Dynamic Tuning Towards Parameter and Inference Efficiency for ViT Adaptation

Wangbo Zhao[1,2] Jiasheng Tang[2,3] Yizeng Han[2,4] Yibing Song[2,3] Kai Wang[1]
Gao Huang[4] Fan Wang[2] Yang You[1]
[1]National University of Singapore [2]DAMO Academy, Alibaba Group
[3]Hupan Laboratory [4]Tsinghua University

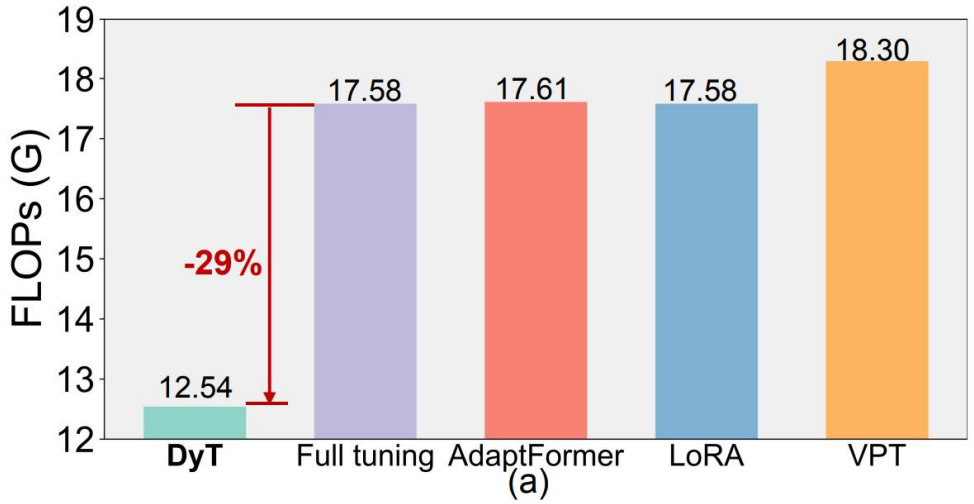
Code: https://github.com/NUS-HPC-AI-Lab/Dynamic-Tuning

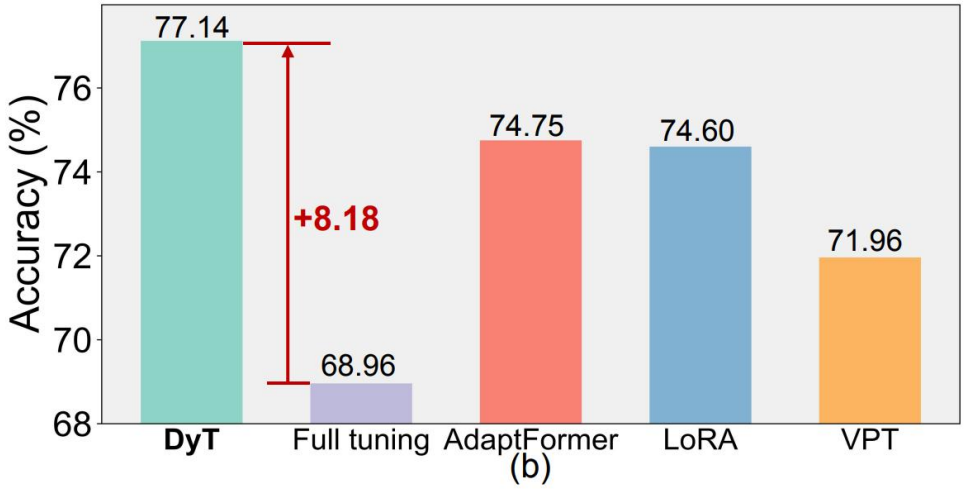
Motivation

- Representative PEFT methods do not reduces the computation during inference compared to full tuning.

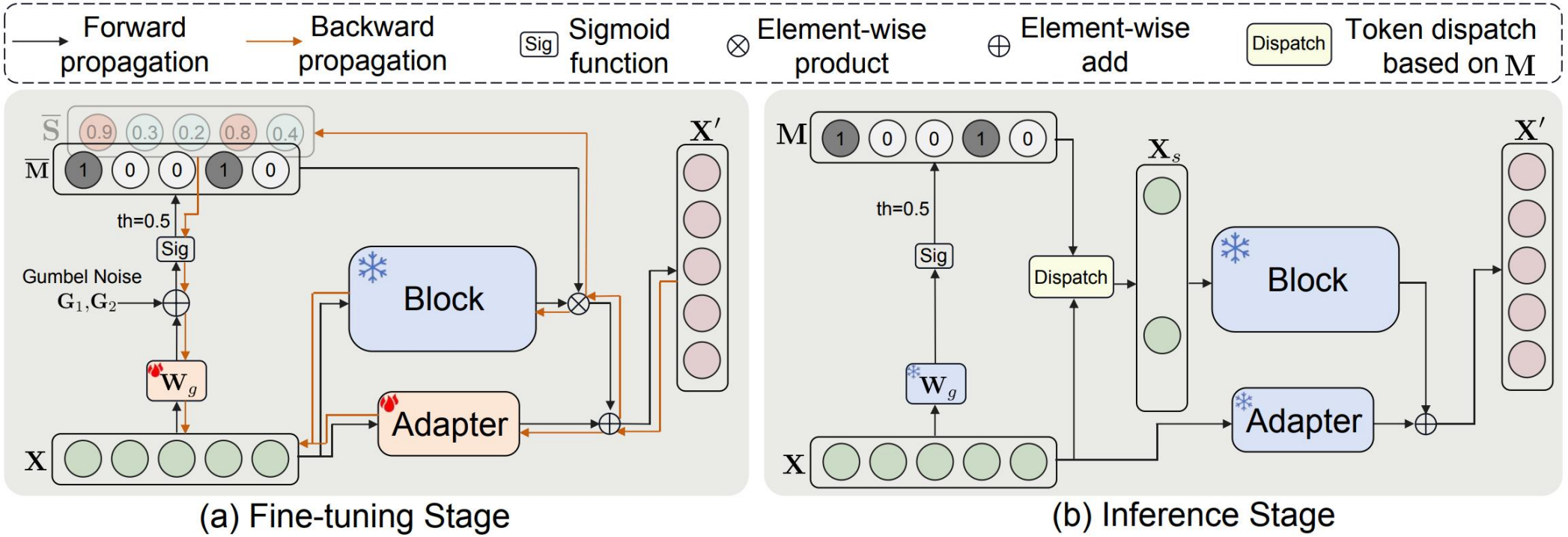
- Dynamic networks helps to reduce model inference costs. However, they primarily focus on pre-t**raining from scratch** or **full tuning on the same dataset,** without considering data domain transfer

- How to develop dynamic modules together with PEFT methods to enhance both parameter and inference efficiency during ViT adaptation





Methodology-Dynamic Tuning

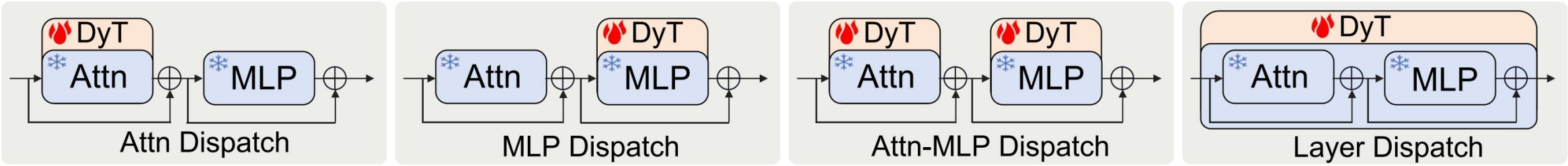


$$\mathbf{X}' = \mathbf{X} + \overline{\mathbf{X}}_s + \text{Adapter}(\mathbf{X})$$

$$\overline{\mathbf{X}}_s = \begin{cases} \text{Block}(\mathbf{X}) \cdot \overline{\mathbf{M}} & \text{Forward Propagation} \\ \text{Block}(\mathbf{X}) \cdot \overline{\mathbf{S}} & \text{Backward Propagation} \end{cases}$$

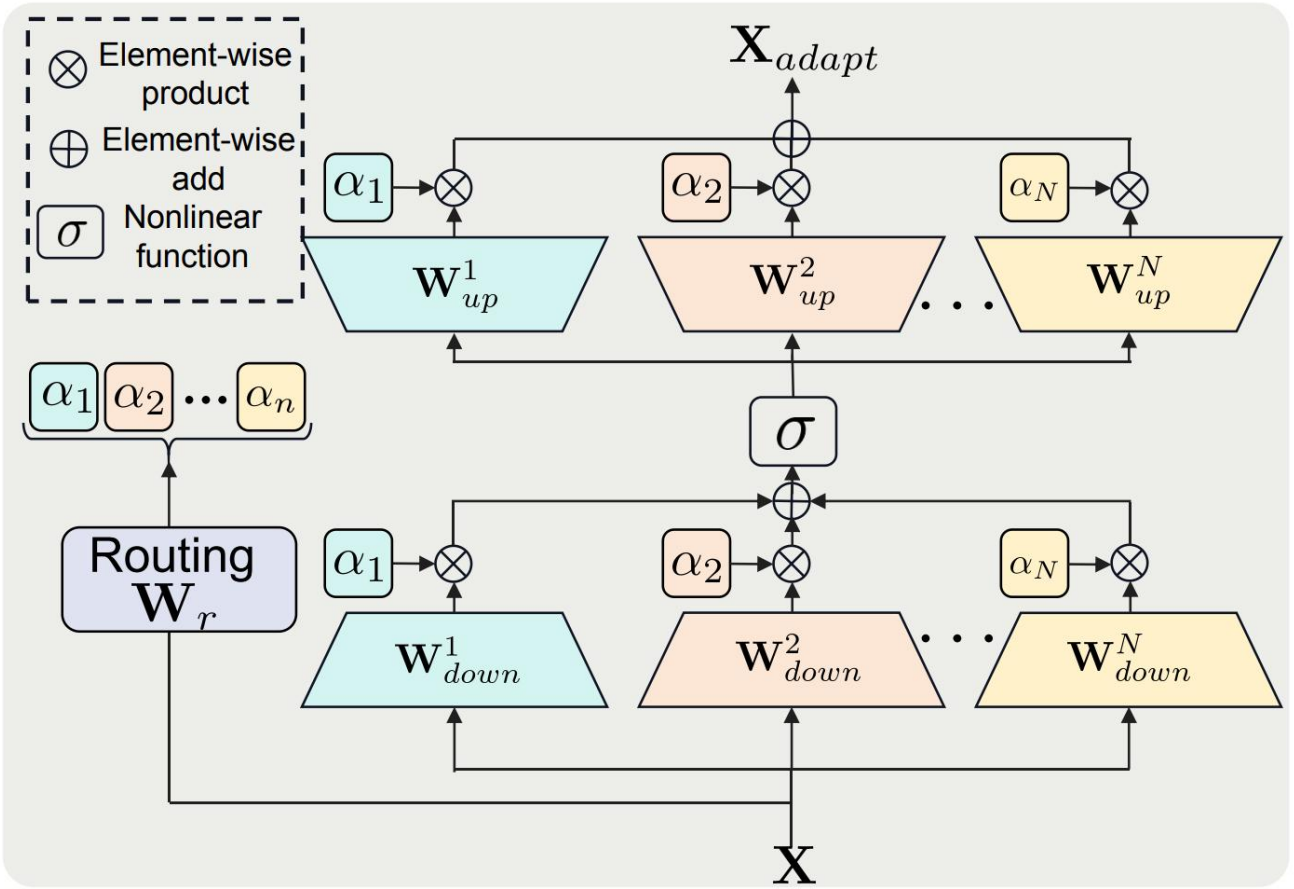
Methodology-Model Variants

Since the impact of skipping tokens in these blocks during the adaptation fine-tuning remains non-trivial to estimate and has not been explored in previous works, we propose four model variants



- Attn Dispatch: Considering the quadratic computational complexity of the Attn block with respect to the toker numbers, skipping tokens before applying Attn can significantly reduce computation.
- MLP Dispatch: MLP takes ~63.5% FLOPs in ViT-B/16 and propose to skip tokens only before MLP.
- Attention-MLP Dispatch: Skipping tokens before both self-attention and MLP blocks.
- Layer Dispatch: we can dispatch tokens before a transformer layer

Methodology-MoE-Adapter



- Adapter is responsible for processing all tokens, requiring it to have enough capability
- MoE-adapter enhances the capability of the adapter with introducing negligible computation cost.

Experiment

Table 3: **Performance and efficiency comparison on VTAB-1K**. "Group Mean" indicates the averaged accuracy of three groups. "Params. (M)" denotes the number of trainable parameters in **backbones**. "FLOPSs (G)" is the average FLOPs across all datasets. **Bold font** and underline denote the best and the second-best performance respectively.

	● Natural							● Specialized				● Structured										
	CIFAR-100	Caltech101	DTD	Flowers102	Pets	SVHN	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Elev	Group Mean	Params. (M)	FLOPs (G)
Traditional methods																						
Full tuning	68.9	87.7	64.3	97.2	86.9	87.4	38.8	79.7	95.7	84.2	73.9	56.3	58.6	41.7	65.5	57.5	46.7	25.7	29.1	68.96	85.80	17.58
Frozen	63.4	85.0	63.2	97.0	86.3	36.6	51.0	78.5	87.5	68.6	74.0	34.3	30.6	33.2	55.4	12.5	20.0	9.6	19.2	57.64	**0.00**	17.58
Parameter-efficient tuning methods																						
Adapter [32]	69.2	90.1	68.0	98.8	89.9	82.8	54.3	84.0	94.9	81.9	75.5	80.9	65.3	48.6	78.3	74.8	48.5	29.9	41.6	73.85	0.16	17.61
BitFit [86]	72.8	87.0	59.2	97.5	85.3	59.9	51.4	78.7	91.6	72.9	69.8	61.5	55.6	32.4	55.9	66.6	40.0	15.7	25.1	65.21	0.10	17.58
LoRA [33]	67.1	91.4	69.4	98.8	90.4	85.3	54.0	84.9	95.3	84.4	73.6	82.9	**69.2**	49.8	78.5	75.7	47.1	31.0	44.0	74.60	0.29	17.58
VPT [35]	**78.8**	90.8	65.8	98.0	88.3	78.1	49.6	81.8	96.1	83.4	68.4	68.5	60.0	46.5	72.8	73.6	47.9	32.9	37.8	71.96	0.53	18.30
SSF [37]	69.0	92.6	**75.1**	**99.4**	91.8	90.2	52.9	87.4	95.9	**87.4**	75.5	75.9	62.3	**53.3**	80.6	77.3	54.9	29.5	37.9	75.69	0.20	17.58
NOAH [90]	69.6	92.7	70.2	99.1	90.4	86.1	53.7	84.4	95.4	83.9	75.8	82.8	68.9	49.9	81.7	81.8	48.3	32.8	44.2	75.48	0.36	17.58*
ConvPass [37]	72.3	91.2	72.2	99.2	90.9	**91.3**	54.9	84.2	**96.1**	85.3	75.6	82.3	67.9	51.3	80.0	**85.9**	53.1	**36.4**	44.4	76.56	0.33	17.64
AdaptFormer [12]	70.8	91.2	70.5	99.1	90.9	86.6	54.8	83.0	95.8	84.4	76.3	81.9	64.3	49.3	80.3	76.3	45.7	31.7	41.1	74.75	0.16	17.61
FacT-TT [38]	71.3	89.6	70.7	98.9	91.0	87.8	54.6	85.2	95.5	83.4	75.7	82.0	69.0	49.8	80.0	79.2	48.4	34.2	41.4	75.30	0.04	17.58
Res-Tuning [36]	75.2	92.7	71.9	99.3	**91.9**	86.7	**58.5**	86.7	95.6	85.0	74.6	80.2	63.6	50.6	80.2	85.4	55.7	31.9	42.0	76.32	0.51	17.67
The proposed Dynamic Tuning																						
DyT $r = 0.5$	73.6	94.8	73.0	99.1	91.4	87.0	56.4	87.3	**96.1**	85.6	**76.7**	82.8	63.8	52.7	**83.7**	83.6	**57.3**	34.6	44.3	77.14	0.16	**12.54**
DyT $r = 0.7$	74.4	**95.5**	73.6	99.2	91.7	87.5	57.4	**88.3**	**96.1**	86.7	**76.7**	**83.5**	63.8	52.9	83.1	**85.7**	54.9	34.3	45.9	**77.57**	0.16	14.92
DyT $r = 0.9$	74.3	94.9	73.1	99.2	91.4	87.8	57.1	87.9	**96.1**	85.9	76.0	83.3	64.8	51.5	83.4	84.0	54.8	35.1	**46.4**	77.30	0.16	17.07

* NOAH cost larger than 17.58 FLOPS since it combines PEFT methods via neural architecture search.

Table 4: **Comparison of throughput.** "VTAB-1K Accuracy ↑" denotes the averaged accuracy of three dataset groups in VTAB-1K [88] benchmark.

Method	VTAB-1K Accuracy ↑	FLOPs (G) ↓	V100 Throughput (img/s) ↑	T4 Throughput (img/s)↑	Xeon(R) 8163 Throughput (img/s) ↑
Full tuning	68.96	17.58	806.24	435.41	2.12
LoRA [33]	74.60	17.58	806.24	435.41	2.12
AdaptFormer [12]	74.75	17.61	767.30	400.42	1.97
VPT [35]	71.96	18.30	762.55	392.13	1.95
DyT $r = 0.5$	**77.14**	**12.54**	**912.30**	**524.93**	**3.89**

Table 5: **Comparison with efficient transformers.** The throughput is measured on a Tesla V100 GPU. "Params. (M) ↓ " denotes the number of trainable parameters in **backbones**.

Method	VTAB-1K Accuracy ↑	FLOPs (G) ↓	Param. (M) ↓	Throughput (img/s) ↑
DynamicViT [66]	60.10	14.05	88.70	1010.40
DynamicViT+AdaptFormer[12]	75.48	14.23	3.10	954.82
EViT [47]	67.42	11.62	85.80	1233.45
EViT+AdaptFormer[12]	74.63	11.77	0.16	1152.38
Full tuning + ToMe [5]	68.68	13.12	85.80	989.29
AdaptFormer [12] + ToMe [5]	74.30	13.29	0.16	941.70
DyT $r = 0.5$	77.14	12.54	0.16	912.39
DyT $r = 0.5$ + ToMe [5]	76.60	9.85	0.16	1114.70

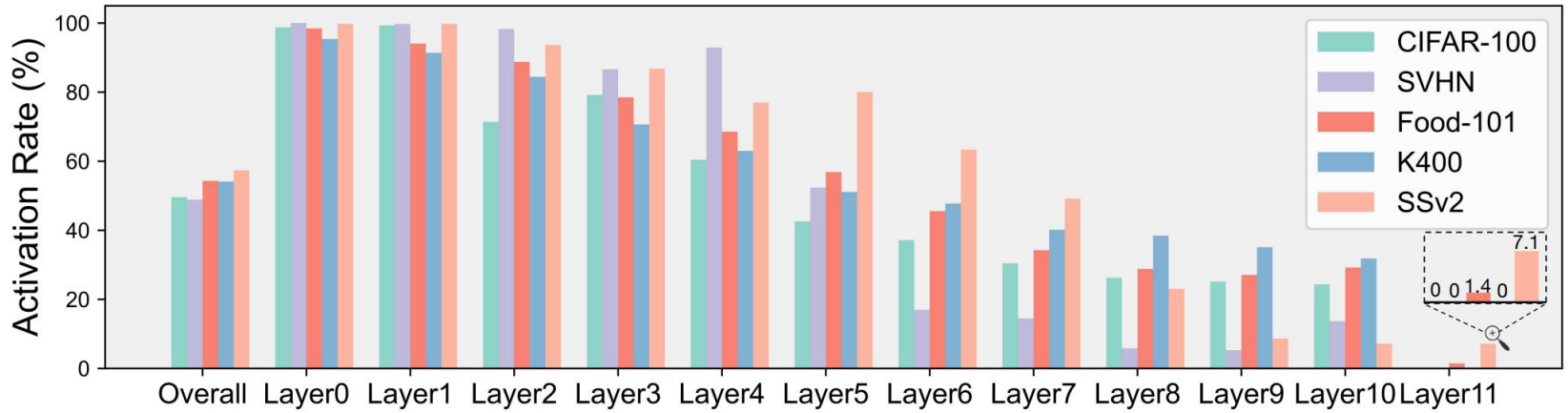


Figure 5: **Token activation rate in different layers.** We visualize the token activation rates in ViT-B/16. "Overall" denotes the mean activation rate in the whole model, which arrives at around 50% when r is set to 0.5. "Layer0" and "Layer11" denote the lowest and highest level, respectively. Notably, the activation rate in the last layer is exactly *0%* on CIFAR-100, SVHN, and K400 datasets.

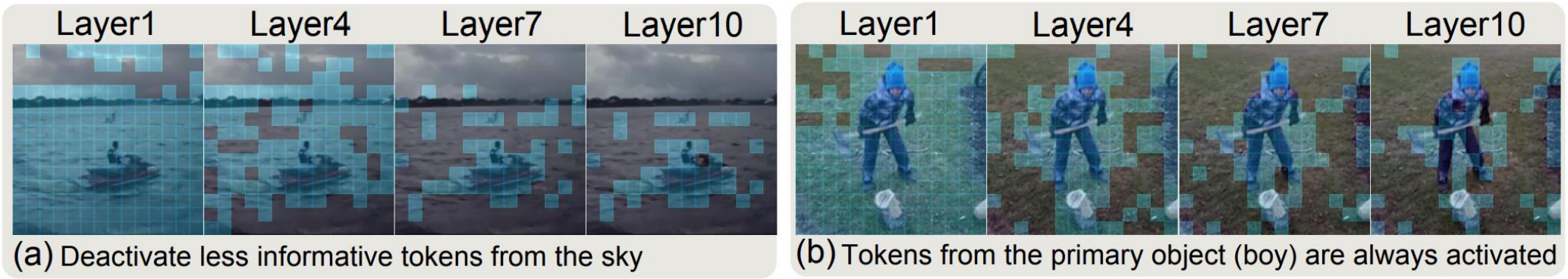


Figure 6: **Visualization of activated tokens.** We present two representative samples from the K400 dataset. **Blue patches** represent the tokens activated in token dispatcher (Detailed in Section 3.2). Results verify that the token dispatcher has learned to identify informative tokens during fine-tuning.

Thanks