer key-value cache size with cross-layer attention. In *NeurIPS*, 2024.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020.

[8] Xuanyao Chen, Zhijian Liu, Haotian Tang, Li Yi, Hang Zhao, and Song Han. Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer. In *CVPR*, 2023.

[9] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024.

[10] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *ICLR*, 2024.

[11] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS*, 2022.

[12] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *ICLR*, 2024.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[15] Qihang Fan, Quanzeng You, Xiaotian Han, Yongfei Liu, Yunzhe Tao, Huaibo Huang, Ran He, and Hongxia Yang. ViTAR: Vision transformer with any resolution. *arXiv preprint arXiv:2403.18361*, 2024.

[16] Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, et al. MoA: Mixture of sparse attention for automatic large language model compression. *arXiv preprint arXiv:2406.14909*, 2024.

[17] Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. SeerAttention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024.

[18] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[19] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

[20] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, 2021.

[21] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir Abdi, Dongsheng Li, Chin-Yew Lin, et al. MInference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. In *NeurIPS*, 2024.

[22] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[23] Lei Ke, Mingqiao Ye, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, Fisher Yu, et al. Segment anything in high quality. In *NeurIPS*, 2023.

[24] 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 *ICCV*, 2023.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[26] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

[27] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *ECCV*, 2022.

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[29] Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. MoBA: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

[30] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[31] Duy-Kien Nguyen, Mahmoud Assran, Unnat Jain, Martin R Oswald, Cees GM Snoek, and Xinlei Chen. An image is worth more than 16x16 patches: Exploring transformers on individual pixels. *arXiv preprint arXiv:2406.09415*, 2024.

[32] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[33] PyTorch. FlexAttention: The flexibility of PyTorch with the performance of FlashAttention. https://pytorch.org/blog/flexattention, 2024.

[34] Lu Qi, Jason Kuen, Weidong Guo, Tiancheng Shen, Jiuxiang Gu, Jiaya Jia, Zhe Lin, and Ming-Hsuan Yang. High-quality entity segmentation. *arXiv preprint arXiv:2211.05776*, 2022.

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[36] 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. *arXiv preprint arXiv:2408.00714*, 2024.

[37] Chaitanya Ryali, Yuan-Ting Hu, Daniel Bolya, Chen Wei, Haoqi Fan, Po-Yao Huang, Vaibhav Aggarwal, Arkabandhu Chowdhury, Omid Poursaeed, Judy Hoffman, et al. Hiera: A hierarchical vision transformer without the bells-and-whistles. In *ICML*, 2023.

[38] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. MaxViT: Multi-axis vision transformer. In *ECCV*, 2022.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[40] Feng Wang, Yaodong Yu, Guoyizhe Wei, Wei Shao, Yuyin Zhou, Alan Yuille, and Cihang Xie. Scaling laws in patchification: An image is worth 50,176 tokens and more. *arXiv preprint arXiv:2502.03738*, 2025.

[41] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021.

[42] Lijie Yang, Zhihao Zhang, Zhuofu Chen, Zikun Li, and Zhihao Jia. TidalDecode: Fast and accurate llm decoding with position persistent sparse attention. *arXiv preprint arXiv:2410.05076*, 2024.

[43] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, YX Wei, Lean Wang, Zhiping Xiao, et al. Native Sparse Attention: Hardware-aligned and natively trainable sparse attention. *arXiv preprint arXiv:2502.11089*, 2025.

[44] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

[45] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *CVPR*, 2022.

[46] Bowen Zhang, Zhi Tian, Quan Tang, Xiangxiang Chu, Xiaolin Wei, Chunhua Shen, et al. SegViT: Semantic segmentation with plain vision transformers. In *NeurIPS*, 2022.

[47] Xuechen Zhang, Xiangyu Chang, Mingchen Li, Amit Roy-Chowdhury, Jiasi Chen, and Samet Oymak. Selective Attention: Enhancing transformer through principled context control. In *NeurIPS*, 2024.

[48] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017.

[49] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. In *IJCV*, 2019.

[50] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.