## Supplementary Material

## A Results on ImageNet-1K classification

**Setup** We also train and evaluate SViT-S on ImageNet-1K [1]. We follow the training settings in DeiT [9] 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 1. 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 2. Although SViT is not originally targeted at classification tasks, it outperforms all models that use gating modules (DynamicViT [8], SPViT [4], AdaViT [6]), the models using special pruning techniques such as adaptive computation time [3] (A-ViT [11]) and reinforcement learning (IA-RED2 [7]), and a model that uses class token's attention (Evo-ViT [10]). 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 [9] | 79.8 | 4.6 | 1524 |
| SViT-S | 79.4 | 3.0 | 2246 |

Table 1. 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 [9] | - | 4.6 | 79.8 |
| DynamicViT ‡ [8] | 30 | 3.0 | 79.3 (-0.5) |
| EViT [5] | 30 | 3.0 | 79.5 (-0.3) |
| Evo-ViT [10] | 300 * | 3.0 | 79.4 (-0.4) |
| Evo-ViT [10] | 30 † | 3.0 | 79.2 (-0.6) |
| A-ViT [11] | 100 | 3.6 | 78.6 (-1.3) |
| ATS [2] | 30 | 3.0 | 79.7 (-0.1) |
| AdaViT [6] | 150 | 2.3 | 77.3 (-2.5) |
| IA-RED2 [7] | 90 | 3.2 | 79.1 (-0.7) |
| SPViT ‡ [4] | 60 | 2.7 | 79.3 (-0.5) |
| SViT (Ours) | 30 | 3.0 | 79.4 (-0.4) |

Table 2. 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*[1] 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 1, 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.

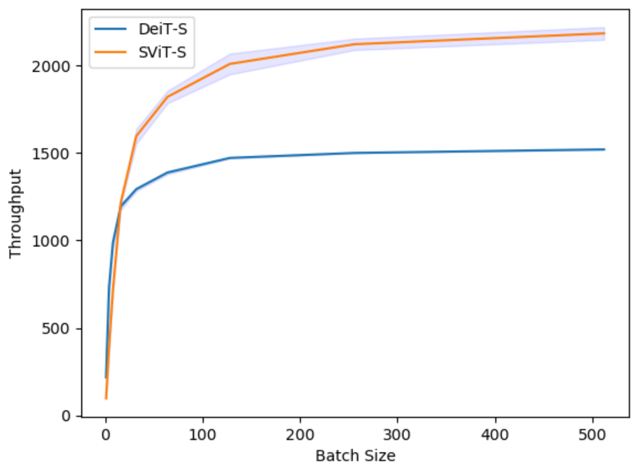


Figure 1. Throughput vs. Batch Sizes of SViT-S and DeiT-S on ImageNet-1K.

[1]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 2, the dense DeiT-S [9] 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 [5], ATS [2], and Evo-ViT [10]), 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 3. 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 3 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 4. 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.

## References

[1] 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. 1

[2] 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] Alex Graves. Adaptive computation time for recurrent neural networks. arXiv preprint arXiv:1603.08983, 2016. 1

[4] 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

[5] 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, 2

[6] 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

[7] 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. 1

[8] 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

[9] 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

[10] 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, 2

[11] 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. 1

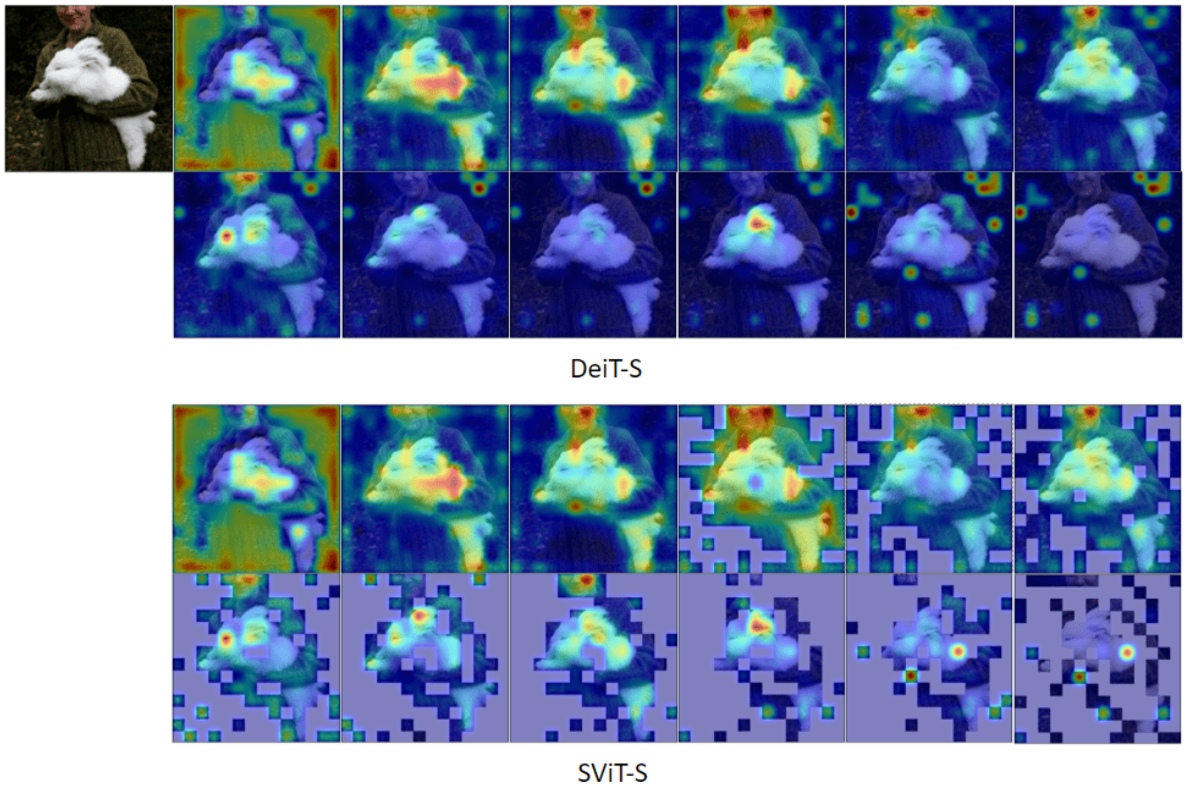


Figure 2. The first row: the attention from the 1st layer to the 6th layer; the second row: the attention maps from the 7th layer to the 12th layer. Compared to the dense DeiT-S model, SViT-S keeps the most important features for each layer, especially the human face, the rabbit eyes, and the rabbit 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.

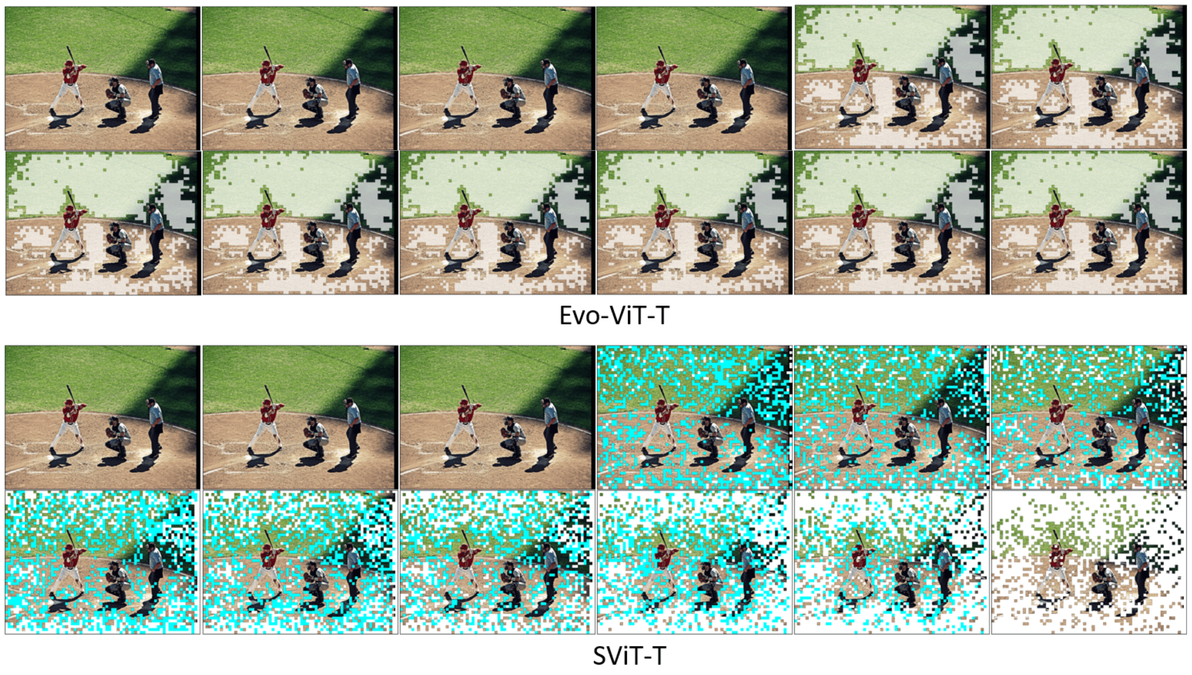
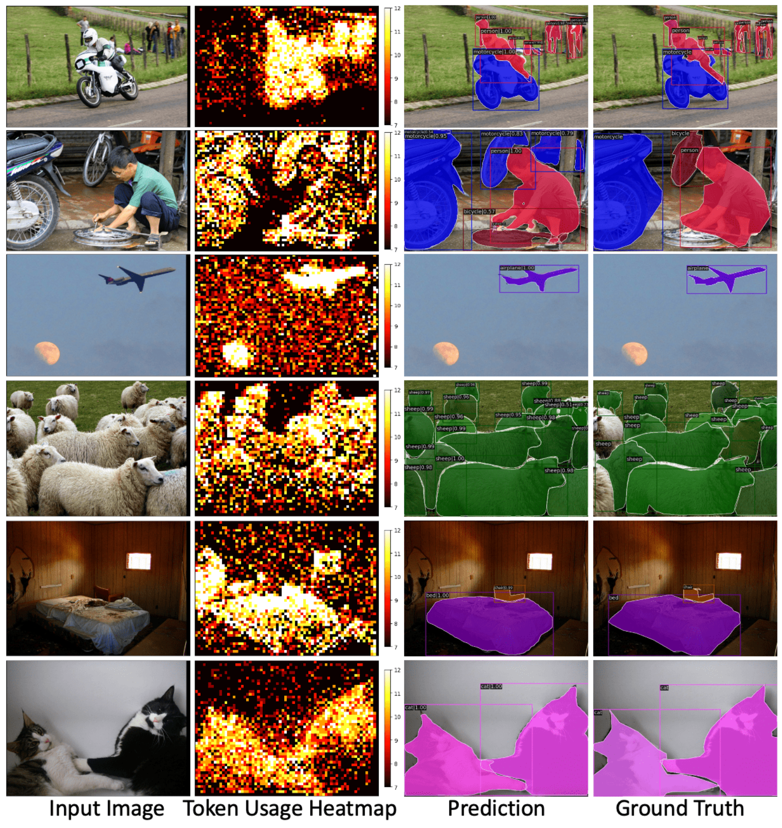


Figure 3. Top: token pruning for Evo-ViT-T. Bottom: token pruning for SViT-T. Evo-ViT has the same keep ratio per layer, and tends to use the same set of tokens for all its pruning layers, so the selection largely depends on its first pruning layer, which may not be optimal. SViT can prune tokens independently for each layer, which is learned by the gating MLPs, and can reuse tokens according as needed.



| Foreground usage | Background usage |
|---|---|
| 91.33 % | 43.49 % |
| 65.41 % | 35.03 % |
| 97.47 % | 50.49 % |
| 59.63 % | 32.88 % |
| 71.32 % | 46.84 % |
| 63.35 % | 42.63 % |

Figure 4. 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.