A Analysis and Discussion

A.1 Model Scaling

In ViTs, the most common method to scale the model is to change the number of channels, while our SPViT provides another perspective to perform token pruning for better complexity/accuracy trade-offs. We illustrate this superior effect of SPViT in Figure 8. First, we train several DeiT [25] models with varying embedding dimensions from 192 to 384. Second, we compress these DeiTs into ones whose channel has one less head than them. However, these new compressed models (SPViT-serious) have better accuracy than DeiT original variants with similar computation. Specifically, the orange line of SPViT-serious is closer to the upper left corner of the Figure 8 than the original DeiT-serious. For DynamicViT, we have also observed uniform benefits, but its efficient models are still slightly inferior to SPViT.

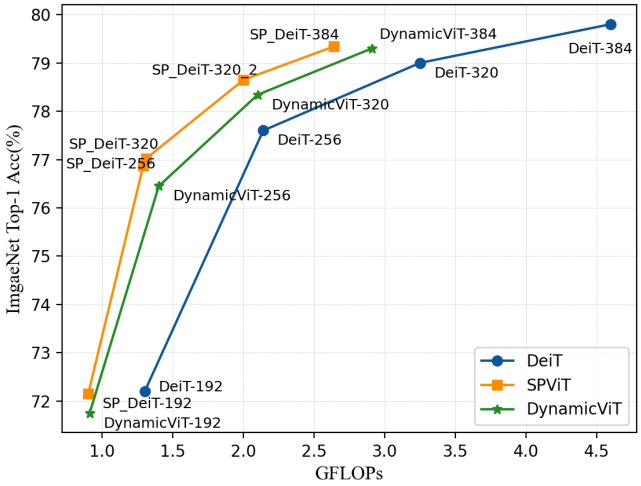

Fig. 8: Comparison of our SPViT method with model scaling. We prune DeiT models with embedding dimensions varying from 192 to 384 and compare with DynamicViT under comparable GFLOPS.

A.2 Progressive Training Sparsity for Each Layer

As a supplement for Figure 7, we show the exact sparsity and accuracy of each layer for our progressive training in Table 7. We start by adding the token selector one by one from the 11th layer to the 1st layer. The layer index indicates the layer before the token selector. Between 6∼8 and 9∼11 layer, each layer has similar accuracy and sparsity, which indicates that these layers can be combined to one pruning phase with a token selector at the front.

Layer	1	2	3	4	5	6	7	8	9	10	11
Accuracy	75.7	77.3	79.4	79.5	79.6	79.7	79.7	79.7	79.8	79.8	79.8
Sparsity	0.15	0.17	0.39	0.47	0.64	0.71	0.77	0.85	0.89	0.92	0.93

Table 7: Progressive training sparsity for each layer

A.3 Model Latency on Hardware

We show all model latency results tested on Samsung Galaxy S20 in Table 8. On the one hand, our models can outperform lightweight models such as DeiT-T by up to 4.8% with even smaller latency (38ms vs. 44ms). On the other hand, we are able to reduce the latency of larger models such as DeiT-S by up to 47% (60ms vs. 113ms) with only 0.46% decrease in accuracy. On LV-ViT-S/M, our SPViT models show better performance, outperforming DynamicViT on both latency and accuracy.

Model	**Method**	**Top-1 Acc. (%)**	**Latency (ms)**
DeiT-T	Baseline	72.20	44
	S^2ViTE	70.12	35
	DynamicViT	71.85	37
	SPViT (Ours)	**72.20**	**33**
	SPViT (Ours)	**72.10**	**26**
	SPViT-256 (Ours)	**76.87**	**36**
	SPViT-320 (Ours)	**77.02**	**38**
DeiT-S	Baseline	79.80	113
	S^2ViTE	79.22	78
	IA-RED^2	79.10	80
	DynamicViT	79.30	72
	SPViT-320 (Ours)	**78.65**	**47**
	SPViT (Ours)	**79.34**	**60**
	SPViT (Ours)	**77.02**	**38**
LV-ViT-S	Baseline	83.30	148
	DynamicViT	83.00	114
	SPViT (Ours)	**83.10**	**89**
LV-ViT-M	Baseline	84.00	269
	DynamicViT	83.61	195
	SPViT (Ours)	**73.71**	**152**

Table 8: Evaluation results on Samsung Galaxy S20 with a Snapdragon 865 processor.

A.4 Number of Package Tokens.

We insert three soft pruning modules for hierarchical pruning for all models. In each module, a new package token is generated. We conduct two ways to pass on

Method	Params (M)	GFLOPs	Top-1 Acc. (%)
Baseline	22.10	4.60	79.80
1 × Package Token	22.13	2.63	79.26
+ Attention-based Branch	22.13	2.64	79.30
3 × Package Token	22.13	2.64	79.28
+ Attention-based Branch	22.13	2.64	**79.34**

Table 9: Package token number comparison on DeiT-S

the package token for subsequent layers: 1) Merge the package token generated from the current module with the existing one from the last module by element-wise addition. Therefore, only 1 additional token is added to the input sequence in total. 2) Concatenate the new package token to the existing ones. Making it 3 additional tokens in total. Table 9 shows a comparison of the two methods.

A.5 Comparison of Different Token Selector Designs

We analyze the performance of our token selector design by replacing it with different operations. More specifically, we replace the original MLP based pipeline in Eq. (1) and (2) with a convolution-based pipeline: $Conv1d \rightarrow BatchNorm1d \rightarrow GELU$. We also evaluated different activation functions: RELU and Hardswish. We compare these variants of the token selector on DeiT-T with a complexity of 0.9 GFLOPs. As shown in Table 10, under the same training settings, MLP based token selectors outperform convolution-based token selectors (72.10% vs 71.56%). Furthermore, GELU function outperforms Hardswish and RELU (71.56% vs. 71.48% vs. 71.13% on Conv1d+3kernel). However, we can not hastily conclude that MLP and GELU are superior to convolution and other activation functions. This may be due to different training difficulties. Hardswish is harder to converge, so larger epochs may be necessary [31]. Additionally, Conv1d with 1kernel can mimic MLP's fully connected layer. The accuracy gap between these two may due to distinct desirable initial learning rates and schedulers. Result also shows that a larger kernel size may be preferable (71.34% vs. 71.56% on Conv1d), which can help learn more local representation. We will further invest in these designs in our future work. More specifically, train each design under different training settings and also combine the advantages of MLP and convolution layers.

A.6 Effectiveness of the Token Packaging Technique w/o Class Token

In many recently proposed vision transformer models, the class token was removed and replaced by doing average pooling on the last output feature to aggregate representation from all patch tokens [31,106]. This process is similar to our token packaging technique, where we also apply average pooling on the removed tokens. We raise a conjecture: Our token packaging technique can be

Token Selector	GFLOPs	Top-1 Acc. (%)
MLP+GELU	0.90	**72.10**
MLP+Hardswish	0.90	71.94
Conv1d+3kernel+RELU	0.90	71.13
Conv1d+3kernel+GELU	0.90	71.56
Conv1d+3kernel+Hardswish	0.90	71.48
Conv1d+1kernel+GELU	0.90	71.34

Table 10: Comparison of different token selector layers and activation functions

more effective on models that do not rely on the class token. To test our hypothesis, we run a simple experiment by first pre-training a DeiT-T without a class token, and then applying our method both with and without the token packaging technique. As shown in Table 11, when training on a DeiT model that contains a class token, our token packaging technique can improve the performance by 0.12% (72.1% vs. 72.0%). When training on a DeiT model without a class token, our token packaging technique can improve the performance by 0.23% (70.75% vs. 70.52%), which verified our assumption.

A.7 Comparison of Different Batch Sizes

We run our SPViT on DeiT-S with different batch sizes for ablation. Results in Table 12 show that accuracy has a slight boost when batch size increases.

A.8 Encoding Redundancy of the Pooling Layer

We make further discussion on Pooling-based ViT (PiT Series). Figure 9 shows that the attention matrix before and after SPViT retains great similarity, which enlightens that the encoding redundancy of the pooling-layer mechanism can be recognized precisely by SPViT.

B Visualization

B.1 Token Pruning Visualization

We further visualize the process of SPViT to describe the performance in the inference phase and make a comparison between the framework with the token packaging technique and without it the token packaging technique. As shown in Figure 10, row 1 and 3 show the results collected from the framework without token packaging, row 2 and 4 are from the framework with token packaging. Take row 1 and 2 for example: At phase 1, there is a 7% difference in the left image groups and 11% in the right ones; At phase 2, 11% difference in the left image groups and 15% in the right ones; At phase 3, 14% difference in the left image

Table 11: Effectiveness of the token packaging technique w/o class token

Model	GFLOPs	Top-1 Acc. (%)
DeiT-T	1.30	72.20
SPViT	0.90	72.10
SPViT w/o package token	0.90	71.98
DeiT-T w/o cls token	1.29	71.42
SPViT	0.87	70.75
SPViT w/o package token	0.87	70.52

Table 12: Different batch size comparison on DeiT-S

Batch Size	GFLOPs	Top-1 Acc. (%)	Top-5 Acc. (%)
96	2.65	79.31	94.64
128	2.65	79.32	94.64
256	2.64	79.34	94.67

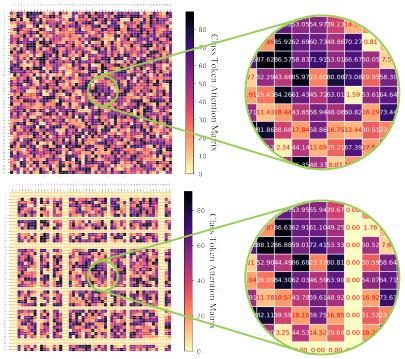

Fig. 9: Illustration of the first attention matrix at the final block. Upper figure is the original PiT-S, lower one is with SPViT.

groups and 18% in the right ones. We can infer token packaging can help to lock the object instead of the background. And this phenomenon is more obvious in complex and multi-object images.

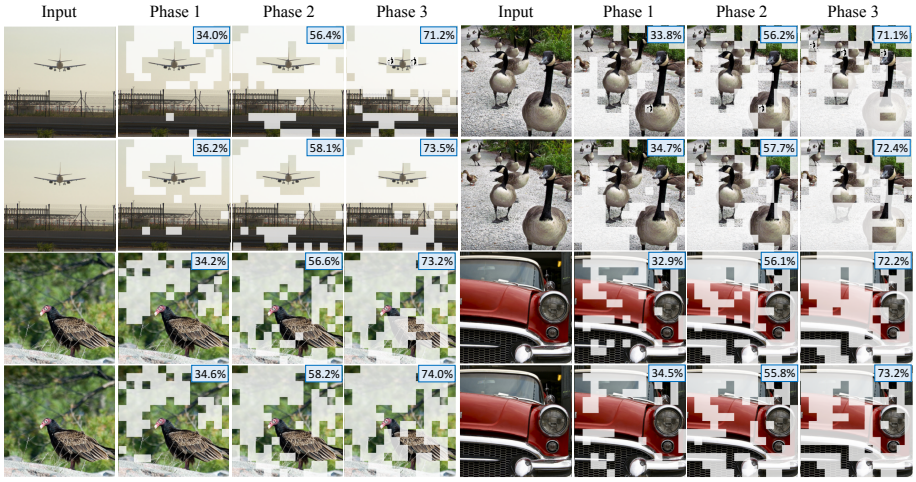

Fig. 10: Visualization of each pruning phase. Row 1 and 3 show the results collected from the framework without token packaging and row 2 and 4 show the results from the framework with token packaging.

B.2 Self-attention Head Heatmap

Figure 11 shows the heatmaps of informative region detected by each self-attention head in DeiT-S. Each attention head focuses on encoding different im-

age features and visual receptive fields. Therefore, token importance is different for each head. This demonstrates the need for obtaining token score individually.

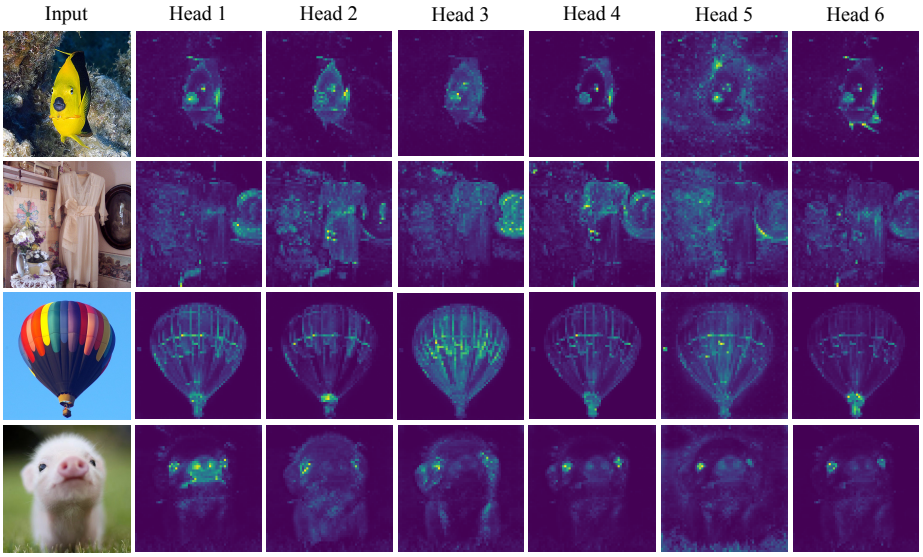

Fig. 11: Heatmaps showing the informative region detected by each head in DeiT-S.

C Main Results

We compare our method with several representative methods including DynamicViT [69], IA-RED^2 [64], RegNetY [67], CrossViT [7], VTP [112], ATS [28], CvT [85], PVT [83], T2T-ViT [104], UP-DeiT [95], PS-ViT [78], Evo-ViT [89], TNT [34], HVT [65], Swin [49], CoaT [88], CPVT [18], EViT [46], UVC [97], MD-DeiT [39],and S^{2}ViTE [12]. As shown in Table 13, we report the top-1 accuracy and GFLOPs for each model. Note that "*" refers to the results reproduced with similar GFLOPs for comparison.

Model	Method	Params (M)	GFLOPs	GFLOPs ↓ (%)	Top-1 Acc. (%)
	Baseline/192	5.60	1.30	0.00	72.20
	Baseline/160*	4.00	0.90	30.77	68.10
	NViT-T	6.40	1.30	0.00	73.91
	UP-DeiT-T	5.70	1.30	0.00	75.79
	MD-DeiT-T	5.70	1.30	0.00	75.06
	T2T-ViT-10	5.90	2.13	-63.85	75.00
	UVC	-	0.64	-50.74	71.3
DeiT-T	PVT-Tiny	13.20	2.64	-103.08	75.00
	DynamicViT	5.90	0.91	30.00	71.85
	PS-ViT	5.60	0.93	28.46	72.00
	S^2ViTE	4.20	0.95	26.92	70.12
	ATS+DeiT/258	10.13	1.50	-15.38	76.90
	SPViT (Ours)	5.70	1.00	23.08	**72.20**
	SPViT (Ours)	5.70	0.90	30.77	**72.10**
	SPViT-256 (Ours)	10.16	1.29	0.77	76.87
	SPViT-320 (Ours)	15.44	1.30	0.00	**77.02**
	Baseline/384	22.10	4.60	0.00	79.80
	Baseline/320*	15.40	3.25	29.35	79.00
	Baseline/288*	12.60	2.65	42.39	78.53
	Baseline/256*	10.00	2.14	53.48	77.21
	HVT-S-1	22.10	2.40	47.82	78.00
	S^2ViTE	14.60	3.14	31.74	79.22
DeiT-S	IA-RED^2	-	3.15	31.52	79.10
	DynamicViT	22.80	2.91	36.74	79.30
	DynamicViT*	22.80	2.71	41.09	79.12
	SPViT-320 (Ours)	15.44	2.00	56.52	78.65
	SPViT (Ours)	22.13	3.86	16.09	**79.80**
	SPViT (Ours)	22.13	2.64	42.61	**79.34**
	Baseline/384	26.15	6.55	0.00	83.30
	Swin-T	29.00	4.50	31.29	81.30
	CvT-13/224	20.00	4.50	31.29	81.60
	MD-DeiT-S	22.10	4.60	29.77	81.48
	CPVT-Small-GAP	23.00	4.60	29.77	81.50
	NViT-S	23.00	4.70	28.24	81.22
LV-ViT-S	ATS+CvT-21	32.00	5.10	22.14	82.30
	T2T-ViT-14	22.00	5.20	20.61	81.50
	CrossViT-S	26.70	5.60	14.50	81.00
	PVT-Medium	44.20	6.70	-2.29	81.20
	CoaT Mini	10.00	6.80	-3.82	80.80
	DynamicViT	26.90	4.57	30.22	83.00
	SPViT (Ours)	26.17	4.28	34.65	**83.10**
	Baseline/512	55.83	12.67	0.00	84.00
	CvT-21/224	32.00	7.10	43.96	82.50
	DynamicViT*	57.10	7.35	41.99	83.61
	DynamicViT	57.10	8.45	33.31	83.80
	RegNetY-8G	39.00	8.00	36.86	81.70
	Swin-S	50.00	8.70	31.33	83.00
	T2T-ViT-19	39.20	8.90	29.76	81.90
	Evo-ViT	87.30	9.07	28.41	81.11
	VTP	48.00	10.00	21.07	80.70
	ATS+CvT-13/384	20.00	11.70	7.66	82.90
LV-ViT-M	T2T-ViT-24	64.10	14.10	-11.29	82.30
	TNT-B	66.00	14.10	-11.29	82.80
	Swin-B	88.00	15.40	-21.55	83.30
	RegNetY-16G	84.00	16.00	-26.28	82.90
	CvT-13/384	20.00	16.30	-28.65	83.00
	ATS+CvT-21/384	32.00	17.40	-37.33	83.10
	DeiT-B	86.60	17.60	-38.91	81.80
	CrossViT-B	104.70	21.20	-67.32	82.20
	CvT-21/384	32.00	24.90	-96.53	83.30
	SPViT (Ours)	55.85	7.32	42.23	**83.71**

Table 13: Results of different ViTs on ImageNet-1K. We compare the proposed SPViT with existing ViT pruning methods under comparable GFLOPs and the number of parameters. Note that "*" refers to our reproduced results to obtain models with similar GFLOPs for comparison. Negative values in FLOPs reduction mean FLOP increases. Baseline/160/192/288/384 indicates the embedding dimensions. SPViT-256/320 indicates pruning from DeiT scaling model of 256/320 embedding dimensions.