# Vision Transformer with Super Token Sampling

In this supplementary material, we first describe the implementation details of Super Token Attention. Then we provide the architecture details of STViT, following by the experimental settings. We also provide ablation study under isotropic settings. At last, we briefly discuss about the limitations and broader impacts of our study.

## A. 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:

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

$$S^t = (\hat{Q}^t)^{\mathrm{T}} X, \tag{2}$$

where $\hat{Q}^t$ is the column-normalized $Q^t$. For $v$ iterations, the complexity of the above steps is $2vmNC$, which is relatively high. To reduce the complexity, we set $v$ to 1 and compute the association $Q^t$ in a sparse manner. For each token, only its $3 \times 3$ 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 $3 \times 3$ 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 $3 \times 3$ 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.

**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)
```

## B. Architecture Details

The architecture details are illustrated in Table I. For the convolution stem, we adopt four $3 \times 3$ convolutions to embed

Table I. Architectures for ImageNet classification with resolution $224 \times 224$.

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

the input image into tokens. GELU and batch normalization are used after each convolution. $3 \times 3$ convolutions with stride 2 are used between stages to reduce the feature resolution. $3 \times 3$ depth-wise convolutions are adopted in CPE and ConvFFN to enhance the capacity of local modeling. The projection layer is composed of a $1 \times 1$ convolution, a batch-normalization layer, and a Swish activation. Global average pooling is used after the projection layer, following by the fully-connected classifier.

## C. Experimental Settings

**ImageNet Image Classification.** We adopt the training strategy proposed in DeiT [9]. In particular, our models are trained from scratch for 300 epochs with the input resolution of $224 \times 224$. 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 [9]. Specifically, the augmentation settings are RandAugment [4] (randm9-mstd0.5-inc1) , Mixup [12] (prob = 0.8), CutMix [11] (prob = 1.0), Random Erasing [13] (prob = 0.25). We do not use Exponential Moving Average (EMA) [8]. The maximum rates of increasing stochastic depth [6] are set to 0.1, 0.4, 0.6 for STViT-S, STViT-B, STViT-L, respectively. For $384 \times 384$ 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 [5] and Cascaded Mask R-CNN [1] as the detection heads based on MMDetection [2]. 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

Table II. Comparing isotropic STViT and Swin without Conv Stem, Projection, CPE and ConvFFN. Swin$^{\dagger}$ uses global SA in 3rd stage.

| Blocks | Channels | RPE | Model | #Params | FLOPs | Acc. |
|---|---|---|---|---|---|---|
| [96, 192, 384, 768] | [2, 2, 6, 2] | √ | Swin | 28M | 4.5G | 81.3% |
| [96, 192, 384, 768] | [2, 2, 6, 2] | √ | STViT | 30M | 4.3G | **82.3%** |
| [64, 128, 320, 512] | [3, 5, 9, 3] | √ | Swin | 23M | 4.2G | 81.4% |
| [64, 128, 320, 512] | [3, 5, 9, 3] | √ | Swin$^{\dagger}$ | 23M | 4.3G | 81.7% |
| [64, 128, 320, 512] | [3, 5, 9, 3] | √ | STViT | 24M | 4.0G | **82.6%** |
| [64, 128, 320, 512] | [3, 5, 9, 3] | × | Swin | 23M | 4.2G | 80.2% |
| [64, 128, 320, 512] | [3, 5, 9, 3] | × | STViT | 24M | 4.0G | **82.3%** |

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 [10] as the segmentation head based on MMSegmentation [3]. We follow the setting used in Swin Transformer [7] and train the model for 160k iterations with the input resolution of $512 \times 512$. We set the stochastic depth rate to 0.3 for our STViT-B backbone.

## D. Isotropic comparison

We have conducted comparisons under isotropic settings. For fair comparison, as shown in Tab. II, we compare STViT against Swin with three variants according to the number of blocks, channels, and using relative position encoding (RPE) or not. STViT outperforms Swin under similar settings, implying that STA itself can bring large performance gain.

## E. Limitations and Broader Impacts

One possible limitation of the proposed STViT 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.

The proposed STViT is a general vision transformer that can be applied on different vision tasks, e.g., image classification, object detection and semantic segmentation. It has no direct negative social impact. Possible malicious uses of STViT as a general-purpose backbone are beyond the scope of our study to discuss.

## References


[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, pages 6154–6162, 2018. 2

[2] 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. 2

[3] MMSegmentation Contributors. Mmsegmentation, an open source semantic segmentation toolbox, 2020. 3

[4] 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. 2

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 2

[6] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661, 2016. 2

[7] 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. 3

[8] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 2

[9] 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. 2

[10] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, pages 418–434, 2018. 3

[11] 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. 2

[12] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 2

[13] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, volume 34, pages 13001–13008, 2020. 2