MLP Can Be A Good Transformer Learner

Supplementary Material

8. Code Asset

Acknowledgement. In Sec. 4.1, we introduce the used benchmarks. The code of this work is built upon previous works (Tab. 6). The authors thank their open sourcing.

9. Performance May Not Show The Full Picture.

In Sec. 3.3, we adopt the idea of transfer entropy and propose the NOSE to measure the interaction between an ordered array of attention layers and the final output layer. The associated combination of attention layers with minimum transfer entropy is selected for removal.

One can mask certain attention layers (*i.e.* set to identical mapping) and measure the performance, namely *remained performance*. This metric is plausible to reflect the interaction between the corresponding attention layers and the final output layer, where higher remained performance indicates less interaction. We argue that the remained performance does not show the full picture of the network. We sample some combinations of attention layers and visualize their transfer entropy together with the remained performance in Fig. 7. We find that two metrics are in part correlated. Specifically, most of the points are scattered on the right side. Typically, a combination with lower transfer entropy has a higher remained performance. In contrast, given several combinations with the same remained performance, their transfer entropy varies largely. Since transfer entropy is more consistent, we use it to determine the correlation among multiple layers. We also perform a case study to show the superiority of transfer entropy against the remained performance in Tab. 7. Although layer index [0,1,3,4,6] has a lower remained performance compared to layer index [1,2,3,4,6], the resulting performance is more favorable.

10. More Experiments

More details. For ImageNet-1k, we use 8 GPUs with a batch size of 128 per GPU. The learning rate is set to 1e-3 and a cosine scheduler is used to regulate the learning rate till it reaches at 1e-5. We use the AdamW optimizer where beta=(0.9,0.999). For CIFAR-100, we adpot a batch size of 384 for each GPU. The image resolution is resized to 224×224. And we use the SGD optimizer with a learning rate 0.1. For ADE20k, we also use 8 GPU and each GPU processes 2 input images. The optimizer is SGD and learning rate is 0.01. The polynomial scheduler with power 1.0 is used to decay the learning rate at each iteration.

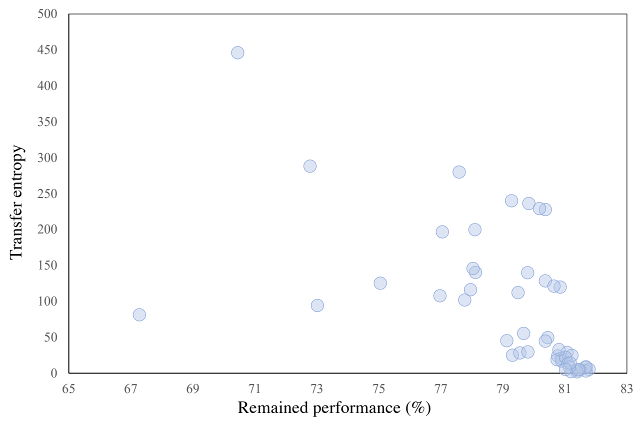


Figure 7. Correlation between remained performance and transfer entropy. Each point is a combination of attention layers with two metrics: transfer entropy and remained performance.

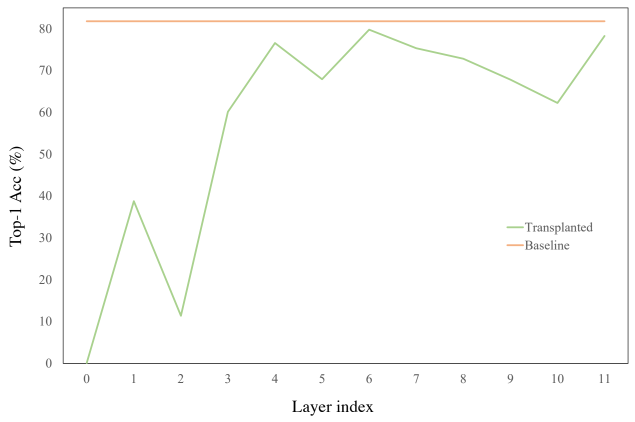


Figure 8. We transplant the transformer blocks of the original DeiT-B into the corresponding blocks of our model and measure the performance to investigate feature compatibility. The top blocks of our model are more compatible with the original model.

Feature space compatibility. We are interested in the feature space learned by our method. Inspired by network transplant [40], we propose to *transplant* the original transformer blocks, indexed from 0 to 11, of a pre-trained Deit-B into the corresponding blocks of our model. The classification accuracy is used to measure the compatibility. As shown in Fig. 8, we find that our model, starting from block 3, is more compatible with the feature space learned by the full architecture in the top blocks. We conjecture that in bottom blocks indexed by [0,1,2], transformer would learn low-level semantics that are not very generalized. In particular, even when the attention layers are removed, blocks 4 and 6 exhibit high compatibility, indicating our model learns the feature space close to the original architecture.

Table 6. Used code asset in our work.

Exp.	URL	Version	License
ImageNet-1k	https://github.com/facebookresearch/deit	263a3f	Apache-2.0
	https://github.com/huggingface/pytorch-image-models	0.3.2	Apache-2.0
CIFAR-100	https://github.com/facebookresearch/ToMe	af95e4	Creative Commons
ADE20k	https://github.com/OliverRensu/TinyMIM	d08470	NA
FLOPs	https://github.com/facebookresearch/fvcore	9d683a	Apache-2.0

Table 7. Case study regarding transfer entropy and remained performance.

Removed index	Transfer entropy $\downarrow$	Remained performance (%)$\uparrow$	Top-1 (%)$\uparrow$	Top-5 (%)$\uparrow$
$[1, 2, 3, 4, 6]$	446.0	**70.45**	81.2	95.4
$[0, 1, 3, 4, 6]$	**81.2**	67.28	**81.8**	**95.6**

Table 8. More experiments on DeiT-S and DeiT-T. The number in brackets indicates the ratio of attention layers removed.

Method	Top-1 (%)$\uparrow$	FLOPs (G)$\downarrow$	Params (M)$\downarrow$	Throughput (images/s)$\uparrow$	Memory bound (images/10GB)$\uparrow$
DeiT-S [33] (baseline)	79.9	4.6	22.1	1318	1168
Evo-ViT [37]	79.4	3.0	22.1	1914	1168
EViT [17]	79.5	3.0	22.1	1921	1168
ToMe [5]	79.5	2.9	22.1	1905	1168
DiffRate [7]	79.6	2.9	22.1	1805	1168
TPS [35]	79.7	3.0	22.1	1896	1168
Ours (25%)	**80.1**	4.2	**20.3**	1502	**1382**
Ours (30%)	**79.8**	4.0	**19.7**	1588	**1388**
Ours (40%)	79.6	3.9	**19.1**	1648	**1392**
Ours (25%)+ToMe	**79.9**	**2.7**	**20.3**	**2128**	**1352**
Ours (30%)+ToMe	79.6	3.0	**19.7**	**1932**	**1354**
DeiT-T [33] (baseline)	72.2	1.3	5.7	3487	2320
EViT [17]	71.9	**0.8**	5.7	5178	2320
Evo-ViT [37]	72.0	**0.8**	5.9	5258	2320
ToMe [5]	71.2	0.9	5.7	4508	2320
ToMe [5]	70.9	**0.8**	5.7	4949	2320
TPS [35]	**72.3**	**0.8**	5.7	5012	2320
Ours (25%)	**72.5**	1.1	**5.3**	4001	**2610**
Ours (30%)	71.9	1.1	**5.1**	4196	**2610**
Ours (25%)+ToMe	**72.3**	**0.8**	**5.3**	**5313**	**2600**
Ours (30%)+ToMe	71.7	0.9	**5.1**	4846	**2604**

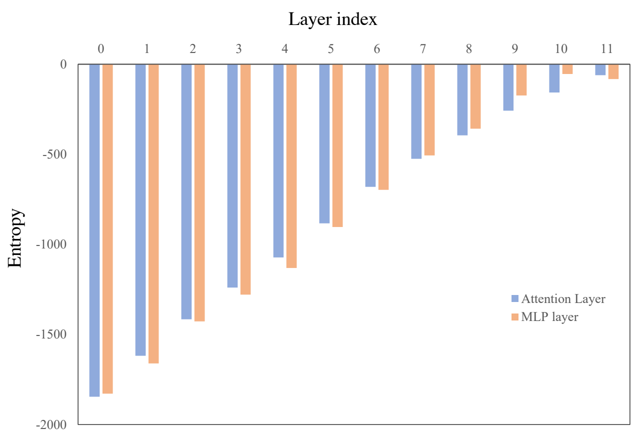
More backbones. We assess our model on two additional backbones: DeiT-S and DeiT-T. We visualize their entropy distribution in Fig. 9. We observe that the two entropy distributions have a similar pattern to that of DeiT-B. The number in the brackets indicates the ratio of attention layers removed. As shown in Tab. 8, for DeiT-S, our method generally improves the memory bound by ∼18.5% and the throughput[1] by 19.5% . When cooperated with an unsupervised token merging method, our method, while removing 25% attention layers, can further improve the throughput by 54% and outperforms other methods without performance compromise. Note that when combined with token merging, the working load of our model slightly decreases. This is because the tensor manipulation introduced by token matching will consume a quantity of memory [20]. A similar experiment result is observed for DeiT-T.

[1]Measured on a RTX 3090 GPU with batch size 256.

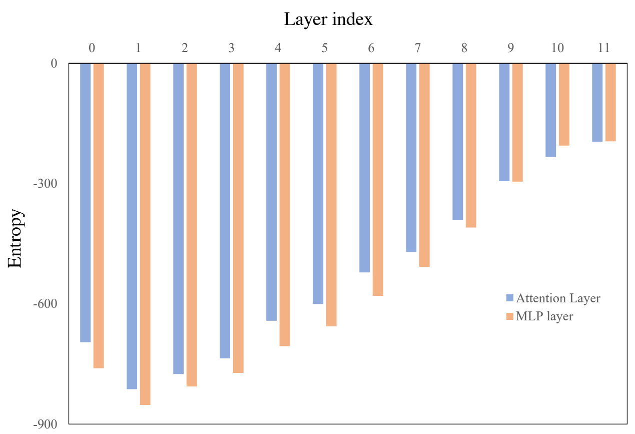
Table 9. Removing first N attention layers on DeiT-B.

Remove Num.		1	2	3	4	5	6	7
First-N	T.E.	140	167	211	333	498	636	645
	Top-1 (%)	81.8	81.8	81.7	81.4	80.8	79.8	77.6
NOSE	T.E.	3	14	20	78	380	433	532
	Top-1 (%)	81.8	81.8	81.8	81.8	81.8	81.5	81.0

Removing first N attention layers. In the main text, we compare NOSE to the random selection strategy. Here, we implement First-N as another baseline, where the first N consecutive attention layers are removed. As shown in Tab. 9, First-N deteriorates quickly with the increase of N, while NOSE maintains good performance yet with less transfer entropy (T.E.).



(a) DeiT-S



(b) DeiT-T

Figure 9. Entropy distribution of DeiT-S and Deit-T. We observe that the two distributions have a similar pattern to that of DeiT-B.

Table 10. Experiments of removal ratio.

Num.	1	2	3	4	5	6	7	8	9	10
Top-1(%)	81.8	81.8	81.8	81.8	81.8	81.5	81.0	79.4	76.3	72.8

Removal rates. We investigate the removal rates on DeiT-B as in Tab. 10. When it comes to 75% removal rate (*i.e*. 9 layer), the performance starts to drop drastically.