PVT v2: Improved Baselines with Pyramid Vision Transformer

Wenhai Wang¹,², Enze Xie³, Xiang Li⁴, Deng-Ping Fan⁵, Kaitao Song⁴, Ding Liang⁶, Tong Lu², Ping Luo³, Ling Shao⁵
¹Shanghai AI Laboratory  ²Nanjing University  ³The University of Hong Kong  
⁴Nanjing University of Science and Technology  ⁵IIAI  ⁶SenseTime Research

wangwenhai@pjlab.org.cn,

Abstract

Transformer recently has presented encouraging progress in computer vision. In this work, we present new baselines by improving the original Pyramid Vision Transformer (PVT v1) by adding three designs, including (1) linear complexity attention layer, (2) overlapping patch embedding, and (3) convolutional feed-forward network. With these modifications, PVT v2 reduces the computational complexity of PVT v1 to linear and achieves significant improvements on fundamental vision tasks such as classification, detection, and segmentation. Notably, the proposed PVT v2 achieves comparable or better performances than recent works such as Swin Transformer. We hope this work will facilitate state-of-the-art Transformer researches in computer vision. Code is available at https://github.com/whai362/PVT.

1. Introduction

Recent studies on vision Transformer are converging on the backbone network [8, 31, 33, 34, 23, 36, 10, 5] 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) [8] treats each image as a sequence of tokens (patches) with a fixed length, and then feeds them to multiple Transformer layers to perform classification. It is the first work to prove that a pure Transformer can archive state-of-the-art performance in image classification when training data is sufficient (e.g., ImageNet-22k [7], JFT-300M). DeiT [31] further explores a data-efficient training strategy and a distillation approach for ViT.

To improve image classification performance, recent methods make tailored changes to ViT. T2T ViT [37] concatenates tokens within an overlapping sliding window into one token progressively. TNT [11] utilizes inner and outer Transformer blocks to generate pixel and patch embeddings respectively. CPVT [6] 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 [20] incorporates depth-wise convolution into vision Transformers to improve the local continuity of features.

2. Related Work

We mainly discuss transformer backbones related to this work. ViT [8] treats each image as a sequence of tokens (patches) with a fixed length, and then feeds them to multiple Transformer layers to perform classification. It is the first work to prove that a pure Transformer can archive state-of-the-art performance in image classification when training data is sufficient (e.g., ImageNet-22k [7], JFT-300M). DeiT [31] further explores a data-efficient training strategy and a distillation approach for ViT.

To improve image classification performance, recent methods make tailored changes to ViT. T2T ViT [37] concatenates tokens within an overlapping sliding window into one token progressively. TNT [11] utilizes inner and outer Transformer blocks to generate pixel and patch embeddings respectively. CPVT [6] 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 [20] incorporates depth-wise convolution into vision Transformers to improve the local continuity of features.

₃PVT v2 has 6 different size variants, from B0 to B5 according to the parameter number.
tection, instance and semantic segmentation, there are also some methods [33, 23, 34, 36, 10, 5] to introduce the pyramid structure in CNNs to the design of Transformer backbones. PVT v1 is the first pyramid structure Transformer, which presents a hierarchical Transformer with four stages, showing that a pure Transformer backbone can be as versatile as CNN counterparts and performs better in detection and segmentation tasks. After that, some improvements [23, 34, 36, 10, 5] are made to enhance the local continuity of features and to remove fixed size position embedding. For example, Swin Transformer [23] replaces fixed size position embedding with relative position biases, and restricts self-attention within shifted windows. CvT [34], CoaT [36], and LeViT [10] introduce convolution-like operations into vision Transformers. Twins [5] combines local attention and global attention mechanisms to obtain stronger feature representation.

3. Methodology

3.1. Limitations in PVT v1

There are three main limitations in PVT v1 [33] as follows: (1) Similar to ViT [8], when processing high-resolution input (e.g., shorter side being 800 pixels), the computational complexity of PVT v1 is relatively large. (2) PVT v1 [33] treats an image as a sequence of non-overlapping patches, which loses the local continuity of the image to a certain extent; (3) The position encoding in PVT v1 is fixed-size, which is inflexible for process images of arbitrary size. These problems limit the performance of PVT v1 on vision tasks.

To address these issues, we propose PVT v2, which improves PVT v1 through three designs, which are listed in Sec 3.2, 3.3, and 3.4.

3.2. Linear Spatial Reduction Attention

First, to reduce the high computational cost caused by attention operations, we propose linear spatial reduction attention (SRA) layer as illustrated in Fig. 1. Different from SRA [33] which uses convolutions for spatial reduction, linear SRA uses average pooling to reduce the spatial dimension (i.e., $h \times w$) to a fixed size (i.e., $P \times P$) before the attention operation. So 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:

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

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

where $R$ is the spatial reduction ratio of SRA [33]. $P$ is the pooling size of linear SRA, which is set to 7.

3.3. Overlapping Patch Embedding

Second, to model the local continuity information, we utilize overlapping patch embedding to tokenize images. As shown in Fig. 2(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 $\frac{h}{S} \times \frac{w}{S} \times c'$.
3.4. Convolutional Feed-Forward

Third, inspired by [17, 6, 20], we remove the fixed-size position encoding [8], and introduce zero padding position encoding into PVT. As shown in Fig. 2(b), we add a $3 \times 3$ depth-wise convolution [16] with the padding size of 1 between the first fully-connected (FC) layer and GELU [15] in feed-forward networks.

3.5. Details of PVT v2 Series

We scale up PVT v2 from B0 to B5 By changing the hyper-parameters, which are listed as follows:

- $S_i$: 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 Efficient Self-Attention in Stage $i$;
- $E_i$: the expansion ratio of the feed-forward layer [32] in Stage $i$;

Tab. 1 shows the detailed information of PVT v2 series. “-Li” denotes PVT v2 with linear SRA.

| Stage | Output Size | Layer Name | Pyramid Vision Transformer v2 |
|-------|-------------|------------|------------------------------|
|       |             | Overlapping Patch Embedding | $C_i$ |
| Stage 1 | $\frac{H}{4} \times \frac{W}{4}$ | $S_1 = 4$ |
| Transformer | $R_1 = 8$ | $N_1 = 1$ |
| Encoder | $E_1 = 8$ | $L_1 = 2$ |
| Stage 2 | $\frac{H}{8} \times \frac{W}{8}$ | $S_2 = 2$ |
| Transformer | $R_2 = 4$ | $N_2 = 2$ |
| Encoder | $E_2 = 8$ | $L_2 = 2$ |
| Stage 3 | $\frac{H}{16} \times \frac{W}{16}$ | $S_3 = 2$ |
| Transformer | $R_3 = 2$ | $N_3 = 5$ |
| Encoder | $E_3 = 4$ | $L_3 = 2$ |
| Stage 4 | $\frac{H}{32} \times \frac{W}{32}$ | $S_4 = 2$ |
| Transformer | $R_4 = 1$ | $N_4 = 8$ |
| Encoder | $E_4 = 4$ | $L_4 = 2$ |

Table 1: Detailed settings of PVT v2 series. “-Li” denotes PVT v2 with linear SRA.

4. Experiment

4.1. Image Classification

Settings. Image classification experiments are performed on the ImageNet-1K dataset [27], 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 [31] and apply random cropping, random horizontal flipping [29], label-smoothing regularization [30], mixup [38], and random erasing [40] as data augmentations. During training, we employ AdamW [25] with a momentum of 0.9, a mini-batch size of 128, and a weight decay of $5 \times 10^{-2}$ to optimize models. The initial learning rate is set to $1 \times 10^{-3}$ and decreases following the cosine schedule [24]. 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$^2$ patch is cropped to evaluate the classification accuracy.

Results. In Tab. 2, we see that PVT v2 is the state-of-the-art method on ImageNet-1K classification. Compared to PVT, PVT v2 has similar flops and parameters, but the image classification accuracy is greatly improved. For example, PVT v2-B1 is 3.6% higher than PVT v1-Tiny, and PVT v2-B4 is 1.9% higher than PVT-Large.

Compared to other recent counterparts, PVT v2 series also has large advantages in terms of accuracy and model size. For example, PVT v2-B5 achieves 83.8% ImageNet top-1 accuracy, which is 0.5% higher than Swin
### 4.2 Object Detection

**Settings.** Object detection experiments are conducted on the challenging COCO benchmark [22]. All models are trained on COCO \texttt{train2017} (118k images) and evaluated on \texttt{val2017} (5k images). We verify the effectiveness of PVT v2 backbones on top of mainstream detectors, including RetinaNet [21], Mask R-CNN [12], Cascade Mask R-CNN [1], ATSS [39], GFL [19], and Sparse R-CNN [28]. Before training, we use the weights pre-trained on ImageNet to initialize the backbone and Xavier [9] to initialize the newly added layers. We train all the models with batch size 16 on 8 V100 GPUs, and adopt AdamW [25] with an initial learning rate of $1 \times 10^{-4}$ as optimizer. Following common practices [21, 12, 3], we adopt 1× or 3× 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× 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 Tab. 3, PVT v2 significantly outperforms PVT v1 on both one-stage and two-stage object detectors with similar model size. For example, PVT v2-B4 archive 46.1 AP on top of RetinaNet [21], and 47.5 AP$^b$ on top of Mask R-CNN [12], surpassing the models with PVT v1 by 3.5 AP and 4.6 AP$^b$, respectively. We present some qualitative object detection and instance segmentation results on COCO \texttt{val2017} [22] in Fig. 3, which also shows the good performance of our models.

For a fair comparison between PVT v2 and Swin Transformer [23], we keep all settings the same, including ImageNet-1K pre-training and COCO fine-tuning strategies. We evaluate Swin Transformer and PVT v2 on four state-of-the-arts detectors, including Cascade R-CNN [1], ATSS [39], GFL [19], and Sparse R-CNN [28]. We see PVT v2 obtain much better AP than Swin Transformer among all the detectors, showing its better feature representation ability. For example, on ATSS, PVT v2 has similar parameters and flops compared to Swin-T, but PVT v2 achieves 49.9 AP, which is 2.7 higher than Swin-T. Our PVT v2-Li can largely reduce the computation from 258 to 194 GFLOPs, while only sacrificing a little performance.

### 4.3. Semantic Segmentation

**Settings.** Following PVT v1 [33], we choose ADE20K [41] to benchmark the performance of semantic segmentation. For a fair comparison, we test the performance of PVT v2 backbones by applying it to Semantic FPN [18]. In the training phase, the backbone is initialized with the weights pre-trained on ImageNet [7], and the newly added layers are initialized with Xavier [9]. We optimize our models using AdamW [25] with an initial learning rate of 1e-4. Following common practices [18, 4], we train our models for 40k iterations with a batch size of 16 on 4 V100 GPUs. The learning rate is decayed following the polynomial decay schedule ($t^{1/10}$), 12 or 36 epochs) to train all detection models. In the training phase, we randomly resize and crop the image to 512 × 512 for training, and rescale to have a shorter side of 512 pixels during testing.

**Results.** As shown in Tab. 5, when using Semantic FPN [18] for semantic segmentation, PVT v2 consistently outperforms PVT v1 [33] and other counterparts. For example, with almost the same number of parameters

| Method             | #Param (M) | GFLOPs  | Top-1 Acc (%) |
|--------------------|------------|---------|---------------|
| PVTv2-B0 (ours)    | 3.4        | 0.6     | 70.5          |
| ResNet18           | 13.7       | 1.8     | 69.8          |
| Det-Tiny/16 [31]   | 5.7        | 1.3     | 72.2          |
| PVTv1-Tiny [33]    | 13.2       | 1.9     | 75.1          |
| PVTv2-B1 (ours)    | 13.1       | 2.1     | 78.7          |
| ResNet50 [14]      | 25.6       | 4.1     | 76.1          |
| ResNet101          | 44.7       | 7.9     | 77.4          |
| RegNetY-4G [26]    | 21.0       | 4.0     | 80.0          |
| Det-T-Small/16 [31]| 22.1       | 4.6     | 79.9          |
| T2T-ViT-14 [37]    | 22.0       | 6.1     | 80.7          |
| PVTv1-Small [33]   | 24.5       | 3.8     | 79.8          |
| TNT-S [11]         | 23.8       | 5.2     | 81.3          |
| Swin-T [23]        | 29.0       | 4.5     | 81.3          |
| CvT-13 [34]        | 20.0       | 4.5     | 81.6          |
| CoaT-Lite Small    | 20.0       | 4.0     | 81.9          |
| Twins-SVT-S        | 24.0       | 2.8     | 81.7          |
| PVTv2-B2-Li (ours) | 22.6       | 3.9     | 82.1          |
| PVTv2-B2 (ours)    | 25.4       | 4.0     | 82.0          |
| ResNet101          | 44.7       | 7.9     | 77.4          |
| ResNet101-32x4d [35]| 44.2      | 8.0     | 78.8          |
| RegNetY-9G [26]    | 39.0       | 8.0     | 81.7          |
| T2T-ViT-19 [37]    | 39.0       | 9.8     | 81.4          |
| PVTv1-Medium [33]  | 44.2       | 6.7     | 81.2          |
| CvT-21 [34]        | 32.0       | 7.1     | 82.5          |
| PVTv2-B3 (ours)    | 45.2       | 6.9     | 83.2          |
| ResNet152 [14]     | 60.2       | 11.6    | 78.3          |
| T2T-ViT-24 [37]    | 64.0       | 15.0    | 82.2          |
| PVTv1-Large [33]   | 61.4       | 9.8     | 81.7          |
| TNT-B [11]         | 66.0       | 14.1    | 82.8          |
| Swin-S [3]         | 50.0       | 8.7     | 83.0          |
| Twins-SVT-B [5]    | 56.0       | 8.3     | 83.2          |
| PVTv2-B4 (ours)    | 62.6       | 10.1    | 83.6          |
| ResNetXt101-64x4d [35]| 83.5   | 15.6    | 79.6          |
| RegNetY-16G [26]   | 84.0       | 16.0    | 82.9          |
| VIT-Base/16 [8]    | 86.6       | 17.6    | 81.8          |
| DeiT-Base/16 [31]  | 86.6       | 17.6    | 81.8          |
| Swin-B [23]        | 88.0       | 15.4    | 83.3          |
| Twins-SVT-L [5]    | 99.2       | 14.8    | 83.7          |
| PVTv2-B5 (ours)    | 82.0       | 11.8    | 83.8          |

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 × 224. “**” indicates the performance of the method trained under the strategy of its original paper. “-Li” denotes PVT v2 with linear SRA.
| Backbone       | RetinaNet 1× | Mask R-CNN 1× |
|---------------|-------------|--------------|
|               | #P (M) | AP | AP₀ | AP₇₅ | AP₅₀ | AP₀ | AP₇₅ | AP₀ | AP₇₅ |
| PVTv2-B0      | 13.0   | 37.2 | 57.2 | 39.5 | 23.6 | 40.4 | 49.7 |
| ResNet18 [14] | 21.3   | 31.8 | 49.6 | 33.6 | 16.6 | 34.3 | 43.2 |
| PVTv1-Tiny [33]| 23.0   | 36.7 | 56.9 | 38.9 | 22.6 | 38.8 | 50.0 |
| PVTv2-B1 (ours)| 23.8   | 41.2 | 61.9 | 43.9 | 25.4 | 44.5 | 54.3 |
| ResNet50 [14] | 37.7   | 36.3 | 55.3 | 38.6 | 19.3 | 40.0 | 48.8 |
| PVTv1-Small [33]| 34.2   | 40.4 | 61.3 | 43.0 | 25.0 | 42.9 | 55.7 |
| PVTv2-B2-Li (ours)| 32.3   | 43.6 | 64.7 | 46.8 | 28.3 | 47.6 | 57.4 |
| PVTv2-B2 (ours)| 35.1   | 44.6 | 65.6 | 47.6 | 27.4 | 48.8 | 58.6 |
| ResNet101 [14]| 56.7   | 38.5 | 57.8 | 41.2 | 21.4 | 42.6 | 51.1 |
| ResNeXt101-32x4d [35]| 56.4 | 39.9 | 59.6 | 42.7 | 22.3 | 44.2 | 52.5 |
| PVTv1-Medium [33]| 53.9   | 41.9 | 63.1 | 44.3 | 25.0 | 44.9 | 57.6 |
| PVTv2-B3 (ours)| 55.0   | 45.9 | 66.8 | 49.3 | 28.6 | 49.8 | 61.4 |
| PVTv1-Large [33]| 71.1   | 42.6 | 63.7 | 45.4 | 25.8 | 46.0 | 58.4 |
| PVTv2-B4 (ours)| 71.1   | 46.1 | 66.9 | 49.2 | 28.4 | 50.0 | 62.2 |
| ResNeXt101-64x4d [35]| 71.1 | 41.0 | 60.9 | 44.4 | 23.9 | 45.2 | 54.0 |
| PVTv2-B5 (ours)| 91.7   | 46.2 | 67.1 | 49.5 | 28.5 | 50.0 | 62.5 |

Table 3: Object detection and instance segmentation on COCO val2017. “#P” refers to parameter number. AP₀ and AP₇₅ denote bounding box AP and mask AP, respectively. “-Li” denotes PVT v2 with linear SRA.

| Backbone       | Method         | AP₀ | AP₅₀ | AP₀ | AP₇₅ | AP₀ | AP₅₀ | AP₀ | AP₇₅ | #P (M) | GFLOPs |
|---------------|----------------|-----|------|-----|------|-----|------|-----|------|-------|--------|
| PVTv2-B0 (ours)| Cascade Mask  | 13.0| 37.2 | 57.2 | 39.5 | 23.6 | 40.4 | 49.7 |
| ResNet50 [14] | PVTv2-B2-Li (ours) | 21.3 | 31.8 | 49.6 | 33.6 | 16.6 | 34.3 | 43.2 |
| PVTv2-B2 (ours) | 23.0 | 36.7 | 56.9 | 38.9 | 22.6 | 38.8 | 50.0 |
| ResNet18 [14] | PVTv1-Small [33] | 34.2 | 40.4 | 61.3 | 43.0 | 25.0 | 42.9 | 55.7 |
| PVTv2-B2-Li (ours) | 32.3 | 43.6 | 64.7 | 46.8 | 28.3 | 47.6 | 57.4 |
| PVTv2-B2 (ours) | 35.1 | 44.6 | 65.6 | 47.6 | 27.4 | 48.8 | 58.6 |

Table 4: Compare with Swin Transformer on object detection. “AP₀” denotes bounding box AP. “#P” refers to parameter number. “GFLOPs” is calculated under the input scale of 1280×800. “-Li” denotes PVT v2 with linear SRA.

GFLOPs, PVT v2-B1/B2/B3/B4 are at least 5.3% higher than PVT v1-Tiny/Small/Medium/Large. Moreover, although the GFLOPs of PVT-Large are 12% lower than those of ResNeXt101-64x4d, the mIoU is still 8.5 points higher (48.7 vs 40.2). In Fig. 3, we also visualize some qualitative segmentation results on ADE20K [41]. These results demonstrate that PVT v2 backbones can extract powerful features for semantic segmentation, benefiting from the improved designs.

4.4. Ablation Study

4.4.1 Model Analysis

Ablation experiments of PVT v2 is reported in Tab. 6. We see that all three designs can improve the model in terms of performance, parameter number, or computation overhead. Overlapping patch embedding (OPE) is important. Comparing #1 and #2 in Tab. 6, the model with OPE obtains better top-1 accuracy (81.1% vs. 79.8%) on ImageNet and better AP (42.2% vs. 40.4%) on COCO than the one with original patch embedding (PE) [8]. OPE is effective because it can model the local continuity of images and feature map via the overlapping sliding window.
Convolutional feed-forward network (CFFN) matters. Compared to original feed-forward network (FFN) [8], our CFFN contains a zero-padding convolutional layer, which can capture the local continuity of the input tensor. In addition, due to the positional information introduced by zero-padding in OPE and CFFN, we can remove the fixed-size positional embeddings used in PVT v1, making the model flexible to handle variable resolution inputs. As reported in #2 and #3 in Tab. 6, CFFN brings 0.9 points improvement on ImageNet (82.0% vs. 81.1%) and 2.4 points improvement on COCO, which demonstrates its effectiveness.

Linear SRA (LSRA) contributes to a better model. As reported in #3 and #4 in Tab. 6, compared to SRA [33], our LSRA significantly reduces the computation overhead (GFLOPs) of the model by 22%, while keeping a comparable top-1 accuracy on ImageNet (82.1% vs. 82.0%), and only 1 point lower AP on COCO (43.6 vs. 44.6). These results show the low computational cost and good effect of LSRA.

Figure 3: Qualitative results of object detection and instance segmentation on COCO val2017 [22], and semantic segmentation on ADE20K [41]. The results (from left to right) are generated by PVT v2-B2-based RetinaNet [21], Mask R-CNN [12], and Semantic FPN [18], respectively.

Table 6: Ablation experiments of PVT v2. “OPE”, “CFFN”, and “LSRA” represent overlapping patch embedding, convolutional feed-forward network, and linear SRA, respectively.
4.4.2 Computation Overhead Analysis

As shown in Figure 4, with increasing input scale, the GFLOPs growth rate of the proposed PVT v2-B2-Li is much lower than that of PVT v1-Small [33], and is similar to that of ResNet-50 [14]. This result proves that our PVT v2-Li successfully addresses the high computational overhead problem caused by the attention layer.

5. Conclusion

We study the limitations of Pyramid Vision Transformer (PVT v1) and improve it with three designs, which are overlapping patch embedding, convolutional feed-forward network, and linear spatial reduction attention layer. Extensive experiments on different tasks, such as image classification, object detection, and semantic segmentation demonstrate that the proposed PVT v2 is stronger than its predecessor PVT v1 and other state-of-the-art transformer-based backbones, under comparable numbers of parameters. We hope these improved baselines will provide a reference for future research in vision Transformer.

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. 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. 1
[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiuru Xu, et al. Mndetection: Open mmlab detection tool- box and benchmark. arXiv preprint arXiv:1906.07155, 2019. 4
[4] 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. 4
[5] 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, 4
[6] 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. 1, 3
[7] 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. 1, 4
[8] 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, 4, 5, 6, 7
[9] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Proc. Int. Conf. Artificial Intel. & Stat., 2010. 4
[10] 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. In Proc. IEEE Int. Conf. Comp. Vis., 2021. 1, 2
[11] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. arXiv preprint arXiv:2103.00112, 2021. 1, 4
[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proc. IEEE Int. Conf. Comp. Vis., 2017. 4, 6
[13] 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. Vis. Comp. Vis., 2015. 1, 7
[14] 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, 7
[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016. 3
[16] 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. 3
[17] Md. Amirul Islam, Sen Jia, and Neil D. B. Bruce. How much position information do convolutional neural networks encode? In Proc. Int. Conf. Learn. Representations, 2020. 3
[18] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In Proc. IEEE Conf. Comp. Vis. Patt. Recogn., 2019. 4, 6
Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 1, 2, 4, 5, 6, 7

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

[35] 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

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

[37] 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. 1, 4

[38] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proc. Int. Conf. Learn. Representations*, 2018. 3

[39] 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 *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020. 4, 5

[40] Zhun Zhong, Liang Zhong, Guoliang Kang, Shaosi Li, and Yi Yang. Random erasing data augmentation. In *Proc. AAAI Conf. Artificial Intell.*, 2020. 3

[41] 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. 1, 4, 5, 6