PVTv2: Improved Baselines with Pyramid Vision Transformer

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,

Abstract

Transformer in computer vision has recently shown encouraging progress. In this work, we improve the original Pyramid Vision Transformer (PVTv1) by adding three improvement designs, which include (1) overlapping patch embedding, (2) convolutional feed-forward networks, and (3) linear complexity attention layers.

With these simple modifications, our PVTv2 significantly improves PVTv1 on classification, detection, and segmentation. Moreover, PVTv2 achieves better performance than recent works, including Swin Transformer. We hope this work will make state-of-the-art vision Transformer research more accessible. Code is available at https://github.com/whai362/PVT.

1. Introduction

Recent studies on vision Transformer are converging on the backbone network [7, 29, 31, 32, 21, 34, 9, 4] designed for downstream vision tasks, such as image classification, object detection, instance and semantic segmentation. To date, there have been some promising results. For example, Vision Transformer (ViT) [7] first proves that a pure Transformer can archive state-of-the-art performance in image classification. Pyramid Vision Transformer (PVT) [31] shows that a pure Transformer backbone can also surpass CNN counterparts in several detection and segmentation tasks. After that, Swin Transformer [21], CoaT [34], LeViT [9], and Twins [4] further improve the classification, detection, and segmentation performance with Transformer backbones.

This work aims to establish stronger and more feasible baselines built on the PVTv1 framework. We report that three design improvements, namely (1) overlapping patch embedding, (2) convolutional feed-forward networks, and (3) linear complexity attention layers are orthogonal to the PVTv1 framework, and when used with PVT, they can bring better image classification, object detection, instance and semantic segmentation performance. Specifically, PVTv2-B5[1] yields 83.8% top-1 error on ImageNet, which is significantly better than Swin-B [21] and Twins-SVT-L [4], while PVTv2-B5 has fewer parameters and GFLOPS. Moreover, GFL [17] with PVT-B2 archives 50.2 AP on COCO `val2017`, 2.6 AP higher than the one with Swin-T [21], 5.7 AP higher than the one with ResNet50 [12]. We hope these improved baselines will provide a reference for future research in vision Transformer.

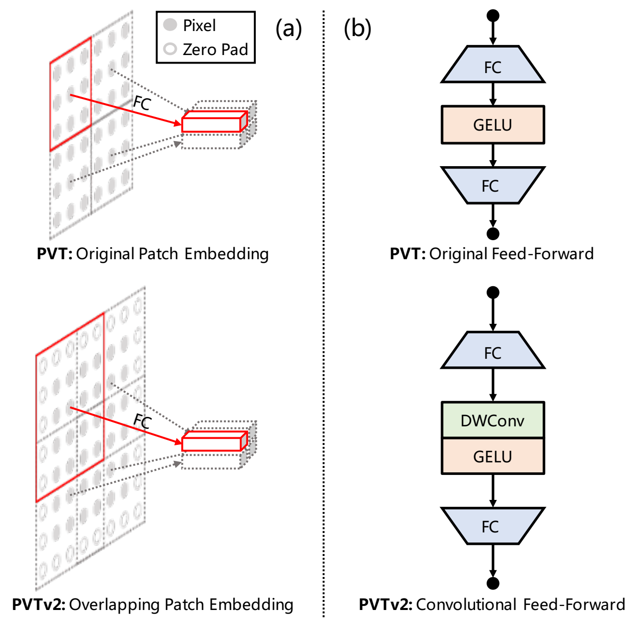

Figure 1: **Two improvements in PVTv2.** (1) Overlapping Patch Embedding. (2) Convolutional Feed Forward Network.

[1]PVTv2 has 6 different size variants, from B0 to B5 according to the parameter number.

2. Related Work

Transformer Backbones. ViT [7] treats each image as a sequence of tokens (patches) with fixed length, and then feed them to multiple Transformer layers to make classification. It is the first work to prove that a pure Transformer can also archive state-of-the-art performance in image classification when training data is sufficient (*e.g.*, ImageNet-22k [6], JFT-300M). DeiT [29] further explores a data-efficient training strategy and a distillation approach for ViT.

To improve image classification performance, some recent methods make tailored changes to ViT. T2T ViT [35] concatenates tokens within an overlapping sliding window into one token progressively. TNT [10] utilizes inner and outer Transformer blocks to generate pixel embeddings and patch embeddings respectively. CPVT [5] replaces the fixed size position embedding in ViT with conditional position encodings, making it easier to process images of arbitrary resolution. CrossViT [2] processes image patches of different sizes via a dual-branch Transformer. LocalViT [18] incorporates depth-wise convolution into vision Transformers to improve the local continuity of features.

To adapt to dense prediction tasks such as object detection, instance and semantic segmentation, there are also some methods [31, 21, 32, 34, 9, 4] to introduce the pyramid structure in CNNs to the design of Transformer backbones. PVTv1 is the first pyramid structure Transformer, which presents a hierarchical Transformer with four stages, showing that a pure Transformer backbone can as versatile as CNN counterparts and performs better in detection and segmentation tasks. After that, some improvements [21, 32, 34, 9, 4] are made to enhance the local continuity of features and to remove fixed size position embedding. For example, Swin Transformer [21] replaces fixed size position embedding with relative position biases, and restricts self-attention within shifted windows. CvT [32], CoaT [34], and LeViT [9] introduce convolution-like operations into vision Transformers. Twins [4] combines local attention and global attention mechanisms to obtain stronger feature representation.

3. Improved Pyramid Vision Transformer

Limitations in PVTv1. (1) Similar to ViT [7], PVTv1 [31] treats an image as a sequence of non-overlapping patches, which loses the local continuity of the image to a certain extent. (2) The position encoding in PVTv1 is fixed-size, which is inflexible for process images of arbitrary size. (3) When processing high-resolution input (*e.g.*, shorter side being 800 pixels), the computational complexity of PVTv1 is relatively large. These problems limit the performance of PVTv1 on vision tasks.

To address these problems, we propose PVTv2, which improves PVTv1 through designs as follows:

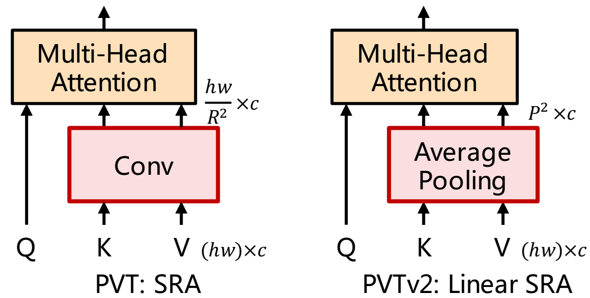

Figure 2: Comparison of SRA in PVTv1 and linear SRA in PVTv2.

Overlapping Patch Embedding. We utilize overlapping patch embedding to tokenize images. As shown in Figure 1(a), we enlarge the patch window, making adjacent windows overlap by half of the area, and pad the feature map with zeros to keep the resolution. In this work, we use convolution with zero paddings to implement overlapping patch embedding. Specifically, given an input of size $h \times w \times c$, we feed it to a convolution with the stride of S, the kernel size of $2S - 1$, the padding size of $S - 1$, and the kernel number of c'. The output size is $\frac{h}{S} \times \frac{w}{S} \times C'$.

Convolutional Feed-Forward. Inspired by [16, 5, 18], we remove the fixed-size position encoding [7], and introduce zero padding position encoding into PVT. As shown in Figure 1(b), we add a 3×3 depth-wise convolution [15] with the padding size of 1 between the first fully-connected (FC) layer and GELU [14] in feed-forward networks.

Linear Spatial Reduction Attention. To further reduce the computation cost of PVT, we propose linear spatial reduction attention (SRA) as illustrated in Figure 2. Different from SRA [31], linear SRA enjoys linear computational and memory costs like a convolutional layer. Specifically, given an input of size $h \times w \times c$, the complexity of SRA and linear SRA are:

$$\Omega(\mathrm{SRA}) = \frac{2h^2w^2c}{R^2} + hwc^2R^2, \tag{1}$$

$$\Omega(\text{Linear SRA}) = 2hwP^2c, \tag{2}$$

where R is the spatial reduction ratio of SRA [31]. P is the pooling size of linear SRA, which is set to 7 by default.

Combining the three improvements, PVTv2 can (1) obtain more local continuity of images and feature maps; (2) process variable-resolution input more flexibly; (3) enjoy the same linear complexity as CNN.

4. Details of PVTv2 Series

We scale up PVTv2 from B0 to B5 By changing the hyper-parameters. which are listed as follows:

	Output Size	Layer Name	Pyramid Vision Transformer v2						
			B0	B1	B2	B2-Li	B3	B4	B5
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	Overlapping Patch Embedding	$S_1 = 4$						
			$C_1 = 32$	$C_1 = 64$					
		Transformer Encoder	$R_1 = 8$ $N_1 = 1$ $E_1 = 8$ $L_1 = 2$	$R_1 = 8$ $N_1 = 1$ $E_1 = 8$ $L_1 = 2$	$R_1 = 8$ $N_1 = 1$ $E_1 = 8$ $L_1 = 3$	$P_1 = 7$ $N_1 = 1$ $E_1 = 8$ $L_1 = 3$	$R_1 = 8$ $N_1 = 1$ $E_1 = 8$ $L_1 = 3$	$R_1 = 8$ $N_1 = 1$ $E_1 = 8$ $L_1 = 3$	$R_1 = 8$ $N_1 = 1$ $E_1 = 4$ $L_1 = 3$
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	Overlapping Patch Embedding	$S_2 = 2$						
			$C_2 = 64$	$C_2 = 128$					
		Transformer Encoder	$R_2 = 4$ $N_2 = 2$ $E_2 = 8$ $L_2 = 2$	$R_2 = 4$ $N_2 = 2$ $E_2 = 8$ $L_2 = 2$	$R_2 = 4$ $N_2 = 2$ $E_2 = 8$ $L_2 = 3$	$P_2 = 7$ $N_2 = 2$ $E_2 = 8$ $L_2 = 3$	$R_2 = 4$ $N_2 = 2$ $E_2 = 8$ $L_2 = 3$	$R_2 = 4$ $N_2 = 2$ $E_2 = 8$ $L_2 = 8$	$R_2 = 4$ $N_2 = 2$ $E_2 = 4$ $L_2 = 6$
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	Overlapping Patch Embedding	$S_3 = 2$						
			$C_3 = 160$	$C_3 = 320$					
		Transformer Encoder	$R_3 = 2$ $N_3 = 5$ $E_3 = 4$ $L_3 = 2$	$R_3 = 2$ $N_3 = 5$ $E_3 = 4$ $L_3 = 2$	$R_3 = 2$ $N_3 = 5$ $E_3 = 4$ $L_3 = 6$	$P_3 = 7$ $N_3 = 5$ $E_3 = 4$ $L_3 = 6$	$R_3 = 2$ $N_3 = 5$ $E_3 = 4$ $L_3 = 18$	$R_3 = 2$ $N_3 = 5$ $E_3 = 4$ $L_3 = 27$	$R_3 = 2$ $N_3 = 5$ $E_3 = 4$ $L_3 = 40$
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	Overlapping Patch Embedding	$S_4 = 2$						
			$C_4 = 256$	$C_4 = 512$					
		Transformer Encoder	$R_4 = 1$ $N_4 = 8$ $E_4 = 4$ $L_4 = 2$	$R_4 = 1$ $N_4 = 8$ $E_4 = 4$ $L_4 = 2$	$R_4 = 1$ $N_4 = 8$ $E_4 = 4$ $L_4 = 3$	$P_4 = 7$ $N_4 = 8$ $E_4 = 4$ $L_4 = 3$	$R_4 = 1$ $N_4 = 8$ $E_4 = 4$ $L_4 = 3$	$R_4 = 1$ $N_4 = 8$ $E_4 = 4$ $L_4 = 3$	$R_4 = 1$ $N_4 = 8$ $E_4 = 4$ $L_4 = 3$

Table 1: **Detailed settings of PVTv2 series.** "-Li" denotes PVTv2 with linear SRA.

- S_i: the stride of the overlapping patch embedding in 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;
- P_i: the adaptive average pooling size of the linear SRA in Stage i;
- N_i: the head number of the Efficient Self-Attention in Stage i;
- E_i: the expansion ratio of the feed-forward layer [30] in Stage i;

Table 1 shows the detailed information of PVTv2 series. Our design follows the principles of ResNet [13]. (1) the channel dimension increase while the spatial resolution shrink with the layer goes deeper. (2) Stage 3 is assigned to most of the computation cost.

5. Experiment

5.1. Image Classification

Settings. Image classification experiments are performed on the ImageNet-1K dataset [25], which comprises 1.28 million training images and 50K validation images from 1,000 categories. All models are trained on the training set for fair comparison and report the top-1 error on the validation set. We follow DeiT [29] and apply random cropping, random horizontal flipping [27], label-smoothing regularization [28], mixup [36], and random erasing [38] as data augmentations. During training, we employ AdamW [23] 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 [22]. All models are trained for 300 epochs from scratch on 8 V100 GPUs. We apply a center crop on the validation set to benchmark, where a 224× 224 patch is cropped to evaluate the classification accuracy.

Results. In Table 2, we see that PVTv2 is the state-of-the-art method on ImageNet-1K classification. Compared to PVT, PVTv2 has similar flops and parameters, but the image classification accuracy is greatly improved. For example, PVTv2-B1 is 3.6% higher than PVTv1-Tiny, and PVTv2-B4 is 1.9% higher than PVT-Large.

Compared to other recent counterparts, PVTv2 series also has large advantages in terms of accuracy and model size. For example, PVTv2-B5 achieves 83.8% ImageNet top-1 accuracy, which is 0.5% higher than Swin Transformer [21] and Twins [4], while our parameters and FLOPS are fewer.

5.2. Object Detection

Settings. Object detection experiments are conducted on the challenging COCO benchmark [20]. All models are trained on COCO `train2017` (118k images) and evaluated on `val2017` (5k images). We verify the effectiveness of PVTv2 backbones on top of mainstream detectors, including RetinaNet [19], Mask R-CNN [11], Cascade Mask R-CNN [1], ATSS [37], GFL [17], and Sparse R-CNN [26]. Before training, we use the weights pre-trained on ImageNet to initialize the backbone and Xavier [8] to initialize the newly added layers. We train all the models with batch size 16 on 8 V100 GPUs, and adopt AdamW [23] with an initial learning rate of 1×10^{-4} as optimizer. Following common practices [19, 11, 3], we adopt 1× or 3× training schedule (*i.e.*, 12 or 36 epochs) to train all detection

Method	#Param (M)	GFLOPS	Top-1 Acc (%)
PVTv2-B0 (ours)	**3.4**	**0.6**	**70.5**
ResNet18 [13]	11.7	1.8	69.8
DeiT-Tiny/16 [29]	**5.7**	**1.3**	72.2
PVT-Tiny [31]	13.2	1.9	75.1
PVTv2-B1 (ours)	13.1	2.1	**78.7**
ResNet50 [13]	25.6	4.1	76.1
ResNeXt50-32x4d [33]	25.0	4.3	77.6
RegNetY-4G [24]	21.0	4.0	80.0
DeiT-Small/16 [29]	22.1	4.6	79.9
T2T-ViT$_t$-14 [35]	22.0	6.1	80.7
PVT-Small [31]	24.5	3.8	79.8
TNT-S [10]	23.8	5.2	81.3
Swin-T [21]	29.0	4.5	81.3
CvT-13 [32]	**20.0**	4.5	81.6
CoaT-Lite Small [34]	**20.0**	4.0	81.9
Twins-SVT-S [4]	24.0	**2.8**	81.3
PVTv2-B2-Li (ours)	22.6	3.9	**82.1**
PVTv2-B2 (ours)	25.4	4.0	82.0
ResNet101 [13]	44.7	7.9	77.4
ResNeXt101-32x4d [33]	44.2	8.0	78.8
RegNetY-8G [24]	39.0	8.0	81.7
T2T-ViT$_t$-19 [35]	39.0	9.8	81.4
PVT-Medium [31]	44.2	6.7	81.2
CvT-21 [32]	**32.0**	7.1	82.5
PVTv2-B3 (ours)	45.2	**6.9**	**83.2**
ResNet152 [13]	60.2	11.6	78.3
T2T-ViT$_t$-24 [35]	64.0	15.0	82.2
PVT-Large [31]	61.4	9.8	81.7
TNT-B [10]	66.0	14.1	82.8
Swin-S [21]	**50.0**	8.7	83.0
Twins-SVT-B [4]	56.0	**8.3**	83.1
PVTv2-B4 (ours)	62.6	10.1	**83.6**
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**

Table 2: **Image classification performance on the ImageNet validation set**. "-Li" denotes PVTv2 with linear SRA. "#Param" refers to the number of parameters. "GFLOPs" is calculated under the input scale of 224×224. "*" indicates the performance of the method trained under the strategy of its original paper.

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 reported in Table 3, PVTv2 significantly outperforms PVTv1 on both one-stage and two-stage object detectors with similar model size. For example, PVTv2-B4 archive 46.1 AP on top of RetinaNet [19], and 47.5 AP$^{\text{b}}$ on top of Mask R-CNN [11], surpassing the models with PVTv1 by 3.5 AP and 3 AP$^{\text{b}}$, respectively.

For fair comparison between PVTv2 and Swin Transformer [21], we keep all settings the same, including ImageNet-1K pre-training and COCO fine-tuning strategies. We evaluate Swin Transformer and PVTv2 on four state-of-the-arts detectors, including Cascade R-CNN [1], ATSS [37], GFL [17], and Sparse R-CNN [26]. We see PVTv2 obtain much better AP than Swin Transformer among all the detectors, showing its better feature representation ability. For example, on ATSS, PVTv2 has similar parameters and flops compared to Swin-T, but PVTv2 achieves 49.9 AP, which is 2.7 higher than Swin-T. Our PVTv2-Li can largely reduce the computation from 258 to 194 GFlops, while only sacrifice a little performance.

Backbone	RetinaNet 1×							Mask R-CNN 1×						
	#P (M)	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	#P (M)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
PVTv2-B0	**13.0**	**37.2**	**57.2**	**39.5**	**23.1**	**40.4**	**49.7**	**23.5**	**38.2**	**60.5**	**40.7**	**36.2**	**57.8**	**38.6**
ResNet18 [13]	**21.3**	31.8	49.6	33.6	16.3	34.3	43.2	**31.2**	36.9	57.1	40.0	33.6	53.9	35.7
PVT-Tiny	23.0	36.7	56.9	38.9	22.6	38.8	50.0	32.9	39.8	62.2	43.0	37.4	59.3	39.9
PVTv2-B1 (ours)	23.8	**41.2**	**61.9**	**43.9**	**25.4**	**44.5**	**54.3**	33.7	**41.8**	**64.3**	**45.9**	**38.8**	**61.2**	**41.6**
ResNet50 [13]	37.7	36.3	55.3	38.6	19.3	40.0	48.8	44.2	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small	34.2	40.4	61.3	43.0	25.0	42.9	55.7	44.1	43.0	65.3	46.9	39.9	62.5	42.8
PVTv2-B2-Li (ours)	**32.3**	43.6	64.7	46.8	28.3	47.6	57.4	**42.2**	44.1	66.3	48.4	40.5	63.2	43.6
PVTv2-B2 (ours)	35.1	**44.6**	**65.6**	**47.6**	**27.4**	**48.8**	**58.6**	45.0	**45.3**	**67.1**	**49.6**	**41.2**	**64.2**	**44.4**
ResNet101 [13]	56.7	38.5	57.8	41.2	21.4	42.6	51.1	63.2	42.8	63.2	47.1	38.5	60.1	41.3
ResNeXt101-32x4d [33]	56.4	39.9	59.6	42.7	22.3	44.2	52.5	**62.8**	44.0	64.4	48.0	39.2	61.4	41.9
PVT-Medium (ours)	**53.9**	41.9	63.1	44.3	25.0	44.9	57.6	63.9	44.2	66.0	48.2	40.5	63.1	43.5
PVTv2-B3 (ours)	55.0	**45.9**	**66.8**	**49.3**	**28.6**	**49.8**	**61.4**	64.9	**47.0**	**68.1**	**51.7**	**42.5**	**65.7**	**45.7**
PVT-Large	**71.1**	42.6	63.7	45.4	25.8	46.0	58.4	**81.0**	44.5	66.0	48.3	40.7	63.4	43.7
PVTv2-B4 (ours)	72.3	**46.1**	**66.9**	**49.2**	**28.4**	**50.0**	**62.2**	82.2	**47.5**	**68.7**	**52.0**	**42.7**	**66.1**	**46.1**
ResNeXt101-64x4d [33]	95.5	41.0	60.9	44.0	23.9	45.2	54.0	101.9	44.4	64.9	48.8	39.7	61.9	42.6
PVTv2-B5 (ours)	**91.7**	**46.2**	**67.1**	**49.5**	**28.5**	**50.0**	**62.5**	**101.6**	**47.4**	**68.6**	**51.9**	**42.5**	**65.7**	**46.0**

Table 3: **Object detection and instance segmentation on COCO `val2017`.** "-Li" denotes PVTv2 with linear SRA. "#P" refers to parameter number. AP^b and AP^m denote bounding box AP and mask AP, respectively.

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	728
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

Table 4: **Compare with Swin Transformer on object detection.** "-Li" denotes PVTv2 with linear SRA. AP^b denotes bounding box AP. "#P" refers to parameter number. "GFLOPs" is calculated under the input scale of 1280×800.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. 3, 4, 5

[2] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *arXiv preprint arXiv:2103.14899*, 2021. 2

[3] 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. 3

[4] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv preprint arXiv:2104.13840*, 2021. 1, 2, 3, 4

[5] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021. 2

[6] 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] 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, 4

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010. 3

[9] Ben Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. *arXiv preprint arXiv:2104.01136*, 2021. 1, 2

[10] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 2, 4

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2017. 3, 4

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2015. 1

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2016. 3, 4, 5

[14] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 2

[15] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[16] Md. Amirul Islam, Sen Jia, and Neil D. B. Bruce. How much position information do convolutional neural networks encode? In *Proceedings of the International Conference on Learning Representations*, 2020. 2

[17] 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. 1, 3, 4, 5

[18] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021. 2

[19] 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. 3, 4

[20] 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. 3

[21] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021. 1, 2, 3, 4, 5

[22] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 3

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 3

[24] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 4

[25] 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

[26] 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. *arXiv preprint arXiv:2011.12450*, 2020. 3, 4, 5

[27] 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

[28] 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

[29] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 1, 2, 3, 4

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 3

[31] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 1, 2, 4

[32] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 1, 2, 4

[33] 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. 4, 5

[34] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. *arXiv preprint arXiv:2104.06399*, 2021. 1, 2, 4

[35] 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. 2, 4

[36] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 3

[37] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9759–9768, 2020. 3, 4, 5

[38] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020. 3