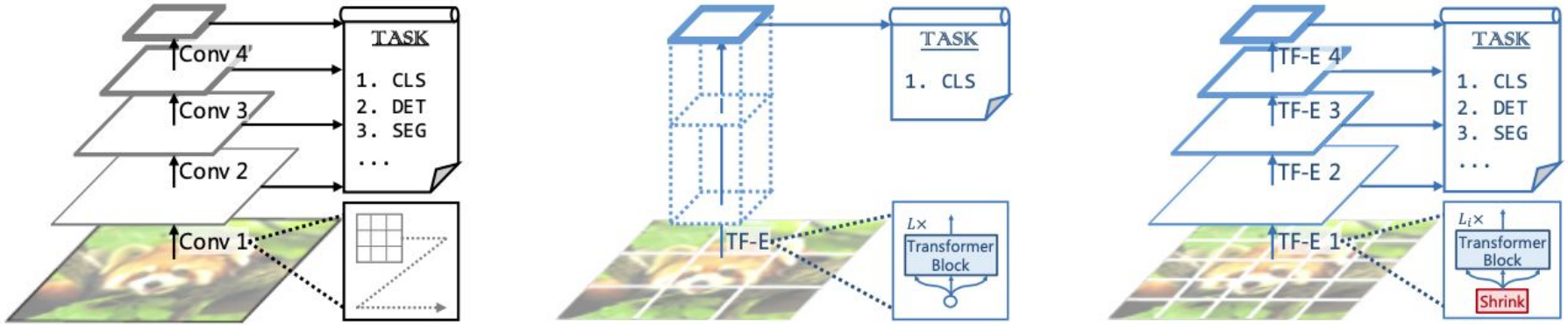
Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions

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

https://github.com/whai362/PVT

Overview

- Adapt ViT for dense prediction tasks by introducing pyramidal structure
- Computationally efficient while convolution-free
- Replace CNN backbones with performance boost for detection, segmentation



(a) CNNs: VGG [54], ResNet [22], *etc*.

(b) Vision Transformer [13]

(c) Pyramid Vision Transformer (ours)

Figure 1. Comparison of different architectures

(a) CNN backbones for dense prediction tasks commonly have pyramidal shape

(b) Columnar structure of ViT ill-suited for dense prediction

(c) PVT has properties of both

Architecture

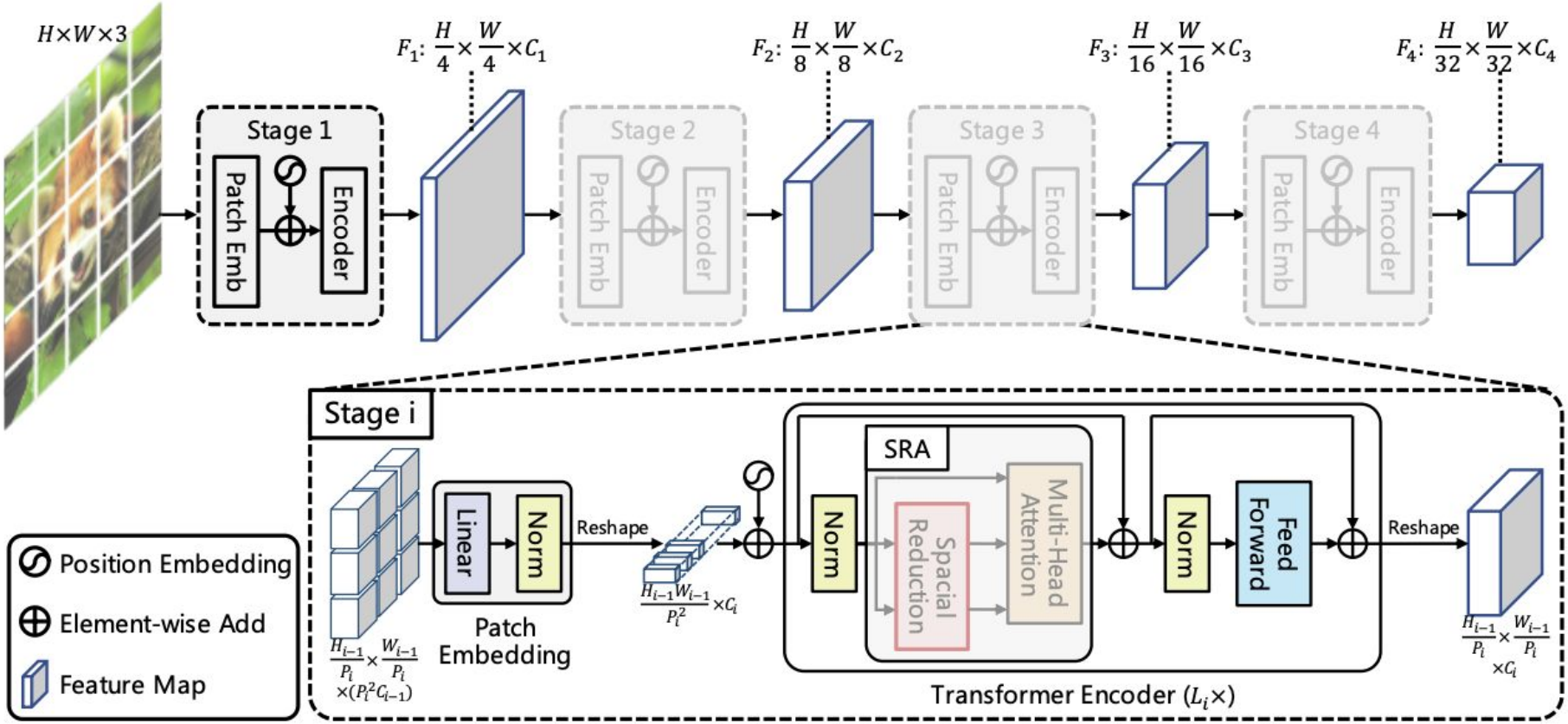


Figure 3: **Overall architecture of Pyramid Vision Transformer (PVT).** The entire model is divided into four stages, each of which is comprised of a patch embedding layer and a L_i-layer Transformer encoder. Following a pyramid structure, the output resolution of the four stages progressively shrinks from high (4-stride) to low (32-stride).

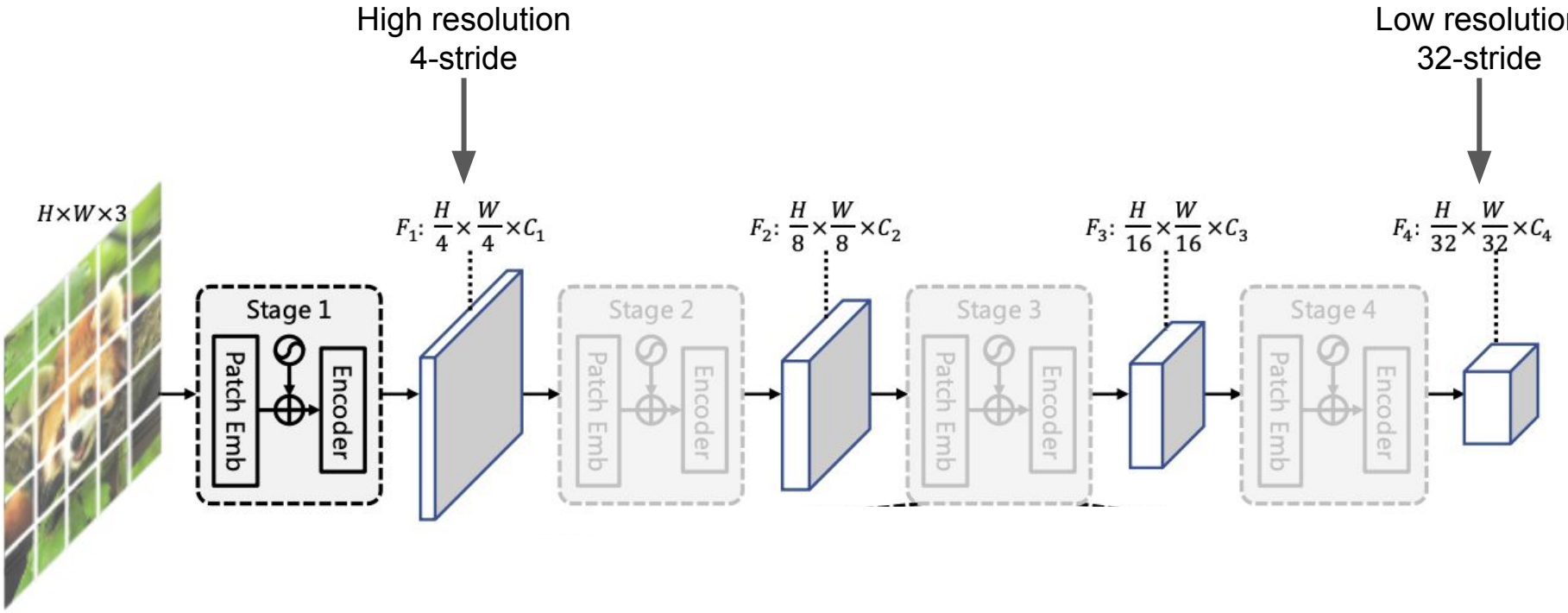


Figure 3. Overall architecture of Pyramid Vision Transformer (PVT)
4 stages generate feature maps of different scales

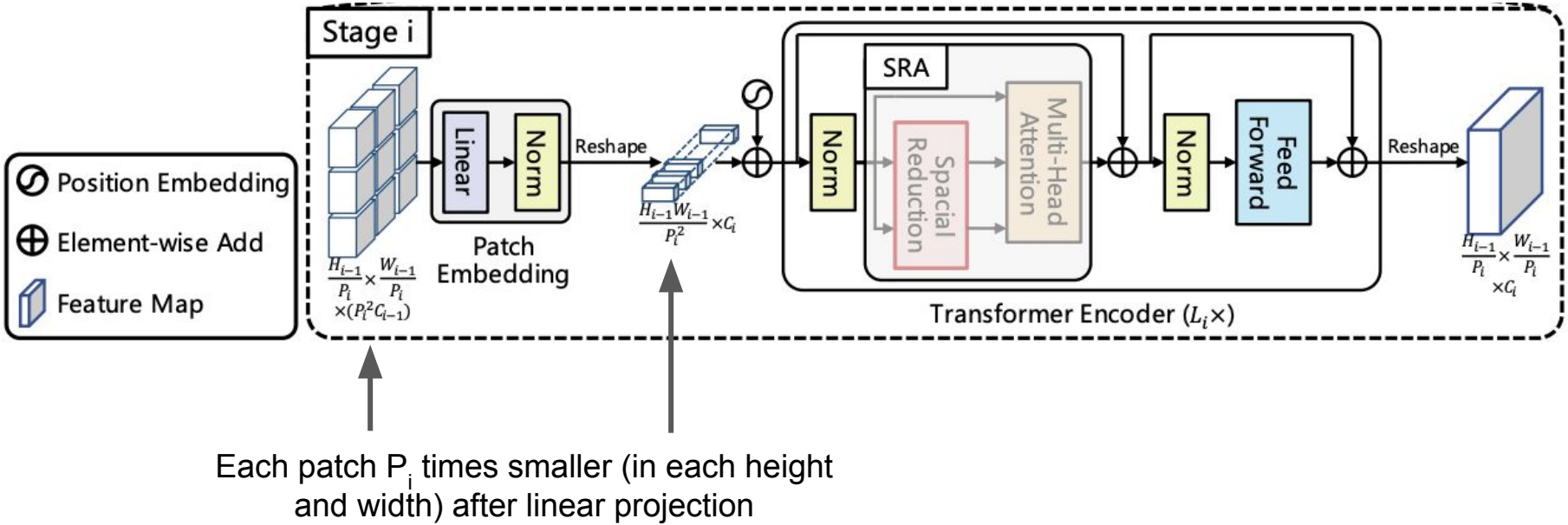


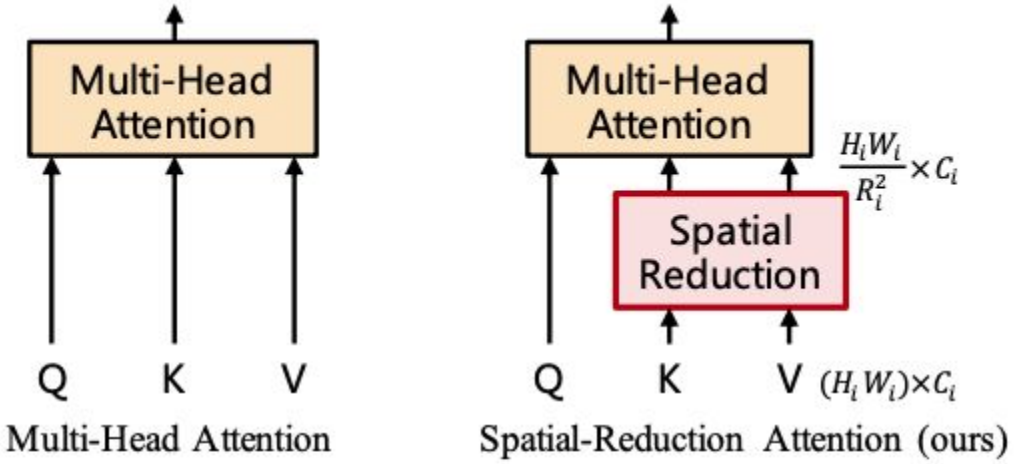
Figure 3. Overall architecture of Pyramid Vision Transformer (PVT)
Flattened patches → linear projection and positional embedding → Transformer encoder with L_i layers → reshaped feature map

Feature Pyramid

- CNN backbones use different convolutional strides to obtain multi-scale feature maps
- PVT uses progressive shrinking strategy in linear embedding step

Transformer Encoder

- Each encoder layer consists of attention + feed-forward layer
- New spatial reduction attention (SRA) replaces multi-headed attention (MHA)
  - SRA reduces spatial scale of K and V before attention operation
  - R_i^2 lower computational/memory cost than MHA (reduction ratio R_i)



Spatial Reduction Attention (SRA)

$$\mathrm{SRA}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_0, ..., \mathrm{head}_{N_i})W^O, \quad (1)$$

$$\mathrm{head}_j = \mathrm{Attention}(QW_j^Q, \mathrm{SR}(K)W_j^K, \mathrm{SR}(V)W_j^V), \quad (2)$$

With linear projection parameters

$$W_j^Q \in \mathbb{R}^{C_i \times d_{\mathrm{head}}}, W_j^K \in \mathbb{R}^{C_i \times d_{\mathrm{head}}}, W_j^V \in \mathbb{R}^{C_i \times d_{\mathrm{head}}}, W^O \in \mathbb{R}^{C_i \times C_i}$$

and N_i heads, resulting in $d_{\mathrm{head}} = C_i / N_i$

Spatial Reduction Attention (SRA)

Spatial reduction of input sequence **x** with reduction ratio R_i

$$\mathrm{SR}(\mathbf{x}) = \mathrm{Norm}(\mathrm{Reshape}(\mathbf{x}, R_i)W^S). \quad (3)$$

Reshape(**x**, R_i) reshapes **x** into $\frac{H_i W_i}{R_i^2} \times (R_i^2 C_i)$

Linear projection $W_S \in \mathbb{R}^{(R_i^2 C_i) \times C_i}$ reduces input sequence to C_i

Layer normalization Norm(.)

Spatial Reduction Attention (SRA)

Standard attention operation

$$\text{Attention}(\mathbf{q}, \mathbf{k}, \mathbf{v}) = \text{Softmax}(\frac{\mathbf{q}\mathbf{k}^{\mathsf{T}}}{\sqrt{d_{\text{head}}}})\mathbf{v}. \qquad (4)$$

Architecture Summary

Parameters:

- P_i: patch size of Stage i
- C_i: channel number of output of Stage i
- L_i: number of encoder layers in Stage i
- R_i: reduction ratio of SRA in Stage i
- N_i: head number of SRA in Stage i
- E_i: expansion ratio of feed-forward layer in Stage i

Experiments select scales to match parameter numbers in ResNets

Comparable to:

	Output Size	Layer Name	PVT-Tiny (ResNet18)	PVT-Small (ResNet50)	PVT-Medium (ResNet101)	PVT-Large (ResNet152)
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 1: **Detailed settings of PVT series.** The design follows the two rules of ResNet [22]: (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.

Experiments

Application to Downstream Tasks

- Image-level prediction
  - Follow ViT/DeiT to append learnable classification token to input of last stage
  - Fully connected layer conducts classification on top of token
- Dense prediction
  - Initialize PVT backbone with weights pre-trained on ImageNet
  - Use output feature pyramid $\{F_1, F_2, F_3, F_4\}$ as input to convolutional Feature Pyramidal Network (FPN)
  - Refined feature pyramid output from FPN fed to follow-up detection/segmentation head
  - Bilinear interpolation on pre-trained positional embeddings prevents loss of meaning on non-ImageNet data

Image Classification

- ImageNet-1k
- All models trained on training set, report top-1 error on validation
- Follow data augmentation from DeiT
  - Random crop, random horizontal flipping, label-smoothing regularization, mixup, CutMix, random erasing

Method	#Param (M)	GFLOPs	Top-1 Err (%)
ResNet18* [22]	11.7	1.8	30.2
ResNet18 [22]	11.7	1.8	31.5
DeiT-Tiny/16 [63]	5.7	1.3	27.8
PVT-Tiny (ours)	13.2	1.9	24.9
ResNet50* [22]	25.6	4.1	23.9
ResNet50 [22]	25.6	4.1	21.5
ResNeXt50-32x4d* [73]	25.0	4.3	22.4
ResNeXt50-32x4d [73]	25.0	4.3	20.5
T2T-ViT$_t$-14 [75]	22.0	6.1	19.3
TNT-S [19]	23.8	5.2	18.7
DeiT-Small/16 [63]	22.1	4.6	20.1
PVT-Small (ours)	24.5	3.8	20.2
ResNet101* [22]	44.7	7.9	22.6
ResNet101 [22]	44.7	7.9	20.2
ResNeXt101-32x4d* [73]	44.2	8.0	21.2
ResNeXt101-32x4d [73]	44.2	8.0	19.4
T2T-ViT$_t$-19 [75]	39.0	9.8	18.6
ViT-Small/16 [13]	48.8	9.9	19.2
PVT-Medium (ours)	44.2	6.7	18.8
ResNeXt101-64x4d* [73]	83.5	15.6	20.4
ResNeXt101-64x4d [73]	83.5	15.6	18.5
ViT-Base/16 [13]	86.6	17.6	18.2
T2T-ViT$_t$-24 [75]	64.0	15.0	17.8
TNT-B [19]	66.0	14.1	17.2
DeiT-Base/16 [63]	86.6	17.6	18.2
PVT-Large (ours)	61.4	9.8	18.3

Image Classification

- With similar GFLOPs, PVT outperforms ResNet
- Comparable performance to transformer-based models
- Pyramidal structure not expected to improve image-level prediction performance

Method	#Param (M)	GFLOPs	Top-1 Err (%)
ResNet18* [22]	11.7	1.8	30.2
ResNet18 [22]	11.7	1.8	31.5
DeiT-Tiny/16 [63]	5.7	1.3	27.8
PVT-Tiny (ours)	13.2	1.9	24.9
ResNet50* [22]	25.6	4.1	23.9
ResNet50 [22]	25.6	4.1	21.5
ResNeXt50-32x4d* [73]	25.0	4.3	22.4
ResNeXt50-32x4d [73]	25.0	4.3	20.5
T2T-ViT$_t$-14 [75]	22.0	6.1	19.3
TNT-S [19]	23.8	5.2	18.7
DeiT-Small/16 [63]	22.1	4.6	20.1
PVT-Small (ours)	24.5	3.8	20.2
ResNet101* [22]	44.7	7.9	22.6
ResNet101 [22]	44.7	7.9	20.2
ResNeXt101-32x4d* [73]	44.2	8.0	21.2
ResNeXt101-32x4d [73]	44.2	8.0	19.4
T2T-ViT$_t$-19 [75]	39.0	9.8	18.6
ViT-Small/16 [13]	48.8	9.9	19.2
PVT-Medium (ours)	44.2	6.7	18.8
ResNeXt101-64x4d* [73]	83.5	15.6	20.4
ResNeXt101-64x4d [73]	83.5	15.6	18.5
ViT-Base/16 [13]	86.6	17.6	18.2
T2T-ViT$_t$-24 [75]	64.0	15.0	17.8
TNT-B [19]	66.0	14.1	17.2
DeiT-Base/16 [63]	86.6	17.6	18.2
PVT-Large (ours)	61.4	9.8	18.3

Object Detection

- COCO dataset
- Test PVT backbone on standard RetinaNet and Mask R-CNN detectors
- Data augmentation via random resizing
- Outperforms comparable ResNet/ResNeXT backbones

Backbone	#Param (M)	RetinaNet 1x						RetinaNet 3x + MS					
		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet18 [22]	21.3	31.8	49.6	33.6	16.3	34.3	43.2	35.4	53.9	37.6	19.5	38.2	46.8
PVT-Tiny (ours)	23.0	36.7(+4.9)	56.9	38.9	22.6	38.8	50.0	39.4(+4.0)	59.8	42.0	25.5	42.0	52.1
ResNet50 [22]	37.7	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 (ours)	34.2	40.4(+4.1)	61.3	43.0	25.0	42.9	55.7	42.2(+3.2)	62.7	45.0	26.2	45.2	57.2
ResNet101 [22]	56.7	38.5	57.8	41.2	21.4	42.6	51.1	40.9	60.1	44.0	23.7	45.0	53.8
ResNeXt101-32x4d [73]	56.4	39.9(+1.4)	59.6	42.7	22.3	44.2	52.5	41.4(+0.5)	61.0	44.3	23.9	45.5	53.7
PVT-Medium (ours)	53.9	41.9(+3.4)	63.1	44.3	25.0	44.9	57.6	43.2(+2.3)	63.8	46.1	27.3	46.3	58.9
ResNeXt101-64x4d [73]	95.5	41.0	60.9	44.0	23.9	45.2	54.0	41.8	61.5	44.4	25.2	45.4	54.6
PVT-Large (ours)	71.1	42.6(+1.6)	63.7	45.4	25.8	46.0	58.4	43.4(+1.6)	63.6	46.1	26.1	46.0	59.5

Table 3: **Object detection performance on COCO `val2017`.** "MS" means that multi-scale training [39, 21] is used.

Backbone	#Param (M)	Mask R-CNN 1x						Mask R-CNN 3x + MS					
		AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet18 [22]	31.2	34.0	54.0	36.7	31.2	51.0	32.7	36.9	57.1	40.0	33.6	53.9	35.7
PVT-Tiny (ours)	32.9	36.7(+2.7)	59.2	39.3	35.1(+3.9)	56.7	37.3	39.8(+2.9)	62.2	43.0	37.4(+3.8)	59.3	39.9
ResNet50 [22]	44.2	38.0	58.6	41.4	34.4	55.1	36.7	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small (ours)	44.1	40.4(+2.4)	62.9	43.8	37.8(+3.4)	60.1	40.3	43.0(+2.0)	65.3	46.9	39.9(+2.8)	62.5	42.8
ResNet101 [22]	63.2	40.4	61.1	44.2	36.4	57.7	38.8	42.8	63.2	47.1	38.5	60.1	41.3
ResNeXt101-32x4d [73]	62.8	41.9(+1.5)	62.5	45.9	37.5(+1.1)	59.4	40.2	44.0(+1.2)	64.4	48.0	39.2(+0.7)	61.4	41.9
PVT-Medium (ours)	63.9	42.0(+1.6)	64.4	45.6	39.0(+2.6)	61.6	42.1	44.2(+1.4)	66.0	48.2	40.5(+2.0)	63.1	43.5
ResNeXt101-64x4d [73]	101.9	42.8	63.8	47.3	38.4	60.6	41.3	44.4	64.9	48.8	39.7	61.9	42.6
PVT-Large (ours)	81.0	42.9(+0.1)	65.0	46.6	39.5(+1.1)	61.9	42.5	44.5(+0.1)	66.0	48.3	40.7(+1.0)	63.4	43.7

Table 4: **Object detection and instance segmentation performance on COCO `val2017`.** AP^b and AP^m denote bounding box AP and mask AP, respectively.

Semantic Segmentation

- ADE20K: scene parsing dataset
- Test backbones on Semantic FPN segmentation network
- Achieves better performance than ResNet/ResNeXT backbones with comparable or fewer parameters
- PVT extracts better features than CNN backbones due to global attention mechanism

Backbone	Semantic FPN		
	#Param (M)	GFLOPs	mIoU (%)
ResNet18 [22]	15.5	32.2	32.9
PVT-Tiny (ours)	17.0	33.2	35.7(+2.8)
ResNet50 [22]	28.5	45.6	36.7
PVT-Small (ours)	28.2	44.5	39.8(+3.1)
ResNet101 [22]	47.5	65.1	38.8
ResNeXt101-32x4d [73]	47.1	64.7	39.7(+0.9)
PVT-Medium (ours)	48.0	61.0	41.6(+2.8)
ResNeXt101-64x4d [73]	86.4	103.9	40.2
PVT-Large (ours)	65.1	79.6	42.1(+1.9)
PVT-Large* (ours)	65.1	79.6	44.8

Table 5: **Semantic segmentation performance of different backbones on the ADE20K validation set.** "GFLOPs" is calculated under the input scale of 512×512. "*" indicates 320K iterations training and multi-scale flip testing.

Pure Transformer Detection

- Object detection: PVT+DE:TR
  - Outperforms ResNet-based DE:TR by 2.4 AP points (34.7 vs 32.3) on COCO2017
  - Pure transformer detector can work well for object detection

Method	DETR (50 Epochs)					
	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet50 [22]	32.3	53.9	32.3	10.7	33.8	53.0
PVT-Small (ours)	34.7(+2.4)	55.7	35.4	12.0	36.4	56.7

Table 6: **Performance of the pure Transformer object detection pipeline.** We build a pure Transformer detector by combining PVT and DETR [6], whose AP is 2.4 points higher than the original DETR based on ResNet50 [22].

Pure Transformer Segmentation

- Semantic segmentation: PVT+Trans2Seg
  - Outperforms ResNet-based Trans2Seg with lower GFLOPs
  - Pure transformer detector can work for semantic segmentation

Method	#Param (M)	GFLOPs	mIoU (%)
ResNet50-d8+DeeplabV3+ [9]	26.8	120.5	41.5
ResNet50-d16+DeeplabV3+ [9]	26.8	45.5	40.6
ResNet50-d16+Trans2Seg [72]	56.1	79.3	39.7
PVT-Small+Trans2Seg	32.1	31.6	42.6(+2.9)

Table 7: **Performance of the pure Transformer semantic segmentation pipeline.** We build a pure Transformer detector by combining PVT and Trans2Seg [72]. It is 2.9% higher than ResNet50-d16+Trans2Seg and 1.1% higher than ResNet50-d8+DeeplabV3+ with lower GFlops. "d8" and "d16" means dilation 8 and 16, respectively.

Ablation Studies

- Deeper more effective than wider models with comparable parameters
- Pre-training on ImageNet for dense prediction backbone improves convergence (vs no pre-training)
- Pure Transformer backbone with multi-headed attention can be more expressive than convolution
- With increasing input scale, PVT GFLOPs grow faster than ResNet but lower than ViT
    - PVT may be most suitable for medium-resolution images

Summary

- Progressive shrinking pyramid and spatial-reduction attention obtain high-resolution and multi-scale feature maps with reduced computation/memory resources
- PVT outperforms comparable ResNet/ResNeXT backbones for object detection and semantic segmentation

Discussion Questions

- Has this paper convinced you to consider using PVT going forward?
- The pure transformer object detection network still underperforms PVT+convolutional detection network. Why do you think that is?