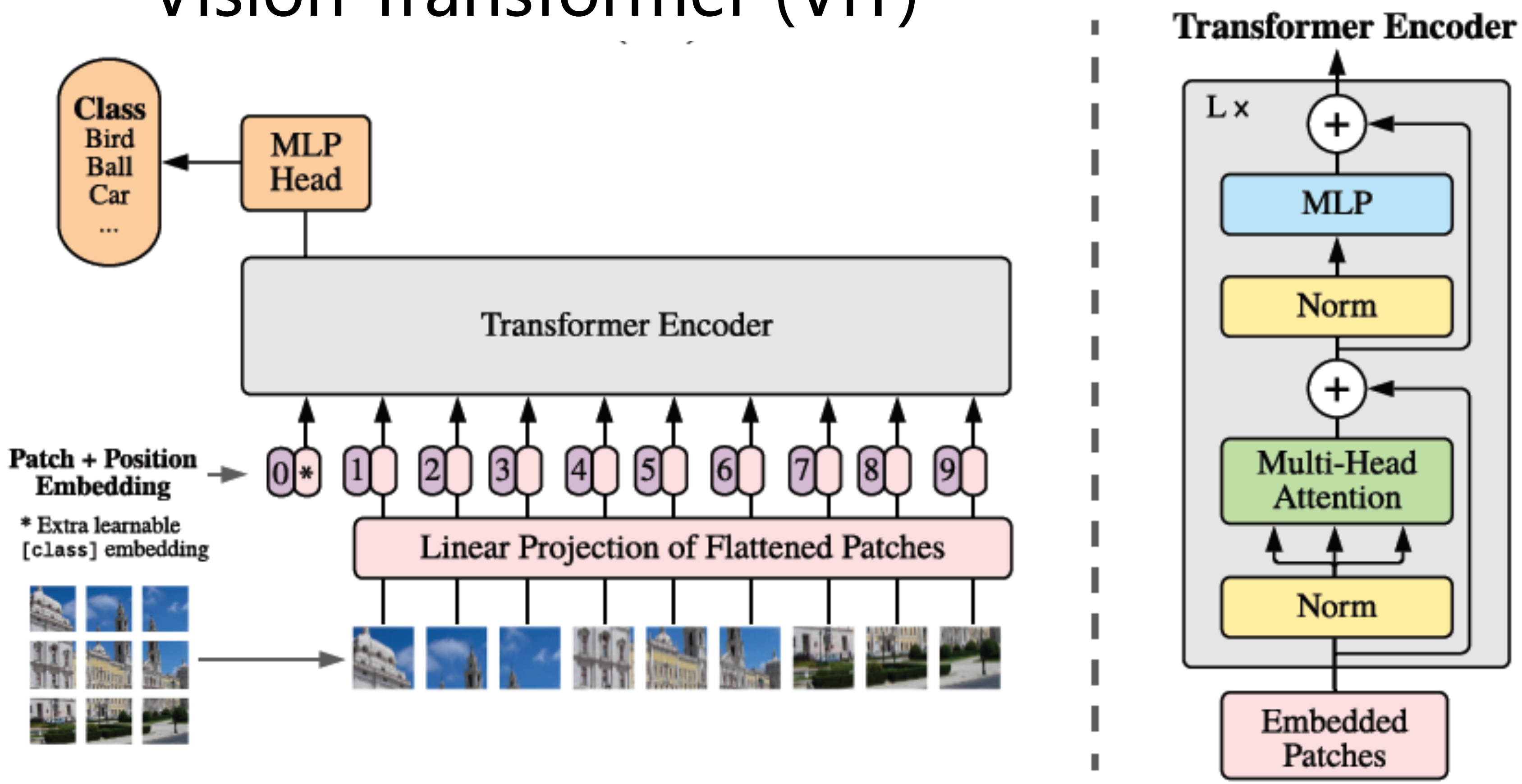
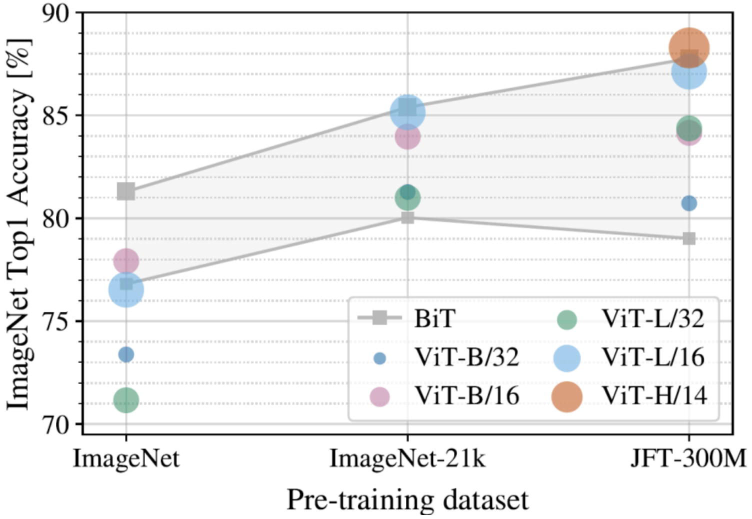
Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions

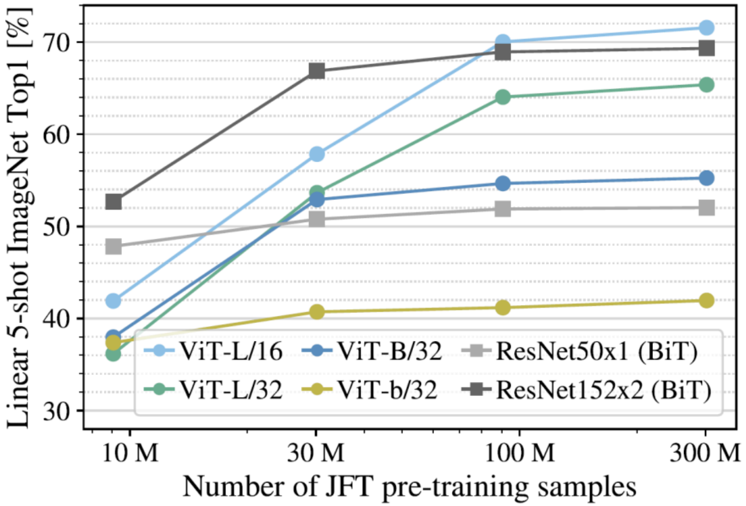
Wenhai Wang[1], Enze Xie[2], Xiang Li[3], Deng-Ping Fan[4],
Kaitao Song[3], Ding Liang[5], Tong Lu[1], Ping Luo[2], Ling Shao[4]
[1]Nanjing University [2]The University of Hong Kong
[3]Nanjing University of Science and Technology [4]IIAI [5]SenseTime Research
wangwenhai362@smail.nju.edu.cn

Related Work

Vision Transformer (ViT)

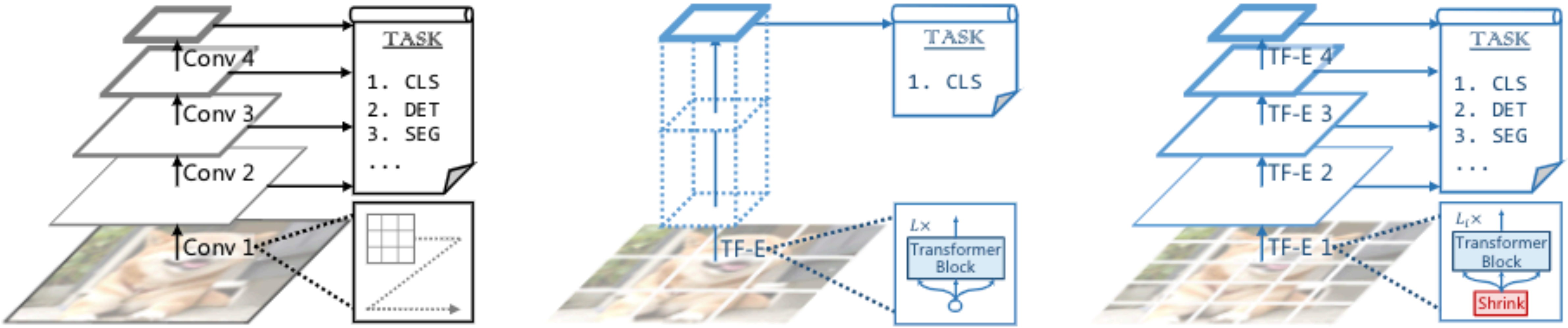






Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." in ICLR, 2020.

Pyramid Vision Transformer (PVT)



(a) CNNs: VGG [41], ResNet [15], *etc.* (b) Vision Transformer [10] (c) Pyramid Vision Transformer (ours)

CNN's Limitations

- Local Receptive Field
- Fixed Weights

ViT's Limitations

- Columnar Structure
- Low-Resolution Output
- Unsuitable for Det/Seg

PVT (ours)

- A Transformer backbone as versatile as CNN

Wang, Wenhai, et al. "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions." arXiv preprint arXiv:2102.12122 (2021).

Overall Architecture

- Key Points

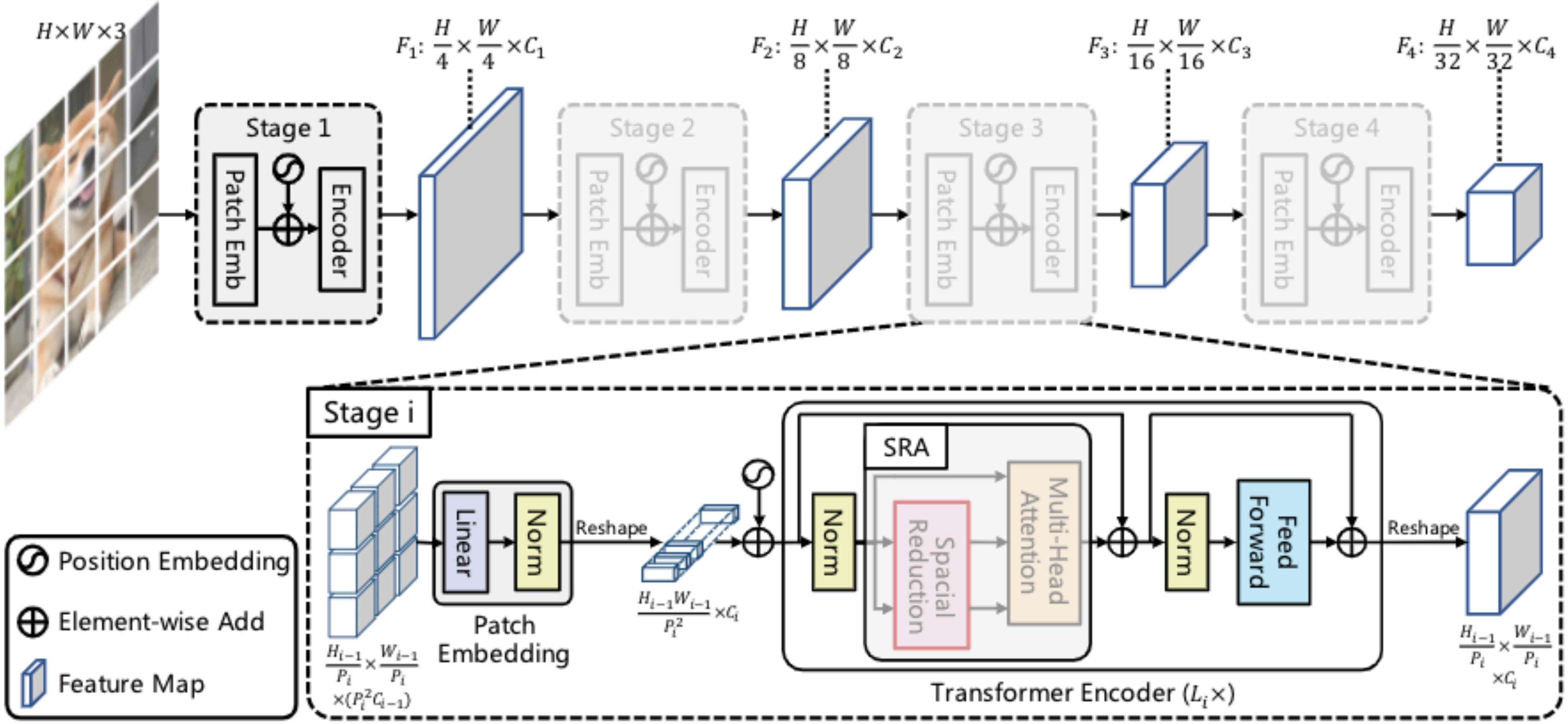
- Four Stages

- Each Stage:

  (1) Patch Emb.

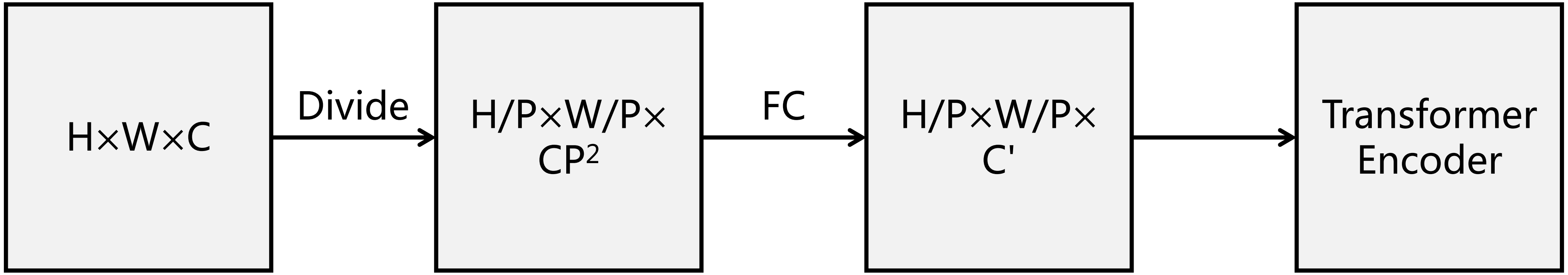
  (2) Transformer Enc.

- Spatial-Reduction Attention (SRA) for high-resolution input



How PVT obtains the feature pyramid?

- Adjusting the patch size (P_i) in Stage i



The process of the patch embedding in Stage i

Spatial-Reduction Attention

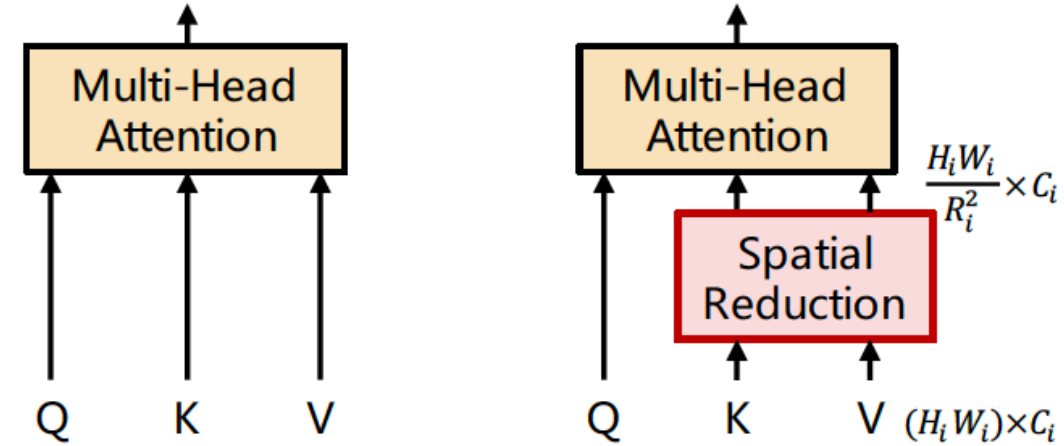
- SRA

$$\mathrm{SRA}(Q,K,V)=\mathrm{Concat}(\mathrm{head}_0,\ldots,\mathrm{head}_{N_i})W^O$$

$$\mathrm{head}_j=\mathrm{Attention}(QW_j^Q,\mathrm{SR}(K)W_j^K,\mathrm{SR}(V)W_j^V)$$

$$\boxed{\mathrm{SR}(\mathbf{x})=\mathrm{Norm}(\mathrm{Reshape}(\mathbf{x},R_i)W^S)}$$

$$\mathrm{Attention}(\mathbf{q},\mathbf{k},\mathbf{v})=\mathrm{Softmax}\left(\frac{\mathbf{q}\mathbf{k}^{\mathsf{T}}}{\sqrt{d_{\mathbf{head}}}}\right)\mathbf{v}$$



Multi-Head Attention Spatial-Reduction Attention (ours)

Compared to original multi-head attention, the complexity of SRA is R_i^2 times lower!

Detailed settings

- P_i: the patch size of the stage i;
- C_i: the channel number of the output of the stage i;
- L_i: the number of encoder layers in the stage i;
- R_i: the reduction ratio of the SRA in the stage i;
- N_i: the head number of the SRA in the stage i;
- E_i: the expansion ratio of the feed-forward layer [51] in the stage i;

	Output Size	Layer Name	PVT-Tiny	PVT-Small	PVT-Medium	PVT-Large
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	Patch Embedding	$P_1 = 4;\ C_1 = 64$			
		Transformer Encoder	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_1 = 8 \\ N_1 = 1 \\ E_1 = 8 \end{bmatrix} \times 3$
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	Patch Embedding	$P_2 = 2;\ C_2 = 128$			
		Transformer Encoder	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} R_2 = 4 \\ N_2 = 2 \\ E_2 = 8 \end{bmatrix} \times 8$
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	Patch Embedding	$P_3 = 2;\ C_3 = 320$			
		Transformer Encoder	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 18$	$\begin{bmatrix} R_3 = 2 \\ N_3 = 5 \\ E_3 = 4 \end{bmatrix} \times 27$
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	Patch Embedding	$P_4 = 2;\ C_4 = 512$			
		Transformer Encoder	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 2$	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} R_4 = 1 \\ N_4 = 8 \\ E_4 = 4 \end{bmatrix} \times 3$

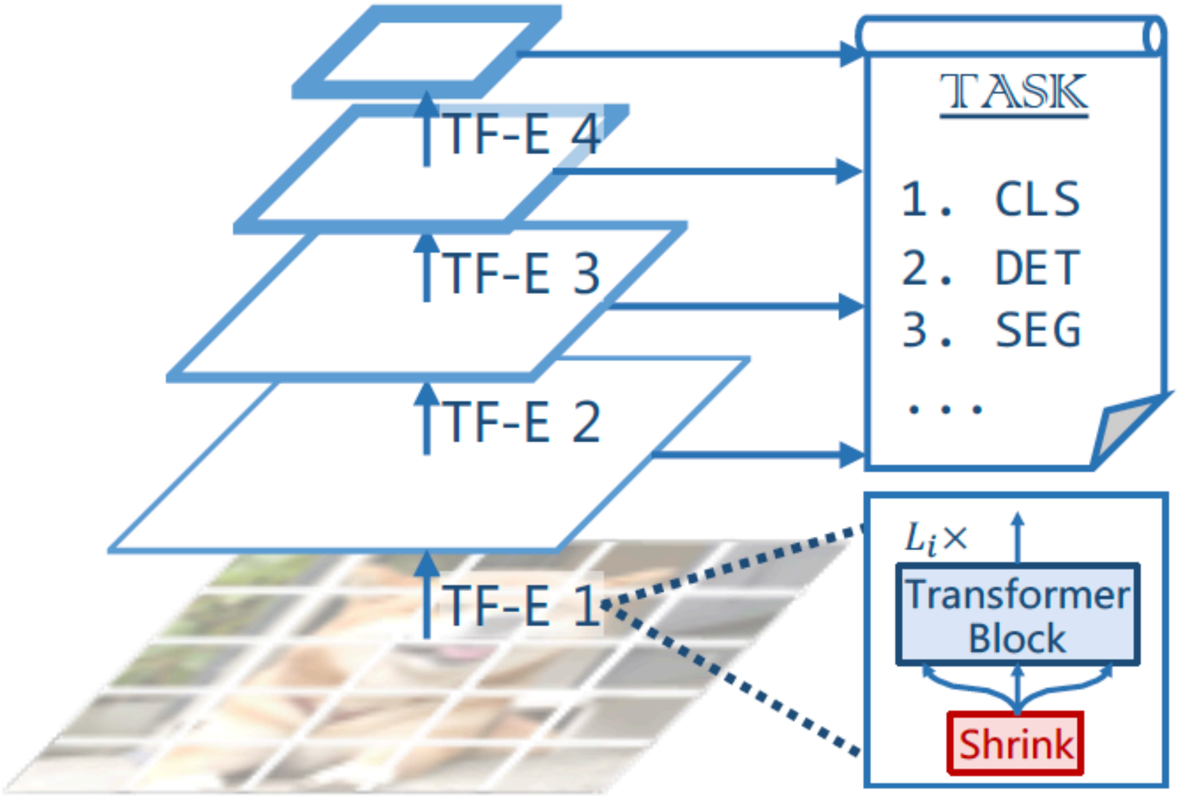
Table A1: **Detailed settings of Pyramid Vision Transformer (PVT) series.** The design follows the two rules of ResNet [5]. (1) With the growth of network depth, the hidden dimension gradually increases, and the output resolution progressively shrinks; (2) The major computation resource is concentrated in Stage 3.

Advantages

- Multi-Scale/High-Resolution Output

- As versatile as CNN, can be apply to detection/segmentation

- Making pure Transformer detection/ segmentation possible, for example
  (1) PVT + DETR
  (2) PVT + Trans2Seg



Performance

- PVT-S vs. R50

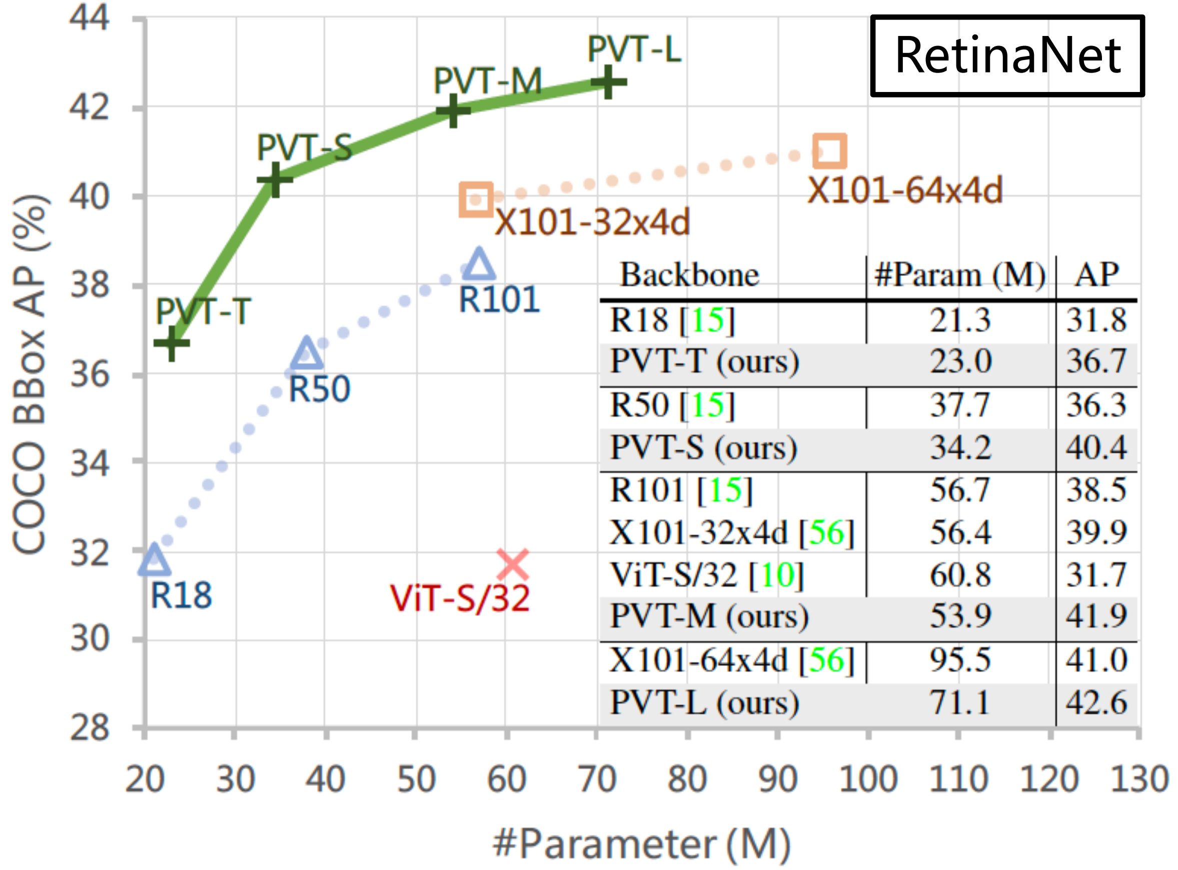
AP: 40.4 vs. 36.3 (+4.1)

#Param: 34.2 vs. 37.7

- PVT-L vs. X101-64x4d

AP: 42.6 vs. 41.0 (+1.6)

#Param: 71.1 vs. 95.5 (20% fewer)



Backbone	#Param (M)	AP
R18 [15]	21.3	31.8
PVT-T (ours)	23.0	36.7
R50 [15]	37.7	36.3
PVT-S (ours)	34.2	40.4
R101 [15]	56.7	38.5
X101-32x4d [56]	56.4	39.9
ViT-S/32 [10]	60.8	31.7
PVT-M (ours)	53.9	41.9
X101-64x4d [56]	95.5	41.0
PVT-L (ours)	71.1	42.6

Deeper vs. Wider & Pretrained

- Deeper vs. Wider

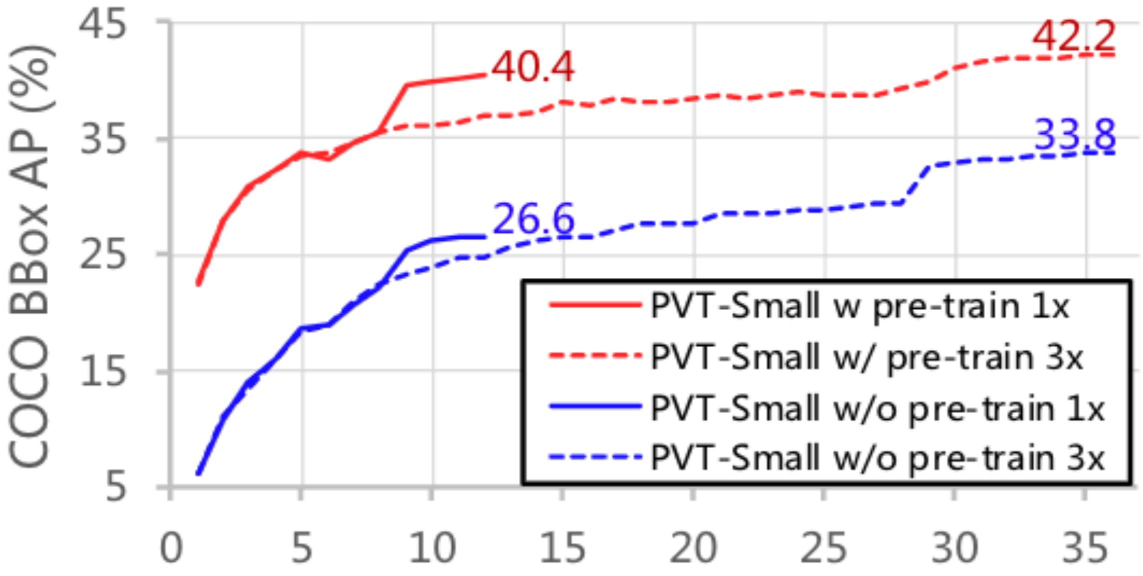
Going deeper is better than going wider.

Method	#Param (M)	Top-1	RetinaNet 1x		
			AP	AP_{50}	AP_{75}
Wider PVT-Small	46.8	19.3	40.8	61.8	43.3
Deeper PVT-Small	44.2	18.8	41.9	63.1	44.3

Table A3: **Deeper *vs.* Wider.** "Top-1" denotes the top-1 error on the ImageNet validation set. "AP" denotes the bounding box AP on COCO `val2017`. The deeper model obtains better performance than the wider model under comparable parameter number.

- Pretrained

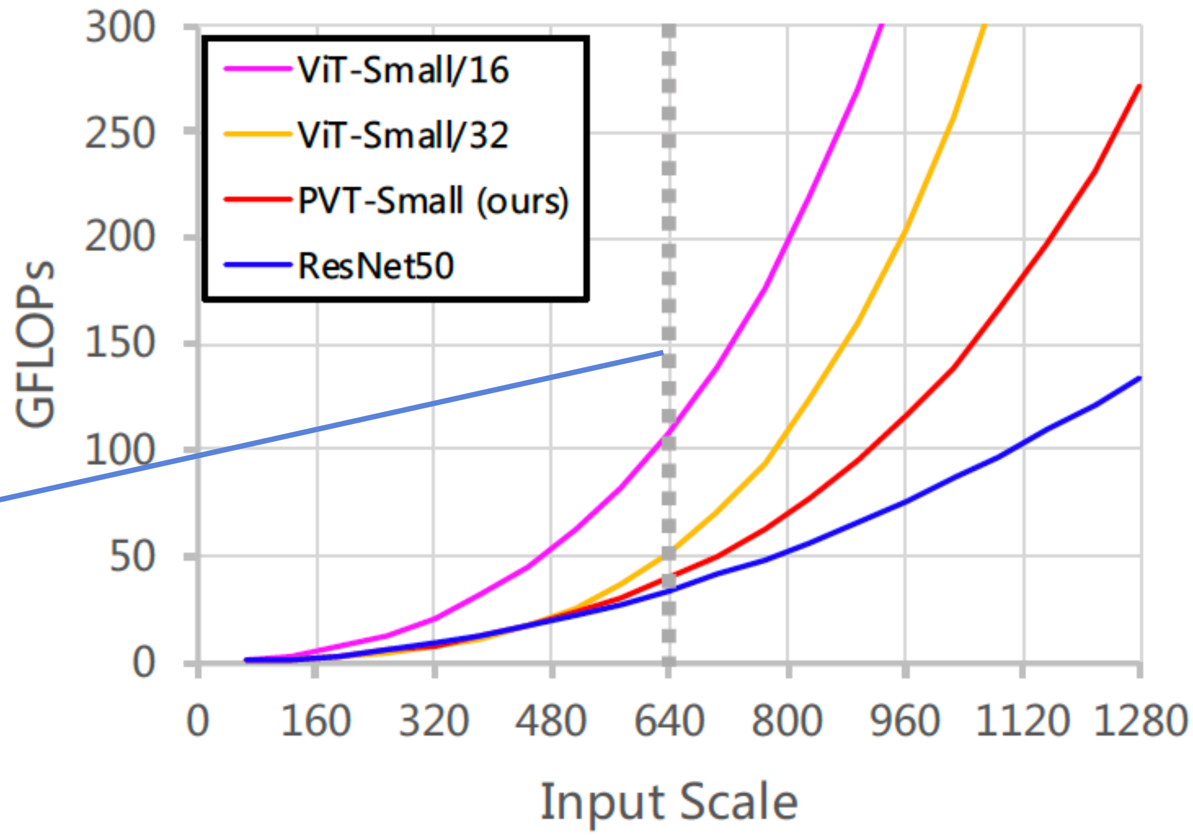
Pretrained weights can help PVT converge faster and better.



Computation Cost

- FLOPs Growth Rate

ViT-S/16 > ViT-S/32 > PVT-S > R50

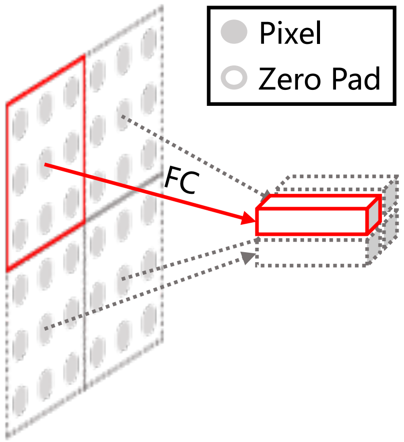


PVT is more suitable for medium-resolution.

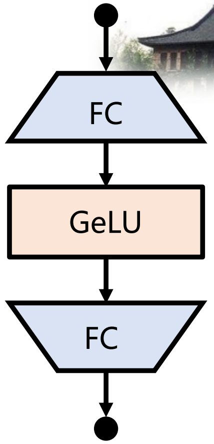
PVTv2 solve this problem by Linear SRA!!!

Improvements

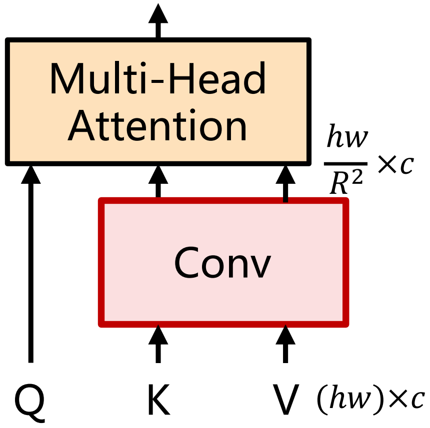
- Overlapping Patch Embedding
- Convolutional Feed-Forward
- Linear SRA



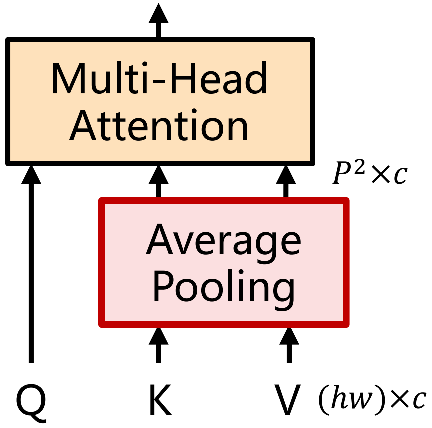
Before: Original Patch Embedding



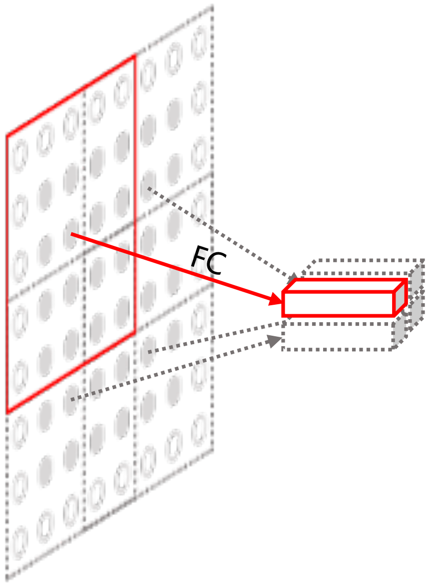
Before: Original Feed-Forward



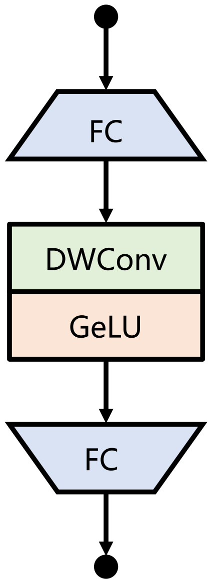
PVTv1: SRA



PVTv2: Linear SRA



Improved: Overlapping Patch Embedding



Improved: Convolutional Feed-Forward

Performance

Classification on ImageNet

Method	#Param (M)	GFLOPS	Top-1 Acc (%)
ResNeXt101-64x4d [33]	83.5	15.6	79.6
RegNetY-16G [24]	84.0	16.0	82.9
ViT-Base/16 [7]	86.6	17.6	81.8
DeiT-Base/16 [29]	86.6	17.6	81.8
Swin-B [21]	88.0	15.4	83.3
Twins-SVT-L [4]	99.2	14.8	83.3
PVTv2-B5 (ours)	**82.0**	**11.8**	**83.8**

Detection on COCO

Backbone	Method	AP^b	AP^b_{50}	AP^b_{75}	#P (M)	GFLOPS
ResNet50 [13]	Cascade Mask R-CNN [1]	46.3	64.3	50.5	82	739
Swin-T [21]		50.5	69.3	54.9	86	745
PVTv2-B2-Li (ours)		50.9	69.5	55.2	**80**	**725**
PVTv2-B2 (ours)		**51.1**	**69.8**	**55.3**	83	788
ResNet50 [13]	ATSS [37]	43.5	61.9	47.0	32	205
Swin-T [21]		47.2	66.5	51.3	36	215
PVTv2-B2-Li (ours)		48.9	68.1	53.4	**30**	**194**
PVTv2-B2 (ours)		**49.9**	**69.1**	**54.1**	33	258
ResNet50 [13]	GFL [17]	44.5	63.0	48.3	32	208
Swin-T [21]		47.6	66.8	51.7	36	215
PVTv2-B2-Li (ours)		49.2	68.2	53.7	**30**	**197**
PVTv2-B2 (ours)		**50.2**	**69.4**	**54.7**	33	261
ResNet50 [13]	Sparse R-CNN [26]	44.5	63.4	48.2	106	166
Swin-T [21]		47.9	67.3	52.3	110	172
PVTv2-B2-Li (ours)		48.9	68.3	53.4	**104**	**151**
PVTv2-B2 (ours)		**50.1**	**69.5**	**54.9**	107	215

Future Direction

- Efficient Attention Layer

Deformable Attention, Linear SRA, ...

- Position Embedding for 2D/3D Images

CPVT, Local ViT, ...

- Pure Transformer Vision Models

Segformer, YOLOS, ...

- Transformer + NAS/Pruning/Distillation/Quantification

Visual Transformer Pruning, Patch Slimming, ...

- Multimodal Transformer (*e.g.*, CV+NLP)

CLIP, Kaleido-BERT, ...

Thanks

Code: https://github.com/whai362/PVT