Fast Vision Transformers with HiLo Attention

Key Ideas

This paper introduces an efficient attention block called Hi-Lo attention which is more efficient compared to Multi-Head Self-attention as well as some of the theoretically efficient attentions like dilated window attention, multi-scale window partition attention etc.,

This attention is based on the idea that in natural images, the high frequencies capture the local fine details (edges, textures etc) while low frequencies capture the global context well. Based on this intuition, they devise a new attention module callled Hi-Lo.

▼ **Multi Head self attention is computationally expensive**

Each self attention head calculates the Query, Key and Value by linear transformation on X

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q, \mathbf{K} = \mathbf{X}\mathbf{W}_k, \mathbf{V} = \mathbf{X}\mathbf{W}_v, \tag{1}$$

Next, the output of the self attention head is :

$$\mathrm{SA}_h(\mathbf{X}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_h}})\mathbf{V}. \tag{2}$$

And for a MHSA block with N heads, the final output is calculated by linear projection of the concatenated outputs from each self-attention heads and is given by:

$$\mathrm{MSA}(\mathbf{X}) = \underset{h\in[N_h]}{\mathrm{concat}}[\mathrm{SA}_h(\mathbf{X})]\mathbf{W}_o, \tag{3}$$

The time complexity here is

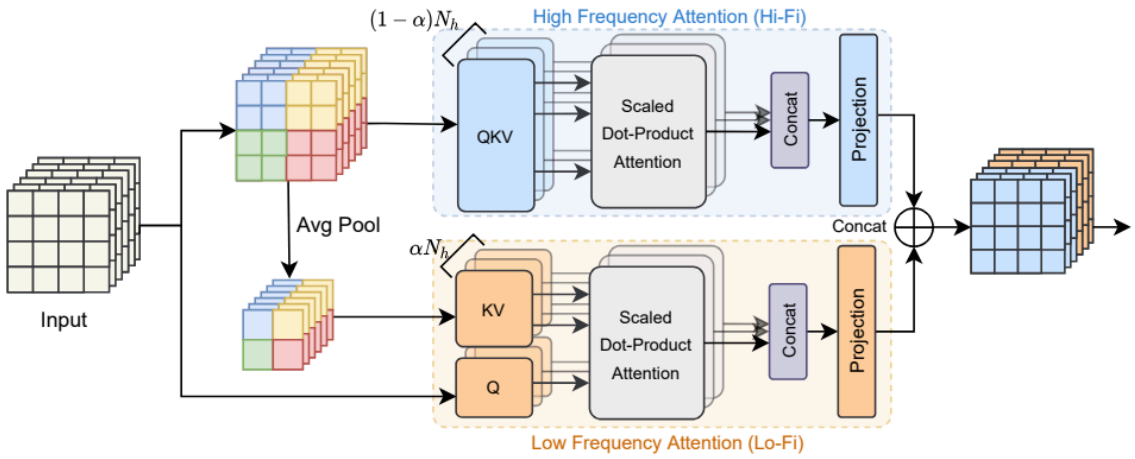
Equation (1) has : $3\ (ND^2)$ where D is the hidden dim and N is the sequence length

Equation (2) has : $2\ (N^2D)$ where D is the hidden dim and N is the sequence length

Equation (3) has : (ND^2) where D is the hidden dim and N is the sequence length

Hence the total complexity can be written as $4\ (ND^2)$ + $2\ (N^2D)$

Introducing HiLo attention:



The mechanism processes the feature maps in two paths. High frequency path for local features and a low frequency path for global features.

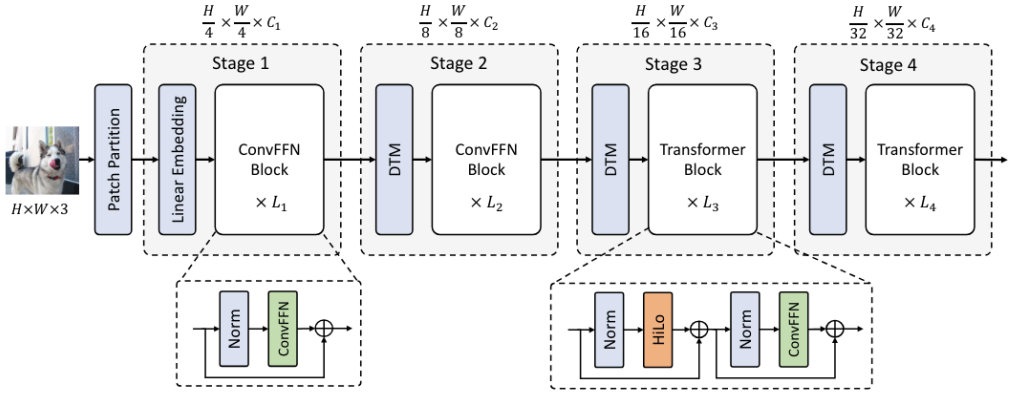
High Frequency Attention: In this path, a local window self attention is applied on the high resolution feature maps. The idea is, high frequencies encode the local details of the object (edges etc). For the window attention, a window of size typically 2 is chosen. The number of heads for the window attention is determined by (1- $\alpha)N_h$ where h is the total number of heads assigned for the block.

Low Frequency Attention: In this path, the feature map is downsampled by average pooling the windows. The keys and values come from the downsampled feature representation. The query still comes from the original resolution space. The number of heads for the self attention is determined by $(\alpha)N_h$ where h is the total number of heads assigned for the block.

Head Splitting: Use an α parameter to control the number of heads for Hi Frequency attention and Low Frequency attention. They use $(1-\alpha)N_h$ for High frequency attention and αN_h for low frequency attention

Post applying the attention, the outputs from both the branches are concatenated

Overall architecture :



In the architecture, in the early network they use convolution blocks and then use the transformer block at later end of the network. For patch merging, they use Deformable token merging, so that the merge tokens that contributes the most by learning the grid offsets. Lastly, they remove positional encodings and add depth-wise convolutions with zero padding in each FFN. The intuition is that the zero padding preserves spatial knowledge in convnets from the paper “How Much Position Information Do Convolutional Neural Networks Encode?” .

Impressions

They show the effectiveness of their attention in solving different problems like classification, detection and semantic segmentation:

▼ Image classification on Imagenet 1K

For throughput calculation ：Default resolution：224×224, batch size：64, Nvidia RTX 3090 averaged over 30 runs

In the table below, LITv2 version **S, M, B** all have better throughput than the other counterparts while still maintaining performance.

Model	Param (M)	FLOPs (G)	Throughput (imgs/s)	Train Mem (GB)	Test Mem (GB)	Top-1 (%)
ResNet-50 [53]	26	4.1	1,279	7.9	2.8	80.4
ConvNext-Ti [33]	28	4.5	1,079	8.3	1.7	82.1
PVT-S [51]	25	3.8	1,007	6.8	1.3	79.8
Swin-Ti [32]	28	4.5	961	6.1	1.5	81.3
CvT-13 [54]	20	4.5	947	6.1	1.5	81.6
Focal-Tiny [60]	29	4.9	384	12.2	3.3	**82.2**
Twins-PCPVT-S [12]	24	3.8	998	6.8	1.2	81.2
LITv1-S [36]	27	4.1	1,298	5.8	1.2	81.5
LITv2-S	28	3.7	**1,471**	**5.1**	**1.2**	82.0
ResNet-101 [53]	45	7.9	722	10.5	3.0	81.5
ConvNext-S [33]	50	8.7	639	12.3	1.8	83.1
PVT-M [51]	44	6.7	680	9.3	1.5	81.2
Twins-SVT-B [12]	56	8.3	621	9.8	1.9	83.2
Swin-S [32]	50	8.7	582	9.7	1.7	83.0
LITv1-M [36]	48	8.6	638	12.0	1.4	83.0
LITv2-M	49	7.5	**812**	**8.8**	**1.4**	**83.3**
ResNet-152 [53]	60	11.6	512	13.4	2.9	82.0
ConvNext-B [33]	89	15.4	469	16.9	2.9	**83.8**
Twins-SVT-L [12]	99	14.8	440	13.7	3.1	83.7
Swin-B [32]	88	15.4	386	13.4	2.4	83.3
LITv1-B [36]	86	15.0	444	16.4	2.1	83.4
LITv2-B	87	13.2	**602**	**12.2**	**2.1**	83.6
DeiT-B↑ 384 [45]	86	55.4	159	39.9	**2.5**	83.1
Swin-B↑ 384 [32]	88	47.1	142	OOM	6.1	84.5
LITv2-B↑ 384	87	39.7	**198**	**35.8**	4.6	**84.7**

▼ Object detection and instance segmentation

In the table below, LITv2 version **S, M, B** all have better throughput than the other counterparts while still maintaining performance.

Backbone	RetinaNet				Mask R-CNN				
	Params	FLOPs (G)	FPS	AP^b	Params	FLOPs (G)	FPS	AP^b	AP^m
ResNet-50 [23]	38M	239	18.5	36.3	44M	260	27.1	38.0	34.4
PVT-S [51]	34M	273	13.0	40.4	44M	292	16.2	40.4	37.8
Swin-T [32]	38M	251	17.0	41.5	48M	270	21.1	42.2	39.1
Twins-SVT-S [12]	34M	225	15.5	43.0	44M	244	20.4	43.4	40.3
LITv1-S [36]	39M	305	3.3	41.6	48M	324	3.2	42.9	39.6
LITv2-S	38M	242	18.7	**44.0**	47M	261	18.7	**44.9**	**40.8**
LITv2-S*	38M	230	20.4	**43.7**	47M	249	21.9	**44.7**	**40.7**
ResNet-101 [23]	57M	315	15.2	38.5	63M	336	20.9	40.4	36.4
PVT-M [51]	54M	348	10.5	41.9	64M	367	10.8	42.0	39.0
Swin-S [32]	60M	343	13.3	44.5	69M	362	15.8	44.8	40.9
Twins-SVT-B [12]	67M	358	10.8	45.3	76M	377	12.7	45.2	41.5
LITv2-M	59M	348	12.2	**46.0**	68M	367	12.6	**46.8**	**42.3**
LITv2-M*	59M	312	14.8	**45.8**	68M	315	16.0	**46.5**	**42.0**
ResNeXt101-64x4d [58]	96M	473	10.3	41.0	102M	493	12.4	42.8	38.4
PVT-L [51]	71M	439	9.5	42.6	81M	457	8.3	42.9	39.5
Swin-B [32]	98M	488	11.0	44.7	107M	507	11.3	45.5	41.3
Twins-SVT-L [12]	111M	504	9.9	45.7	120M	524	10.1	45.9	41.6
LITv2-B	97M	481	9.5	**46.7**	106M	500	9.3	**47.3**	**42.6**
LITv2-B*	97M	430	11.8	**46.3**	106M	449	11.5	**46.8**	**42.3**

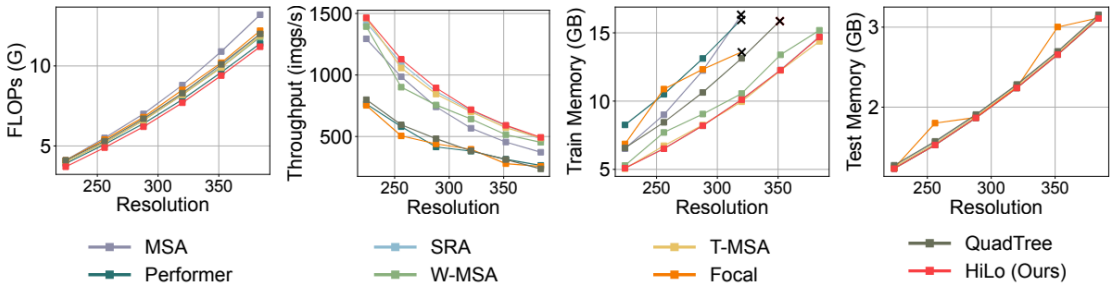
Ablation studies:

▼ Comparing Hi-Lo with other attentions

These sets of experiments were conducted by replacing the Hi-Lo attention with different attention mechanisms:

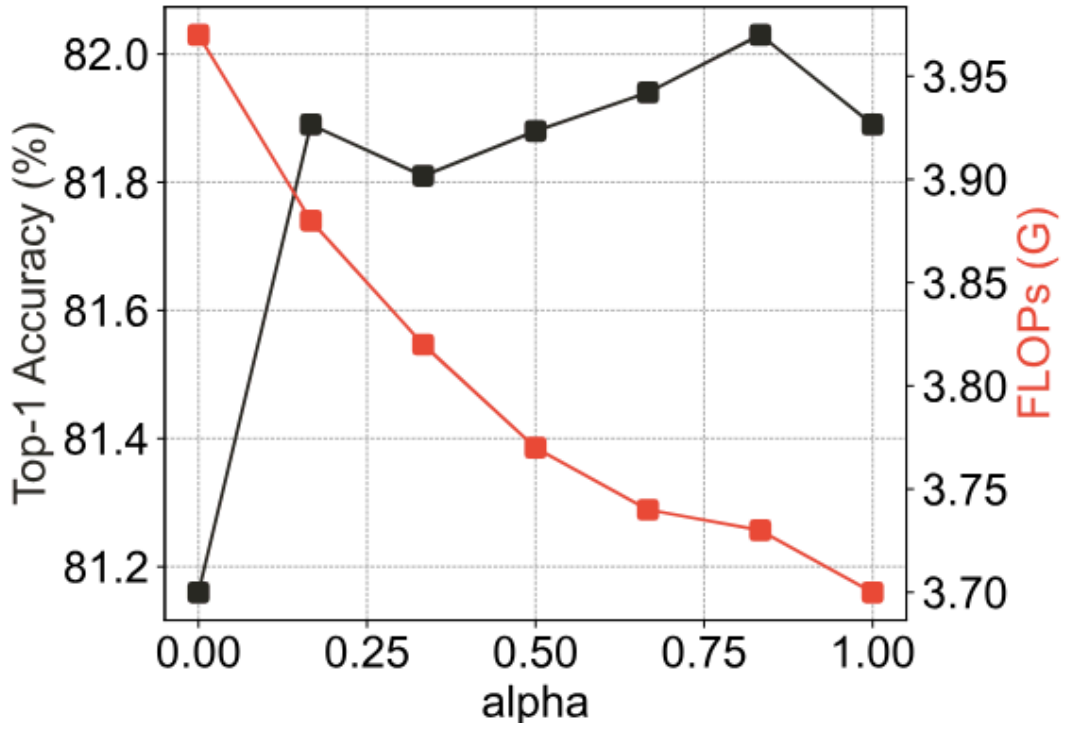
Method	ImageNet-1K						ADE20K		
	Params (M)	FLOPs (G)	Throughput (images/s)	Train Mem (GB)	Test Mem (GB)	Top-1 (%)	Params (M)	FLOPs (G)	mIoU (%)
MSA	28	4.1	1,293	6.5	1.2	82.3	32	46.5	43.7
SRA [51]	32	4.0	1,425	5.1	1.3	81.7	35	**42.4**	42.8
W-MSA [32]	28	4.0	1,394	5.3	1.2	81.9	32	42.7	41.9
T-MSA [12]	30	4.0	1,462	**5.0**	1.3	81.8	33	42.5	44.0
HiLo	**28**	**3.7**	**1,471**	5.1	**1.2**	**82.0**	**31**	42.6	**44.3**

From the table above, we can see that HiLo reduces more flops while having better performance and higher throughput.



▼ Effect of α

Here when the alpha value is set to 1, which means only Lo-Fi attention is active, it is basically applying MHSA on downsample keys and values and hence it gets good performance on imagenet1k.



Method	ImageNet-1K						ADE20K		
	Params (M)	FLOPs (G)	Throughput (images/s)	Train Mem (GB)	Test Mem (GB)	Top-1 (%)	Params (M)	FLOPs (G)	mIoU (%)
MSA	28	4.1	1,293	6.5	1.2	82.3	32	46.5	43.7
SRA [51]	32	4.0	1,425	5.1	1.3	81.7	35	**42.4**	42.8
W-MSA [32]	28	4.0	1,394	5.3	1.2	81.9	32	42.7	41.9
T-MSA [12]	30	4.0	1,462	**5.0**	1.3	81.8	33	42.5	44.0
HiLo	**28**	**3.7**	**1,471**	5.1	**1.2**	**82.0**	**31**	42.6	**44.3**