Vision Transformer with Super Token Sampling

Huaibo Huang[1,2] Xiaoqiang Zhou[1,2,4] Jie Cao[1,2] Ran He[1,2,3*] Tieniu Tan[1,2,5]
[1]Center for Research on Intelligent Perception and Computing, CASIA, Beijing, China
[2]National Laboratory of Pattern Recognition, CASIA, Beijing, China
[3]School of Artificial Intelligence, UCAS, Beijing, China
[4]University of Science and Technology of China, Hefei, China
[5]Nanjing University, Nanjing, China
{huaibo.huang, jie.cao}@cripac.ia.ac.cn, xq525@mail.ustc.edu.cn, {rhe, tnt}@nlpr.ia.ac.cn

Abstract

*Vision transformer has achieved impressive performance for many vision tasks. However, it may suffer from high redundancy in capturing local features for shallow layers. Local self-attention or early-stage convolutions are thus utilized, which sacrifice the capacity to capture long-range dependency. A challenge then arises: can we access efficient and effective global context modeling at the early stages of a neural network? To address this issue, we draw inspiration from the design of superpixels, which reduces the number of image primitives in subsequent processing, and introduce super tokens into vision transformer. Super tokens attempt to provide a semantically meaningful tessellation of visual content, thus reducing the token number in self-attention as well as preserving global modeling. Specifically, we propose a simple yet strong super token attention (STA) mechanism with three steps: the first samples super tokens from visual tokens via sparse association learning, the second performs self-attention on super tokens, and the last maps them back to the original token space. STA decomposes vanilla global attention into multiplications of a sparse association map and a low-dimensional attention, leading to high efficiency in capturing global dependencies. Based on STA, we develop a hierarchical vision transformer. Extensive experiments demonstrate its strong performance on various vision tasks. In particular, without any extra training data or label, it achieves **86.4%** top-1 accuracy on ImageNet-1K with less than 100M parameters. It also achieves **53.9** box AP and **46.8** mask AP on the COCO detection task, and **51.9** mIOU on the ADE20K semantic segmentation task. Code will be released at https://github.com/hhb072/SViT.*

1. Introduction

Transformer [42] has dominated the natural language processing field and shown excellent capability of capturing long-range dependency with self-attention. Recent studies [15, 31] have demonstrated that transformer can be successfully transplanted to vision scenarios. Dosovitskiy et al. [15] present the pioneering vision transformer (ViT), where self-attention performs global comparison among all visual tokens. ViT and the following works [39, 40] exhibit the strong capacity of learning global dependency in visual content, achieving impressive performance in many vision tasks [5, 38, 45, 60, 63]. Nevertheless, the computational complexity of self-attention is quadratic to the number of tokens, resulting in huge computational cost for high-resolution vision tasks, e.g., object detection and segmentation.

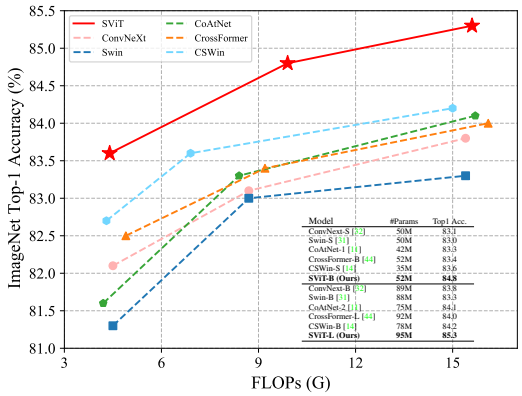
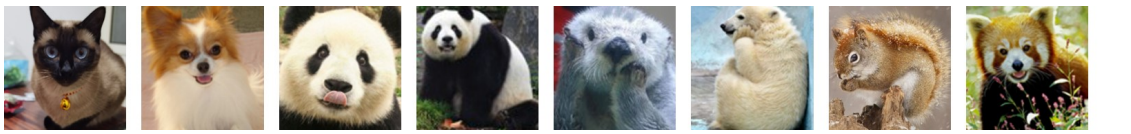


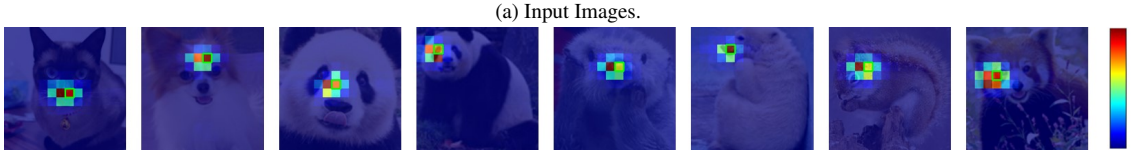
Figure 1. FLOPs vs. Top-1 accuracy on ImageNet-1K with 224 × 224 resolution.

Recent studies [26, 34] observe that ViTs tend to capture local features at shallow layers with high redundancy. Specifically, as shown in Fig. 2(b), given an anchor token, shallow-layer global attention concentrates on a few adjacent tokens (filled with red color) whereas neglects most of the tokens of far distance. Thus, global comparisons among all the tokens result in huge unnecessary computation cost in capturing such local correlations. In order to reduce computation costs, Swin Transformer [31] adopts window-based local attention to restrict attention to local regions. For local attention, as shown in Fig. 2(c), redundancy is reduced but still exists in the shallow layers, where only a few nearby tokens

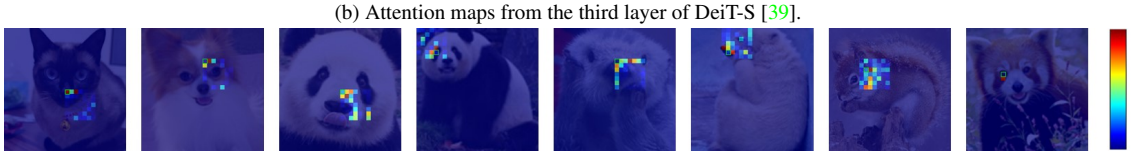
* Ran He is the corresponding author.



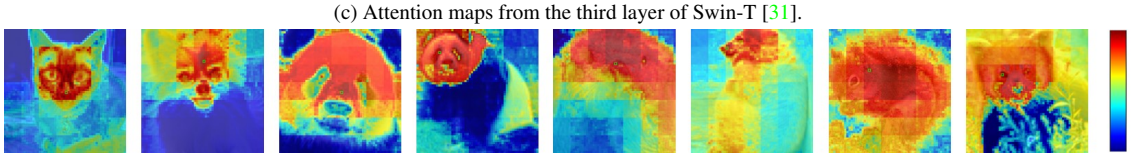
(a) Input Images.



(b) Attention maps from the third layer of DeiT-S [39].



(c) Attention maps from the third layer of Swin-T [31].



(d) Attention maps from the third layer of SViT-S (Ours).

Figure 2. Visualization of early-stage attention maps for different vision transformers. For global attention in DeiT [39] and local attention in Swin [31], only a few neighboring tokens (filled with red color) work for an anchor token (green box), resulting in local representations with high redundancy. Compared with such ViTs, our method can learn global representations even for shallow layers.

obtain high weights. In another way, Uniformer [26] utilizes convolutions in the shallow layers and effectively reduces the computation redundancy for local features. Nevertheless, both the local attention [31] and early-stage convolution [26] schemes sacrifice the capacity of capturing global dependency that is crucial for transformer. A challenge then arises: can we access efficient and effective global representations at the early stages of a neural network?

To address this problem, inspired by the idea of superpixels [23], we present super token attention to learn efficient global representations in vision transformer, especially for the shallow layers. As an over-segmentation of image, superpixels perceptually group similar pixels together, reducing the number of image primitives for subsequent processing. We borrow the idea of superpixels from the pixel space to the token space and assume super tokens as a compact representation of visual content. We propose a simple yet strong super token attention (STA) mechanism with three steps. Firstly, we apply a fast sampling algorithm to predict super tokens via learning sparse associations between tokens and super tokens. Then we perform self-attention in the super token space to capture long-range dependency among super tokens. Compared to self-attention in the token space, such self-attention can reduce computational complexity significantly meanwhile learn global contextual information thanks to the representational and computational efficiency of super tokens. At last, we map the super tokens back to the original token space by using the learned associations in the first step. As shown in Fig. 2(d), the presented super token attention can learn global representations even in the shallow layers. For example, given an anchor token, like the green box in the Siamese cat's nose, most of the related tokens (i.e., those in the cat face) contribute to the representation learning.

Based on the super token attention mechanism, we present a general vision backbone named Super Token Vision Transformer (SViT) in this paper. As shown in Fig. 3, it is designed as a hierarchical ViT hybrid with convolutional layers. The convolutional layers are adopted to compensate for the capacity of capturing local features. In each stage, we use a stack of super token transformer (STT) blocks for efficient and effective representation learning. Specifically, the STT block consists of three key modules, i.e., Convolutional Position Embedding (CPE), Super Token Attention (STA), and Convolutional Feed-Forward Network (ConvFFN). The presented STA can efficiently learn global representations, especially for shallow layers. The CPE and ConvFFN with depth-wise convolutions can enhance the representative capacity of local features with a low computation cost.

Extensive experiments demonstrate the superiority of SViT on a broad range of vision tasks, including image classification, object detection, instance segmentation and semantic segmentation. For example, without any extra training data, our large model SViT-L achieves **86.4%** top-1 accuracy on ImageNet-1K image classification. Our base model SViT-B achieves **53.9** box AP and **46.8** mask AP on the COCO detection task, **51.9** mIOU on the ADE20K semantic segmentation task, surpassing the Swin Transformer [31] counterpart by **+2.1**, **+2.1** and **+2.4**, respectively.

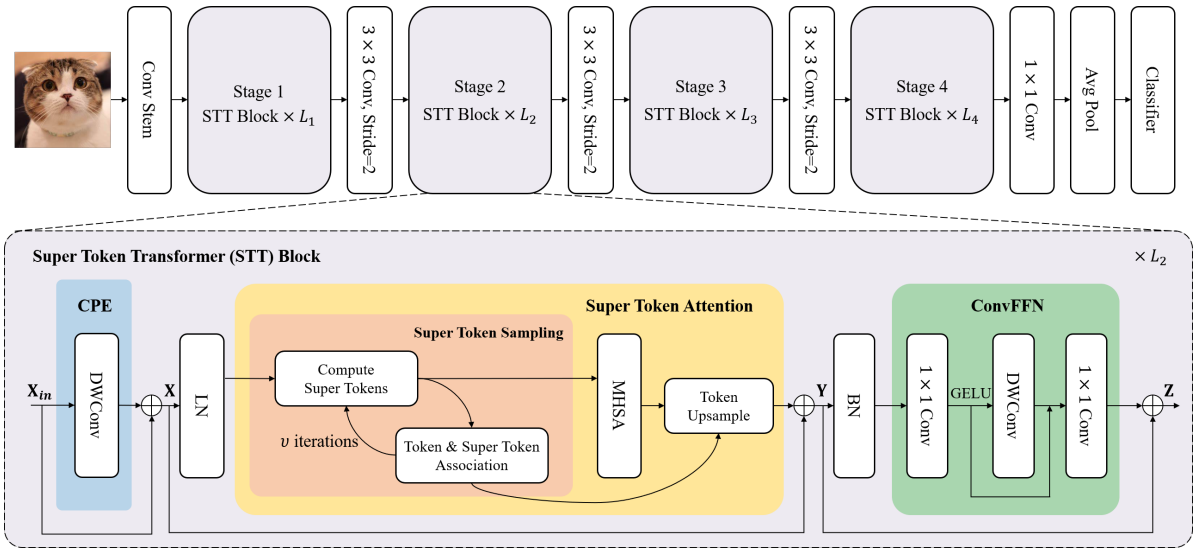


Figure 3. The architecture of Super Token Vision Transformer (SViT).

2. Related Works

Vision Transformers. As the Transformer network achieves tremendous improvements in many NLP tasks [13, 42], ViT [15] is the pioneering work that adapts Transformer architecture to vision tasks and gets promising results. After that, given the great ability on long-range dependency modeling, researchers begin to pay more attention on designing general vision Transformer backbones [25, 51]. Among them, many novel hierarchical architectures [31, 43], self-attention variants [17, 27, 35, 47, 54] and positional encodings [8, 14, 37, 42] are developed to accommodate the characteristics of vision tasks.

Many efficient self-attention mechanisms are introduced to alleviate the great computational costs. One way is restricting the attention region to spatially neighbor tokens, such as Swin Transformer [31], CSWin [14] and criss-cross attention [22]. Besides, PVT [43] designs a pyramid architecture with downsampled key and value tokens. GG Transformer [53] and CrossFormer [44] resort to dilated token sampling [58]. Reformer [24] distributes tokens to buckets by hashing functions and perform dense attention within each bucket. ACT [59] and TCFormer [56] take the merged tokens as queries and the original tokens as keys and values to reduce the computation complexity. Different from these works, the presented STA conducts sparse mapping between tokens and super tokens via soft associations and perform self-attention in the super token space. It decomposes global attention into multiplications of sparse and small matrices to learn efficient global representations.

Superpixel Algorithms. Traditional superpixel algorithms can be divided into two categories, i.e., graph-based and clustering-based methods. The graph-based methods view image pixels as graph nodes and partition nodes by the edge connectivity between adjacent pixels [16, 29, 36]. The clustering-based methods leverage traditional clustering techniques to construct superpixels, such as k-means clustering on different feature representation [1, 30]. With the prosperity of deep learning in recent years, more and more deep clustering approaches [2, 3, 50, 52] are proposed to exploit deep features and improve the clustering efficiency. SEAL [41] proposes to learn the deep feature with a traditional superpixel algorithm. SSN [23] develops an end-to-end differentiable superpixel segmentation framework. Our sampling algorithm of super tokens is mostly motivated by SSN [23] but we adopt a more attention-like manner with sparse sampling.

3. Method

3.1. Overall Architecture

An overview of the Super Token Vision Transformer (SViT) architecture is illustrated in Fig. 3. Given an input image, we firstly feed it into a stem consisting of four 3×3 convolutions with stride 2, 1, 2, 1, respectively. Compared with the vanilla non-overlapping tokenization [31, 42], the convolution stem can extract better local representations and is widely applied in recent ViTs [17, 26]. Then the tokens go through four stages of stacked Super Token Transformer (STT) blocks for hierarchical representation extraction. The 3×3 convolutions with stride 2 are used between the stages to reduce the token number. Finally, the 1×1 convolution projection, global average pooling and fully connected layer are used to output the predictions.

A STT block contains three key modules: Convolutional

Position Embedding (CPE), Super Token Attention (STA) and Convolutional Feed-Forward-Network (ConvFFN):

$$X = \mathrm{CPE}(X_{in}) + X_{in}, \tag{1}$$
$$Y = \mathrm{STA}(\mathrm{LN}(X)) + X, \tag{2}$$
$$Z = \mathrm{ConvFFN}(\mathrm{BN}(Y)) + Y. \tag{3}$$

Given the input token tensor $X_{in} \in \mathbb{R}^{C\times H\times W}$, we firstly use CPE, i.e., a 3×3 depth-wise convolution, to add position information into all the tokens. Compared with the absolute positional encoding (APE) [42] and relative positional encoding (RPE) [31, 37], CPE is more flexible for arbitrary input resolutions and can learn better local representations. Then, we use STA to extract global contextual representations by efficiently exploring and fully exploiting long-range dependencies. The details of STA are elaborated in the following subsection. Finally, we adopt a convolution FFN to enhance local representations. It consists of two 1×1 convolutions, one 3×3 depth-wise convolution and one non-linear function, i.e., GELU. Note that the two depth-wise convolutions in CPE and ConvFFN can compensate the capacity of local correlation learning. Thereby, the combination of CPE, STA, and ConvFFN enables the STT block to capture both local and long-range dependencies.

3.2. Super Token Attention

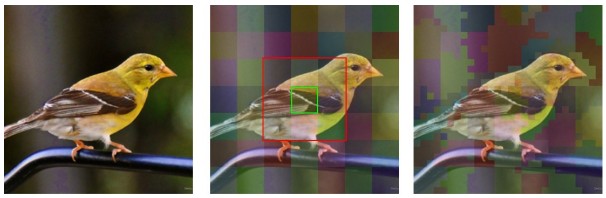
As shown in Fig. 3, the Super Token Attention (STA) module consists of three processes, i.e., Super Token Sampling (STS), Multi-Head Self-Attention (MHSA), and Token Upsampling (TU). Specifically, we firstly aggregate tokens into super tokens by the STS algorithm, then perform MHSA to model global dependencies in the super token space and finally map the super tokens back to visual tokens by the TU algorithm.

3.2.1 Super Token Sampling

In the STS process, we adapt the soft k-means based superpixel algorithm in SSN [23] from the pixel space to the token space. Given the visual tokens $X \in \mathbb{R}^{N\times C}$ (where $N = H\times W$ is the token number), each token $X_i \in \mathbb{R}^{1\times C}$ is assumed to belong to one of m super tokens $S \in \mathbb{R}^{m\times C}$, making it necessary to compute the X-S association map $Q \in \mathbb{R}^{N\times m}$. First, we sample initial super tokens S^0 by averaging tokens in regular grid regions. If the grid size is $h\times w$, then the number of super tokens is $m = \frac{H}{h} * \frac{W}{w}$. Then we run the sampling algorithm iteratively with the following two steps:

Token & Super Token Association. In SSN [23], the pixel-superpixel association at iteration t is computed as

$$Q^t_{ij} = e^{-\|X_i - S^{t-1}_j\|^2}. \tag{4}$$



(a) Input Image (b) Initial Super Tokens (c) Learned Super Tokens

Figure 4. Visualization of super tokens from initial grid to learned ones. Only the surrounding super tokens in the red box are used to compute the associations for every token in the green box.

Different from SSN [23], we apply a more attention-like manner to compute the association map Q^t, defined as

$$Q^t = \mathrm{Softmax}(\frac{X S^{t-1\mathrm{T}}}{\sqrt{d}}), \tag{5}$$

where d is the channel number C.

Super Token Update. The super tokens are updated as the weighted sum of tokens, defined as

$$S = (\hat{Q}^t)^{\mathrm{T}} X, \tag{6}$$

where $\hat{Q}^t$ is the column-normalized Q^t. The computational complexity of the above sampling algorithm is

$$\Omega(\mathrm{STS}) = 2vmNC, \tag{7}$$

where v is the number of iterations. It is time-consuming even for a small number of super tokens m. To speed up the sampling process, following SSN [23], we constrain the association computations from each token to only 9 surrounding super tokens as shown in Fig. 4. For each token in the green box, only the super tokens in the red box are used to compute the association. Moreover, we only update the super tokens once with $v = 1$. Consequently, the complexity is significantly reduced to

$$\Omega(\mathrm{STS}) = 19NC, \tag{8}$$

where the complexities for obtaining initial super-tokens, computing sparse associations and updating super tokens are NC, $9NC$ and $9NC$, respectively. We provide the details of the sparse computation of STS in Appendix A.1.

3.2.2 Self-Attention for Super Tokens

Since super tokens are compact representations of visual content, applying self-attention to them can focus more on global contextual dependencies rather than local features. We apply the standard self-attention to the sampled super tokens $S \in \mathbb{R}^{m\times C}$, defined as

$$\mathrm{Attn}(S) = \mathrm{Softmax}(\frac{\mathbf{q}(S)\mathbf{k}^{\mathrm{T}}(S)}{\sqrt{d}})\mathbf{v}(S) \tag{9}$$
$$= \mathbf{A}(S)\mathbf{v}(S) \tag{10}$$

Table 1. Architecture variants of SViT. The FLOPs are measured at resolution 224×224.

Model	Blocks	Channels	Heads	Params	FLOPs
SViT-S	[3,5,9,3]	[64,128,320,512]	[1,2,5,8]	25M	4.4G
SViT-B	[4,6,14,6]	[96,192,384,512]	[2,3,6,8]	52M	9.9G
SViT-L	[4,7,19,8]	[96,192,448,640]	[2,3,7,10]	95M	15.6G

where $\mathbf{A}(S) = \mathrm{Softmax}(\frac{\mathbf{q}(S)\mathbf{k}^{\mathrm{T}}(S)}{\sqrt{d}}) \in \mathbb{R}^{m\times m}$ is the attention map, $\mathbf{q}(S) = SW_q$, $\mathbf{k}(S) = SW_k$ and $\mathbf{v}(S) = SW_v$ are linear functions with parameters W_q, W_k and W_v, respectively. We omit the multi-head setting for clarity.

3.2.3 Token Upsampling

Although super tokens can capture better global representations with self-attention, they have lost most of local details in the sampling process. Therefore, instead of directly using them as the input to the subsequent layers, we map them back into visual tokens and add them to the original tokens X. We use the association map Q (we omit the iteration index since we only adopt one iteration) to upsample tokens from the super tokens S, which can be defined as

$$\mathrm{TU}(\mathrm{Attn}(S)) = Q\mathrm{Attn}(S). \tag{11}$$

3.2.4 Analysis of Complexity and Redundancy

In the following we compare our STA with the standard global self-attention and analyze the reason for its strong capacity of learning efficient global representations.

Global Self-Attention. The definition of the standard global self-attention (GSA) is rewritten as

$$\mathrm{GSA}(X) = \mathbf{A}(X)\mathbf{v}(X), \tag{12}$$

where $\mathbf{A}(X) = \mathrm{Softmax}(\frac{\mathbf{q}(X)\mathbf{k}^{\mathrm{T}}(X)}{\sqrt{d}}) \in \mathbb{R}^{N\times N}$ is the attention map for the input tokens. The computational complexity of GSA is

$$\Omega(\mathrm{GSA}) = 2N^2C + 4NC^2. \tag{13}$$

Super Token Attention. Considering the Eq. (6), Eq. (10) and Eq. (11), the overall process of STA is reformulated as

$$\begin{aligned}\mathrm{STA}(X) &= Q(\mathbf{A}(S)(\hat{Q}^{\mathrm{T}}X)W_v) &(14)\\ &= \tilde{\mathbf{A}}(X)\mathbf{v}(X), &(15)\end{aligned}$$

where $\tilde{\mathbf{A}}(X) = Q\mathbf{A}(S)\hat{Q}^{\mathrm{T}} \in \mathbb{R}^{N\times N}$ is the corresponding attention map for the input tokens. The computational complexity of STA is

$$\begin{aligned}\Omega(\mathrm{STA}) &= \Omega(\mathrm{STS}) + \Omega(\mathrm{MHSA}) + \Omega(\mathrm{TU}) &(16)\\ &= 19NC + (2m^2C + 4mC^2) + 9NC &(17)\\ &= 2m^2C + 4mC^2 + 28NC. &(18)\end{aligned}$$

Obviously, given m smaller than N, STA has a much lower computational cost than the global attention.

Redundancy Discussion. Images have large local redundancy, e.g., visual content tends to be similar in a local region. As shown in Fig. 2(b), for a certain anchor token in the shallow layer, global attention would highlight a few tokens in a local region, resulting in high redundancy in the comparisons among all the tokens. Compared with visual tokens, the presented super tokens tend to have distinct patterns and suppress local redundancy. For example, as shown in Fig. 4, there is a significant discrepancy with the super tokens at the bird head and surrounding regions. Thus, global dependencies can be better captured by self-attention in the super token space, as illustrated in Fig. 2(d). Moreover, considering Eq. (15), our STA can be viewed as a specific global attention that decomposes a computationally expensive $N \times N$ attention $\mathbf{A}(X)$ into multiplications of sparse and small matrices, i.e., the sparse association Q and the small $m \times m$ attention $\mathbf{A}(S)$, with far lower computation cost. Therefore, we can conclude that STA can learn effective global representations with high efficiency.

3.3. Implementation Details

As shown in Table 8, we build three SViT backbones with different settings of block number and channel number in each stage. The expansion ratios in ConvFFN are set to 4. For the first two stages, the sampling grid sizes (as shown in Fig. 4(b)) are set to 8×8 and 4×4, respectively. Therefore, the number of super tokens m is 49 for input resolution 224×224. For the last two stages, the local redundancy is already low after the preceding two stages of representation learning. So we set the grid sizes to 1×1 and directly use the tokens as super tokens without the STS and TU processes. The iteration number in STS is set to 1 to reduce the sampling cost. More details are described in the Appendix A.2.

4. Experiments

We conduct experiments on a broad range of vision tasks, including image classification on ImageNet-1K [12], object detection and instance segmentation on COCO 2017 [28], and semantic segmentation on ADE20K [62]. We also conduct ablation studies to examine the role of each component.

4.1. Image Classification

Settings. We train our models from scratch on the ImageNet-1K [12] data. For a fair comparison, we follow

Table 2. Performance comparison on ImageNet-1K classification. The throughput is measured on a single V100 GPU with batch size 16.

Model Size	Model	#Param	Flops	Throughput	Test Size	Top-1
small model size (~25M)	ConvNeXt-T [32]	28M	4.5G	720	224	82.1
	DeiT-S [39]	22M	4.6G	922	224	79.9
	PVT-S [43]	25M	3.8G	722	224	79.8
	Swin-T [31]	29M	4.5G	712	224	81.3
	CoAtNet-0 [11]	25M	4.2G	943	224	81.6
	Focal-T [51]	29M	4.9G	323	224	82.2
	CrossFormer-S [44]	31M	4.9G	636	224	82.5
	DAT-T [47]	29M	4.6G	575	224	82.0
	CSwin-T [14]	23M	4.3G	515	224	82.7
	UniFormer-S	24M	4.2G	824	224	82.9
	MPViT-S [25]	23M	4.7G	352	224	83.0
	CMT-S [17]	25M	4.0G	481	224	83.5
	SViT-S	25M	4.4G	564	224	**83.6**
	CvT-13 [46]	20M	16.3G	-	384	83.0
	CaiT-xs24 [40]	27M	19.3G	86	384	83.8
	CoAtNet-0 [11]	25M	13.4G	262	384	83.9
	CSwin-T [14]	23M	14.0G	204	384	84.3
	SViT-S	25M	14.1G	188	384	**85.0**
medium model size (~50M)	ConvNeXt-S [32]	50M	8.7G	404	224	83.1
	PVT-L [43]	61M	9.8G	321	224	81.7
	Swin-S [31]	50M	8.7G	406	224	83.0
	CaiT-s24 [40]	47M	9.4G	326	224	82.7
	CoAtNet-1 [11]	42M	8.4G	471	224	83.3
	Focal-S [51]	51M	9.1G	191	224	83.5
	CrossFormer-B [44]	52M	9.2G	377	224	83.4
	CSwin-S [14]	35M	6.9G	315	224	83.6
	DAT-S [47]	50M	9.0G	312	224	83.7
	UniFormer-B	50M	8.3G	375	224	83.9
	MViTv2-B [27]	52M	10.2G	250	224	84.4
	CMT-B [17]	46M	9.3G	259	256	84.5
	SViT-B	52M	9.9G	286	224	**84.8**
	CaiT-s24 [40]	47M	32.2G	60	384	84.3
	CSwin-S [14]	35M	22.0G	128	384	85.0
	CoAtNet-1 [11]	42M	27.4G	124	384	85.1
	MViTv2-B [27]	52M	36.7G	-	384	85.6
	SViT-B	52M	31.5G	95	384	**86.0**
large model size (~100M)	ConvNeXt-B [32]	89M	15.4G	252	224	83.8
	DeiT-B [39]	86M	17.5G	298	224	81.8
	Swin-B [31]	88M	15.4G	258	224	83.3
	CaiT-s48 [40]	90M	18.6G	162	224	83.5
	Focal-B [51]	90M	16.0G	136	224	83.8
	CrossFormer-L [44]	92M	16.1G	246	224	84.0
	DAT-B [47]	88M	15.8G	211	224	84.0
	CoAtNet-2 [11]	75M	15.7G	298	224	84.1
	CSwin-B [14]	78M	15.0G	216	224	84.2
	MPViT-B [25]	75M	16.4G	185	224	84.3
	CMT-L [17]	75M	19.5G	-	288	84.8
	SViT-L	95M	15.6G	192	224	**85.3**
	Swin-B [31]	88M	47.0G	85	384	84.2
	CaiT-s48 [40]	90M	63.8G	30	384	85.1
	CSwin-B [14]	78M	47.0G	69	384	85.5
	CoAtNet-2 [11]	75M	49.8G	88	384	85.7
	SViT-L	95M	49.7G	65	384	**86.4**

the same training strategy proposed in DeiT [39] and adopt the default data augmentation and regularization. All our models are trained from scratch for 300 epochs with the input size of 224×224. We employ the AdamW optimizer with a cosine decay learning rate scheduler and 5 epochs of linear warm-up. The initial learning rate, weight decay,

Table 3. Object detection and instance segmentation with Mask R-CNN on COCO val2017. The FLOPs are measured at resolution 800×1280. All the models are pre-trained on ImageNet-1K.

Backbone	#Param (M)	FLOPs (G)	Mask R-CNN 1× schedule						Mask R-CNN 3× + MS schedule					
			AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Res50 [19]	44	260	38.0	58.6	41.4	34.4	55.1	36.7	41.0	61.7	44.9	37.1	58.4	40.1
PVT-S [43]	44	245	40.4	62.9	43.8	37.8	60.1	40.3	43.0	65.3	46.9	39.9	62.5	42.8
Swin-T [31]	48	264	42.2	64.6	46.2	39.1	61.6	42.0	46.0	68.2	50.2	41.6	65.1	44.8
Focal-T [51]	49	291	44.8	67.7	49.2	41.0	64.7	44.2	47.2	69.4	51.9	42.7	66.5	45.9
CMT-S [17]	45	249	44.6	66.8	48.9	40.7	63.9	43.4	48.3	70.4	52.3	43.7	67.7	47.1
UniFormer-S [26]	41	269	45.6	68.1	49.7	41.6	64.8	45.0	48.2	70.4	52.5	43.4	67.1	47.0
SViT-S	44	252	**47.6**	**70.0**	**52.3**	**43.1**	**66.8**	**46.5**	**49.2**	**70.8**	**54.4**	**44.2**	**68.0**	**47.7**
Res101 [19]	63	336	40.4	61.1	44.2	36.4	57.7	38.8	42.8	63.2	47.1	38.5	60.1	41.3
PVT-M [43]	64	302	42.0	64.4	45.6	39.0	61.6	42.1	44.2	66.0	48.2	40.5	63.1	43.5
Swin-S [31]	69	354	44.8	66.6	48.9	40.9	63.4	44.2	48.5	70.2	53.5	43.3	67.3	46.6
Focal-S [51]	71	401	47.4	69.8	51.9	42.8	66.6	46.1	48.8	70.5	53.6	43.8	67.7	47.2
DAT-S [47]	69	378	47.1	69.9	51.5	42.5	66.7	45.4	49.0	70.9	53.8	44.0	68.0	47.5
UniFormer-B [26]	69	399	47.4	69.7	52.1	43.1	66.0	46.5	50.3	**72.7**	55.3	44.8	69.0	48.3
SViT-B	70	359	**49.7**	**71.7**	**54.7**	**44.8**	**68.9**	**48.7**	**51.0**	72.3	**56.0**	**45.4**	**69.5**	**49.3**

Table 4. Object detection and instance segmentation with Cascade Mask R-CNN on COCO val2017.

Method	#Params (M)	FLOPs (G)	3× + MS schedule					
			AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
X101-32 [49]	101	819	48.1	66.5	52.4	41.6	63.9	45.2
Swin-S [31]	107	838	51.8	70.4	56.3	44.7	67.9	48.5
Shuffle-S [21]	107	844	51.9	70.9	56.4	44.9	67.8	48.6
CSwin-S [14]	92	820	53.7	72.2	58.4	46.4	69.6	50.6
DAT-S [47]	107	857	52.7	71.7	57.2	45.5	69.1	49.3
Swin-B [31]	145	972	51.9	70.9	57.0	45.3	68.5	48.9
DAT-B [47]	145	1003	53.0	71.9	57.6	45.8	69.3	49.5
SViT-B	108	837	**53.9**	**72.7**	**58.5**	**46.8**	**70.4**	**50.8**

Table 5. Semantic segmentation with Upernet on ADE20K. The FLOPs are measured at resolution 512×2048.

Backbone	#Param (M)	FLOPs (G)	mIoU (%)	MS mIoU (%)
Res101 [19]	86	1029	-	44.9
Twins-B [7]	89	1020	47.7	48.9
Swin-S [31]	81	1038	47.6	49.5
Focal-T [51]	85	1130	48.0	50.0
CrossFormer-B [44]	84	1079	49.2	50.1
UniFormer-B [26]	80	1106	49.5	50.7
Swin-B [31]	121	1188	48.1	49.7
Focal-B [51]	126	1354	49.0	50.5
SViT-B	80	1036	**50.7**	**51.9**

and batch-size are 0.001, 0.05, and 1024, respectively. The maximum rates of increasing stochastic depth [20] are set to 0.1/0.4/0.6 for SViT-S/B/L. When fine-tuning our models on 384 × 384 resolution, the learning rate, weight decay, batch-size and total epoch are set to 5e-6, 1e-8, 512, 30, respectively. More details about the experimental settings are provided in the Appendix A.3.

Results. We compare our SViT against the state-of-the-art models in Table 2. The comparison results clearly show that our SViT outperforms previous models under different settings in terms of FLOPs and model size. Specifically, our small model SViT-S achieves **83.6%** Top-1 accuracy with only 4.4G FLOPs, surpassing Swin-T, CSwin-T and MPViT-S by **2.3%**, **0.9%** and **0.6%**, respectively. It achieves the same accuracy as Focal-S with half of parameters, half of FLOPs and triplet speed. Our base Model SViT-B achieves **84.8%** accuracy, not only surpassing the corresponding counterparts with medium model size but also surpassing those with large model size. As for 384 × 384 input size, our large model SViT-L achieves an accuracy of **86.4%**, surpassing Swin-B and CSwin-B by **2.2%** and **0.9%**, respectively, and outperforming CaiT-s48 by **1.3%** with 22% fewer FLOPs and double speed. The comparisons against the SOTA methods demonstrate the powerful learning capacity of our SViT.

4.2. Object Detection and Instance Segmentation

Settings. Experiments on object detection and instance segmentation are conducted on COCO 2017 dataset [28]. Following [31], we use our models as the backbone network, and take Mask-RCNN [18] and Cascaded Mask R-CNN [4] as the detection and segmentation heads. The backbones are pretrained on ImageNet-1K, and fine-tuned on the COCO training set with the AdamW optimizer. We adopt two common experimental settings: "1 ×" (12 training epochs) and "3 × +MS" (36 training epochs with multi-scale training). The configurations follow the setting used in Swin Transformer [31] and are implemented with MMDetection [6].

Results. We report the comparison results on the object detection task and instance segmentation task in Table 3 and Table 4. The results show that our SViT variants outperform all the other vision backbones. For Mask R-CNN

Table 6. Ablations of SViT.

Model	Params	FLOPs	Top-1
DeiT-S	22M	4.6G	79.9%
Swin-T	29M	4.5G	81.3%
STA-4stage	24.1M	3.97G	82.3%
+ Conv Stem	24.2M	4.26G	82.6%
+ Projection	25.2M	4.29G	82.9%
+ CPE	25.3M	4.30G	83.3%
w/o CPE, + APE	24.2M	4.26G	83.1%
w/o CPE, + RPE	25.3M	4.29G	83.2%
+ ConvFFN	25.4M	4.37G	83.6%
w/o shortcut	25.4M	4.37G	83.4%

Table 7. Ablations of STA.

Grid size	Iteration	FLOPs	Top-1
8,1,1,1	1	4.97G	81.7%
8,4,1,1	1	3.97G	82.3%
8,4,2,1	1	3.27G	81.3%
8,4,1,1	1	3.97G	82.3%
4,2,1,1	1	4.15G	82.1%
8,4,1,1	0	3.93G	81.9%
8,4,1,1	1	3.97G	82.3%
8,4,1,1	2	4.00G	82.1%

framework, our SViT-S outperforms Uniformer-S by **+2.0** box AP, **+1.5** mask AP under the $1\times$ schedule. For Cascade Mask R-CNN framework, our SViT-B not only outperforms other backbones with similar FLOPs but also achieves better performance than those models having more FLOPs, e.g., outperforming Swin-B by **+2.0** box AP, **+1.5** mask AP and outperforming DAT-B by **+0.9** box AP, **+1.0** mask AP.

4.3. Semantic Segmentation

Settings. We conduct semantic segmentation experiments on the ADE20K dataset [62]. We adopt Upernet [48] as the segmentation heads and replace the backbones with our SViT-B. We follow the setting in Swin Transformer [31] to train Upernet for 160k iterations.

Results. We provide the comparison results on semantic segmentation in Table 5. Our SViT-B achieves **+3.1** higher mIOU than Swin transformer with similar model size and can even achieve **+2.6** higher mIOU than Swin transformer with larger model size. It outperforms Uniformer by **+1.2** mIOU and **+1.2** Multi-Scale(MS) mIOU. The superior semantic segmentation performance further validates the effectiveness of our method.

4.4. Ablation Study

We conduct ablation studies to examine the role of each component of SViT. In Table 6, we start with a hierarchical 4-stage network with STA and then progressively add the rest modules of SViT, i.e., the convolution stem, the 1×1 convolution projection, the CPE and the ConvFFN. The performance gradually increases, implying the effectiveness of each component. More discussions about the STT block are detailed as follows.

Super Token Attention. As shown in Table 6, we firstly develop a simple hierarchical ViT backbone with STA, which has 4 stages like SViT-S. With no convolutional modules, it outperforms DeiT-S and Swin-T by 2.4% and 1.0%, respectively. Note that it does not use position embedding while DeiT uses absolute PE, and Swin uses relative PE. The superior performance validate the effectiveness of STA.

Based on the STA-4stage backbone, we compare different variants of STA and report the results in Table 7. Specifically, we vary the grid sizes (the initial sizes of super tokens) and the sampling iterations. Note that when the grid size equals to 1, we ignore the super tokens sampling process and view the tokens themselves as super tokens.

Table 7 shows that applying super token sampling works well in the first two stags but would hurt performance in the third stage. This is reasonable since super tokens are developed to address local redundancy in the shallow layers. We also observe that larger grid sizes yield better performance. This may be attributed to the fact that larger super tokens can cover more tokens and capture global representations better .

As shown in Table 7, using a single iteration achieves the best performance. When the number of iterations is zero, the super tokens are downsampled features by average pooling and may cover different semantic regions, resulting in performance drop. Using two sampling iterations achieves a slightly worse performance with 0.03G more FLOPs. This indicates that a single iteration is enough for STA.

Convolution Position Embedding. The results in Table 6 show that adding position information is vital for SViT. All the PEs can improve the performance, e.g., APE by 0.2%, RPE by 0.3% and CPE by 0.4%. SViT adopts CPE since it is more flexible for arbitrary resolutions.

Convolution FFN. We use ConvFFN to enhance local features and the performance gain by ConvFFN validates its effectiveness. The shortcut connection in ConvFFN is also helpful for the final performance.

5. Conclusion

We present a super token vision transformer (SViT) to learn efficient and effective global representations in the shallow layers. We adapt the design of superpixels into the token space and introduce super tokens that aggregate similar tokens together. Super tokens have distinct patterns, thus reducing local redundancy. The proposed super token attention decomposes vanilla global attention into multiplications

of a sparse association map and a low-dimensional attention, leading to high efficiency in capturing global dependencies. Extensive experiments on various vision tasks, including image classification, object detection, instance segmentation and semantic segmentation, demonstrate the effectiveness and superiority of the proposed SViT backbone.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *TPAMI*, 34(11):2274–2282, 2012. 3

[2] Radhakrishna Achanta and Sabine Susstrunk. Superpixels and polygons using simple non-iterative clustering. In *CVPR*, pages 4651–4660, 2017. 3

[3] Lile Cai, Xun Xu, Jun Hao Liew, and Chuan Sheng Foo. Revisiting superpixels for active learning in semantic segmentation with realistic annotation costs. In *CVPR*, pages 10988–10997, 2021. 3

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, pages 6154–6162, 2018. 7, 12

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229, 2020. 1

[6] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 7, 12

[7] 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. In *NeurIPS*, 2021. 7

[8] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021. 3

[9] MMSegmentation Contributors. Mmsegmentation, an open source semantic segmentation toolbox, 2020. 12

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPRW*, pages 702–703, 2020. 11

[11] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. 34:3965–3977, 2021. 1, 6

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 5

[13] 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. 3

[14] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. In *CVPR*, 2022. 1, 3, 6, 7

[15] 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. 1, 3

[16] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 3

[17] Jianyuan Guo, Kai Han, Han Wu, Yehui Tang, Xinghao Chen, Yunhe Wang, and Chang Xu. Cmt: Convolutional neural networks meet vision transformers. In *CVPR*, pages 12175–12185, 2022. 3, 6, 7

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 7, 12

[19] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 2016. 7

[20] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661, 2016. 7, 12

[21] Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu. Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arXiv preprint arXiv:2106.03650*, 2021. 7

[22] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, pages 603–612, 2019. 3

[23] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel sampling networks. In *ECCV*, pages 352–368, 2018. 2, 3, 4

[24] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020. 3

[25] Youngwan Lee, Jonghee Kim, Jeffrey Willette, and Sung Ju Hwang. Mpvit: Multi-path vision transformer for dense prediction. In *CVPR*, pages 7287–7296, 2022. 3, 6

[26] Kunchang Li, Yali Wang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatiotemporal representation learning. In *ICLR*, 2022. 1, 2, 3, 7

[27] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *CVPR*, pages 4804–4814, 2022. 3, 6

[28] 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 *ECCV*, pages 740–755, 2014. 5, 7

[29] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *CVPR*, pages 2097–2104, 2011. 3

[30] Yong-Jin Liu, Cheng-Chi Yu, Min-Jing Yu, and Ying He. Manifold slic: A fast method to compute content-sensitive superpixels. In *CVPR*, pages 651–659, 2016. 3

[31] 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*, pages 10012–10022, 2021. 1, 2, 3, 4, 6, 7, 8, 12

[32] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, pages 11976–11986, 2022. 1, 6

[33] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 12

[34] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? In *NeurIPS*, volume 34, pages 12116–12128, 2021. 1

[35] Sucheng Ren, Daquan Zhou, Shengfeng He, Jiashi Feng, and Xinchao Wang. Shunted self-attention via multi-scale token aggregation. In *CVPR*, pages 10853–10862, 2022. 3

[36] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *ICCV*, volume 2, pages 10–10, 2003. 3

[37] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL-HLT (2)*, 2018. 3, 4

[38] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, pages 7262–7272, 2021. 1

[39] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021. 1, 2, 6, 11

[40] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *ICCV*, pages 32–42, 2021. 1, 6

[41] Wei-Chih Tu, Ming-Yu Liu, Varun Jampani, Deqing Sun, Shao-Yi Chien, Ming-Hsuan Yang, and Jan Kautz. Learning superpixels with segmentation-aware affinity loss. In *CVPR*, pages 568–576, 2018. 3

[42] 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*, pages 5998–6008, 2017. 1, 3, 4

[43] 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 *ICCV*, pages 568–578, 2021. 3, 6, 7

[44] Wenxiao Wang, Lu Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu. Crossformer: A versatile vision transformer hinging on cross-scale attention. In *ICLR*, 2022. 1, 3, 6, 7

[45] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. In *CVPR*, pages 8741–8750, 2021. 1

[46] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *ICCV*, pages 22–31, 2021. 6

[47] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *CVPR*, pages 4794–4803, 2022. 3, 6, 7

[48] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, pages 418–434, 2018. 8, 12

[49] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995, 2017. 7

[50] Fengting Yang, Qian Sun, Hailin Jin, and Zihan Zhou. Superpixel segmentation with fully convolutional networks. In *CVPR*, pages 13964–13973, 2020. 3

[51] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. In *NeurIPS*, 2021. 3, 6, 7

[52] Donghun Yeo, Jeany Son, Bohyung Han, and Joon Hee Han. Superpixel-based tracking-by-segmentation using markov chains. In *CVPR*, pages 1812–1821, 2017. 3

[53] Qihang Yu, Yingda Xia, Yutong Bai, Yongyi Lu, Alan L Yuille, and Wei Shen. Glance-and-gaze vision transformer. *NeurIPS*, 34, 2021. 3

[54] 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 *ICCV*, pages 558–567, 2021. 3

[55] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, pages 6023–6032, 2019. 11

[56] Wang Zeng, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Not all tokens are equal: Human-centric visual analysis via token clustering transformer. In *CVPR*, pages 11101–11111, 2022. 3

[57] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 11

[58] Long Zhao, Zizhao Zhang, Ting Chen, Dimitris Metaxas, and Han Zhang. Improved transformer for high-resolution gans. *NeurIPS*, 34, 2021. 3

[59] Minghang Zheng, Peng Gao, Renrui Zhang, Kunchang Li, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. In *BMVC*, 2021. 3

[60] 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 *CVPR*, pages 6881–6890, 2021. 1

[61] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, volume 34, pages 13001–13008, 2020. 11

[62] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *IJCV*, 127(3):302–321, 2019. 5, 8

[63] 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*, 2020. 1

A. Appendix

Algorithm 1 Pseudocode of Super Token Attention

```
# input: (B, C, H, W), output: (B, C, H, W)
# grid_size: (h, w)
# super token number: m=p*q, where p=H//h, q=W//w
# number of iterations: n_iter, default as 1
# scale: C**-0.5, eps: 1e-12

import torch.nn.functional as F
from einops import rearrange

def MultiHeadSelfAttention(x): return x

# rearrange the tokens
tokens = rearrange(x, "bc(yh)(xw)->b(yx)(hw)c", y
    =p, x=q)

# compute the initial super tokens
stokens = F.adaptive_avg_pool2d(x, (p, q))

# compute the associations iteratively
for idx in range(n_iter):
    # extract the 9 surrounding super tokens
    stokens = F.unfold(stokens, kernel_size=3)
    stokens = stokens.transpose(1, 2).reshape(B, p
        *q, C, 9)

    # compute sparse associations (B, p*q, h*w, 9)
    association = tokens @ stokens * scale
    association = association.softmax(-1)

    # prepare for association normalization
    association_sum = association.sum(2).transpose
        (1, 2).reshape(B, 9, p*q)
    association_sum = F.fold(association_sum,
        output_size=(p, q), kernel_size=3)

    # compute super tokens
    stokens = tokens.transpose(-1, -2) @
        association
    stokens = stokens.permute(0, 2, 3, 1).reshape(
        B, C*9, p*q)
    stokens = F.fold(stokens, output_size=(p, q),
        kernel_size=3)
    stokens = stokens/(association_sum + eps)

# MHSA for super tokens
stokens = MultiHeadSelfAttention(stokens)

# map super tokens back to tokens
stokens = F.unfold(stokens, kernel_size=3)
stokens = stokens.transpose(1, 2).reshape(B, p*q,
     C, 9)
tokens = stokens @ association.transpose(-1, -2)
output = rearrange(tokens, "b(yx)c(hw)->bc(yh)(xw
    )", y=p, x=q)
```

A.1. Algorithm of Super Token Attention

The proposed Super Token Attention (STA) consists of three processes, i.e., Super Token Sampling (STS), Multi-Head Self-Attention (MHSA), and Token Upsampling (TU). In particular, STS can be further decomposed into two iterative steps, reformulated as:

$$Q^t = \text{Softmax}(\frac{XS^{t-1^{\mathrm{T}}}}{\sqrt{d}}), \tag{19}$$

$$S^t = (\hat{Q}^t)^{\mathrm{T}} X, \tag{20}$$

where $\hat{Q}^t$ is the column-normalized Q^t. For υ iterations, the complexity of the above steps is $2\upsilon mNC$, which is relatively high. To reduce the complexity, we set υ to 1 and compute the association Q^t in a sparse manner. For each token, only its 3×3 surrounding super tokens are used to compute Q (we omit the index for clarity). We use the Unfold and Fold functions to extract and combine the corresponding 3×3 super tokens, respectively.

The computation details of STA are illustrated in Algo. 1. Given the input tokens $X \in \mathbb{R}^{C \times H \times W}$, we first generate the initial super tokens $S \in \mathbb{R}^{C \times p \times q}$ by average pooling, where $p = \frac{H}{h}$, $q = \frac{W}{w}$, and $h \times w$ is the grid size. We then extract the 3×3 super tokens $\bar{S} \in \mathbb{R}^{pq \times C \times 9}$ corresponding to each token via the Unfold function and compute the association $Q \in \mathbb{R}^{(pq \times hw) \times 9}$. We update $\bar{S}$ and combine the surrounding tokens to S via the Fold function. S is divided by the combined sum of Q for normalization. Since the iteration num is set to 1, we perform the multi-head self-attention on S. Finally, we perform the sparse multiplication of Q and S via the Unfold function like above to map super tokens back to the token space.

A.2. Architecture Details

The architecture details are illustrated in Table 8. For the convolution stem, we adopt four 3×3 convolutions to embed the input image into tokens. GELU and batch normalization are used after each convolution. 3×3 convolutions with stride 2 are used between stages to reduce the feature resolution. 3×3 depth-wise convolutions are adopted in CPE and ConvFFN to enhance the capacity of local modeling. The projection layer is composed of a 1×1 convolution, a batch-normalization layer, and a Swish activation. Global average pooling is used after the projection layer, following by the fully-connected classifier.

A.3. Experimental Settings

ImageNet Image Classification. We adopt the training strategy proposed in DeiT [39]. In particular, our models are trained from scratch for 300 epochs with the input resolution of 224×224. The AdamW optimizer is applied with a cosine decay learning rate scheduler and 5 epochs of linear warm-up. The initial learning rate, weight decay, and batch-size are set to 0.001, 0.05, and 1024, respectively. We apply the same data augmentation and regularization used in DeiT [39]. Specifically, the augmentation settings are RandAugment [10] (randm9-mstd0.5-inc1) , Mixup [57] (prob = 0.8), CutMix [55] (prob = 1.0), Random Erasing [61]

Table 8. Architectures for ImageNet classification with resolution 224×224.

Output Size	Layer Name	SViT-S	SViT-B	SViT-L
56×56	Conv Stem	$3 \times 3, 32$, stride 2 $3 \times 3, 32$ $3 \times 3, 64$, stride 2 $3 \times 3, 64$	$3 \times 3, 48$, stride 2 $3 \times 3, 48$ $3 \times 3, 96$, stride 2 $3 \times 3, 96$	$3 \times 3, 48$, stride 2 $3 \times 3, 48$ $3 \times 3, 96$, stride 2 $3 \times 3, 96$
Stage 1	CPE STA ConvFFN	$\begin{bmatrix} 3 \times 3, 64 \\ \text{grid } 8, \text{heads } 1 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 96 \\ \text{grid } 8, \text{heads } 2 \\ 3 \times 3, 96 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 3, 96 \\ \text{grid } 8, \text{heads } 2 \\ 3 \times 3, 96 \end{bmatrix} \times 4$
28×28	Patch Merging	$3 \times 3, 128$, stride 2	$3 \times 3, 192$, stride 2	$3 \times 3, 192$, stride 2
Stage 2	CPE STA ConvFFN	$\begin{bmatrix} 3 \times 3, 128 \\ \text{grid } 4, \text{heads } 2 \\ 3 \times 3, 128 \end{bmatrix} \times 5$	$\begin{bmatrix} 3 \times 3, 192 \\ \text{grid } 4, \text{heads } 3 \\ 3 \times 3, 192 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 3, 192 \\ \text{grid } 4, \text{heads } 3 \\ 3 \times 3, 192 \end{bmatrix} \times 7$
14×14	Patch Merging	$3 \times 3, 320$, stride 2	$3 \times 3, 384$, stride 2	$3 \times 3, 448$, stride 2
Stage 3	CPE STA ConvFFN	$\begin{bmatrix} 3 \times 3, 320 \\ \text{grid } 1, \text{heads } 5 \\ 3 \times 3, 320 \end{bmatrix} \times 9$	$\begin{bmatrix} 3 \times 3, 384 \\ \text{grid } 1, \text{heads } 6 \\ 3 \times 3, 384 \end{bmatrix} \times 14$	$\begin{bmatrix} 3 \times 3, 448 \\ \text{grid } 1, \text{heads } 7 \\ 3 \times 3, 448 \end{bmatrix} \times 19$
7×7	Patch Merging	$3 \times 3, 512$, stride 2	$3 \times 3, 512$, stride 2	$3 \times 3, 640$, stride 2
Stage 4	CPE STA ConvFFN	$\begin{bmatrix} 3 \times 3, 512 \\ \text{grid } 1, \text{heads } 8 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 512 \\ \text{grid } 1, \text{heads } 8 \\ 3 \times 3, 512 \end{bmatrix} \times 6$	$\begin{bmatrix} 3 \times 3, 640 \\ \text{grid } 1, \text{heads } 10 \\ 3 \times 3, 640 \end{bmatrix} \times 8$
7×7	Projection	$1 \times 1, 1024$		
1×1	Classifier	Fully Connected Layer, 1000		
# Params		25 M	52 M	95 M
# FLOPs		4.4 G	9.9 G	15.6 G

(prob = 0.25). We do not use Exponential Moving Average (EMA) [33]. The maximum rates of increasing stochastic depth [20] are set to 0.1, 0.4, 0.6 for SViT-S, SViT-B, SViT-L, respectively. For 384×384 input resolution, the models are fine-tuned for 30 epochs with learning rate of 1e-5, weight decay of 1e-8 and bach-size of 512.

COCO Object Detection and Instance Segmentation. We apply Mask-RCNN [18] and Cascaded Mask R-CNN [4] as the detection heads based on MMDetection [6]. The models are trained under two common settings: "1 $\times$" (12 training epochs) and "3 $\times$ +MS" (36 training epochs with multi-scale training). For the "1 $\times$" setting, images are resized to the shorter side of 800 pixels while the longer side is within 1333 pixels. For the "3 $\times$ +MS", we apply the multi-scale training strategy to randomly resize the shorter side between 480 to 800 pixels. We apply the AdamW optimizer with the initial learning rate of 1e-4 and weight decay of 0.05. We set the stochastic depth rates to 0.1 and 0.3 for our small and base models, respectively.

ADE20K Semantic Segmentation. We apply our model as the backbone network and Upernet [48] as the segmentation head based on MMSegmentation [9]. We follow the setting used in Swin Transformer [31] and train the model for 160k iterations with the input resolution of 512×512. We set the stochastic depth rate to 0.3 for our SViT-B backbone.

A.4. Limitations and Broader Impacts

One possible limitation of the proposed SViT is that it adopts some operations that are not computational efficient for GPU, such as the Fold and Unfold operations in STA and the depth-wise convolutions in CPE and ConvFFN. This makes our models not among the fastest under similar settings of FLOPs. Besides, due to computational constraint, it is not trained on large scale datasets (e.g., ImageNet-21K), which will be explored in the future.

SViT is a general vision transformer that can be applied on different vision tasks. It has no direct negative social impact. Possible malicious uses of SViT as a general-purpose backbone are beyond the scope of our study to discuss.