You Only Need Less Attention at Each Stage in Vision Transformers

Shuoxi Zhang , Hanpeng Liu, Stephen Lin, Kun He
{zhangshuoxi, hanpengliu, brooklet60}@hust.edu.cn, stevelin@microsoft.com

CVPR
JUNE 17-21, 2024
SEATTLE, WA

Background

- The computational complexity of the self-attention mechanism grows quadratically with the number of tokens.

- The computational burden becomes heavier with higher-resolution images.

- Training Vision Transformers (ViTs) often leads to attention saturation phenomena.

Question & Answer

- Q: Is it really necessary to consistently apply the self-attention mechanism throughout each stage of the network, from inception to conclusion?

- A: Considering the attention saturation, we conclude that not all attention computation is necessary. Then we design the Less-Attention Module to alleviate the attention computation and attention saturation.

Architecture

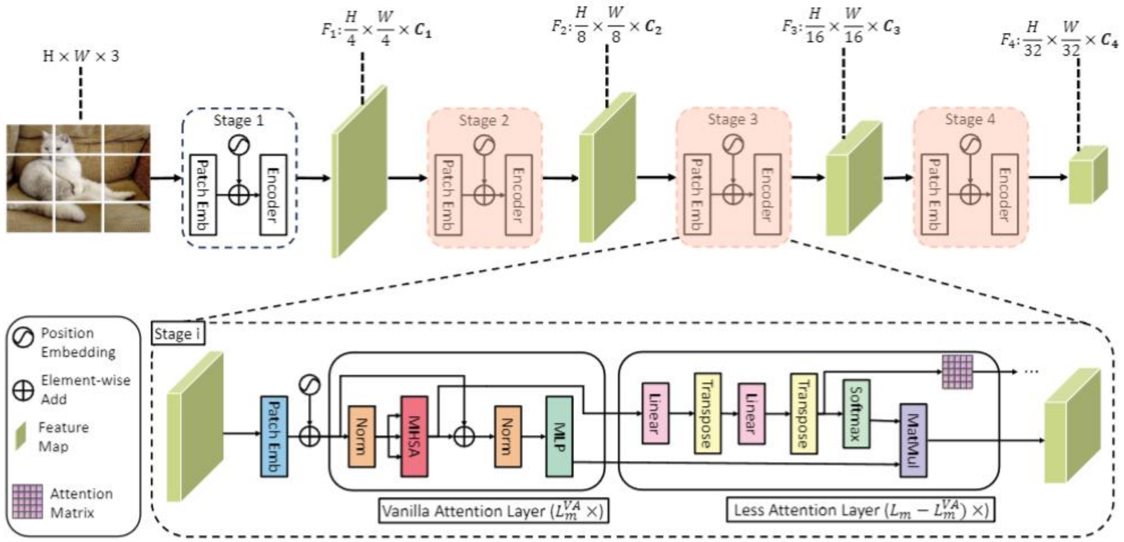


Figure 1. The architecture of our **L**ess-**A**ttention **V**ision **T**ransformer (LaViT). The bottom part: the proposed Less-Attention layer, which together with conventional Transformer blocks in the preceding layers constitutes the feature extraction module of this stage.

$$\mathbf{A}_m^{\mathrm{VA},l} = \frac{\mathbf{Q}_m^l(\mathbf{K}_m^l)^{\mathsf{T}}}{\sqrt{d}}, \quad l \le L_m^{\mathrm{VA}}.$$

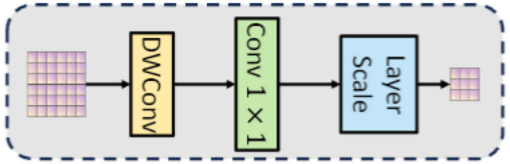
$$\mathbf{A}_m^l = \Psi(\Theta(\mathbf{A}_m^{l-1})^{\mathsf{T}})^{\mathsf{T}}, \quad L_m^{\mathrm{VA}} < l \le L_m,$$

$$\mathbf{Z}^{\mathrm{LA},l} = \mathrm{Softmax}(\mathbf{A}_m^l)\mathbf{V}^l.$$

In each stage, we extract the feature representation in two phases. At the initial several Vanilla Attention (VA) layers, we conduct the standard MHSA operation to capture the overall long-range dependencies. Subsequently, we simulate the attention matrices to mitigate quadratic computation and address attention saturation at the following Less-Attention (LA) layers by applying a linear transformation to the stored attention scores.

Extra designs

- Residual-based Attention Downsampling



- Diagonality Preserving Loss

$$\mathcal{L}_{\mathrm{DP},l} = \sum_{i=1}^{N}\sum_{j=1}^{N} |\mathbf{A}_{ij} - \mathbf{A}_{ji}| + \sum_{i=1}^{N}((N-1)\mathbf{A}_{ii} - \sum_{j \neq i} \mathbf{A}_{j}).$$

Experiments

Model	Params (M)	FLOPs (G)	Throughput (image/s)	Top1 (%)
ResNet-18	11.7	1.8	**4454**	69.8
RegNetY-1.6G	11.2	1.6	1845	78.0
DeiT-T	**5.7**	**1.3**	3398	72.2
PVT-T	13.2	1.9	1768	75.1
PVTv2-b1	13.1	2.1	1231	78.7
LaViT-T	10.9	1.6	2098	**79.2**
ResNet-50	25.0	4.1	1279	76.2
RegNetY-4G	20.6	4.0	1045	79.4
EfficientNet-B4	**19.0**	4.2	387	82.4
EfficientViT-B2	24.0	4.5	1587	82.1
DeiT-S	22.1	4.6	1551	79.9
DeepViT-S	27.0	6.2	1423	82.3
PVT-S	24.5	3.8	1007	79.8
CvT-S	25.8	7.1	636	82.0
Swin-T	28.3	4.5	961	81.2
PVTv2-b2	25.4	4.0	695	82.0
DynamicViT-S (90%)	24.1	4.0	1524	79.8
EViT-S (90%)	23.9	4.1	**1706**	79.7
LiT-S	27.0	4.1	1298	81.5
PPT-S	22.1	3.1	1698	79.8
LaViT-S	22.4	**3.3**	1546	**82.6**
ResNet-101	45.0	7.9	722	77.4
ViT-B	86.6	17.6	270	77.9
DeiT-B	86.6	17.5	582	81.8
Swin-S	49.6	8.7	582	83.1
Swin-B	87.8	15.4	386	**83.4**
DynamicViT-B (90%)	76.6	14.1	732	81.5
EViT-B (90%)	78.6	15.3	852	81.3
LiT-M	48.0	8.6	638	83.0
PPT-B	86.0	14.5	714	81.4
PVT-M	44.2	6.7	680	81.2
PVT-L	61.4	9.8	481	81.7
LaViT-B	**39.6**	**6.1**	**877**	83.1

Table 2. Comparison of different backbones on ImageNet-1K classification. Except for EfficientNet (EfficientNet-B4), all models are trained and evaluated with an input size of 224 × 224. The least computations and fastest throughput appear in **blue bold**, and the best results appear in **bold**.[1]

Backbone	#Param. (M)	FLOPs (G)	RetinaNet 1× AP^b	AP^b_{50}	AP^b_{75}	AP^b_S	AP^b_M	AP^b_L	RetinaNet 3× + MS AP^b	AP^b_{50}	AP^b_{75}	AP^b_S	AP^b_M	AP^b_L
ResNet50	38	239	36.3	55.3	38.6	19.3	40.0	48.8	39.0	58.4	41.8	22.4	42.8	51.6
PVT-Small	34	226	40.4	61.3	43.0	25.0	42.9	55.7	42.2	62.7	45.0	26.2	45.2	57.2
Swin-T	39	245	41.5	62.1	44.2	25.1	44.9	55.5	43.9	64.8	47.1	28.4	47.2	57.8
LaViT-T(ours)	33	202	**46.2**	**67.2**	**49.1**	**29.6**	**50.2**	**61.3**	**48.4**	**69.9**	**51.7**	**31.8**	**52.2**	**64.1**
ResNet101	58	315	38.5	57.8	41.2	21.4	42.6	51.1	40.9	60.1	44.0	23.7	45.0	53.8
PVT-M	54	283	41.9	63.1	44.3	25.0	44.9	57.6	43.2	63.8	46.1	27.3	46.3	58.9
Swin-S	60	335	44.5	65.7	47.5	27.4	48.0	59.9	46.3	67.4	49.8	31.1	50.3	60.9
LaViT-S(ours)	47	290	**46.7**	**68.3**	**49.7**	**29.9**	**50.7**	**61.7**	**48.9**	**70.3**	**52.2**	**33.1**	**52.6**	**65.4**

Table 3. Results on COCO object detection using the RetinaNet [12] framework. 1× refers to 12 epochs, and 3× refers to 36 epochs. MS means multi-scale training. AP^b and AP^m denotes box mAP and mask mAP, respectively. FLOPs are measured at resolution 800 × 1280.

Backbone	Semantic FPN 80k Param (M)	FLOPs (G)	mIOU (%)	UperNet 160K Param (M)	FLOPs (G)	mIOU (%)	MS mIOU (%)
ResNet-50	28.5	183	36.7	-	-	-	-
Swin-T	31.9	182	41.5	59.9	945	44.5	45.8
PVT-S	30.2	146	43.2	-	-	-	-
Twin-S	28.3	144	43.2	54.4	932	46.2	47.1
LiT-S	32.0	172	41.3	57.8	978	44.6	45.9
Focal-T	-	-	-	62.0	998	45.8	47.0
LaViT-S	**25.1**	**122**	**44.1**	**52.0**	**920**	**47.2**	**49.5**

Table 4. Segmentation performance of different backbones in Semantic FPN and UpperNet framework on ADE20K. The least computation appears in **blue bold**, and the best results appear in **bold**.

Experiments

- Extensibility

Backbone	Tiny		Small	
	Top-1 Acc(%)	FLOPs (G)	Top-1 Acc(%)	FLOPs (G)
ViT	72.2	1.4	79.1	4.6
$\text{ViT}_{+\text{LA}}$	73.2($\uparrow$ 1.0)	1.2($\downarrow$ 14.2%)	80.0($\uparrow$ 0.9)	4.0($\downarrow$ 13.1%)
DeiT	72.2	1.4	79.9	4.7
$\text{DeiT}_{+\text{LA}}$	73.4($\uparrow$ 1.2)	1.2($\downarrow$ 14.2%)	80.4($\uparrow$ 0.5)	4.2($\downarrow$ 10.6%)
DeepViT	73.4	1.5	80.9	4.8
$\text{DeepViT}_{+\text{LA}}$	73.8($\uparrow$ 0.4)	1.1($\downarrow$ 25.8%)	81.4($\uparrow$ 0.5)	4.2($\downarrow$ 12.6%)
CeiT	76.2	1.2	82.0	4.5
$\text{CeiT}_{+\text{LA}}$	76.7($\uparrow$ 0.5)	1.1($\downarrow$ 9.0%)	82.4($\uparrow$ 0.4)	4.1($\downarrow$ 8.8%)
HVT	75.7	1.4	80.4	4.6
$\text{HVT}_{+\text{LA}}$	76.2($\uparrow$ 0.5)	1.2($\downarrow$ 15.2%)	80.8($\uparrow$ 0.4)	4.2($\downarrow$ 13.4%)
PVT	75.1	1.9	79.8	3.8
$\text{PVT}_{+\text{LA}}$	75.9($\uparrow$ 0.8)	1.4($\downarrow$ 25.6%)	80.4($\uparrow$ 0.6)	3.2($\downarrow$ 15.7%)
Swin	81.2	4.5	83.2	8.7
$\text{Swin}_{+\text{LA}}$	81.7($\uparrow$ 0.5)	4.0($\downarrow$ 11.1%)	83.5($\uparrow$ 0.3)	7.8($\downarrow$ 10.3%)

The incorporation of the Less-Attention layer into any of the foundational Transformer architectures leads to enhancements in accuracy while concurrently reducing computational demands.

Experiments

- Indispensibility

Model	Module			Tiny	Small
	AR	LA	$\mathcal{L}_{DP}$		
w/o LA	-	-	-	78.7	82.0
w/o AR	-	✓	✓	79.0	82.2
LaViT	✓	✓	✓	79.2	82.6
w/o $\mathcal{L}_{DP}$	✓	✓	-	59.1(↓ 20.1)	57.1(↓ 25.5)

All these experimental findings collectively emphasize the contribution of each component within our model architecture

Conclusion

- Aiming to reduce the costly self-attention computations, we designed a novel plug-in Less-Attention module on ViTs. It leverages the computed dependency in MHSA blocks and bypasses the attention computation by reusing attentions from previous MHSA blocks. Our architecture effectively captures cross-token associations, surpassing the performance of the baseline while maintaining a computationally efficient profile in terms of parameters and floating-point operations per second (FLOPs).

Thank you,
CVPR 2024