This paper has been accepted for publication at the
IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa, 2024. ©IEEE

Revisiting Token Pruning for Object Detection and Instance Segmentation

Yifei Liu Mathias Gehrig Nico Messikommer Marco Cannici Davide Scaramuzza
Robotics and Perception Group, University of Zurich, Switzerland
{yifei.liu@, mgehrig@ifi., nmessi@ifi, cannici@ifi., sdavide@ifi.}uzh.ch

Abstract

Vision Transformers (ViTs) have shown impressive performance in computer vision, but their high computational cost, quadratic in the number of tokens, limits their adoption in computation-constrained applications. However, this large number of tokens may not be necessary, as not all tokens are equally important. In this paper, we investigate token pruning to accelerate inference for object detection and instance segmentation, extending prior works from image classification. Through extensive experiments, we offer four insights for dense tasks: (i) tokens should not be completely pruned and discarded, but rather preserved in the feature maps for later use. (ii) reactivating previously pruned tokens can further enhance model performance. (iii) a dynamic pruning rate based on images is better than a fixed pruning rate. (iv) a lightweight, 2-layer MLP can effectively prune tokens, achieving accuracy comparable with complex gating networks with a simpler design. We assess the effects of these design decisions on the COCO dataset and introduce an approach that incorporates these findings, showing a reduction in performance decline from ∼1.5 mAP to ∼0.3 mAP in both boxes and masks, compared to existing token pruning methods. In relation to the dense counterpart that utilizes all tokens, our method realizes an increase in inference speed, achieving up to 34% faster performance for the entire network and 46% for the backbone.
Code: https://github.com/uzh-rpg/svit/

1. Introduction

Transformers and multi-head self-attention [52] have revolutionized the field of computer vision. Since their first introduction, Vision Transformers (ViTs) [16, 37, 51] have quickly become the leading model architecture for a number of vision tasks, including image classification [21, 37, 51], object detection [4, 9, 33, 37, 75], semantic segmentation [10, 37, 60, 74], and others [23, 29]. Their unique ability to perform global reasoning through pair-wise token attention is, however, both a strength and a weakness. Although it enhances the representational power of these architectures, it also leads to a significant increase in computational footprint, limiting the adoption of ViTs in resource-constrained settings.

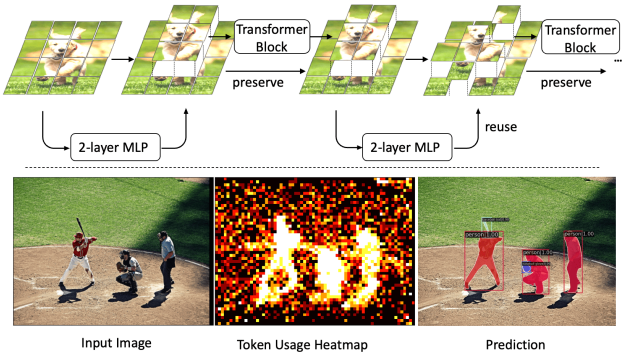


Figure 1. Top: high-level workflow of SViT, the MLP selectively chooses tokens to be processed in the transformer block, and the pruned tokens are preserved in feature maps and can be reactivated in later layers. Bottom: the token useage heatmap represents the number of layers using the tokens, and shows that the computational distribution highly aligns with interested objects.

A viable strategy for mitigating the substantial computational demands involves leveraging input-aware inference to prune less critical features within the input space. While this strategy has previously been applied to CNNs [20], resulting in improved FLOP measurements, the intrinsic regularity of convolution operations makes it difficult to obtain noticeable speedup on hardware. However, the advent of ViTs paves the way for input-space pruning, as the MLPs in ViTs operate pointwise and self-attention inherently accommodates an arbitrary number of tokens. Consequently, pruning tokens can readily attain remarkable speedup without necessitating any additional hardware adaptations.

Initial investigations in the domain of token pruning have encompassed the utilization of gating networks to identify less significant tokens [32, 39, 44] or eliminating tokens receiving minimal attention from the class token [19, 34, 61]. These approaches, while having demonstrated their effectiveness, were only applied to classification and have yet to be applied to other tasks such as object detection and instance segmentation. To the best of our knowledge, the exploration of token pruning in the context of dense tasks

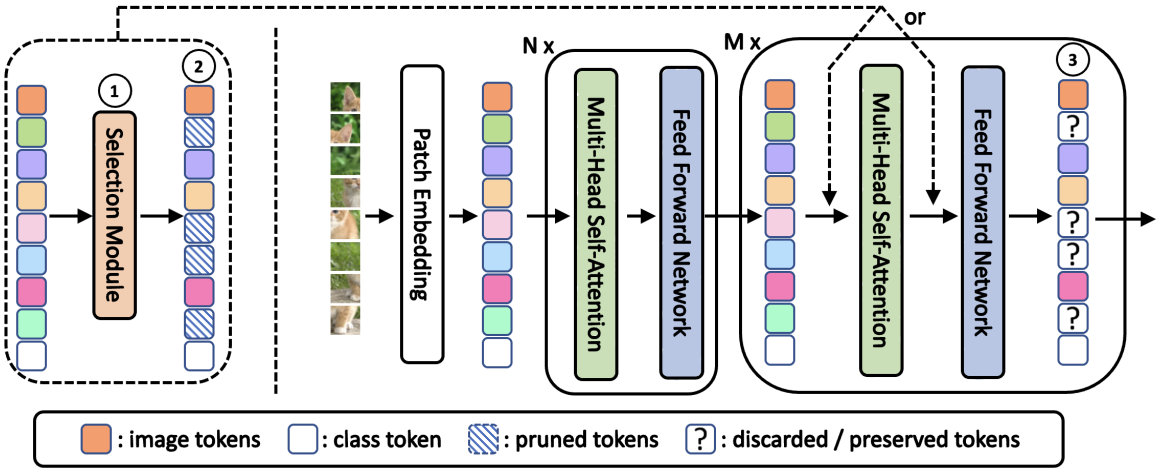


Figure 2. A high-level comparison of the overall workflows for various token pruning methods. ① Selection Module: may utilize a gating module (before self-attention) or be attention-based (after self-attention). ② Number of pruned tokens: can be either dynamic or fixed. ③ Treatment of pruned tokens: either removing or preserving them. If they are preserved within feature maps, there is an additional option to reactivate them.

remains still notably scarce (Section 2)[1]. In this paper, we investigate token pruning for object detection and instance segmentation on isotropic vision transformers, with the aim of bridging the gap between classification and dense tasks. During our preliminary experiments, we adapted prior methods to dense tasks and discovered they have apparent performance loss (Section 4.1). With extensive experiments, we identified four key insights that are beneficial for improving model performance and simplifying model designs (Section 3.2), leading to a method that outperforms previous state-of-the-art token pruning methods by a significant margin on object detection and instance segmentation (Section 4.2). Our insights are as follows:

Token preserving on dense tasks. Unlike classification, where pruned tokens can be removed permanently, dense prediction tasks benefit from preserving them in feature maps for subsequent utilization by the detection head.

Token reactivation as needed. In addition to preserving them, reactivating pruned tokens in the backbone on demand can improve model performance by adapting to layer-wise attention and recovering mis-pruned tokens for better robustness. A token once pruned has the flexibility to be reused at any subsequent layer, including the immediately succeeding one.

Pruning with a dynamic rate. The concept of a dynamic pruning rate, previously introduced for classification tasks in [19, 32, 63], optimizes model performance within the same computation resource by allocating more tokens for complex images and fewer for simple images. It gains additional efficacy when integrated with token reactivation on dense prediciton tasks.

2-layer MLP is sufficient. A lightweight MLP is sufficient to select which tokens should be pruned, delivering almost the same accuracy as more complex gating networks [32, 44] used for classification.

We evaluate these design choices and build upon them to introduce a straightforward model to selectively prune tokens, which we refer to as SViT. We demonstrate that this model surpasses previous state-of-the-art token pruning models by reducing loss in mAP from ∼1.5 to ∼0.3 for both boxes and masks, and accelerates the inference speed of the dense counterpart by up to 34% for the whole network and 46% for the backbone.

2. Related Work

Vision Transformer Originating in the NLP community [3, 15, 36, 42], Transformers [52] have lately acquired popularity also in the field of computer vision for their ability to capture long-range relations [23, 29]. The seminal work on Vision Transformers (ViTs) [16] demonstrated state-of-the-art classification performance, when pre-trained on large-scale datasets.

Since then, several improvements have been proposed to the ViT architecture, including improved tokens' aggregation schemes [24, 27, 51, 66], multi-scale hierarchical designs [7, 13, 37, 56–59], and hybrid architectures combining CNNs [65, 70, 71] . Apart from design improvements, researchers have also investigated their use in more complex vision tasks [9, 33, 37, 64, 72, 73]. This paper fits in between these two lines of research, as we not only focus on archi-

[1]The most related work on dense tasks is an extended version [43] of DynamicViT. However, it focuses on skipping MLPs in hierarchical models for dense tasks.

tecture design choices, but also extend their usage to dense prediction tasks such as object detection and instance segmentation.

Transformer Acceleration Various methods have been explored for optimizing Transformers' high computational cost, including designing alternative lightweight attention formulations [11, 28, 31, 46, 50, 54, 68], removing unnecessary network modules [17, 40, 53] approximating attention multiplications with low-rank decompositions [6, 12, 55], distilling knowledge into a more efficient student network [48, 51, 69], and extending network quantization techniques for Transformers [1, 18, 30, 49, 67]. Furthermore, acceleration techniques specific to ViTs have been proposed [19, 34, 41, 44, 47, 61, 63] by exploiting the redundancy in the input patches to early drop tokens for saving computation.

Input Space Pruning As not all regions in the input image are equally important, pruning redundant areas can save computation without apparent accuracy loss. Spatially ACT [20] prunes pixels for CNNs. Numerous token pruning methods for ViTs have been developed on classification, including using gating networks [32, 39, 44], attention scores [19, 34, 61], reinforcement learning [41] and others [2, 47, 63]. Among them, ToMe [2] proposes to merge tokens rather than remove them. A few works also consider dense tasks: SparseViT [8] prunes coarse windows for pyramid transformers, while we prune finer-grained tokens for isotropic transformers. SparseDETR [45] focuses on improving the efficiency of DETR [4] architecture, while we focus on improving transformer-based backbones. STViT-R [5] sparsifies tokens by repeatedly clustering them into a few semantic tokens and restoring the spatial resolution, while we keep the spatial resolution with detailed position information.

3. Token Pruning on dense prediction tasks

3.1. Revisit prior token pruning approaches

We review the majority of token pruning techniques by illustrating the high-level distinctions in their workflows. As shown in Table 1, these approaches can be classified along four dimensions: the selection module, use of dynamic pruning rate, preservation of pruned tokens, and reactivation of pruned tokens.

The overall workflow of token pruning is depicted in Figure 2 and can be summarized as follows: initially, the input image is partitioned into non-overlapping patches, which are linearly transformed into tokens and subsequently processed by the initial ViT blocks to obtain comprehensive enough feature representations. Then, token selection modules are introduced to identify tokens for pruning, consequently accelerating computations due to the reduced number of tokens. Note that, here, acceleration comes out-of-the-box as self-attention can adaptively process fewer number of tokens without any modification.

3.2. Insights and Observations

Preserve pruned tokens within feature maps. A notable distinction between classification and dense prediction tasks is how the pruned tokens should be treated. In classification, token pruning methods often remove tokens permanently because pruned tokens will no longer influence the result, as the classification solely depends on the class token, which is always kept.

However, on dense prediction tasks, the pruned tokens can still be utilized by subsequent detection heads, even if they are no longer updated in the backbone. Therefore, it is beneficial to keep the already computed features for pruned tokens for later use. When pruned tokens are not preserved, we recover a dense feature map by placing remaining tokens in their original location, and zero-pad the pruned ones [62]. Preserving pruned tokes, instead, built the feature map incrementally, each time replacing updated tokens, but keeping pruned ones unchanged. Preserving pruned tokens can be as fast as removing them (see Table 2), and improves model performance on various models on dense tasks.

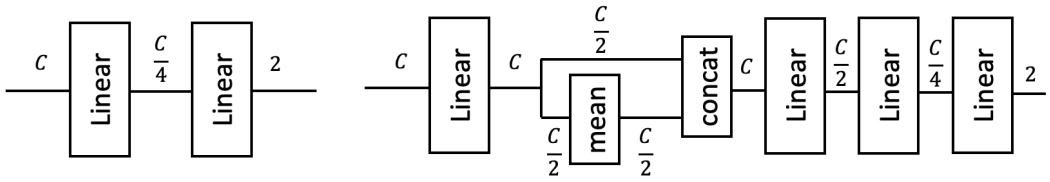
Reuse preserved tokens on demand. As pruned tokens are preserved within feature maps, it is natural to consider whether they should be used again. In the scope of this paper, "token preserving" refers to the utilization of pruned tokens only by detection heads, whereas "token reactivation" implies that these tokens can also be reintroduced into the backbone for subsequent layers. A counterargument to token reactivation may say that ViT should prioritize allocating computing resources to informative tokens as much as possible [61], and reactivating pruned tokens may potentially undermine this principle. However, the definition of "informative" may vary across different layers since ViT could concentrate on distinct regions at each layer, see supplementary material. Thus, the ability to reactivate pruned tokens accommodates the distinct attentions of various ViT layers, enabling the model to prioritize its current focus before returning to other relevant tokens in subsequent blocks. Additionally, this makes pruning more robust, as mis-pruned tokens have the opportunity to become active again, see Figure 5b. Ultimately, these advantages lead to a more effective overall utilization of tokens under the same token usage per block. In Section 4, we allow the model to learn whether and when to reuse pruned tokens by itself, and show that this ability can improve model accuracy by a 0.4 box AP and 0.3 mask AP.

A 2-layer MLP can substitute complex gating networks for pruning tokens. Prior token pruning approaches tend

Selection Module	Dynamic Pruning Rate	Preserve Pruned Tokens	Reactivate Pruned Tokens	Model
gating module	✓	✓	✓	SViT (Ours)
gating module	✗	✗	✗	DynamicViT [44]
attention-based	✓	✗	✗	ATS [19]
attention-based	✗	✓	✗[2]	Evo-ViT [61]
attention-based	✗	✗	✗	EViT [34]

[2] While Evo-ViT is theoretically capable of reusing tokens by design, it tends to use the same tokens throughout the network, details in the supplementary material.

Table 1. A high-level examination of token pruning techniques. The gating module refers to an auxiliary compact network, designed to predict the tokens to be pruned. Attention-based selection involves pruning tokens that receive minimal attention from the class token.



(a) Two-layer gating network (b) Gating network used in DynamicViT

Figure 3. Different types of gating networks for predicting tokens to be pruned. (a) is used by SViT, and (b) is used by DynamicViT [44]. Normalization and activation functions are omitted for conciseness. C represents the dimension of tokens.

to employ complex gating networks for predicting the tokens to be pruned. In DynamicViT [44], several MLPs are utilized in conjunction with mean and concatenation operations to learn both token-specific and global information for determining which tokens should be pruned, as illustrated in Figure 3b. SPViT [32], introduces a more intricate gating network that incorporates an additional head branch to calculate score weights for each individual head. However, in Section 4.1, our study shows that a simple 2-layer MLP in Figure 3a works equally well and simplifies the architecture design.

A dynamic pruning rate is better than a fixed pruning rate. Several studies [19, 32, 63] in the context of classification have implemented dynamic pruning rates, adaptively pruning varying numbers of tokens based on the input images during inference. We further validate its effectiveness in the context of object detection and instance segmentation, and show it is one key components to achieve optimal performance in Section 4.

3.3. SViT: Selective Vision Transformer

In light of the insights, we introduce the Selective Vision Transformer (SViT), a simple yet effective token pruning model, which seamlessly integrates all prior findings.

SViT is depicted in Figure 1. For the selection module, we employ a 2-layer perceptron followed by Gumbel Softmax [26, 38] to make the discrete decision differentiable, as shown in Eq (1). By placing this selection module before the entire ViT block, we facilitate acceleration for both self-attention and the MLP in the transformer encoder:

$$\begin{aligned} &\mathbf{p} = \text{Softmax}(\text{MLP}(\mathbf{x})) \in \mathbb{R}^{N\times 2},\ \mathbf{x} \in \mathbb{R}^{N\times C} \\ &\mathbf{M} = \text{GumbelSoftmax}(\mathbf{p}) \in \{0,1\}^{N}, \\ &\mathbf{x} \leftarrow \mathbf{M} \odot \text{ViTBlock}(\mathbf{x}, \mathbf{M}) + (1-\mathbf{M}) \odot \mathbf{x} \end{aligned} \tag{1}$$

where $\mathbf{x}$ represents the input tokens, $\mathbf{p}$ is the intermediate sampling probability, $\mathbf{M}$ signifies token masks, and $\odot$ is Hadamard product. The MLP transforms token dimensions from C to $\frac{C}{4}$, and $\frac{C}{4}$ to 2. The ViTBlock takes in the masks $\mathbf{M}$ and eliminates the influence of pruned tokens on other tokens during training by setting the corresponding columns in the attention matrix to 0. During inference, we simply gather the active tokens, feed them to the current ViT Block, and then scatter them back to the previous feature map.

For controlling the number of pruned tokens, similar to [32], we use a dynamic pruning ratio loss during training as in Eq (2):

$$\begin{aligned} &\mathcal{L}_{dynamic} = \frac{1}{L}\sum_{l\in L}((\frac{1}{BN}\sum_{b\in B}\sum_{n\in N}\mathbf{M}_n^{b,l}) - \mathbf{t}^l)^2, \\ &\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda\mathcal{L}_{\text{dynamic}} \end{aligned} \tag{2}$$

where $\mathbf{M}_n^{b,l}$ denotes the mask at batch b and layer l for the n-th token, $\mathbf{t}^l$ represents the target keeping ratio at layer l, and λ is a hyper-parameter to weight losses. It is

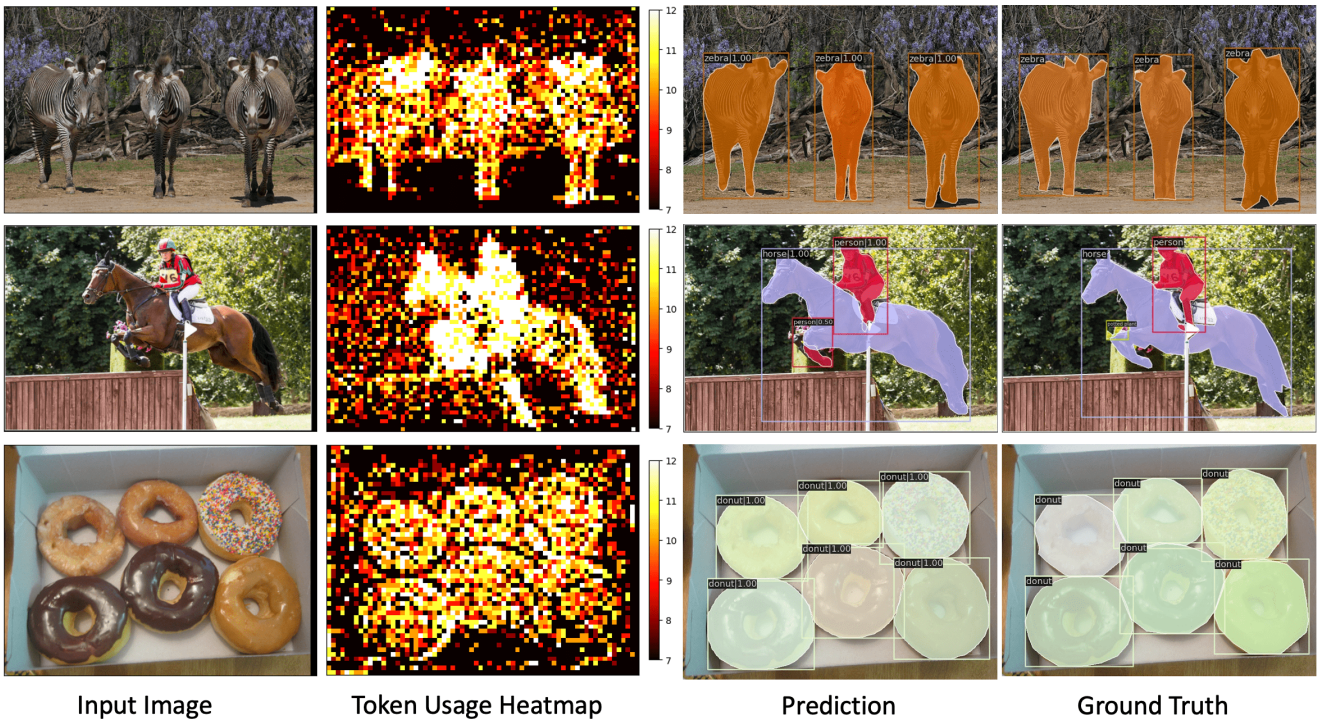


Figure 4. SViT learns to allocate computation to visually more important tokens. The token usage heatmap shows the number of layers used for each token and reflects computational distribution over the input space, which highly align with fine-grained object contours. More visualizations and dataset-level statistics are in the supplementary material.

worth noting that the token usage, i.e averaged mask values: $\frac{1}{BN}\sum_{b\in B}\sum_{n\in N}\mathbf{M}_n^{b,l}$, is averaged not only across all tokens but also across images in a batch, making the loss aware of the trade-off between token usage and accuracy, resulting in more tokens allocated for complex images and fewer tokens for simpler images. For a comparison with a fixed pruning ratio loss, see Section 4.1.

4. Experiments

We conduct experiments on the COCO 2017 object detection and instance segmentation dataset [35], which consists of 118K training images and 5K validation images, and provide experiments on ImageNet-1K [14] classification in the supplementary material A. We use Mask R-CNN [25] as our object detection framework, and employ ViT-Adapter [9] to wrap a ViT as the backbone. The dense backbone utilizes DeiT [51] with global self-attention, while the sparse backbone adopts one of the token pruning models (DynamicViT, EViT, EvoViT, ATS, SViT) with a reduced number of tokens. By default, SViT incorporates nine gating modules, ranging from the 4-th to the 12-th layer to prune tokens from the dense model, and adheres to the target keeping ratio of [70%, 70%, 70%, 49%, 49%, 49%, 34.3%, 34.3%, 34.3%] following conventions [34, 44]. For training, we follow the settings of ViT-Adapter [9] to train the dense model with a 3x schedule (36 epochs). Then we finetune each sparse model for 6 and 4 epochs for tiny and small models, respectively, with an initial learning rate of 1e-5 and the loss hyper-parameter $\lambda = 4$. In the following, we first present experiments for each insight, and then compare the derived SViT with other state-of-the-art token pruning models on object detection and instance segmentation. Finally, we analyse the pattern of pruning and reactivation by providing qualitative and quantitative results.

4.1. Evaluation of the insights and observations

Preserve pruned tokens within feature maps We evaluate the difference between removing and preserving pruned tokens on four state-of-the-art models: EViT [34], Evo-ViT [61], DynamicViT [44] and ATS [19]. Some of these models prune tokens via the attention score from the class token, which does not naturally exist on dense tasks, and we insert an artificial class token and find it still works well for these models. As shown in Table 2, Evo-ViT, which inherently preserves pruned tokens, performs the best among the original models. In addition, by enabling preserving tokens, EViT and ATS both get small increase in performance, and DynamicViT has a boost increase, as gating networks learned end-to-end are sensitive to gradient information kept in pruned tokens, and the gradients cannot

be back-propagated to the backbone if pruned tokens are dropped. Owing to this factor, DynamicViT-S experiences training divergence, as indicated in 5.

Reuse preserved tokens at demand We evaluate the influence of reusing / reactivating pruned tokens on SViT instead of the previous models, as models utilizing attention-based selection cannot really reuse tokens. Attention-based selection happens after Multi-Head Self-Attention (MHSA), and reusing tokens in such case requires all tokens to participate in MHSA, leading to no computational savings. To construct our baseline that is restricted not to reuse pruned tokens, we multiply the mask at l-th layer by its previous mask at $l-1$-th layer following [44]: $\mathbf{M}^l \leftarrow \mathbf{M}^l \odot \mathbf{M}^{l-1}$. This implies that active tokens will consistently be a subset of previous active tokens, and pruned tokens cannot be used again. As the set of active tokens is strictly decreasing, we merge selection modules with the same keeping ratios into one selection module. Table 3 shows that by reusing pruned tokens, SViT-T gets +0.4 box AP and +0.3 mask AP. Reactivation ratio and visualizations samples are in Figure 5.

Dynamic pruning rate outperforms fixed pruning rate To evaluate the influence of dynamic pruning rate vs. fixed pruning rate, we create a baseline by changing the dynamic ratio loss $\mathcal{L}_{dynamic}$ from equation (2) to the fixed ratio loss $\mathcal{L}_{fixed}$ [44] as follows:

$$\mathcal{L}_{fixed} = \frac{1}{LB} \sum_{l \in L} \sum_{b \in B} \left(\left(\frac{1}{N} \sum_{n \in N} \mathbf{M}^l_{b,n} \right) - \mathbf{t}^l \right)^2, \quad (3)$$

This loss does not average token usage across images within a batch, thereby penalizing each image towards the same keeping ratio. As indicated in Table 3, employing just a dynamic pruning rate yields a gain of +0.2 in both box AP and mask AP for SViT-T. When further augmented with token reactivation, these improvements escalate to +0.4 for box AP and +0.5 for mask AP. This substantiates both the efficacy of implementing a dynamic pruning rate in dense tasks and the added benefits of its integration with token reactivation. We also provide throughput experiments with different batch sizes in the supplementary material.

A 2-layer MLP performs as good as complex gating networks We evaluate the designs of the different gating modules, as shown in Figure 3, and experiment with both tiny and small models. Table 4 shows that using a 2-layer MLP to predict tokens for pruning achieves the same box AP and only -0.1 mask AP for tiny models, and -0.1 box AP and the same mask AP for small models. This verifies the role of 2-layer MLP as an effective and simple selection module.

4.2. Comparison with state-of-the-art models

In this section, we compare SViT with prior art pruning models adapted for dense tasks. We evaluate inference speeds on a NVIDIA A100 GPU for both the backbones and the entire networks. As illustrated in Table 5, sparse models exhibit comparable relative speed gains under identical total pruning rates, with the exception of ATS. The latter incurs computational overhead in its inverse transform sampling module for dealing with a large number of tokens in dense tasks. Among sparse models, SViT gets the highest performance for both tiny and small models. SViT-S significantly surpasses all baseline models, narrowing the performance drop with respect to the dense model from a range of -1.3 to -1.8 in box AP and -1.2 to -1.7 in mask AP, down to a mere -0.3 for both metrics. This performance advantage is consistently observed in SViT-T as well. In comparison with the dense counterpart, SViT-S improves inference speed by $\sim$34% and $\sim$46% for the entire network and the backbone, respectively, with negligible -0.3 drop in both box AP and mask AP.

4.3. Additional Analysis

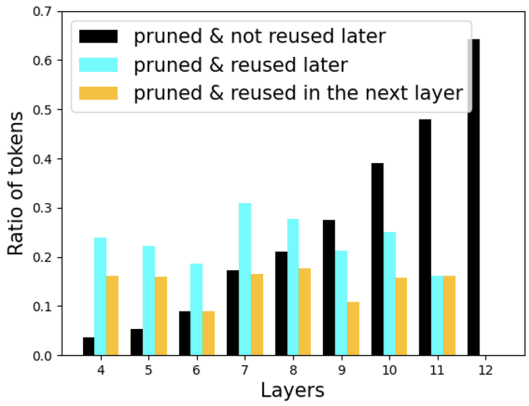
Qualitative Results We show qualitative results of the token pruning in SViT in Figure 4, and refer to supplementary material for more examples. The token-usage heatmap, created by quantifying the number of active layers for each token position, distinctly highlights not only the objects themselves but also their fine-grained contours. For example, the zebra's feet and the contour of the donuts stand out clearly against their respective backgrounds. In the case of background tokens, uniform areas such as the ground in the zebra image are more prone to be pruned, whereas textured backgrounds near objects are kept processed more. We also present the averaged token-usage heatmap on COCO validation set in Figure 6. The heatmap reveals a higher frequency of token usage in the central regions of images, due to the common photographic tendency to place objects at the center.

Different pruning rates We adjust the pruning rate for SViT-S from the default 0.7 to $\{0.5, 0.6, 0.8, 0.9\}$ and plot the mAP vs. pruning rate in Figure 7. As shown in the plot, SViT consistently achieves better speed-accuracy trade-off than DeiT. However, we observe noticeable AP drop when base pruning rate is as low as 0.6 or 0.5, due to too aggressive pruning rates in the last three ViT blocks, i.e., 0.216 and 0.125, which is consistent with findings from classification [44].

Reactivation distribution To further understand the behavior of reactivation across transformer layers, we plot the reactivation ratio at each layer averaged on COCO valida-

	DynamicViT		EViT		ATS		Evo-ViT
	remove	prsv.	remove	prsv.	remove	prsv.	
AP^{box}	41.2	44.1	44.5	44.7	43.9	44.1	44.8
AP^{mask}	37.1	39.3	39.8	39.9	39.1	39.3	39.9
FPS	23.10	22.95	22.76	22.81	16.41	16.52	22.12

Table 2. Effectiveness of removing tokens vs. preserving tokens on COCO 2017. Evo-ViT inherently preserves tokens, and performs the best among these models; Similarly, preserving tokens in feature maps increases the performance of the other three models. As anticipated, the token removal or preservation process has a negligible impact on inference speeds; scattering updated tokens onto either a zero feature map or the previous feature map consumes equivalent computational time.



(a) reactivation ratio per layer

(b) reactivation example

Figure 5. (a) Reactivation ratio at different layers, averaged on COCO validation set. (b) Visualization of reactivated tokens. Cyan tokens will be reactivated in later layers, while white tokens are not. Reactivated tokens are visually more important tokens.

Model	dynamic	reactivation	AP^{box}	AP^{mask}
SViT-T	✓	✓	45.5	40.7
SViT-T	✗	✓	45.1	40.2
SViT-T	✓	✗	45.1	40.4
SViT-T	✗	✗	44.9	40.2

Table 3. The effects of dynamic pruning rate and reactivating pruned tokens. Both can enhance performance individually, and their combination results in a larger improvement.

Gating module	Tiny		Small	
	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}
2-layer MLP	48.2	48.5	45.5	40.7
complex gating network	48.2	48.6	45.6	40.7

Table 4. Evaluation of designs for the gating module on SViT-T and SViT-S. A simple MLP can achieve similar performance with complex gating network, simplifying model design.

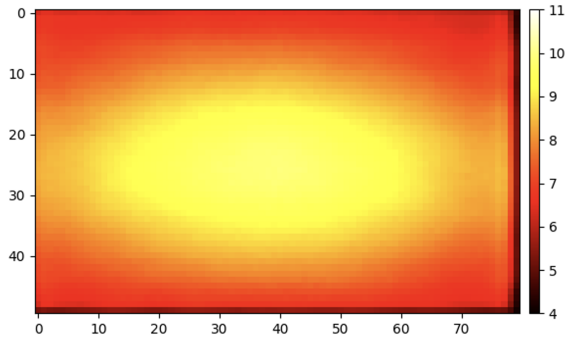


Figure 6. Averaged token-usage heat map of SViT-T showing the number of active layers for each token position, averaged on COCO [35] validation set. The resolution is interpolated to 50 x 80 tokens for all images.

tion set in Figure 5a. In the scope of this paper, the reactivation ratio at a layer (cyan colored) is defined as the ratio of current pruned tokens that are reused in at least one later layer in the backbone. As shown in the plot, most pruned tokens in early layers are reused in later layers. This indicates that it is harmful to fully drop tokens in early layers, and the model chooses to recover them in succeeding layers to alleviate the loss. In deeper layers, although the pruning rate is higher, the reactivation ratio is not apparently increased, as it is more tolerant to drop tokens. We also observe that over 50% of reactivated tokens are immediately reused in the succeeding layer. This observation aligns with the notion that the utility of a token diminishes if it remains unused for an extended period, given that feature characteristics often vary between deep and shallow layers.

Model in ViT-Adapter	Tiny				Small			
	AP^{box}	AP^{mask}	FPS^{w}	FPS^{b}	AP^{box}	AP^{mask}	FPS^{w}	FPS^{b}
DeiT [51]	45.8	40.9	18.45	27.61	48.5	42.8	11.70	14.20
EViT [34]	44.5 (-1.3)	39.8 (-1.1)	22.76	35.80	47.1 (-1.4)	41.6 (-1.2)	15.34	20.01
EvoViT [61]	44.8 (-1.0)	39.9 (-1.0)	22.12	34.33	47.2 (-1.3)	41.6 (-1.2)	15.48	20.26
ATS [19]	43.9 (-1.9)	39.1 (-1.8)	16.41	22.38	46.7 (-1.8)	41.1 (-1.7)	11.63	14.24
DyViT [44]	41.2 (-4.6)	37.1 (-3.8)	**23.10**	**36.45**	diverge	diverge	/	/
DyViT+prsv.	44.1 (-1.7)	39.3 (-1.6)	22.95	36.38	47.2 (-1.3)	41.6 (-1.2)	15.66	**20.79**
SViT (Ours)	**45.5 (-0.3)**	**40.7 (-0.2)**	22.32	34.69	**48.2 (-0.3)**	**42.5 (-0.3)**	**15.75**	20.78

Table 5. Comparison of token pruning methods on COCO object detection and instance segmentation. DeiT is the dense model using all tokens. FPS^{w} and FPS^{b} represents the inference speeds for the whole network and the backbone, respectively, which are measured with batch size 1 on a single A100 GPU.

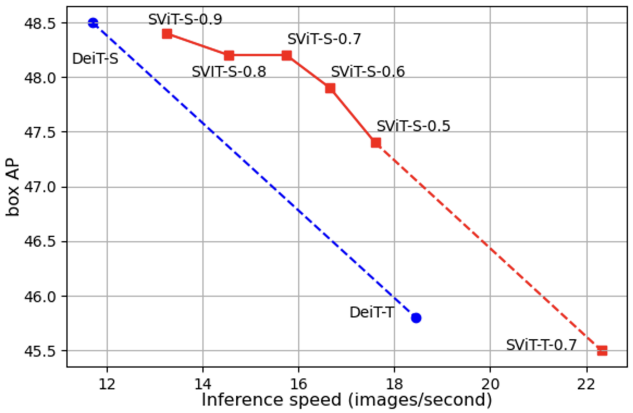


Figure 7. Trade-off between speed and accuracy across various pruning rates in ViT-Adapter with dense DeiT and sparse SViT configurations.

Reactivation areas In the previous section we analysed the reactivation ratio for different model layers, and here we show reactivation regions in images as visualized in Figure 5b. The token pruning is shown at middle layers of SViT. As anticipated, background tokens are predominantly not reactivated (white colored), while pruned tokens in interested objects, such as person, soccer and computers, are selectively reactivated (cyan colored). When faced with a high pruning rate that necessitates the temporary removal of tokens associated with objects of interest, the model strategically reactivates these tokens at later layers. This approach allows for a more expansive set of active tokens compared to scenarios where token reactivation is not an option.

5. Limitations and Societal Impacts

Limitations The aim of our work is to bridge the gap of token pruning between classification and dense tasks for isotropic vision transformers. We do not focus on pyramidal vision transformers, nor on exploring better pruning rates. These topics are covered by some concurrent works and will be further studied in future works.

Societal Impact The fintuning of sparse pruning is conducted after the model is fully trained and will introduce some additional energy consumption for training. However, this cost can be amortized once the model is deployed with improved inference efficiency. The proposed method predicts pruned tokens based on learned statistics from the training dataset, any bias inherent in the training data will be mirrored in the pruning process, and may result in the model disregarding biased content and exacerbating fairness issues.

6. Conclusions

In this work, we revisit the designs of token pruning for vision transformers in the context of object detection and instance segmentation. We provide four insights that can enhance token pruning on dense tasks: the pruned tokens should not be removed but preserved in feature maps; reactivating pruned tokens at demand can boost model performance; a dynamic pruning rate is helpful on dense tasks; and a 2-layer MLP can be as effective as more complex gating networks. By incorporating these insights together, we present a token pruning method that outperforms prior state-of-the-arts by a significant margin and accelerates backbone inference by $\sim$46% with negligible loss in accuracy. We hope these insights and encouraging results can inspire further research on ViT acceleration for dense prediction tasks beyond image classification.

7. Acknowledgements

This work was supported by the National Centre of Competence in Research (NCCR) Robotics (grant agreement No. 51NF40-185543) through the Swiss National Science Foundation (SNSF), and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

Supplementary Material

A Results on ImageNet-1K classification

Setup We also train and evaluate SViT-S on ImageNet-1K [14]. We follow the training settings in DeiT [51] and initialize our model from public pre-trained weights of DeiT-S. We use an AdamW optimizer to train the our model for 30 epochs and set the learning rate as $\frac{\text{batchsize}}{512} \times$ 1e-5. The model is trained on a single machine with 4 V100 GPUs with a batch size of 1024.

Results We compare the throughput of SViT-S and the dense counterpart DeiT-S in Table 6. SViT achieves 47% higher throughput than the dense counter part while only sacrificing -0.4% accuracy, effectively improving the accuracy-speed trade off. We also compare SViT-S with other token pruning models in Table 7. Although SViT is not originally targeted at classification tasks, it outperforms all models that use gating modules (DynamicViT [44], SPViT [32], AdaViT [39]), the models using special pruning techniques such as adaptive computation time [22] (A-ViT [63]) and reinforcement learning (IA-RED2 [41]), and a model that uses class token's attention (Evo-ViT [61]). However, when it comes to classification, EViT and ATS demonstrate superior performance over SViT. This is primarily due to their utilization of the class token, a feature specifically designed for the classification task.

Model	Top-1 Accuracy	GFLOPS	images/s
DeiT-S [51]	79.8	4.6	1524
SViT-S	79.4	3.0	2246

Table 6. Model Performance of DeiT-S and SViT-S. Throughput is measured on a single A100 GPU with batch size 512.

Model	epochs	GFLOPS	Top-1 Acc(%)
DeiT-S [51]	-	4.6	79.8
DynamicViT ‡ [44]	30	3.0	79.3 (-0.5)
EViT [34]	30	3.0	79.5 (-0.3)
Evo-ViT [61]	300 *	3.0	79.4 (-0.4)
Evo-ViT [61]	30 †	3.0	79.2 (-0.6)
A-ViT [63]	100	3.6	78.6 (-1.3)
ATS [19]	30	3.0	79.7 (-0.1)
AdaViT [39]	150	2.3	77.3 (-2.5)
IA-RED2 [41]	90	3.2	79.1 (-0.7)
SPViT ‡ [32]	60	2.7	79.3 (-0.5)
SViT (Ours)	30	3.0	79.4 (-0.4)

Table 7. Model performance on ImageNet-1K. * means training from scratch. † indicates experiments trained by us. ‡ uses additional knowledge distillation. Note that EViT, Evo-ViT and ATS depend on the class token, designed specifically for classification, to help improve their performance. On the other hand, SViT targets more general tasks and does not rely on the attention map of the class token for token pruning.

B Influence of Batch Size on Throughput

It is not straightforward to do batch inference with different number of tokens per image, as the tensor cannot be easily arranged in a regular shape, therefore, we use *pytorch nested tensor*[2] to efficiently process the varying-length sequences of tokens. The tokens to be processed are first gathered into a nested tensor, then passed to a ViT block constructed with *nested tensor* operations, and finally unnested and scattered back to the feature map.

We test the throughput of SViT and DeiT on ImageNet-1k for varying batch sizes as follows: for each batch size, we randomly fetch 30 batches from the validation set, and for each batch we run the inference for 50 times and take the average throughput as the speed for this batch. Then we calculate the mean and standard deviation over the speeds of the batches. As seen in Figure 8, proportional throughput gains can be obtained from increased batch sizes for SViT, which verifies that *nested tensor* could be a promising way of handling the varying-sized tensors in the dynamic scenario. However, note that the *nested tensor* is not fully developed and is still in a prototype stage, and leads to some overhead when the batch size is small.

In the case of object detection and instance segmentation, we abstain from using *nested tensors* due to the difficulties associated with creating a large batch size for high-resolution images. Consequently, we centered our efforts on inference with a batch size of 1 for these dense tasks. Future advancements in this technique, along with other related breakthroughs, may facilitate additional speedups. These improvements could be readily integrated into SViT, as previously demonstrated in classification tasks.

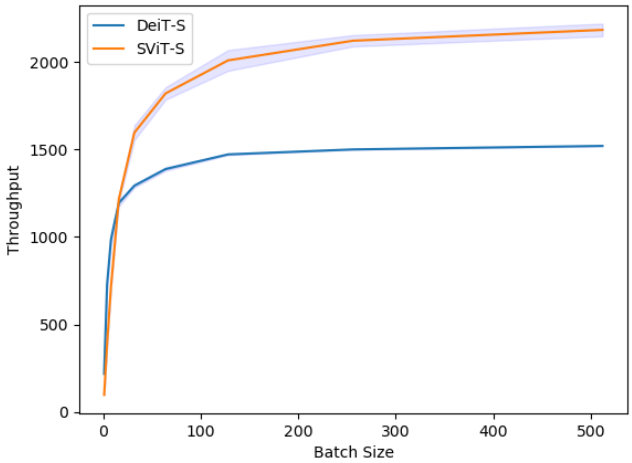


Figure 8. Throughput vs. Batch Sizes of SViT-S and DeiT-S on ImageNet-1K.

[2] https://pytorch.org/docs/1.13/nested.html

C ViT has different layer-wise attention

Vision Transformers do not always attend to the same set of tokens, even for the important ones. An illustrated in the example in Figure 9, the dense DeiT-S [51] first attends to the background tokens in the 1st layer, and then attends to joint regions of the human face and the rabbit in the next three layers. After that, the human face, rabbit eyes, and rabbit ears are attended in different layers, respectively. This inspired us to reactivate previously pruned tokens, as each layer can have its customized preference on tokens.

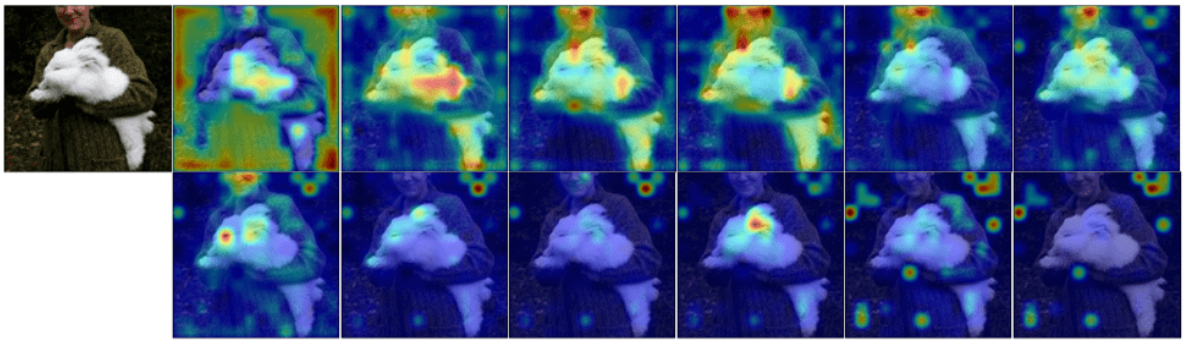
D Discussion on Prior-Art Token Pruning Methods

Since the class token does not originally exist for ViT models on dense tasks, we append a randomly initialize a class token for attention-based token pruning models (EViT [34], ATS [19], and Evo-ViT [61]), which can help them prune tokens reasonably on dense tasks.

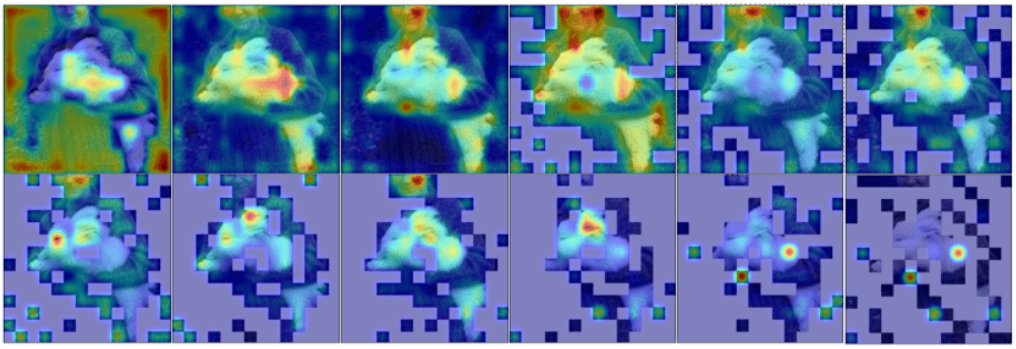
Among these models, Evo-ViT is unique because it preserves pruned tokens in the feature map. However, it has a tendency to converge to a consistent set of tokens and thus does not reuse pruned tokens, as shown in Figure 10. We conjecture this is because Evo-ViT uses a moving average to update the attention scores of processed tokens, which in turn is used to select tokens to be pruned. Since updating the scores is only done for processed tokens, the pruned tokens do not have a chance to change their scores, and thus causing the model to consistently use the same selection scheme. As Figure 10 illustrates, Evo-ViT consistently selects bottom-left tokens from its first layer onward, even though they are irrelevant background tokens. In contrast, our model can dynamically choose different tokens for each layer, and reuse important ones.

E Qualitative Examples

We provide more visualizations of SViT-S on COCO in Figure 11. To better understand the token selection of SViT, we split the tokens into two sets: object tokens and background tokens, according to the prediction mask. Then we compute the token usage for them separately. The first observation is that SViT uses a larger ratio of foreground tokens than background tokens. When the proportion of the object in the image is small, the foreground token usage can be as high as 90%. In cluttered images, such as the sheep example in the 4th row, not all foreground tokens are essential; less discriminative foreground tokens can still be pruned without affecting performance.



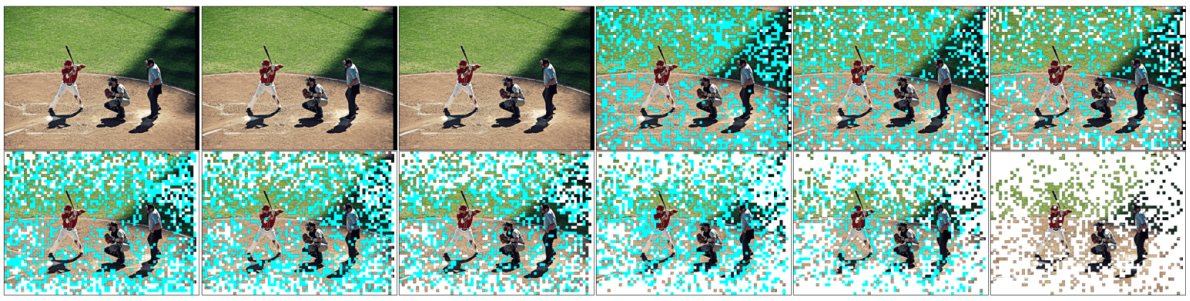
DeiT-S



SViT-S

Figure 9. The first row: the attention from the 1st layer to the 6th layer; the second row: the attention maps from the 7th layer to the 12th layer. Compared to the dense DeiT-S model, SViT-S keeps the most important features for each layer, especially the human face, the rabbit eyes, and the rabbit ear. Since ViT's attentions can be different across different layers, SViT does not keep the same tokens across different layers. The attention maps are visualized as the mean of the attention from class token's heads.

Evo-ViT-T



SViT-T

Figure 10. Top: token pruning for Evo-ViT-T. Bottom: token pruning for SViT-T. Evo-ViT has the same keep ratio per layer, and tends to use the same set of tokens for all its pruning layers, so the selection largely depends on its first pruning layer, which may not be optimal. SViT can prune tokens independently for each layer, which is learned by the gating MLPs, and can reuse tokens according as needed.

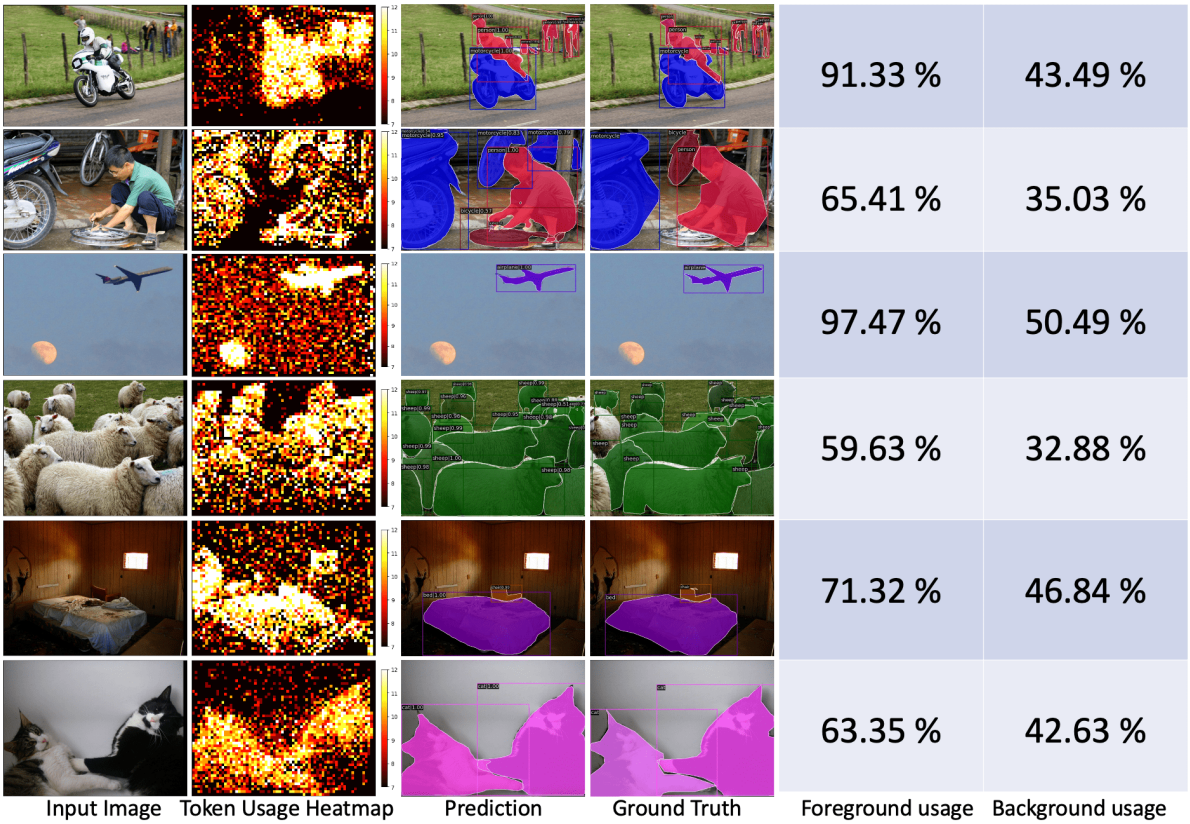


Figure 11. More visualizations of SViT-S on COCO validation set. The token usage heat map shows the number of used layers per token. The tokens are further split into two sets: foreground tokens and background tokens, and token usages are computed for them separately.

References

[1] Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. Efficient 8-bit quantization of transformer neural machine language translation model. arXiv preprint arXiv:1906.00532, 2019. 3

[2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In Int. Conf. Learn. Representations (ICLR), 2023. 3

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901, 2020. 2

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Eur. Conf. Comput. Vis. (ECCV), 2020. 1, 3

[5] Shuning Chang, Pichao Wang, Ming Lin, Fan Wang, David Junhao Zhang, Rong Jin, and Mike Zheng Shou. Making vision transformers efficient from a token sparsification view. In IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), pages 6195–6205, 2023. 3

[6] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention approximation. arXiv preprint arXiv:2110.15343, 2021. 3

[7] Chun-Fu Chen, Rameswar Panda, and Quanfu Fan. Regionvit: Regional-to-local attention for vision transformers. arXiv preprint arXiv:2106.02689, 2021. 2

[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 IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2023. 3

[9] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. In Int. Conf. Learn. Representations (ICLR), 2023. 1, 2, 5

[10] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. Advances in Neural Information Processing Systems, 34:17864–17875, 2021. 1

[11] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509, 2019. 3

[12] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. arXiv preprint arXiv:2009.14794, 2020. 3

[13] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. Advances in Neural Information Processing Systems, 34:9355–9366, 2021. 2

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2009. 5, 9

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018. 2

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In Int. Conf. Learn. Representations (ICLR), 2021. 1, 2

[17] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. arXiv preprint arXiv:1909.11556, 2019. 3

[18] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. arXiv preprint arXiv:2004.07320, 2020. 3

[19] Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Juergen Gall. Adaptive token sampling for efficient vision transformers. In Eur. Conf. Comput. Vis. (ECCV), 2022. 1, 2, 3, 4, 5, 8, 9, 10

[20] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2017. 1, 3

[21] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In Int. Conf. Comput. Vis. (ICCV), 2021. 1

[22] Alex Graves. Adaptive computation time for recurrent neural networks. arXiv preprint arXiv:1603.08983, 2016. 9

[23] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. IEEE Trans. Pattern Anal. Mach. Intell., 45(1):87–110, 2022. 1, 2

[24] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. Advances in Neural Information Processing Systems, 34:15908–15919, 2021. 2

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Int. Conf. Comput. Vis. (ICCV), 2017. 5

[26] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In Int. Conf. Learn. Representations (ICLR), 2017. 4

[27] Zi-Hang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. Advances in neural information processing systems, 34:18590–18602, 2021. 2

[28] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In Proc. Int. Conf. Mach. Learning (ICML), pages 5156–5165. PMLR, 2020. 3

[29] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. ACM computing surveys (CSUR), 54(10s):1–41, 2022. 1, 2

[30] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In Proc. Int. Conf. Mach. Learning (ICML), pages 5506–5518. PMLR, 2021. 3

[31] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451, 2020. 3

[32] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Bin Ren, Minghai Qin, Hao Tang, and Yanzhi Wang. Spvit: Enabling faster vision transformers via soft token pruning. In Eur. Conf. Comput. Vis. (ECCV), 2022. 1, 2, 3, 4, 9

[33] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In Eur. Conf. Comput. Vis. (ECCV), pages 280–296, 2022. 1, 2

[34] Youwei Liang, Chongjian GE, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. EViT: Expediting vision transformers via token reorganizations. In Int. Conf. Learn. Representations (ICLR), 2022. 1, 3, 4, 5, 8, 9, 10

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In Eur. Conf. Comput. Vis. (ECCV), 2014. 5, 7

[36] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019. 2

[37] 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 Int. Conf. Comput. Vis. (ICCV), 2021. 1, 2

[38] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In Int. Conf. Learn. Representations (ICLR), 2017. 4

[39] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. In IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2022. 1, 3, 9

[40] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? Advances in neural information processing systems, 32, 2019. 3

[41] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. In Advances in Neural Information Processing Systems, volume 34, pages 24898–24911, 2021. 3, 9

[42] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 2

[43] Yongming Rao, Zuyan Liu, Wenliang Zhao, Jie Zhou, and Jiwen Lu. Dynamic spatial sparsification for efficient vision transformers and convolutional neural networks. IEEE Trans. Pattern Anal. Mach. Intell., 2023. 2

[44] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In Advances in Neural Information Processing Systems, 2021. 1, 2, 3, 4, 5, 6, 8, 9

[45] Byungseok Roh, JaeWoong Shin, Wuhyun Shin, and Saehoon Kim. Sparse detr: Efficient end-to-end object detection with learnable sparsity. In Int. Conf. Learn. Representations (ICLR), 2022. 3

[46] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. Transactions of the Association for Computational Linguistics, 9:53–68, 2021. 3

[47] Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? arXiv preprint arXiv:2106.11297, 2021. 3

[48] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019. 3

[49] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In AAAI Conf. Artificial Intell., volume 34, pages 8815–8821, 2020. 3

[50] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In Proc. Int. Conf. Mach. Learning (ICML), pages 9438–9447. PMLR, 2020. 3

[51] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In Proc. Int. Conf. Mach. Learning (ICML), July 2021. 1, 2, 3, 5, 8, 9, 10

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, 2017. 1, 2

[53] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. 3

[54] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. Advances in Neural Information Processing Systems, 33:21665–21674, 2020. 3

[55] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768, 2020. 3

[56] Wenxiao Wang, Wei Chen, Qibo Qiu, Long Chen, Boxi Wu, Binbin Lin, Xiaofei He, and Wei Liu. Crossformer++: A versatile vision transformer hinging on cross-scale attention. arXiv preprint arXiv:2303.06908, 2023. 2

[57] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In Int. Conf. Comput. Vis. (ICCV), pages 568–578, 2021. 2

[58] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. Computational Visual Media, 8(3):415–424, 2022. 2

[59] Wenxiao Wang, Lulian Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu. Crossformer: A versatile vision transformer hinging on cross-scale attention. In Int. Conf. Learn. Representations (ICLR), 2021. 2

[60] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. Advances in Neural Information Processing Systems, 34:12077–12090, 2021. 1

[61] Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In AAAI Conf. Artificial Intell., 2022. 1, 3, 4, 5, 8, 9, 10

[62] Botao Ye, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Joint feature learning and relation modeling for tracking: A one-stream framework. In Eur. Conf. Comput. Vis. (ECCV), 2022. 3

[63] Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2022. 2, 3, 4, 9

[64] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In Int. Conf. Comput. Vis. (ICCV), 2021. 2

[65] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. In Int. Conf. Comput. Vis. (ICCV), pages 579–588, 2021. 2

[66] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In Int. Conf. Comput. Vis. (ICCV), 2021. 2

[67] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. In 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS), pages 36–39. IEEE, 2019. 3

[68] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. Advances in neural information processing systems, 33:17283–17297, 2020. 3

[69] Jinnian Zhang, Houwen Peng, Kan Wu, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Minivit: Compressing vision transformers with weight multiplexing. In Int. Conf. Comput. Vis. (ICCV), 2022. 3

[70] Qinglong Zhang and Yu-Bin Yang. Rest: An efficient transformer for visual recognition. Advances in Neural Information Processing Systems, 34:15475–15485, 2021. 2

[71] Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, and Tomas Pfister. Aggregating nested transformers. arXiv preprint arXiv:2105.12723, 2(3):5, 2021. 2

[72] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In Int. Conf. Comput. Vis. (ICCV), 2021. 2

[73] Ce Zheng, Sijie Zhu, Matias Mendieta, Taojiannan Yang, Chen Chen, and Zhengming Ding. 3d human pose estimation with spatial and temporal transformers. In Int. Conf. Comput. Vis. (ICCV), 2021. 2

[74] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In IEEE Conf. Comput. Vis. Pattern Recog. (CVPR), 2021. 1

[75] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159, 2020. 1