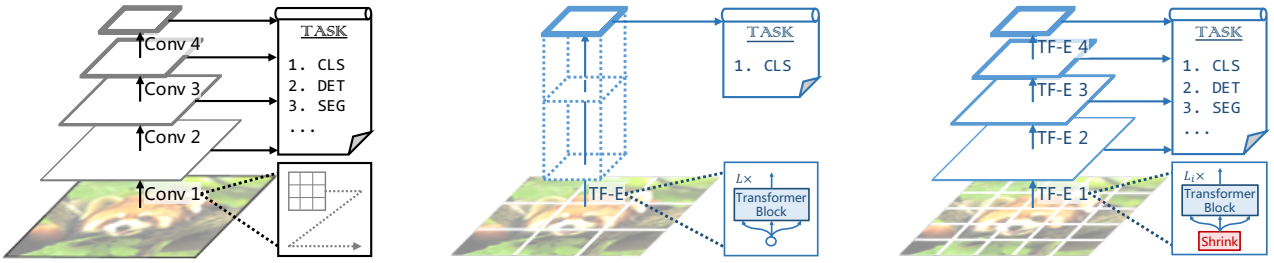
Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions

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



(a) CNNs: VGG [54], ResNet [22], *etc.* (b) Vision Transformer [13] (c) Pyramid Vision Transformer (ours)

Figure 1: **Comparisons of different architectures,** where "Conv" and "TF-E" stand for "convolution" and "Transformer encoder", respectively. (a) Many CNN backbones use a pyramid structure for dense prediction tasks such as object detection (DET), instance and semantic segmentation (SEG). (b) The recently proposed Vision Transformer (ViT) [13] is a "columnar" structure specifically designed for image classification (CLS). (c) By incorporating the pyramid structure from CNNs, we present the Pyramid Vision Transformer (PVT), which can be used as a versatile backbone for many computer vision tasks, broadening the scope and impact of ViT. Moreover, our experiments also show that PVT can easily be combined with DETR [6] to build an end-to-end object detection system without convolutions.

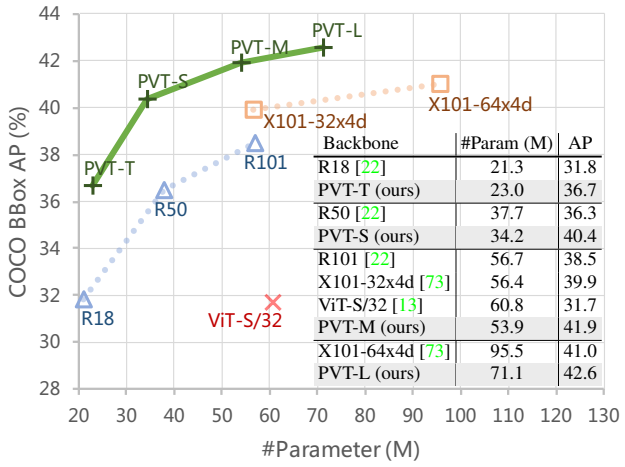
Abstract

Although convolutional neural networks (CNNs) have achieved great success in computer vision, this work investigates a simpler, convolution-free backbone network useful for many dense prediction tasks. Unlike the recently-proposed Vision Transformer (ViT) that was designed for image classification specifically, we introduce the Pyramid Vision Transformer (PVT), which overcomes the difficulties of porting Transformer to various dense prediction tasks. PVT has several merits compared to current state of the arts. (1) Different from ViT that typically yields low-resolution outputs and incurs high computational and memory costs, PVT not only can be trained on dense partitions of an image to achieve high output resolution, which is important for dense prediction, but also uses a progressive shrinking pyramid to reduce the computations of large feature maps. (2) PVT inherits the advantages of both CNN and Transformer, making it a unified backbone for various vision tasks without convolutions, where it can be used as a direct replacement for CNN backbones. (3) We validate PVT through extensive experiments, showing that it boosts the performance of many downstream tasks, including object detection, instance and semantic segmentation. For example, with a comparable number of parameters, PVT+RetinaNet achieves 40.4 AP on the COCO dataset, surpassing ResNet50+RetinNet (36.3 AP) by 4.1 absolute AP (see Figure 2). We hope that PVT could serve as an alternative and useful backbone for pixel-level predictions and facilitate future research.

✉ Corresponding authors: Deng-Ping Fan (dengpfan@gmail.com); Tong Lu (lutong@nju.edu.cn).

1. Introduction

Convolutional neural network (CNNs) have achieved remarkable success in computer vision, making them a versatile and dominant approach for almost all tasks [54, 22, 73, 49, 21, 39, 9, 32]. Nevertheless, this work aims to explore an alternative backbone network beyond CNN, which can be used for dense prediction tasks such as object detec-



Backbone	#Param (M)	AP
R18 [22]	21.3	31.8
PVT-T (ours)	23.0	36.7
R50 [22]	37.7	36.3
PVT-S (ours)	34.2	40.4
R101 [22]	56.7	38.5
X101-32x4d [73]	56.4	39.9
ViT-S/32 [13]	60.8	31.7
PVT-M (ours)	53.9	41.9
X101-64x4d [73]	95.5	41.0
PVT-L (ours)	71.1	42.6

Figure 2: **Performance comparison on COCO `val2017` of different backbones using RetinaNet for object detection**, where "T", "S", "M" and "L" denote our PVT models with tiny, small, medium and large size. We see that when the number of parameters among different models are comparable, PVT variants significantly outperform their corresponding counterparts such as ResNets (R) [22], ResNeXts (X) [73], and ViT [13].

tion [40, 14], semantic [83] and instance segmentation [40], in addition to image classification [12].

Inspired by the success of Transformer [64] in natural language processing, many researchers have explored its application in computer vision. For example, some works [6, 85, 72, 56, 24, 42] model the vision task as a dictionary lookup problem with learnable queries, and use the Transformer decoder as a task-specific head on top of the CNN backbone. Although some prior arts have also incorporated attention modules [70, 48, 80] into CNNs, as far as we know, *exploring a clean and convolution-free Transformer backbone to address dense prediction tasks in computer vision is rarely studied.*

Recently, Dosovitskiy *et al.* [13] introduced the Vision Transformer (ViT) for image classification. This is an interesting and meaningful attempt to replace the CNN backbone with a convolution-free model. As shown in Figure 1 (b), ViT has a columnar structure with coarse image patches as input.[1] Although ViT is applicable to image classification, it is challenging to directly adapt it to pixel-level dense predictions such as object detection and segmentation, because (1) its output feature map is single-scale and low-resolution, and (2) its computational and memory costs are relatively high even for common input image sizes (*e.g.*, shorter edge of 800 pixels in the COCO benchmark [40]).

To address the above limitations, this work proposes a pure Transformer backbone, termed Pyramid Vision Transformer (PVT), which can serve as an alternative to the CNN backbone in many downstream tasks, including image-level prediction as well as pixel-level dense predictions. Specifically, as illustrated in Figure 1 (c), our PVT overcomes the difficulties of the conventional Transformer by (1) taking fine-grained image patches (*i.e.*, 4×4 pixels per patch) as input to learn high-resolution representation, which is essential for dense prediction tasks; (2) introducing a progressive shrinking pyramid to reduce the sequence length of Transformer as the network deepens, significantly reducing the computational cost, and (3) adopting a spatial-reduction attention (SRA) layer to further reduce the resource consumption when learning high-resolution features.

Overall, the proposed PVT possesses the following merits. Firstly, compared to the traditional CNN backbones (see Figure 1 (a)), which have local receptive fields that increase with the network depth, our PVT always produces a global receptive field, which is more suitable for detection and segmentation. Secondly, compared to ViT (see Figure 1 (b)), thanks to its advanced pyramid structure, our method can more easily be plugged into many representative dense prediction pipelines, *e.g.*, RetinaNet [39] and Mask R-CNN [21]. Thirdly, we can build a convolution-free pipeline by combining our PVT with other task-specific Transformer decoders, such as PVT+DETR [6] for object detection. *To our knowledge, this is the first entirely convolution-free object detection pipeline.*

Our main contributions are as follows:

(1) We propose Pyramid Vision Transformer (PVT), which is the first pure Transformer backbone designed for various pixel-level dense prediction tasks. Combining our PVT and DETR, we can construct an end-to-end object detection system without convolutions and handcrafted components such as dense anchors and non-maximum suppression (NMS).

(2) We overcome many difficulties when porting Transformer to dense predictions, by designing a progressive shrinking pyramid and a spatial-reduction attention (SRA). These are able to reduce the resource consumption of Transformer, making PVT flexible to learning multi-scale and high-resolution features.

(3) We evaluate the proposed PVT on several different tasks, including image classification, object detection, instance and semantic segmentation, and compare it with popular ResNets [22] and ResNeXts [73]. As presented in Figure 2, our PVT with different parameter scales can consistently archived improved performance compared to the prior arts. For example, under a comparable number of parameters, using RetinaNet [39] for object detection, PVT-Small achieves 40.4 AP on COCO `val2017`, outper-

[1] Due to resource constraints, ViT cannot use fine-grained image patches (*e.g.*, 4×4 pixels per patch) as input, instead only receive coarse patches (*e.g.*, 32×32 pixels per patch) as input, which leads to its low output resolution (*e.g.*, 32-stride).

forming ResNet50 by 4.1 points (40.4 *vs*. 36.3). Moreover, PVT-Large achieves 42.6 AP, which is 1.6 points better than ResNeXt101-64x4d, with 30% less parameters.

2. Related Work

2.1. CNN Backbones

CNNs are the work-horses of deep neural networks in visual recognition. The standard CNN was first introduced in [34] to distinguish handwritten numbers. The model contains convolutional kernels with a certain receptive field that captures favorable visual context. To provide translation equivariance, the weights of convolutional kernels are shared over the entire image space. More recently, with the rapid development of the computational resources (*e.g.*, GPU), the successful training of stacked convolutional blocks [33, 54] on large-scale image classification datasets (*e.g.*, ImageNet [51]) has become possible. For instance, GoogLeNet [59] demonstrated that a convolutional operator containing multiple kernel paths can achieve very competitive performance. The effectiveness of a multi-path convolutional block was further validated in Inception series [60, 58], ResNeXt [73], DPN [10], MixNet [65] and SKNet [36]. Further, ResNet [22] introduced skip connections into the convolutional block, making it possible to create/train very deep networks and obtaining impressive results in the field of computer vision. DenseNet [25] introduced a densely connected topology, which connects each convolutional block to all previous blocks. More recent advances can be found in recent survey/review papers [31, 53].

Unlike the full-blown CNNs, the vision Transformer backbone is still in its early stage of development. In this work, we try to extend the scope of Vision Transformer by designing a new versatile Transformer backbone suitable for most vision tasks.

2.2. Dense Prediction Tasks

Preliminary. The dense prediction task aims to perform pixel-level classification or regression on a feature map. Object detection and semantic segmentation are two representative dense prediction tasks.

Object Detection. In the era of deep learning, CNNs [34] have become the dominant framework for object detection, which includes single-stage detectors (*e.g.*, SSD [43], RetinaNet [39], FCOS [62], GFL [37, 35], PolarMask [71] and OneNet [55]) and multi-stage detectors (Faster R-CNN [49], Mask R-CNN [21], Cascade R-CNN [4] and Sparse R-CNN [57]). Most of these popular object detectors are built on high-resolution or multi-scale feature maps to obtain good detection performance. Recently, DETR [6] and deformable DETR [85] combined the CNN backbone and the Transformer decoder to build an end-to-end object detector. Likewise, they also require high-resolution or multi-scale feature maps for accurate object detection.

Semantic Segmentation. CNNs also play an important role in semantic segmentation. In the early stages, FCN [44] introduced a fully convolutional architecture to generate a spatial segmentation map for a given image of any size. After that, the deconvolution operation was introduced by Noh *et al.* [47] and achieved impressive performance on the PASCAL VOC 2012 dataset [52]. Inspired by FCN, U-Net [50] was proposed for the medical image segmentation domain specifically, bridging the information flow between corresponding low-level and high-level feature maps of the same spatial sizes. To explore richer global context representation, Zhao *et al.* [81] designed a pyramid pooling module over various pooling scales, and Kirillov *et al.* [32] developed a lightweight segmentation head termed Semantic FPN, based on FPN [38]. Finally, the DeepLab family [8, 41] applies dilated convolutions to enlarge the receptive field while maintaining the feature map resolution. Similar to object detection methods, semantic segmentation models also rely on high-resolution or multi-scale feature maps.

2.3. Self-Attention and Transformer in Vision

As convolutional filter weights are usually fixed after training, they cannot be dynamically adapted to different inputs. Many methods have been proposed to alleviate this problem using dynamic filters [30] or self-attention operations [64]. The non-local block [70] attempts to model long-range dependencies in both space and time, which has been shown beneficial for accurate video classification. However, despite its success, the non-local operator suffers from the high computational and memory costs. Criss-cross [26] further reduces the complexity by generating sparse attention maps through a criss-cross path. Ramachandran *et al.* [48] proposed the stand-alone self-attention to replace convolutional layers with local self-attention units. AANet [3] achieves competitive results when combining the self-attention and convolutional operations. LambdaNetworks [2] uses the lambda layer, an efficient self-attention to replace the convolution in the CNN. DETR [6] utilizes the Transformer decoder to model object detection as an end-to-end dictionary lookup problem with learnable queries, successfully removing the need for handcrafted processes such as NMS. Based on DETR, deformable DETR [85] further adopts a deformable attention layer to focus on a sparse set of contextual elements, obtaining faster convergence and better performance. Recently, Vision Transformer (ViT) [13] employs a pure Transformer [64] model for image classification by treating an image as a sequence of patches. DeiT [63] further extends ViT using a novel distillation approach. Different from previous models, this work introduces the pyramid structure into Transformer to present a pure Transformer

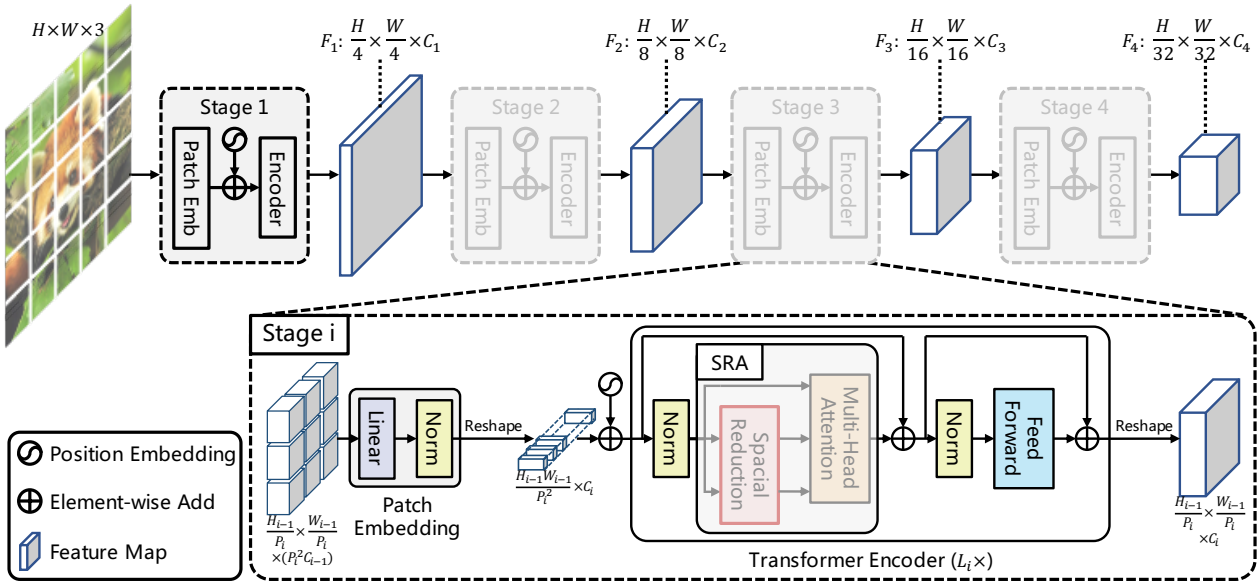


Figure 3: **Overall architecture of Pyramid Vision Transformer (PVT).** The entire model is divided into four stages, each of which is comprised of a patch embedding layer and a L_i-layer Transformer encoder. Following a pyramid structure, the output resolution of the four stages progressively shrinks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

backbone for dense prediction tasks, rather than a task-specific head or an image classification model.

3. Pyramid Vision Transformer (PVT)

3.1. Overall Architecture

Our goal is to introduce the pyramid structure into the Transformer framework, so that it can generate multi-scale feature maps for dense prediction tasks (*e.g.*, object detection and semantic segmentation). An overview of PVT is depicted in Figure 3. Similar to CNN backbones [22], our method has four stages that generate feature maps of different scales. All stages share a similar architecture, which consists of a patch embedding layer and L_i Transformer encoder layers.

In the first stage, given an input image of size $H\times W\times 3$, we first divide it into $\frac{HW}{4^2}$ patches,[2] each of size $4\times 4\times 3$. Then, we feed the flattened patches to a linear projection and obtain embedded patches of size $\frac{HW}{4^2}\times C_1$. After that, the embedded patches along with a position embedding are passed through a Transformer encoder with L_1 layers, and the output is reshaped to a feature map F_1 of size $\frac{H}{4}\times\frac{W}{4}\times C_1$. In the same way, using the feature map from the previous stage as input, we obtain the following feature maps: F_2, F_3, and F_4, whose strides are 8, 16, and 32 pixels with respect to the input image. With the feature pyramid $\{F_1, F_2, F_3, F_4\}$, our method can be easily applied to most downstream tasks, including image classification, object detection, and semantic segmentation.

[2] As done for ResNet, we keep the highest resolution of our output feature map at 4-stride.

3.2. Feature Pyramid for Transformer

Unlike CNN backbone networks [54, 22], which use different convolutional strides to obtain multi-scale feature maps, our PVT uses a *progressive shrinking strategy* to control the scale of feature maps by patch embedding layers.

Here, we denote the patch size of the i-th stage as P_i. At the beginning of stage i, we first evenly divide the input feature map $F_{i-1}\in\mathbb{R}^{H_{i-1}\times W_{i-1}\times C_{i-1}}$ into $\frac{H_{i-1}W_{i-1}}{P_i^2}$ patches, and then each patch is flatten and projected to a C_i-dimensional embedding. After the linear projection, the shape of the embedded patches can be viewed as $\frac{H_{i-1}}{P_i}\times\frac{W_{i-1}}{P_i}\times C_i$, where the height and width are P_i times smaller than the input.

In this way, we can flexibly adjust the scale of the feature map in each stage, making it possible to construct a feature pyramid for Transformer.

3.3. Transformer Encoder

The Transformer encoder in the stage i has L_i encoder layers, each of which is composed of an attention layer and a feed-forward layer [64]. Since PVT needs to process high-resolution (*e.g.*, 4-stride) feature maps, we propose a spatial-reduction attention (SRA) layer to replace the traditional multi-head attention (MHA) layer [64] in the encoder.

Similar to MHA, our SRA receives a query Q, a key K, and a value V as input, and outputs a refined feature. The difference is that our SRA reduces the spatial scale of K

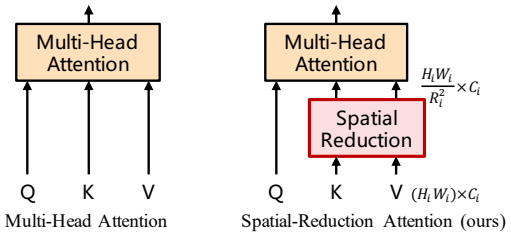


Figure 4: **Multi-head attention (MHA) *vs*. spatial-reduction attention (SRA).** With the spatial-reduction operation, the computational/memory cost of our SRA is much lower than that of MHA.

and V before the attention operation (see Figure 4), which largely reduces the computational/memory overhead. Details of the SRA in the stage i can be formulated as follows:

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

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

where $\mathrm{Concat}(\cdot)$ is the concatenation operation as in [64]. $W_j^Q\in\mathbb{R}^{C_i\times d_{\mathrm{head}}}$, $W_j^K\in\mathbb{R}^{C_i\times d_{\mathrm{head}}}$, $W_j^V\in\mathbb{R}^{C_i\times d_{\mathrm{head}}}$, and $W^O\in\mathbb{R}^{C_i\times C_i}$ are linear projection parameters. N_i is the head number of the attention layer in Stage i. Therefore, the dimension of each head (*i.e.*, d_{head}) is equal to $\frac{C_i}{N_i}$. $\mathrm{SR}(\cdot)$ is the operation for reducing the spatial dimension of the input sequence (*i.e.*, K or V), which is written as:

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

Here, $\mathbf{x}\in\mathbb{R}^{(H_iW_i)\times C_i}$ represents a input sequence, and R_i denotes the reduction ratio of the attention layers in Stage i. $\mathrm{Reshape}(\mathbf{x},R_i)$ is an operation of reshaping the input sequence $\mathbf{x}$ to a sequence of size $\frac{H_iW_i}{R_i^2}\times(R_i^2C_i)$. $W_S\in\mathbb{R}^{(R_i^2C_i)\times C_i}$ is a linear projection that reduces the dimension of the input sequence to C_i. $\mathrm{Norm}(\cdot)$ refers to layer normalization [1]. As in the original Transformer [64], our attention operation $\mathrm{Attention}(\cdot)$ is calculated as:

$$\mathrm{Attention}(\mathbf{q},\mathbf{k},\mathbf{v})=\mathrm{Softmax}(\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{d_{\mathrm{head}}}})\mathbf{v}. \quad (4)$$

Through these formulas, we can find that the computational/memory costs of our attention operation are R_i^2 times lower than those of MHA, so our SRA can handle larger input feature maps/sequences with limited resources.

3.4. Model Details

In summary, the hyper parameters of our method are listed as follows:

- P_i: the patch size of Stage i;
- C_i: the channel number of the output of Stage i;
- L_i: the number of encoder layers in Stage i;
- R_i: the reduction ratio of the SRA in Stage i;
- N_i: the head number of the SRA in Stage i;
- E_i: the expansion ratio of the feed-forward layer [64] in Stage i;

Following the design rules of ResNet [22], we (1) use small output channel numbers in shallow stages; and (2) concentrate the major computation resource in intermediate stages.

To provide instances for discussion, we describe a series of PVT models with different scales, namely PVT-Tiny, -Small, -Medium, and -Large, in Table 1, whose parameter numbers are comparable to ResNet18, 50, 101, and 152 respectively. More details of employing these models in specific downstream tasks will be introduced in Section 4.

3.5. Discussion

The most related work to our model is ViT [13]. Here, we discuss the relationship and differences between them. First, both PVT and ViT are pure Transformer models without convolutions. The primary difference between them is the pyramid structure. Similar to the traditional Transformer [64], the length of ViT's output sequence is the same as the input, which means that the output of ViT is single-scale (see Figure 1 (b)). Moreover, due to the limited resource, the input of ViT is coarse-grained (*e.g.*, the patch size is 16 or 32 pixels), and thus its output resolution is relatively low (*e.g.*, 16-stride or 32-stride). As a result, it is difficult to directly apply ViT to dense prediction tasks that require high-resolution or multi-scale feature maps.

Our PVT breaks the routine of Transformer by introducing a progressive shrinking pyramid. It can generate multi-scale feature maps like a traditional CNN backbone. In addition, we also designed a simple but effective attention layer—SRA, to process high-resolution feature maps and reduce computational/memory costs. Benefiting from the above designs, our method has the following advantages over ViT: 1) more flexible—can generate feature maps of different scales/channels in different stages; 2) more versatile—can be easily plugged and played in most downstream task models; 3) more friendly to computation/memory—can handle higher resolution feature maps or longer sequences.

4. Application to Downstream Tasks

4.1. Image-Level Prediction

Image classification is the most classical task of image-level prediction. To provide instances for discussion, we design a series of PVT models with different scales, namely PVT-Tiny, -Small, -Medium, and -Large, whose parameter numbers are similar to ResNet18, 50, 101, and 152, respec-

	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 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.

tively. Detailed hyper-parameter settings of the PVT series are provided in the *supplementary material (SM)*.

For image classification, we follow ViT [13] and DeiT [63] to append a learnable classification token to the input of the last stage, and then employ a fully connected (FC) layer to conduct classification on top of the token.

4.2. Pixel-Level Dense Prediction

In addition to image-level prediction, dense prediction that requires pixel-level classification or regression to be performed on the feature map, is also often seen in downstream tasks. Here, we discuss two typical tasks, namely object detection, and semantic segmentation.

We apply our PVT models to three representative dense prediction methods, namely RetinaNet [39], Mask R-CNN [21], and Semantic FPN [32]. RetinaNet is a widely used single-stage detector, Mask R-CNN is the most popular two-stage instance segmentation framework, and Semantic FPN is a vanilla semantic segmentation method without special operations (*e.g.*, dilated convolution). Using these methods as baselines enables us to adequately examine the effectiveness of different backbones.

The implementation details are as follows: (1) Like ResNet, we initialize the PVT backbone with the weights pre-trained on ImageNet; (2) We use the output feature pyramid $\{F_1, F_2, F_3, F_4\}$ as the input of FPN [38], and then the refined feature maps are fed to the follow-up detection/segmentation head; (3) When training the detection/segmentation model, none of the layers in PVT are frozen; (4) Since the input for detection/segmentation can be an arbitrary shape, the position embeddings pre-trained on ImageNet may no longer be meaningful. Therefore, we perform bilinear interpolation on the pre-trained position embeddings according to the input resolution.

5. Experiments

We compare PVT with the two most representative CNN backbones, *i.e.*, ResNet [22] and ResNeXt [73], which are widely used in the benchmarks of many downstream tasks.

5.1. Image Classification

Settings. Image classification experiments are performed on the ImageNet 2012 dataset [51], which comprises 1.28 million training images and 50K validation images from 1,000 categories. For fair comparison, all models are trained on the training set, and report the top-1 error on the validation set. We follow DeiT [63] and apply random cropping, random horizontal flipping [59], label-smoothing regularization [60], mixup [78], CutMix [76], and random erasing [82] as data augmentations. During training, we employ AdamW [46] with a momentum of 0.9, a mini-batch size of 128, and a weight decay of 5×10^{-2} to optimize models. The initial learning rate is set to 1×10^{-3} and decreases following the cosine schedule [45]. All models are trained for 300 epochs from scratch on 8 V100 GPUs. To benchmark, we apply a center crop on the validation set, where a 224$\times$ 224 patch is cropped to evaluate the classification accuracy.
Results. In Table 2, we see that our PVT models are superior to conventional CNN backbones under similar parameter numbers and computational budgets. For example, when

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

Table 2: **Image classification performance on the ImageNet validation set**. "#Param" refers to the number of parameters. "GFLOPs" is calculated under the input scale of 224 $\times$ 224. "*" indicates the performance of the method trained under the strategy of its original paper.

the GFLOPs are roughly similar, the top-1 error of PVT-Small reaches 20.2, which is 1.3 points higher than that of ResNet50 [22] (20.2 *vs*. 21.5). Meanwhile, under similar or lower complexity, PVT models archive performances comparable to the recently proposed Transformer-based models, such as ViT [13] and DeiT [63] (PVT-Large: 18.3 *vs*. ViT(DeiT)-Base/16: 18.3). Here, we clarify that these results are within our expectations, because the pyramid structure is beneficial to dense prediction tasks, but brings little improvements to image classification.

Note that ViT and DeiT have limitations as they are specifically designed for classification tasks, and thus are not suitable for dense prediction tasks, which usually require effective feature pyramids.

5.2. Object Detection

Settings. Object detection experiments are conducted on the challenging COCO benchmark [40]. All models are trained on COCO `train2017` (118k images) and evaluated on `val2017` (5k images). We verify the effectiveness of PVT backbones on top of two standard detectors, namely RetinaNet [39] and Mask R-CNN [21]. Before training, we use the weights pre-trained on ImageNet to initialize the backbone and Xavier [18] to initialize the newly added layers. Our models are trained with a batch size of 16 on 8 V100 GPUs and optimized by AdamW [46] with an initial learning rate of 1×10^{-4}. Following common practices [39, 21, 7], we adopt $1\times$ or $3\times$ training schedule (*i.e.*, 12 or 36 epochs) to train all detection models. The training image is resized to have a shorter side of 800 pixels, while the longer side does not exceed 1,333 pixels. When using the $3\times$ training schedule, we randomly resize the shorter side of the input image within the range of $[640, 800]$. In the testing phase, the shorter side of the input image is fixed to 800 pixels.

Results. As shown in Table 3, when using RetinaNet for object detection, we find that under comparable number of parameters, the PVT-based models significantly surpasses their counterparts. For example, with the $1\times$ training schedule, the AP of PVT-Tiny is 4.9 points better than that of ResNet18 (36.7 *vs*. 31.8). Moreover, with the $3\times$ training schedule and multi-scale training, PVT-Large archive the best AP of 43.4, surpassing ResNeXt101-64x4d (43.4 *vs*. 41.8), while our parameter number is 30% fewer. These results indicate that our PVT can be a good alternative to the CNN backbone for object detection.

Similar results are found in instance segmentation experiments based on Mask R-CNN, as shown in Table 4. With the $1\times$ training schedule, PVT-Tiny achieves 35.1 mask AP (AP^m), which is 3.9 points better than ResNet18 (35.1 *vs*. 31.2) and even 0.7 points higher than ResNet50 (35.1 *vs*. 34.4). The best AP^m obtained by PVT-Large is 40.7, which is 1.0 points higher than ResNeXt101-64x4d (40.7 *vs*. 39.7), with 20% fewer parameters.

5.3. Semantic Segmentation

Settings. We choose ADE20K [83], a challenging scene parsing dataset, to benchmark the performance of semantic segmentation. ADE20K contains 150 fine-grained semantic categories, with 20,210, 2,000, and 3,352 images for training, validation, and testing, respectively. We evaluate our PVT backbones on the basis of Semantic FPN [32], a simple segmentation method without dilated convolutions [74]. In the training phase, the backbone is initialized with the weights pre-trained on ImageNet [12], and other newly added layers are initialized with Xavier [18]. We optimize our models using AdamW [46] with an initial learning rate of 1e-4. Following common practices [32, 8], we train our models for 80k iterations with a batch size of 16 on 4 V100 GPUs. The learning rate is decayed following the polynomial decay schedule with a power of 0.9. We randomly resize and crop the image to 512 $\times$ 512 for training, and rescale to have a shorter side of 512 pixels during testing.

Results. As shown in Table 5, when using Semantic FPN [32] for semantic segmentation, PVT-based models consistently outperforms the models based on ResNet [22] or ResNeXt [73]. For example, with al-

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.

.

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.

most the same number of parameters and GFLOPs, our PVT-Tiny/Small/Medium are at least 2.8 points higher than ResNet-18/50/101. In addition, although the parameter number and GFLOPs of our PVT-Large are 20% lower than those of ResNeXt101-64x4d, the mIoU is still 1.9 points higher (42.1 *vs*. 40.2). With a longer training schedule and multi-scale testing, PVT-Large+Semantic FPN archives the best mIoU of 44.8, which is very close to the state-of-the-art performance of the ADE20K benchmark. Note that Semantic FPN is just a simple segmentation head. These results demonstrate that our PVT backbones can extract better features for semantic segmentation than the CNN backbone, benefiting from the global attention mechanism.

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].

5.4. Pure Transformer Detection & Segmentation

PVT+DETR. To reach the limit of no convolution, we build a pure Transformer pipeline for object detection by simply combining our PVT with a Transformer-based detection head—DETR [6]. We train models on COCO `train2017` for 50 epochs with an initial learning rate of 1×10^{-4}. The learning rate is divided by 10 at the 33rd epoch. We use random flipping and multi-scale training as data augmentation. All other experimental settings is the same as those in Sec. 5.2. As reported in Table 6, PVT-based DETR archieves 34.7 AP on COCO `val2017`, outperforming the original ResNet50-based DETR by 2.4 points (34.7 *vs*. 32.3). These results prove that *a pure Transformer detector can also works well in the object detection task*.

PVT+Trans2Seg. We build a pure Transformer model for semantic segmentation by combining our PVT with

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.

Method	#Param (M)	RetinaNet 1x					
		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ViT-Small/4 [13]	60.9	Out of Memory					
ViT-Small/32 [13]	60.8	31.7	51.3	32.3	14.8	33.7	47.9
PVT-Small (ours)	34.2	40.4	61.3	43.0	25.0	42.9	55.7

Table 8: **Performance comparison between ViT and our PVT using RetinaNet for object detection.** ViT-Small/4 runs out of GPU memory due to small patch size (*i.e.*, 4×4 per patch). ViT-Small/32 obtains 31.7 AP on COCO `val2017`, which is 8.7 points lower than our PVT-Small.

Trans2Seg [72], a Transformer-based segmentation head. According to the experimental settings in Sec. 5.3, we perform experiments on ADE20K [83] with 40k iterations training, single scale testing, and compare it with ResNet50+Trans2Seg [72] and DeeplabV3+ [9] with ResNet50-d8 (dilation 8) and -d16(dilation 8) in Table 7. We find that our PVT-Small+Trans2Seg achieves 42.6 mIoU, outperforming ResNet50-d8+DeeplabV3+ (41.5). Note that, ResNet50-d8+DeeplabV3+ has 120.5 GFLOPs due to the high computation cost of dilated convolution, and our method has only 31.6 GFLOPs, which is 4 times fewer. In addition, our PVT-Small+Trans2Seg performs better than ResNet50-d16+Trans2Seg (mIoU: 42.6 *vs*. 39.7, GFlops: 31.6 *vs*. 79.3). These results prove that *a pure Transformer segmentation network is workable.*

5.5. Ablation Study

Settings. We conduct ablation studies on ImageNet [12] and COCO [40] datasets. The experimental settings on ImageNet are the same as the settings in Sec. 5.1. For COCO, all models are trained with a $1\times$ training schedule (*i.e.*, 12 epochs) and without multi-scale training, and other settings follow those in Sec. 5.2.

Pyramid Structure. A Pyramid structure is crucial when applying Transformer to dense prediction tasks. ViT (see Figure 1 (b)) is a columnar framework, whose output is single-scale. This results in a low-resolution output feature map when using coarse image patches (*e.g.*, 32×32 pixels per patch) as input, leading to poor detection performance (31.7 AP on COCO `val2017`),[3] as shown in Table 8. When using fine-grained image patches (*e.g.*, 4×4 pixels per patch) as input like our PVT, ViT will exhaust the GPU memory (32G). Our method avoids this problem through a progressive shrinking pyramid. Specifically, our model can process high-resolution feature maps in shallow stages and low-resolution feature maps in deep stages. Thus, it obtains a promising AP of 40.4 on COCO `val2017`, 8.7 points higher than ViT-Small/32 (40.4 *vs*. 31.7).

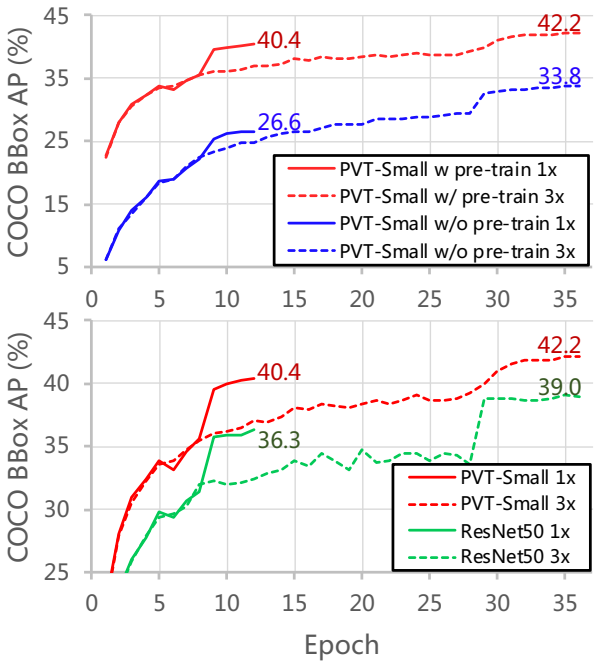


Figure 5: **AP curves of RetinaNet on COCO `val2017` under different backbone settings.** Top: using weights pre-trained on ImageNet *vs*. random initialization. Bottom: PVT-S *vs*. R50 [22].

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 9: **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 deep model (*i.e.*, PVT-Medium) obtains better performance than the wide model (*i.e.*, PVT-Small-Wide) under comparable parameter number.

Deeper *vs*. Wider. The problem of whether the CNN backbone should go deeper or wider has been extensively discussed in previous work [22, 77]. Here, we explore this

[3] For adapting ViT to RetinaNet, we extract the features from the layer 2, 4, 6, and 8 of ViT-Small/32, and interpolate them to different scales.

Method	#Param (M)	GFLOPs	Mask R-CNN 1x		
			AP^m	AP^m_{50}	AP^m_{75}
ResNet50+GC r4 [5]	54.2	279.6	36.2	58.7	38.3
PVT-Small (ours)	44.1	304.4	37.8	60.1	40.3

Table 10: **PVT *vs*. CNN w/ non-local.** AP^m denotes mask AP. Under similar parameter nubmer and GFLOPs, our PVT outperform the CNN backbone w/ Non-Local (ResNet50+GC r4) by 1.6 AP^m (37.8 *vs*. 36.2).

problem in our PVT. For fair comparisons, we multiply the hidden dimensions $\{C_1, C_2, C_3, C_4\}$ of PVT-Small by a scale factor 1.4 to make it have an equivalent parameter number to the deep model (*i.e.*, PVT-Medium). As shown in Table 9, the deep model (*i.e.*, PVT-Medium) consistently works better than the wide model (*i.e.*, PVT-Small-Wide) on both ImageNet and COCO. Therefore, going deeper is more effective than going wider in the design of PVT. Based on this observation, in Table 1, we develop PVT models with different scales by increasing the model depth.

Pre-trained Weights. Most dense prediction models (*e.g.*, RetinaNet [39]) rely on the backbone whose weights are pre-trained on ImageNet. We also discuss this problem in our PVT. In the top of Figure 5, we plot the validation AP curves of RetinaNet-PVT-Small w/ (red curves) and w/o (blue curves) pre-trained weights. We find that the model w/ pre-trained weights converges better than the one w/o pre-trained weights, and the gap between their final AP reaches 13.8 under the $1\times$ training schedule and 8.4 under the $3\times$ training schedule and multi-scale training. Therefore, like CNN-based models, pre-training weights can also help PVT-based models converge faster and better. Moreover, in the bottom of Figure 5, we also see that the convergence speed of PVT-based models (red curves) is faster than that of ResNet-based models (green curves).

PVT *vs*. "CNN w/ Non-Local" To obtain a global receptive field, some well-engineered CNN backbones, such as GCNet [5], integrate the non-local block in the CNN framework. Here, we compare the performance of our PVT (pure Transformer) and GCNet (CNN w/ non-local), using Mask R-CNN for instance segmentation. As reported in Table 10, we find that our PVT-Small outperforms ResNet50+GC r4 [5] by 1.6 points in AP^m (37.8 *vs*. 36.2), and 2.0 points in AP^m_{75} (38.3 *vs*. 40.3), under comparable parameter number and GFLOPs. There are two possible reasons for this result:

(1) Although a single global attention layer (*e.g.*, non-local [70] or multi-head attention (MHA) [64]) can acquire global-receptive-field features, the model performance keeps improving as the model deepens. This indicates that *stacking multiple MHAs can further enhance the representation capabilities of features.* Therefore, as a pure Transformer backbone with more global attention layers, our PVT tends to perform better than the CNN backbone equipped with non-local blocks (*e.g.*, GCNet).

(2) Regular convolutions can be deemed as special instantiations of spatial attention mechanisms [84]. In other words, the format of MHA is more flexible than the regular convolution. For example, for different inputs, the weights of the convolution are fixed, but the attention weights of MHA change dynamically with the input. Thus, *the features learned by the pure Transformer backbone full of MHA layers, could be more flexible and expressive.*

Method	Scale	GFLOPs	Time (ms)	RetinaNet 1x		
				AP	AP_{50}	AP_{75}
ResNet50 [22]	800	239.3	55.9	36.3	55.3	38.6
PVT-Small (ours)	640	157.2	51.7	38.7	59.3	40.8
	800	285.8	76.9	40.4	61.3	43.0

Table 11: **Latency and AP under different input scales.** "Scale" and "Time" denote the input scale and time cost per image. When the shorter side is 640 pixels, the PVT-Small+RetinaNet has a lower GFLOPs and time cost (on a V100 GPU) than ResNet50+RetinaNet, while obtaining 2.4 points better AP (38.7 *vs*. 36.3).

Computation Overhead. With increasing input scale, the growth rate of the GFLOPs of our PVT is greater than ResNet [22], but lower than ViT [13], as shown in Figure 6. However, when the input scale does not exceed 640×640 pixels, the GFLOPs of PVT-Small and ResNet50 are similar. This means that our PVT is more suitable for tasks with medium-resolution input.

On COCO, the shorter side of the input image is 800 pixels. Under this condition, the inference speed of RetinaNet based on PVT-Small is slower than the ResNet50-based model, as reported in Table 11. (1) *A direct solution for this problem is to reduce the input scale.* When reducing the shorter side of the input image to 640 pixels, the model based on PVT-Small runs faster than the ResNet50-based model (51.7ms *vs*., 55.9ms), with 2.4 higher AP (38.7 *vs*. 36.3). 2) *Another solution is to develop a self-attention layer with lower computational complexity.* This is a worth exploring direction, we recently propose a solution PVTv2 [67].

Detection & Segmentation Results. In Figure 7, we also present some qualitative object detection and instance segmentation results on COCO `val2017` [40], and semantic segmentation results on ADE20K [83]. These results indicate that a pure Transformer backbone (*i.e.*, PVT) without convolutions can also be easily plugged in dense prediction models (*e.g.*, RetinaNet [39], Mask R-CNN [21], and Semantic FPN [32]), and obtain high-quality results.

6. Conclusions and Future Work

We introduce PVT, a pure Transformer backbone for dense prediction tasks, such as object detection and seman-

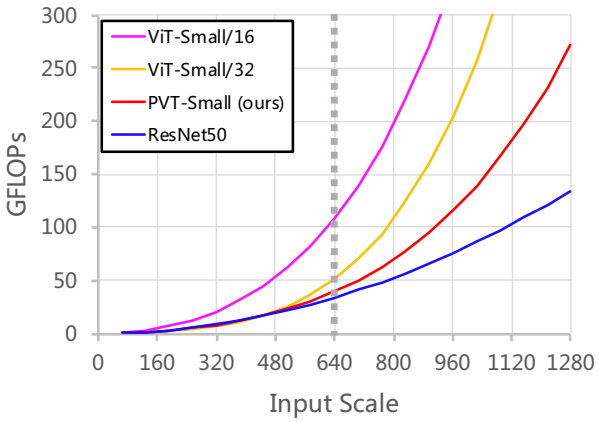


Figure 6: **Models' GFLOPs under different input scales.** The growth rate of GFLOPs: ViT-Small/16 [13]>ViT-Small/32 [13]>PVT-Small (ours)>ResNet50 [22]. When the input scale is less than 640×640, the GFLOPs of PVT-Small and ResNet50 [22] are similar.

tic segmentation. We develop a progressive shrinking pyramid and a spatial-reduction attention layer to obtain high-resolution and multi-scale feature maps under limited computation/memory resources. Extensive experiments on object detection and semantic segmentation benchmarks verify that our PVT is stronger than well-designed CNN backbones under comparable numbers of parameters.

Although PVT can serve as an alternative to CNN backbones (*e.g.*, ResNet, ResNeXt), there are still some specific modules and operations designed for CNNs and not considered in this work, such as SE [23], SK [36], dilated convolution [74], model pruning [20], and NAS [61]. Moreover, with years of rapid developments, there have been many well-engineered CNN backbones such as Res2Net [17], EfficientNet [61], and ResNeSt [79]. In contrast, the Transformer-based model in computer vision is still in its early stage of development. Therefore, we believe there are many potential technologies and applications (*e.g.*, OCR [68, 66, 69], 3D [28, 11, 27] and medical [15, 16, 29] image analysis) to be explored in the future, and hope that PVT could serve as a good starting point.

Acknowledgments

This work was supported by the Natural Science Foundation of China under Grant 61672273 and Grant 61832008, the Science Foundation for Distinguished Young Scholars of Jiangsu under Grant BK20160021, Postdoctoral Innovative Talent Support Program of China under Grant BX20200168, 2020M681608, the General Research Fund of Hong Kong No. 27208720.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[2] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *Proc. Int. Conf. Learn. Representations*, 2021. 3

[3] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2019. 3

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 3

[5] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2019. 10

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proc. Eur. Conf. Comp. Vis.*, 2020. 1, 2, 3, 8

[7] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 7

[8] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2017. 3, 7

[9] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. Eur. Conf. Comp. Vis.*, 2018. 1, 9

[10] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. *Proc. Advances in Neural Inf. Process. Syst.*, 2017. 3

[11] Mingmei Cheng, Le Hui, Jin Xie, and Jian Yang. SSPC-Net: Semi-supervised semantic 3D point cloud segmentation network. In *Proc. AAAI Conf. Artificial Intell.*, 2021. 11

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009. 2, 7, 9

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *Proc. Int. Conf. Learn. Representations*, 2021. 1, 2, 3, 5, 6, 7, 9, 10, 11

[14] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, 2010. 2

Figure 7: **Qualitative results of object detection and instance segmentation on COCO `val2017` [40], and semantic segmentation on ADE20K [83].** The results (from left to right) are generated by PVT-Small-based RetinaNet [39], Mask R-CNN [21], and Semantic FPN [32], respectively.

[15] Deng-Ping Fan, Ge-Peng Ji, Ming-Ming Cheng, and Ling Shao. Concealed object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021. 11

[16] Deng-Ping Fan, Ge-Peng Ji, Tao Zhou, Geng Chen, Huazhu Fu, Jianbing Shen, and Ling Shao. Pranet: Parallel reverse attention network for polyp segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2020. 11

[17] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr. Res2net: A new multi-scale backbone architecture. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019. 11

[18] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. Int. Conf. Artificial Intell. & Stat.*, 2010. 7

[19] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 7

[20] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 11

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017. 1, 2, 3, 6, 7, 8, 10, 12

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11

[23] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 11

[24] Ronghang Hu and Amanpreet Singh. Transformer is all you need: Multimodal multitask learning with a unified trans-

former. *arXiv preprint arXiv:2102.10772*, 2211. 2

[25] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017. 3

[26] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2019. 3

[27] Le Hui, Mingmei Cheng, Jin Xie, and Jian Yang. Efficient 3D point cloud feature learning for large-scale place recognition. *arXiv preprint arXiv:2101.02374*, 2021. 11

[28] Le Hui, Rui Xu, Jin Xie, Jianjun Qian, and Jian Yang. Progressive point cloud deconvolution generation network. In *Proc. Eur. Conf. Comp. Vis.*, 2020. 11

[29] Ge-Peng Ji, Yu-Cheng Chou, Deng-Ping Fan, Geng Chen, Huazhu Fu, Debesh Jha, and Ling Shao. Progressively normalized self-attention network for video polyp segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2021. 11

[30] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Proc. Advances in Neural Inf. Process. Syst.*, 2016. 3

[31] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, 2020. 3

[32] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. 1, 3, 6, 7, 10, 12

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Proc. Advances in Neural Inf. Process. Syst.*, 2012. 3

[34] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. 1998. 3

[35] Xiang Li, Wenhai Wang, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2021. 3

[36] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. 3, 11

[37] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. In *Proc. Advances in Neural Inf. Process. Syst.*, 2020. 3

[38] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017. 3, 6

[39] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017. 1, 2, 3, 6, 7, 8, 10, 12

[40] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. Eur. Conf. Comp. Vis.*, 2014. 2, 7, 9, 10, 12

[41] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. 3

[42] Nian Liu, Ni Zhang, Kaiyuan Wan, Ling Shao, and Junwei Han. Visual saliency transformer. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2021. 2

[43] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proc. Eur. Conf. Comp. Vis.*, 2016. 3

[44] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015. 3

[45] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *Proc. Int. Conf. Learn. Representations*, 2017. 6

[46] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proc. Int. Conf. Learn. Representations*, 2019. 6, 7

[47] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015. 3

[48] Niki Parmar, Prajit Ramachandran, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Proc. Advances in Neural Inf. Process. Syst.*, 2019. 2, 3

[49] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Advances in Neural Inf. Process. Syst.*, 2015. 1, 3

[50] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015. 3

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 2015. 3, 6

[52] Suyash Shetty. Application of convolutional neural network for image classification on pascal voc challenge 2012 dataset. *arXiv preprint arXiv:1607.03785*, 2016. 3

[53] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019. 3

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *Proc. Int. Conf. Learn. Representations*, 2015. 1, 3, 4

[55] Peize Sun, Yi Jiang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Onenet: Towards end-to-end one-stage object detection. *arXiv preprint arXiv:2012.05780*, 2020. 3

[56] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple-object tracking with transformer. *arXiv preprint arXiv:2012.15460*, 2020. 2

[57] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2021. 3

[58] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proc. AAAI Conf. Artificial Intell.*, 2017. 3

[59] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015. 3, 6

[60] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016. 3, 6

[61] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proc. Int. Conf. Mach. Learn.*, 2019. 11

[62] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2019. 3

[63] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Proc. Int. Conf. Mach. Learn.*, 2021. 3, 6, 7

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. Advances in Neural Inf. Process. Syst.*, 2017. 2, 3, 4, 5, 10

[65] Wenhai Wang, Xiang Li, Jian Yang, and Tong Lu. Mixed link networks. *Proc. Int. Joint Conf. Artificial Intell.*, 2018. 3

[66] Wenhai Wang, Xuebo Liu, Xiaozhong Ji, Enze Xie, Ding Liang, ZhiBo Yang, Tong Lu, Chunhua Shen, and Ping Luo. Ae textspotter: Learning visual and linguistic representation for ambiguous text spotting. In *Proc. Eur. Conf. Comp. Vis.*, 2020. 11

[67] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *arXiv preprint arXiv:2106.13797*, 2021. 10

[68] Wenhai Wang, Enze Xie, Xiang Li, Wenbo Hou, Tong Lu, Gang Yu, and Shuai Shao. Shape robust text detection with progressive scale expansion network. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. 11

[69] Wenhai Wang, Enze Xie, Xiang Li, Xuebo Liu, Ding Liang, Yang Zhibo, Tong Lu, and Chunhua Shen. Pan++: Towards efficient and accurate end-to-end spotting of arbitrarily-shaped text. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021. 11

[70] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 2, 3, 10

[71] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. Polarmask: Single shot instance segmentation with polar representation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020. 3

[72] Enze Xie, Wenjia Wang, Wenhai Wang, Peize Sun, Hang Xu, Ding Liang, and Ping Luo. Segmenting transparent object in the wild with transformer. In *Proc. Int. Joint Conf. Artificial Intell.*, 2021. 2, 9

[73] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017. 1, 2, 3, 6, 7, 8

[74] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In Yoshua Bengio and Yann LeCun, editors, *Proc. Int. Conf. Learn. Representations*, 2016. 7, 11

[75] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 7

[76] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6023–6032, 2019. 6

[77] Erwan Zerhouni, Dávid Lányi, Matheus Viana, and Maria Gabrani. Wide residual networks for mitosis detection. In *IEEE International Symposium on Biomedical Imaging*, 2017. 9

[78] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. Int. Conf. Learn. Representations*, 2018. 6

[79] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020. 11

[80] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020. 2

[81] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017. 3

[82] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proc. AAAI Conf. Artificial Intell.*, 2020. 6

[83] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017. 2, 7, 9, 10, 12

[84] Xizhou Zhu, Dazhi Cheng, Zheng Zhang, Stephen Lin, and Jifeng Dai. An empirical study of spatial attention mechanisms in deep networks. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2019. 10

[85] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *Proc. Int. Conf. Learn. Representations*, 2021. 2, 3