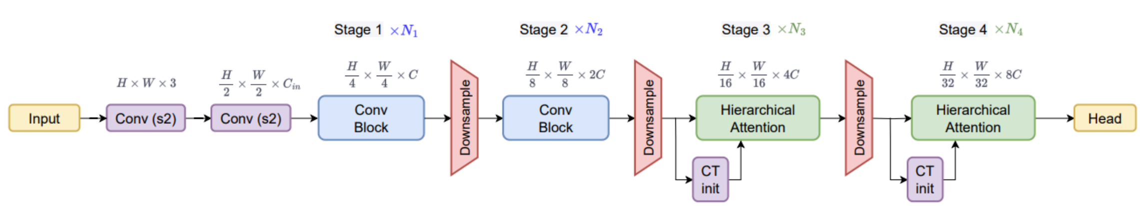
FasterVit: Fast Vision Transformer with Hierarchal Attention

Key Ideas

The paper introduces a hybrid CNN-VIT architecture with focus on throughput. They introduce an new attention module - Hierarchal Attention (HAT) that decomposes the global self-attention which has quadratic complexity into a multi-level attention that reduces the computational complexity. They further claim that their architecture is faster and more accurate than all the other counterparts on high resolution images.

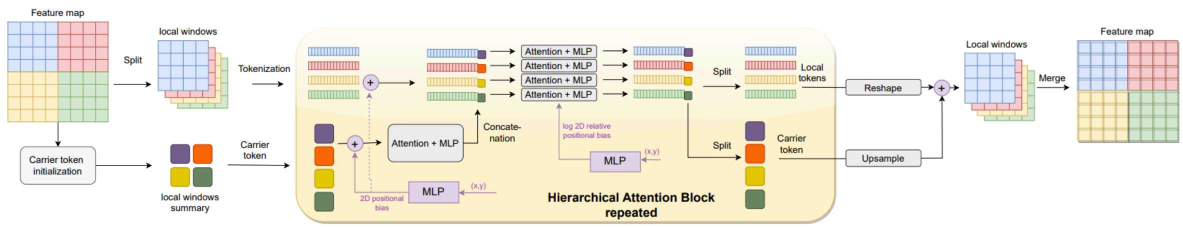
Architecture:



The **FasterVit** has a multi-scale architecture with CNN in the early 2 stages and Hierarchal attention blocks in later 2 stages. The CNN in early stages operate on high resolution features

The architecture is quite simpler. You have a series of convolution layers in the beginning to patchify the image. Post which there are 4 blocks in the network. The first two blocks are convolution blocks that operate on high resolution images. The downsampler layer is again a convolution layer with stride 2. The last two blocks are Hierarchal attention (HAT) blocks.

Hierarchal Attention Module:



In the hierarchal attention module is as follows:

To process a feature map, the following steps are applied:

Local Windows Creation: Begin by dividing the feature map into local windows, as shown in the figure above. These windows serve as localized segments of the image, each focusing on a specific region.

Carrier Tokens Initialization: For each window, generate a set of **carrier tokens** by applying a pooling operation. These carrier tokens act as compact representations or summaries of the local windows, capturing essential information.

Global Information Exchange (Self-Attention on Carrier Tokens): After creating the carrier tokens, gather them across all windows for the entire image. Then, apply **self-attention** to the carrier tokens. This allows the carrier tokens to exchange global information, sharing information from different windows.

Recombining and Local Attention: Once global information is shared via self-attention, concatenate the updated carrier tokens back to their corresponding windows. Then, apply

window-level self-attention on these augmented windows. This step enables an exchange of both local and global information within each window, blending the carrier tokens with the original local tokens.

After applying window-level self-attention, the carrier tokens are separated from the window tokens. The process is repeated n times in the attention block. The carrier tokens are just initialized once at the beginning and then for every repeat, you reuse the carrier tokens processed in the last step.

In the final step, the carrier tokens are upsampled and added back to the local windows. This produces the final processed feature map, where both local and global information have been thoroughly exchanged.

This attention ensures that the feature map benefits from both the local information as well as the global context coming from the carrier tokens.

Impressions

Classification:

Table 1: Comparison of classification benchmarks on **ImageNet-1K** dataset (Deng et al., 2009). Image throughput is measured on A100 GPUs with batch size of 128.

Model	Image Size (Px)	#Param (M)	FLOPs (G)	Throughput (Img/Sec)	Top-1 (%)
Conv-Based					
ConvNeXt-T Liu et al. (2022b)	224	28.6	4.5	3196	82.0
ConvNeXt-S Liu et al. (2022b)	224	50.2	8.7	2008	83.1
ConvNeXt-B Liu et al. (2022b)	224	88.6	15.4	1485	83.8
RegNetY-040 Radosavovic et al. (2020)	288	20.6	6.6	3227	83.0
ResNetV2-101 Wightman et al. (2021)	224	44.5	7.8	4019	82.0
EfficientNetV2-S Tan & Le (2021)	384	21.5	8.0	1735	83.9
Transformer-Based					
Swin-T Liu et al. (2021)	224	28.3	4.4	2758	81.3
Swin-S Liu et al. (2021)	224	49.6	8.5	1720	83.2
SwinV2-T Liu et al. (2022a)	256	28.3	4.4	1674	81.8
SwinV2-S Liu et al. (2022a)	256	49.7	8.5	1043	83.8
SwinV2-B Liu et al. (2022a)	256	87.9	15.1	535	84.6
Twins-B Chu et al. (2021a)	224	56.1	8.3	1926	83.1
DeiT3-L	224	304.4	59.7	535	84.8
PoolFormer-M58 Yu et al. (2022)	224	73.5	11.6	884	82.4
Hybrid					
CoaT-Lite-S Xu et al. (2021a)	224	19.8	4.1	2269	82.3
CrossViT-B Chen et al. (2021a)	240	105.0	20.1	1321	82.2
Visformer-S Chen et al. (2021d)	224	40.2	4.8	3676	82.1
EdgeViT-S Pan et al. (2022)	224	13.1	1.9	4254	81.0
EfficientFormer-L7 Li et al. (2022)	224	82.2	10.2	1359	83.4
MaxViT-B Tu et al. (2022)	224	120.0	23.4	507	84.9
MaxViT-L Tu et al. (2022)	224	212.0	43.9	376	85.1
FasterViT					
FasterViT-0	224	31.4	3.3	**5802**	**82.1**
FasterViT-1	224	53.4	5.3	**4188**	**83.2**
FasterViT-2	224	75.9	8.7	**3161**	**84.2**
FasterViT-3	224	159.5	18.2	**1780**	**84.9**
FasterViT-4	224	424.6	36.6	**849**	**85.4**
FasterViT-5	224	957.5	113.0	**449**	**85.6**
FasterViT-6	224	1360.0	142.0	**352**	**85.8**

They compare the performance of the Fastervit with other transformer and conv based architerctures. When compared with convolutional architectures, for the same throughput, the Fastervit has a slightly better performance. In comparison with transformer like models for example swin, the fastervit model is significantly faster. Finally, comparing the performance with hybrid networks like Edgevit and max-vit, the fastervit has higher throughput and better top1 accuracy on imagenet.

Dense tasks:

Table 2: Object detection and instance segmentation benchmarks using Cascade Mask R-CNN (He et al., 2017) on **MS COCO** dataset (Lin et al., 2014). All models employ 3× schedule. All model statistics are reported using a input test resolution of 1280 × 800.

Backbone	Throu. im/sec	AP^{box} Box	50	75	AP^{mask} Mask	50	75
Swin-T Liu et al. (2021)	161	50.4	69.2	54.7	43.7	66.6	47.3
ConvNeXt-T Liu et al. (2022b)	166	50.4	69.1	54.8	43.7	66.5	47.3
DeiT-Small/16 Touvron et al. (2021a)	269	48.0	67.2	51.7	41.4	64.2	44.3
FasterViT-2	**287**	**52.1**	**71.0**	**56.6**	**45.2**	**68.4**	**49.0**
Swin-S Liu et al. (2021)	119	51.9	70.7	56.3	45.0	68.2	48.8
X101-32 Xie et al. (2017)	124	48.1	66.5	52.4	41.6	63.9	45.2
ConvNeXt-S Liu et al. (2022b)	128	51.9	70.8	56.5	45.0	68.4	49.1
FasterViT-3	**159**	**52.4**	**71.1**	**56.7**	**45.4**	**68.7**	**49.3**
X101-64 Xie et al. (2017)	86	48.3	66.4	52.3	41.7	64.0	45.1
Swin-B Liu et al. (2021)	90	51.9	70.5	56.4	45.0	68.1	48.9
ConvNeXt-B Liu et al. (2022b)	101	52.7	71.3	57.2	45.6	68.9	49.5
FasterViT-4	**117**	**52.9**	**71.6**	**57.7**	**45.8**	**69.1**	**49.8**

From the dense experiments, it is shown that, FasterVit has a better accuracy - throughput tradeoff compared to other architectures. In the table above, we see FasterVit consistently being better both throughput as well as the AP.

Ablation studies

Varying carrier token size :

Window Size	Carrier Token Size	Latency Ratio	Top-1 (%)
7	2	1	84.2
7	1	1.05	83.9
7	9	0.47	84.9
14	0	0.9	84.4

Impact of conv blocks on throughput:

Model	Top-1	Throughput
FasterViT-0	82.1	5802
FasterViT-0 wo Conv-block	81.7	3616
FasterViT-1	83.2	4188
FasterViT-1 wo Conv-block	82.8	3280
FasterViT-2	84.2	3161
FasterViT-2 wo Conv-block	83.8	2085
FasterViT-3	84.9	1780
FasterViT-3 wo Conv-block	84.5	1397
FasterViT-4	85.4	849
FasterViT-4 wo Conv-block	84.9	712

As expected, the conv blocks are more efficient than transformer block and training without them impacts the model throughput by a large margin. It also effects the top1 accuracy on imagenet.

Scaling to different resolution:

	Pretrain		Finetune					
	W8, I256		W12, I384		W16, I512		W24, I768	
Model	acc	im/s	acc	im/s	acc	im/s	acc	im/s
SwinV2-T Liu et al. (2022a)	81.8	1674	83.2	573	83.8	168	84.2	72
SwinV2-S Liu et al. (2022a)	83.7	633	84.8	338	85.4	153	-	-
FasterViT-2	**84.3**	**2500**	**85.3**	**984**	**85.5**	**489**	**85.6**	**155**
SwinV2-B Liu et al. (2022a)	84.2	499	85.1	251	85.6	115	-	-
FasterViT-4 256	**85.3**	**653**	**86.0**	**254**	**86.1**	**133**	**86.0**	**44**

In the table above, they compare the performance of the SwinV2 model with FasterViT on high-resolution images. Both models are initially pretrained on ImageNet-1K for 300 epochs with an image resolution of 256^2 pixels. They are then fine-tuned at a larger resolution (denoted as I) for 30 additional epochs, using various window sizes (W). Across all configurations, FasterViT consistently achieves higher image throughput, often outperforming SwinV2 by a notable margin.