The Supplementary of "Training-free Transformer Architecture Search"

In the supplementary, we provide additional detailed information for the readers to better understand the proposed DSS-indicator and its working mechanism. The content of this supplementary is listed as follows:

- In Sec. A, we list the performance of searched optimal ViT networks via different zero-cost proxies to better demonstrate the superiority of our DSS-indicator.
- In Sec. B, we provide the pseudo-code of the DSS-indicator to present the implementation details and encourage further investigation.
- In Sec. C, we provide further specific information about the PiT search space.
- In Sec. D, we show some optimal ViT architectures searched on AutoFormer search space and PiT search space with the help of the proposed DSS-indicator.

A. Additional Comparison of Zero-Cost Proxy.

In Tab. A, we list the classification performance of the searched optimal networks with the help of different zero-cost proxy methods. As it is indicated in Table.5 in the manuscript, the ViT architecture searched by our DSS-indicator is better than that of the cutting-edge counterparts [3–5], which have achieved competitive performance in popular CNN search spaces. The results also verify the necessity to design a ViT-oriented performance indicator.

Table A. The classification accuracies on ImageNet of the searched optimal ViT architectures via different zero-cost proxies from the AutoFormer search space.

Proxy	#Param (M)	FLOPs (B)	Top-1 (%)	Top-5 (%)
SNIP [3]	5.8	1.4	74.8	92.7
GraSP [5]	5.5	1.3	74.3	92.2
TE-score [1]	5.6	1.3	74.6	92.6
NASWOT [4]	5.9	1.5	74.8	92.8
DSS-indicator (ours)	5.9	1.4	**75.3**	**92.8**

B. The Pseudo-code of DSS-indicator.

To make it easier to understand and reproduce the proposed ViT-oriented performance indicator, we present the implementation details of DSS-indicator in Algorithm 1. It should be noted that: in the step to `construct the input`, we prepare one tensor, whose values are all 1 (as we mentioned in lines $486-488$ in the main text). Besides, we don't measure the influence from convolution module directly, because we mainly focus on zero-cost performance estimation for vision transformer architectures.

Algorithm 1 DSS-indicator (PyTorch-like)

```
# net: a ViT, input_dim: input dimension
########## definition of DSS-indicator ##########
def DSS_indicator(layer):
    if isinstance(layer, MSA):
        if layer.weight.grad is not None:
            return torch.abs(torch.norm(layer.
                   weight.grad,'nuc') * torch.
                   norm(layer.weight,'nuc'))
        else:
            return torch.zeros_like(layer.weight)

    else if isinstance(layer, MLP):
        if layer.weight.grad is not None:
            return torch.abs(layer.weight.grad
                   * layer.weight)
        else:
            return torch.zeros_like(layer.weight)

############ construct the input ############
inputs = torch.ones([1] + input_dim)

############ forward and backward ############
net.train()
output = net.forward(inputs)
torch.sum(output).backward()

########## measure the DSS-indicator ##########
proxy_array = []
for layer in net.modules():
    proxy_array.append(DSS_indicator(layer))

S_DSS = torch.sum(proxy_array)
```

C. The detail of the PiT search space.

To verify the generalization of TF-TAS, we construct a PiT search space based on PiT [2], which adopts some depth-wise convolution operations as pooling to obtain deep-narrow ViT networks. The detailed configurations of the PiT search space are listed in Tab. C. We set three levels (*i.e.*, *tiny*, *extra small*, and *small*) in the PiT search space. The patch size and stride of these levels are set to 16 and

Table B. The details of the searched architectures on AutoFormer search space via the proposed DSS-indicator.

Embed Dim	MLP Ratio	Head Num	Depth Num
Tiny: TF-TAS-Ti in Table.1			
192	3.5, 4.0, 3.5, 3.5, 4.0, 3.5, 3.5, 3.5, 4.0, 4.0, 4.0, 4.0, 3.5	4, 3, 3, 3, 3, 4, 4, 3, 3, 3, 4, 3, 3	13
Small: TF-TAS-S in Table.1			
384	4.0, 4.0, 4.0, 3.0, 4.0, 3.0, 3.0, 3.5, 4.0, 3.5, 3.5, 4.0, 3.0, 3.5	6, 6, 6, 6, 7, 6, 5, 7, 5, 5, 5, 7, 6, 7	14
Base: TF-TAS-B in Table.1			
576	4.0, 4.0, 4.0, 3.5, 3.0, 3.0, 3.0, 4.0, 3.0, 3.0, 3.5, 4.0, 3.5, 4.0, 3.0	9, 10, 10, 10, 10, 9, 10, 9, 9, 10, 9, 9, 9, 10, 9	15

Table C. The details of the PiT search space. The patch_size and stride are follow the settings in PiT [2]. The depth includes 3 parts, which are correspond to 3 stages.

Patch size	Stride	Base Dim	Depth Num	Head Num	MLP Ratio
Tiny					
16	8	{16,24,32,40}	{(1, 2, 3), (4, 6, 8), (2, 4, 6)}	{2,4,8}	{2,4,6,8}
Extra Small					
16	8	{40,48,56,64}	{(1, 2, 3), (4, 6, 8), (2, 4, 6)}	{2,4,8}	{2,4,6,8}
Small					
16	8	{36,48,60,72}	{(1, 2, 3), (4, 6, 8), (2, 4, 6)}	{3,6,12}	{2,4,6,8}
# (Possible Architectures) $= 3.4 \times 10^6$					

8, respectively. The important dimensions we are going to search include base dimension, depth number, the number of heads, and MLP ratio. In total, there are about 3.4×10^6 architectures in the PiT search space.

D. The searched optimal ViT Architectures

We provide samples of architectures searched on AutoFormer search space and PiT search spaces. The optimal networks are shown in Tab. B and Tab. D. The architectures in AutoFormer search space are layer-based, of which the MLP ratio and head number in Tab. B have the value of the number of depth number and other searched dimensions are consistent for each layer. Different from AutoFormer search space, the architectures in PiT search space are stage-based, of which the depth number and head number in Tab. D have 3 values representing the searched results in the three stages. In addition, the patch size, stride, base dimension, and MLP ratio are consistent for each stage.

Table D. The optimal architectures searched by our DSS-indicator on PiT search space.

Patch size	Stride	Base Dim	Depth Num	Head Num	MLP Ratio
Tiny: TF-TAS-Ti in Table.2					
16	8	40	2, 4, 4	2, 4, 8	2
Extra Small: TF-TAS-XS in Table.2					
16	8	56	1, 4, 2	2, 8, 8	2
Small: TF-TAS-S in Table.2					
16	8	36	1, 4, 2	3, 12, 12	4

References

[1] Wuyang Chen, Xinyu Gong, and Zhangyang Wang. Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective. In *ICLR*, 2021. 1

[2] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *ICCV*, 2021. 1, 2

[3] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. In *ICLR*, 2019. 1

[4] Joe Mellor, Jack Turner, Amos Storkey, and Elliot J Crowley. Neural architecture search without training. In *ICML*, 2021. 1

[5] Chaoqi Wang, Guodong Zhang, and Roger Grosse. Picking winning tickets before training by preserving gradient flow. In *ICLR*, 2020. 1