P2T: Pyramid Pooling Transformer for Scene Understanding

Yu-Huan Wu, Yun Liu, Xin Zhan, and Ming-Ming Cheng

Abstract—Recently, the vision transformer has achieved great success by pushing the state-of-the-art of various vision tasks. One of the most challenging problems in the vision transformer is that the large sequence length of image tokens leads to high computational cost (quadratic complexity). A popular solution to this problem is to use a single pooling operation to reduce the sequence length. This paper considers how to improve existing vision transformers, where the pooled feature extracted by a single pooling operation seems less powerful. To this end, we note that pyramid pooling has been demonstrated to be effective in various vision tasks owing to its powerful ability in context abstraction. However, pyramid pooling has not been explored in backbone network design. To bridge this gap, we propose to adapt pyramid pooling to Multi-Head Self-Attention (MHSA) in the vision transformer, simultaneously reducing the sequence length and capturing powerful contextual features. Plugged with our pooling-based MHSA, we build a universal vision transformer backbone, dubbed Pyramid Pooling Transformer (P2T). Extensive experiments demonstrate that, when applied P2T as the backbone network, it shows substantial superiority in various vision tasks such as image classification, semantic segmentation, object detection, and instance segmentation, compared to previous CNN- and transformer-based networks. The code will be released at https://github.com/yuhuan-wu/P2T.

Index Terms—Transformer, backbone network, efficient self-attention, pyramid pooling, scene understanding

✦

1 INTRODUCTION

IN the past decade, convolutional neural networks (CNNs) have dominated computer vision and achieved many great stories [1]–[9]. The state-of-the-art of various vision tasks on many large-scale datasets has been significantly pushed forward [10]–[14]. In an orthogonal field, *i.e.*, natural language processing (NLP), the dominating technique is transformer [15]. Transformer entirely relies on self-attention to capture the long-range global relationships and has achieved brilliant successes. Considering that global information is also essential for vision tasks, a proper adaption of the transformer [15] should be useful to overcome the limitation of CNNs, *i.e.*, CNNs usually enlarge the receptive field by stacking more layers.

Lots of efforts are dedicated to exploring such a proper adaption of the transformer [15]. Some early attempts use CNNs [2], [4] to extract deep features that are fed into transformers for further processing and regressing the targets [16]–[18]. Dosovitskiy *et al.* [19] made a thorough success by applying a pure transformer network for image classification. They split an image into patches and took each patch as a word/token in an NLP application so that transformer can then be directly adopted. This simple method attains competitive performance on ImageNet [10]. Therefore, a
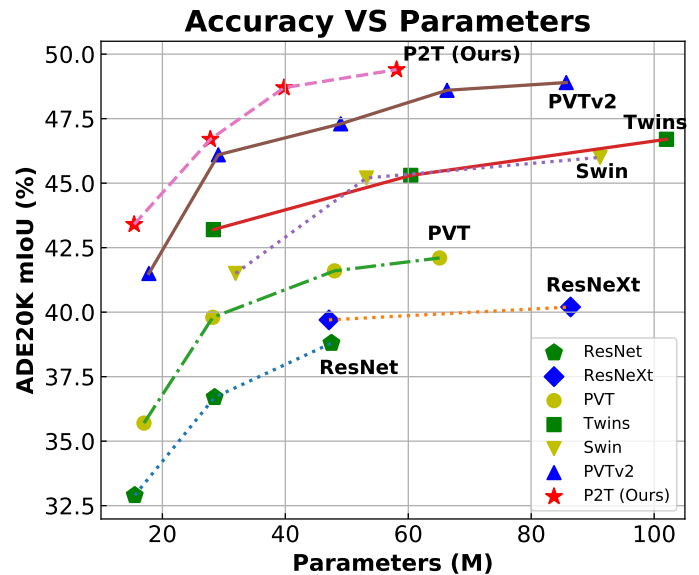

Fig. 1. **Experimental results for semantic segmentation on the ADE20K dataset [14].** Following PVT [21], Semantic FPN [26] is chosen as the basic method, equipped with different backbone networks, including ResNet [4], ResNeXt [27], PVT [21], Twins [28], Swin Transformer [22], PVTv2 [29], and our P2T.

• Y.-H. Wu and M.-M. Cheng are with TMCC, College of Computer Science, Nankai University, Tianjin, China. (E-mail: wuyuhuan@mail.nankai.edu.cn, cmm@nankai.edu.cn)
• Y. Liu is with Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore. (E-mail: vagrantlyun@gmail.com)
• X. Zhan is with Alibaba DAMO Academy, Hangzhou, China.
• The first two authors contributed equally to this work.
• Corresponding author: M.-M. Cheng. (E-mail: cmm@nankai.edu.cn)
• This work is done while Y.-H. Wu is a research intern at Alibaba DAMO Academy.

new concept of the vision transformer appears. In a very short period, a large amount of literature has emerged to improve the vision transformer [19], and much better performance than CNNs has been achieved [20]–[25].

Nevertheless, there is still one challenging problem in vision transformers, *i.e.*, the length of the data sequence. When viewing image patches as tokens, the sequence length is much longer than in NLP applications. For example, in

NLP, the well-known WMT2014 English-German dataset [30] has 50M English words with 2M sentences, with an average sequence length of 25. In contrast, in computer vision, we usually use the image resolution of 224×224 for image classification on the ImageNet dataset [10], resulting in the sequence length of 3136 if we use the common patch size of 4×4. Since the computational and space complexity of Multi-Head Self-Attention (MHSA) in the transformer is *quadratic* (rather than *linear* in CNNs) to the image size, directly applying the transformer to vision tasks has a high requirement for computational resources. To make a pure transformer network possible for image classification, ViT [19] uses large patch sizes, *e.g.*, 16 and 32, to reduce the sequence length, and achieves great success for image classification. Later, many transformer works significantly improve the performance of ViT [19] by introducing the pyramid structure [20]–[24], [28], [31], [32], where the input layer first uses the small patch size of 4×4 and the sequence length is gradually reduced by merging adjacent image patches.

To further reduce the computational cost of MHSA, PVT [21] and MViT [23] use a single pooling operation to downsample the feature map in the computation of MHSA. With the pooled features, they model token-to-region relationship rather than the expected token-to-token relationship. Swin Transformer [22] proposes to compute MHSA within small windows rather than across the whole input, modeling local relationships. It uses a window shift strategy to gradually enlarge the receptive field, like CNNs, enlarging the receptive field through stacking more layers [33]. However, an essential characteristic of the vision transformer is its *direct global relationship modeling*, which is also why we transfer from CNNs to transformers.

Here, we consider how to improve PVT [21] and MViT [23] where the pooled feature extracted by a single pooling operation seems less powerful. If we can squeeze the input feature into a powerful representation with a short sequence length, we may achieve better performance. To this end, we note that pyramid pooling [34]–[37] is a long-history technique for computer vision, which extracts contextual information and utilizes multiple pooling operations with different receptive fields and strides onto the input feature map. This simple technique has been demonstrated to be effective in various downstream vision tasks such as semantic segmentation [37] and object detection [36]. Nevertheless, recent pyramid pooling approaches highly rely on a pretrained CNN backbone, and thus they are limited to a specific task. In another word, the pyramid pooling technique **has not been** explored in the backbone network design that has broad applications. Motivated by this, we bridge this gap by adapting pyramid pooling to the vision transformer block for **simultaneously** reducing the sequence length and learning powerful contextual representations. Pyramid pooling is also very efficient and thus will only induce negligible computational cost for the vision transformer.

We achieve this goal by proposing a new transformer backbone network, *i.e.*, **Pyramid Pooling Transformer (P2T)**. We adapt the idea of pyramid pooling to the computation of multi-head self-attention (MHSA) in the vision transformer, reducing the computational cost of MHSA

and capturing rich contextual information simultaneously. By applying the new pooling-based MHSA, P2T exhibits stronger ability in feature representation learning and visual recognition than PVT [21] and MViT [23] which are based on a single pooling operation. We evaluate P2T for various typical vision tasks, like image classification, semantic segmentation, object detection, and instance segmentation. Extensive experiments demonstrate that P2T performs better than all previous CNN- and transformer-based backbone networks for these fundamental vision tasks (see Fig. 1 for comparisons on semantic segmentation).

In summary, our main contributions include:

- We encapsulate pyramid pooling to MHSA, simultaneously reducing the sequence length of image tokens and extracting powerful contextual features.
- We plug our pooling-based MHSA into the vision transformer to build a new backbone network, *i.e.*, P2T, making it flexible and powerful for visual recognition.
- We conduct extensive experiments to demonstrate that, when applied as a backbone network for various scene understanding tasks, P2T achieves substantially better performance than previous CNN- and transformer-based networks.

2 RELATED WORK

2.1 Convolutional Neural Networks

Since AlexNet [1] won the champion in the ILSVRC-2012 competition [10], numerous advanced techniques have been invented for improving CNNs, achieving many successful stories in computer vision. VGG [2] and GoogleNet [3] first try to deepen CNNs for better image recognition. Then, ResNets [4] succeed in building very deep CNNs with the help of residual connections. ResNeXts [27] and Res2Nets [9] improve ResNets [4] by exploring its cardinal operation. DenseNets [5] introduce dense connections that connect each layer to all its subsequent layers for easing optimization. MobileNets [38], [39] decompose a vanilla convolution into a 1×1 convolution and a depthwise separable convolution to build lightweight CNNs for mobile and embedded vision applications. ShuffleNets [40], [41] further reduce the latency of MobileNets [38], [39] by replacing the 1×1 convolution with the grouped 1×1 convolution and the channel shuffle operation. EfficientNet [7] and MnasNet [42] adopt neural architecture search (NAS) to search for optimal network architectures. Since our work focuses on the transformer [15], a comprehensive survey of CNNs is beyond the scope of this paper. Please refer to [43] and [44] for a more extensive survey.

2.2 Vision Transformer

Transformer is initially proposed for machine translation in NLP [15]. Through MHSA, transformer entirely relies on self-attention to model global token-to-token dependencies. Considering that global relationship is also highly required by computer vision tasks, it is a natural idea to adopt transformer for improving vision tasks. However, transformer is designed to process sequence data and thus cannot process images directly. Hence, some researchers use CNNs to
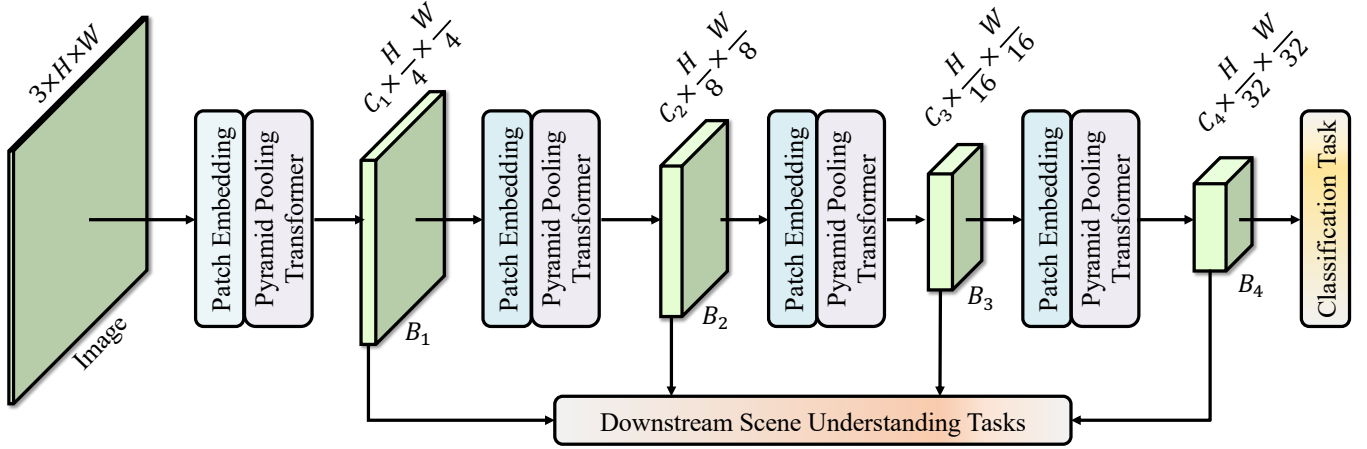
Fig. 2. **Architecture of the proposed P2T network.** We replace the traditional MHSA with our pooling-based MHSA. The feature maps $\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4\}$ can be used for downstream scene understanding tasks.

extract 2D representation that is then flattened and fed into the transformer [16], [17], [45]–[47]. DETR [16] is a milestone in this direction.

Instead of relying on the CNN backbone for feature extraction, Dosovitskiy *et al.* [19] proposed the first vision transformer (ViT). They split an image into small patches, and each patch is viewed as a word/token in an NLP application. Thus, a pure transformer network can be directly adopted with the class token used for image classification. Their method achieved competitive performance on ImageNet [10]. Then, DeiT [48] alleviates the required resources for training ViT [19] via knowledge distillation. T2T-ViT [49] proposes to split an image with overlapping to better preserve local structures. CvT [31] introduces depthwise convolution for generating the query, key, and value in the computation of MHSA. CPVT [50] proposes to replace the absolute positional encoding with the conditional positional encoding via a depthwise convolution. Some efforts are contributed to building the pyramid structure for the vision transformer using pooling operations [20]–[23]. Among them, PVT [21] and MViT [23] first adopt the single pooling operation to reduce the number of tokens when computing MHSA. In this way, they actually conduct token-to-region relationship modeling, not the expected token-to-token modeling. Since this paper also resolves the long sequence length problem through pooling, we adopt PVT [21] and MViT [23] as strong baselines in this paper. Swin Transformer [22] reduces the computational load of MHSA by computing it within small windows. However, Swin Transformer gradually achieves global relationship modeling by window shift, somewhat like CNNs that enlarge the receptive field by stacking more layers [33]. Hence, we think that Swin Transformer [22] sacrifices an essential characteristic of the vision transformer, *i.e.*, *direct global relationship modeling*.

Different from PVT [21] and MViT [23] where the pooled feature extracted by a single pooling operation seems less powerful, we adapt the idea of pyramid pooling to the vision transformer, simultaneously reducing the sequence length and learning powerful contextual representations. With more powerful representations, it is intuitive that pyramid pooling may work better than single pooling for

computing self-attention in MHSA. Pyramid pooling is very efficient and thus will only induce negligible computational cost. Experiments show that the proposed P2T performs substantially better performance than previous CNN- and transformer-based networks. Besides, our design is also compatible with other transformer techniques such as patch embedding [51], positional encoding [50], and feed-forward network [24], [52], [53].

2.3 Pyramid Pooling

In computer vision, pyramid pooling is a long-history and widely-acknowledged technique for extracting feature presentations. Before the renaissance of deep CNNs [1], there emerged several well-known works that applied pyramid pooling for recognizing natural scenes [34], [35]. Inspired by [34], [35], He *et al.* [36] introduced pyramid pooling to deep CNNs for image classification and object detection. They adopted several pooling operations to pool the final convolutional feature map of a CNN backbone into several fixed-size maps. These resulting maps are then flattened and concatenated into a fixed-length representation for robust visual recognition. Then, Zhao *et al.* [37] applied pyramid pooling for semantic segmentation. Instead of flattening in [36], they upsampled the pooled fixed-size maps into the original size and concatenated the upsampled maps for prediction. Their success suggests the effectiveness of pyramid pooling in dense prediction. After that, pyramid pooling has been widely applied to various vision tasks such as semantic segmentation [37], [54]–[57] and object detection [36], [58]–[60].

Unlike existing literature that explores pyramid pooling in CNNs for specific tasks, we propose to adapt the concept of pyramid pooling to the vision transformer backbone network. With this idea, we first embed the pyramid pooling into the basic pooling-based attention block of our P2T backbone, which can simultaneously reduce the sequence length and learn powerful contextual feature representations. P2T can be easily used by various vision tasks for feature representation learning, while previous works about pyramid pooling are limited to a specific vision task. Extensive experiments on image classification, semantic segmentation,
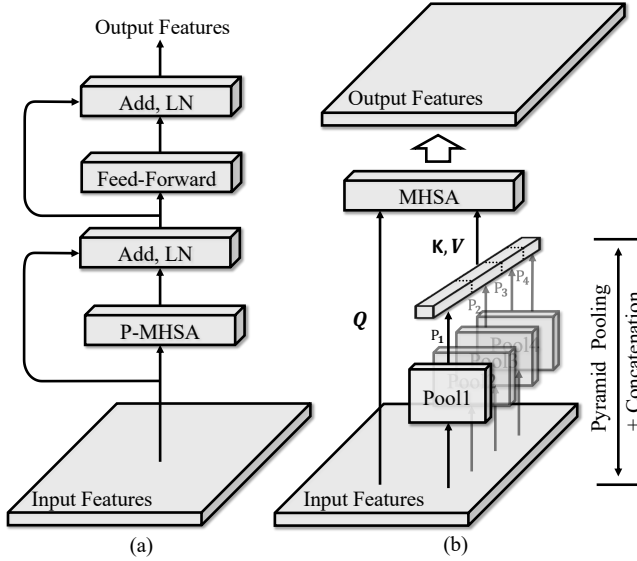
Fig. 3. **Illustration of Pyramid Pooling Transformer.** (a) The brief structure of Pyramid Pooling Transformer. (b) The detailed structure of the pooling-based MHSA.

object detection, and instance segmentation demonstrate the superiority of P2T compared with existing CNN- and transformer-based networks. Therefore, this work is distinctive and would benefit the research on various vision tasks.

3 METHODOLOGY

In this section, we first provide an overview of our P2T networks in §3.1. Then, we present the architecture of P2T with pooling-based MHSA in §3.2. Finally, we introduce some implementation details of our networks in §3.3.

3.1 Overview

The overall architecture of P2T is illustrated in Fig. 2. With a natural color image as input, P2T first splits it into $\frac{H}{4} \times \frac{W}{4}$ patches, each flattened to 48 ($4 \times 4 \times 3$) elements. Following [21], we feed these flattened patches to a patch embedding module, which consists of a linear projection layer followed by the addition with a learnable positional encoding. The patch embedding module will expand the feature dimension from 48 to C_1. Then, we stack the proposed pyramid pooling transformer blocks that will be introduced in §3.2. The whole network can be divided into four stages with feature dimensions of C_i ($i = \{1, 2, 3, 4\}$), respectively. Between every two stages, each 2×2 patch group is concatenated and linearly projected from $4 \times C_i$ to C_{i+1} dimension ($i = \{1, 2, 3\}$). In this way, the scales of four stages become $\frac{H}{4} \times \frac{W}{4}$, $\frac{H}{8} \times \frac{W}{8}$, $\frac{H}{16} \times \frac{W}{16}$, and $\frac{H}{32} \times \frac{W}{32}$, respectively. From four stages, we can derive four feature representations $\{\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3, \mathbf{B}_4\}$, respectively. Only $\mathbf{B}_4$ will be used for final prediction for image classification, while all pyramid features can be utilized for downstream scene understanding tasks.

3.2 Pyramid Pooling Transformer

Pyramid pooling has been widely used in many scene understanding tasks collaborating with CNNs [36], [37], [61]–[68]. However, existing literature usually applies pyramid

pooling on top of CNN backbones for extracting global and contextual information for a specific task. In contrast, this paper is the first to explore pyramid pooling in transformers and backbone networks, targeting for improving various scene understanding tasks generically. To this end, we adapt the idea of pyramid pooling to the transformer, simultaneously reducing the computational load of MHSA and capturing rich contextual information.

Let us continue by introducing the proposed P2T, the structure of which is illustrated in Fig. 3 (a). The input first passes through the pooling-based MHSA, whose output is added with the residual identity, followed by LayerNorm [69]. Like the traditional transformer block [19], [21], [48], a feed-forward network (FFN) follows for feature projection. A residual connection and LayerNorm [69] are applied again. The above process can be formulated as

$$\begin{aligned} \mathbf{X}_{att} &= \text{LayerNorm}(\mathbf{X} + \text{P-MHSA}(\mathbf{X})), \\ \mathbf{X}_{out} &= \text{LayerNorm}(\mathbf{X}_{att} + \text{FFN}(\mathbf{X}_{att})), \end{aligned} \quad (1)$$

where $\mathbf{X}$, $\mathbf{X}_{att}$, and $\mathbf{X}_{out}$ are the input, the output of MHSA, and the output of the transformer block, respectively. **P-MHSA** is the abbreviation of pooling-based MHSA.

3.2.1 Pooling-based MHSA

Here, we present the design of our pooling-based MHSA. Its structure is shown in Fig. 3 (b). First, the input $\mathbf{X}$ is reshaped into the 2D space. Then, we apply multiple average pooling layers with various ratios onto the reshaped $\mathbf{X}$ to generate pyramid feature maps, like

$$\begin{aligned} \mathbf{P}_1 &= \text{AvgPool}_1(\mathbf{X}), \\ \mathbf{P}_2 &= \text{AvgPool}_2(\mathbf{X}), \\ &\cdots, \\ \mathbf{P}_n &= \text{AvgPool}_n(\mathbf{X}), \end{aligned} \quad (2)$$

where $\{\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_n\}$ denote the generated pyramid feature maps and n is the number of pooling layers. Next, we feed pyramid feature maps to the depthwise convolution for relative positional encoding:

$$\mathbf{P}_i^{enc} = \text{DWConv}(\mathbf{P}_i) + \mathbf{P}_i, \quad i = 1, 2, \cdots, n, \quad (3)$$

where $\text{DWConv}(\cdot)$ indicates the depthwise convolution with the kernel size 3×3, and $\mathbf{P}_i^{enc}$ is $\mathbf{P}_i$ with the relative positional encoding. Since $\mathbf{P}_i$ is the pooled feature, the operation in Equ. 3 only has a little computational cost. After that, we flatten and concatenate these pyramid feature maps:

$$\mathbf{P} = \text{LayerNorm}(\text{Concat}(\mathbf{P}_1^{enc}, \mathbf{P}_2^{enc}, ..., \mathbf{P}_n^{enc})), \quad (4)$$

where the flattening operation is omitted for simplicity. In this way, $\mathbf{P}$ can be a shorter sequence than the input $\mathbf{X}$ if pooling ratios are large enough. Besides, $\mathbf{P}$ contains the contextual abstraction of the input $\mathbf{X}$ and can thus serve as a strong substitute for the input $\mathbf{X}$ when computing MHSA.

Suppose the query, key, and value tensors in MHSA [19] are $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, respectively. Instead of using traditional

$$(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = (\mathbf{X}\mathbf{W}^q, \mathbf{X}\mathbf{W}^k, \mathbf{X}\mathbf{W}^v), \quad (5)$$

we propose to use

$$(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = (\mathbf{X}\mathbf{W}^q, \mathbf{P}\mathbf{W}^k, \mathbf{P}\mathbf{W}^v), \quad (6)$$

TABLE 1
Detailed settings of the proposed P2T. The parameters of building blocks are shown in brackets, with the numbers of blocks stacked. For the first stage, we apply a 7×7 convolution with C output channels and a stride of S for patch embedding. Each IRB uses an expansion ratio of E. For simplicity, we omit the patch embedding operation, *i.e.*, a 3×3 convolution with a stride of $S = 2$, after the t-th stage ($t = \{2, 3, 4\}$). "#Params" refers to the number of parameters. "#Flops" denotes the computational cost with the input size of 224×224.

Stage	Input Size	Operator	P2T-Tiny	P2T-Small	P2T-Base	P2T-Large
1	224×224	7×7 conv.	$C = 48, S = 4$	$C = 64, S = 4$		
2	56×56	P-MHSA IRB	$\begin{array}{c} C = 48 \\ E = 8 \end{array} \times 2$	$\begin{array}{c} C = 64 \\ E = 8 \end{array} \times 2$	$\begin{array}{c} C = 64 \\ E = 8 \end{array} \times 3$	$\begin{array}{c} C = 64 \\ E = 8 \end{array} \times 3$
3	28×28	P-MHSA IRB	$\begin{array}{c} C = 96 \\ E = 8 \end{array} \times 2$	$\begin{array}{c} C = 128 \\ E = 8 \end{array} \times 2$	$\begin{array}{c} C = 128 \\ E = 8 \end{array} \times 4$	$\begin{array}{c} C = 128 \\ E = 8 \end{array} \times 8$
4	14×14	P-MHSA IRB	$\begin{array}{c} C = 240 \\ E = 4 \end{array} \times 6$	$\begin{array}{c} C = 320 \\ E = 4 \end{array} \times 9$	$\begin{array}{c} C = 320 \\ E = 4 \end{array} \times 18$	$\begin{array}{c} C = 320 \\ E = 4 \end{array} \times 27$
5	7×7	P-MHSA IRB	$\begin{array}{c} C = 384 \\ E = 4 \end{array} \times 3$	$\begin{array}{c} C = 512 \\ E = 4 \end{array} \times 3$	$\begin{array}{c} C = 512 \\ E = 4 \end{array} \times 3$	$\begin{array}{c} C = 640 \\ E = 4 \end{array} \times 3$
	1×1	-	Global Average Pooling, 1000-d FC, Softmax			
#Params			11.6M	24.1M	36.1M	54.5M
#Flops			1.8G	3.7G	6.5G	9.8G

in which $\mathbf{W}^q$, $\mathbf{W}^k$, and $\mathbf{W}^v$ denote the weight matrices of linear transformations for generating the query, key, and value tensors, respectively. Then, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are fed into the attention module to compute the attention $\mathbf{A}$, which can be formulated as below:

$$\mathbf{A} = \text{Softmax}(\frac{\mathbf{Q} \times \mathbf{K}^{\mathbf{T}}}{\sqrt{d_K}}) \times \mathbf{V}, \qquad (7)$$

where d_K is the channel dimension of $\mathbf{K}$, and $\sqrt{d_K}$ can serve as an approximate normalization. The Softmax function is applied along the rows of the matrix. Equ. 7 omits the concept of multiple heads [15], [19] for simplicity.

Since $\mathbf{K}$ and $\mathbf{V}$ have a smaller length than $\mathbf{X}$, the proposed P-MHSA is more efficient than traditional MHSA [15], [19]. Besides, since $\mathbf{K}$ and $\mathbf{V}$ contains highly-abstracted multi-scale information, the proposed P-MHSA has a stronger capability in global contextual dependency modeling, which is helpful for scene understanding [36], [37], [62], [65]–[68]. From a different perspective, pyramid pooling is usually used as an effective technique connected upon backbone networks; in contrast, this paper first exploits pyramid pooling within backbone networks through transformers, thus providing powerful feature representation learning for scene understanding. With the above analyses, P-MHSA is expected to be more efficient and more effective than traditional MHSA [15], [19].

Analysis of computational complexity. As described in Equ. 2, the proposed pooling-based attention leverages several pooling operations to generate pyramid feature maps. The pyramid pooling operation only has negligible $O(NC)$ computational complexity, where N and C represent the sequence length and the feature dimension, respectively. Hence, the computational complexity for computing self-attention can be formulated as

$$O(\text{P-MHSA}) = (N + 2M)C^2 + 2NMC, \qquad (8)$$

where M is the concatenated sequence length of all pooled features. For the default pooling ratios of $\{12, 16, 20, 24\}$, we have $M \approx \frac{N}{66.3} \approx \frac{N}{8^2}$, which is comparable with the computational cost of MHSA in PVT [21].

3.2.2 Feed-Forward Network

Feed-Forward Network (FFN) is an essential component of transformers for feature enhancement [15], [70]. Previous transformers usually apply an MLP as the FFN [15], [19], [21] and entirely rely on attention to capture inter-pixel dependencies. Though effective, this architecture is not good at learning 2D locality, which plays a critical role in scene understanding. To this end, we follow [52], [53] to insert the depthwise convolution into FFN so that the resulting transformer can inherit the merits of both transformer (*i.e.*, long-range dependency modeling) and CNN (*i.e.*, 2D locality). Specifically, we adopt the Inverted Bottleneck Block (IRB), proposed in MobileNetV2 [39], as the FFN.

To adapt IRB for the vision transformer, we first transform the input sequence $\mathbf{X}_{att}$ to a 2D feature map $\mathbf{X}_{att}^I$:

$$\mathbf{X}_{att}^I = \text{Seq2Image}(\mathbf{X}_{att}), \qquad (9)$$

where $\text{Seq2Image}(\cdot)$ is to reshape the 1D sequence to a 2D feature map. Given the input $\mathbf{X}_{att}^I$, IRB can be directly applied, like

$$\begin{aligned} \mathbf{X}_{\text{IRB}}^1 &= \text{Act}(\mathbf{X}_{att}^I \mathbf{W}_{\text{IRB}}^1), \\ \mathbf{X}_{\text{IRB}}^{\text{out}} &= \text{Act}(\text{DWConv}(\mathbf{X}_{\text{IRB}}^1))\mathbf{W}_{\text{IRB}}^2, \end{aligned} \qquad (10)$$

where $\mathbf{W}_{\text{IRB}}^1$, $\mathbf{W}_{\text{IRB}}^2$ indicate the weight matrices of 1×1 convolutions, "Act" indicates the nonlinear activation function, $\mathbf{X}_{\text{IRB}}^{\text{out}}$ is the output of IRB. Since $\mathbf{X}_{\text{IRB}}^{\text{out}}$ is a 2D feature map, we finally transform it to a 1D sequence:

$$\mathbf{X}_{\text{IRB}}^S = \text{Image2Seq}(\mathbf{X}_{\text{IRB}}^{\text{out}}), \qquad (11)$$

where $\text{Image2Seq}(\cdot)$ is the operation that reshapes the 2D feature map to a 1D sequence. $\mathbf{X}_{\text{IRB}}^S$ is the output of FFN, with the same shape as $\mathbf{X}_{att}$.

3.3 Implementation Details

P2T with different depths. Following previous backbone architectures [4], [21], [22], [24], [27], we build P2T with different depths via stacking the different number of pyramid pooling transformers at each stage. In this manner, we

propose four versions of P2T, *i.e.*, P2T-Tiny, P2T-Small, P2T-Base, and P2T-Large with similar numbers of parameters to ResNet-18 [4], ResNet-50 [4], ResNet-101 [4], and PVT-Large [21], respectively. Each head of P-MHSA has 64 feature channels except that each head has 48 feature channels in P2T-Tiny. Other configurations for different versions of P2T are shown in Table 1.

Pyramid pooling settings. We empirically set the number of parallel pooling operations in P-MHSA as 4. At different stages, the pooling ratios of pyramid pooling transformers are different. The pooling ratios for the first stage are empirically set as $\{12, 16, 20, 24\}$. Pooling ratios in each next stage are divided by 2 except that, in the last stage, they are set as $\{1, 2, 3, 4\}$. In each transformer block, all depthwise convolutions (Equ. 3) in P-MHSA share the same parameters.

Other settings. Although a larger kernel size (*e.g.*, 5×5) of depthwise convolution (in Equ. 3) can bring better performance, the kernel size of all depthwise convolutions is set to 3×3 for efficiency. We choose Hardswish [71] as the non-linear activation function because it saves much memory compared with GELU [72]. Hardswish [71] also empirically works well. Same with PVT2 [29], we apply overlapped patch embedding. That is, we use 3×3 convolution with a stride of 2 for patch embedding from the second to the last stage, while we apply a 7×7 convolution with a stride of 4 for patch embedding in the first stage.

4 EXPERIMENTS

We first introduce the experiments on image classification in §4.1. Then we validate P2T's effectiveness on several scene understanding tasks, *i.e.*, semantic segmentation, object detection, and instance segmentation in §4.2, §4.3, and §4.4, respectively. At last, we conduct ablation studies for better understanding our method in §4.5.

4.1 Image Classification

Image classification is the most common task for evaluating the capability of backbone networks. It aims to assign a class label to each natural image input. Many other tasks build on top of image classification via applying classification networks as the backbones for feature extraction.

Experimental setup. As described in §3.1, only the output feature $\mathbf{B}_4$ of the last stage is utilized here. Following regular CNN networks [4], [5], [9], we append a global average pooling layer and a fully-connected layer on top of $\mathbf{B}_4$ to obtain the final classification scores. We train our network on the ImageNet-1K dataset [10], which has 1.28M training images and 50k validation images. For a fair comparison, we follow PVT [21] to adopt the same training protocols as DeiT [48] (without knowledge distillation), which is a standard choice for training vision transformers. Specifically, we use AdamW [76] as the optimizer, with the initial learning rate of 1e-3, weight decay of 0.05, and a mini-batch of 1024 images. We train P2T for 300 epochs with the cosine learning rate decay strategy. Images are resized to the size of 224×224 for training and testing. Models are warmed up

TABLE 2
Image classification results on the ImageNet-1K dataset [10].
"Top-1" indicates the top-1 accuracy rate. "*" indicates the results with knowledge distillation [48]. "#P (M)" denotes the number of parameters (M). Both the number of computational cost (GFlops) and running speed (frames per second, FPS) are reported with the default 224×224 input size for each network except that the speed of ViT-B [19] is tested with the input size of 384×384. FPS is tested on a single RTX 2070 GPU. The results of the proposed P2T are marked in **bold**.

Method	#P (M) ↓	GFlops ↓	Top-1 (%) ↑	FPS ↑
ResNet-18 [4]	11.7	1.8	68.5	1410
DeiT-Tiny/16* [48]	5.7	1.3	72.2	1212
ViL-Tiny [73]	6.7	1.3	76.7	441
PVT-Tiny [21]	13.2	1.9	75.1	608
PVTv2-B1 [29]	13.1	2.1	78.7	502
P2T-Tiny (Ours)	**11.6**	**1.8**	**79.8**	**473**
ResNet-50 [4]	25.6	4.1	78.5	483
ResNeXt-50-32x4d [27]	25.0	4.3	79.5	407
Res2Net-50 [9]	25.7	4.5	80.3	430
DeiT-Small/16* [48]	22.1	4.6	79.9	489
PVT-Small [21]	24.5	3.8	79.8	336
T2T-ViT$_t$-14 [49]	21.5	5.2	80.7	305
Swin-T [22]	29.0	4.5	81.3	349
Twins-SVT-S [28]	24.0	2.9	81.7	439
ViL-Small [73]	25.0	4.9	82.4	187
PVTv2-B2 [29]	25.4	4.0	82.0	284
P2T-Small (Ours)	**24.1**	**3.7**	**82.4**	**284**
ResNet-101 [4]	44.7	7.9	79.8	288
ResNeXt-101-32x4d [27]	44.2	8.0	80.6	228
Res2Net-101 [9]	45.2	8.3	81.2	265
PVT-Medium [21]	44.2	6.7	81.2	216
T2T-ViT$_t$-19 [49]	39.2	8.4	81.4	202
Swin-S [22]	50.0	8.7	83.0	207
ViL-Medium [73]	40.4	8.7	83.5	114
MViT-B-16 [23]	37.0	7.8	83.1	222
PVTv2-B3 [29]	45.2	6.9	83.2	189
P2T-Base (Ours)	**36.1**	**6.5**	**83.5**	**182**
ResNeXt-101-64x4d [27]	83.5	15.6	81.5	147
MViT-B-24 [23]	53.5	10.9	83.0	151
ViL-Base [73]	57.0	13.4	83.7	67
PVT-Large [21]	61.4	9.8	81.7	152
DeiT-Base/16* [48]	86.6	17.6	81.8	161
ViT-Base/16 [19]	86.6	17.6	77.9	49
Swin-B [22]	88.0	15.4	83.3	140
Twins-SVT-L [28]	99.2	14.8	83.3	143
PVTv2-B4 [29]	62.6	10.1	83.6	133
PVTv2-B5 [29]	82.0	11.8	83.8	120
P2T-Large (Ours)	**54.5**	**9.8**	**83.9**	**128**

for the first five epochs. The data augmentation is also the same as [21], [48].

Experimental results. The quantitative comparisons are summarized in Table 2. All models are trained and evaluated with the input size of 224×224 except that we follow the official ViT [19] to train and evaluate it with the input size of 384×384. P2T largely outperforms regular CNN models like ResNets [4] and ResNeXts [27]. For example, although the running time of P2T-Tiny/Small/Base/Large is 2.98/1.70/1.58/1.15 times that of ResNet-18/50/101 [4] and ResNeXt-101-64x4d [27], the top-1 accuracy of P2T-Tiny/Small/Base/Large is 11.3%/3.9%/3.7%/2.4% better than that of ResNet-18/50/101 [4] and ResNeXt-101-64x4d [27], respectively. As can be seen, our P2T also achieves superior results compared with recent state-of-the-art transformer models. For example, P2T-Small/Base/Large is 1.1%/0.5%/0.6%

TABLE 3
Experimental results on the validation set of the ADE20K dataset [14] for semantic segmentation. We replace the backbone of Semantic FPN [26] with various network architectures. The number of GFlops is calculated with the input size of 512×512. FPS is tested on a single RTX 2070 GPU. The results of P2T backbones are marked in **bold**.

Backbone	Semantic FPN [26]			
	#Param (M) ↓	GFlops ↓	mIoU (%) ↑	FPS ↑
ResNet-18 [4]	15.5	31.9	32.9	68
PVT-Tiny [21]	17.0	32.1	35.7	36
PVTv2-B1 [29]	17.8	33.1	41.5	30
P2T-Tiny (Ours)	**15.4**	**31.6**	**43.4**	**31**
ResNet-50 [4]	28.5	45.4	36.7	35
PVT-Small [21]	28.2	42.9	39.8	26
Swin-T [22]	31.9	46	41.5	26
Twins-SVT-S [28]	28.3	37	43.2	27
PVTv2-B2 [29]	29.1	44.1	46.1	21
P2T-Small (Ours)	**27.8**	**42.7**	**46.7**	**24**
ResNet-101 [4]	47.5	64.8	38.8	26
ResNeXt-101-32x4d [27]	47.1	64.6	39.7	20
PVT-Medium [21]	48.0	59.4	41.6	19
Swin-S [22]	53.2	70	45.2	18
Twins-SVT-B [28]	60.4	67	45.3	17
PVTv2-B3 [29]	49.0	60.7	47.3	15
P2T-Base (Ours)	**39.8**	**58.5**	**48.7**	**16**
ResNeXt-101-64x4d [27]	86.4	104.2	40.2	15
PVT-Large [21]	65.1	78.0	42.1	15
Swin-B [22]	91.2	107	46.0	13
Twins-SVT-L [28]	102	103.7	46.7	13
PVTv2-B4 [29]	66.3	79.6	48.6	11
PVTv2-B5 [29]	85.7	89.4	48.9	10
P2T-Large (Ours)	**58.1**	**77.7**	**49.4**	**12**

better than Swin Transformer [22] with fewer network parameters and less computational cost. Although PVTv2 [29] has large improvement over PVT [21], our P2T-Tiny/Small/Base/Large still have 1.1%/0.4%/0.3%/0.3% improvement over PVTv2-B1/B2/B3/B4 [29] with fewer parameters and less computational cost. P2T applies four parallel pooling operations when computing self-attention, still achieving competitive speed with PVTv2 [29]. ViL [73] achieves competitive performance with P2T. Nevertheless, the speed of ViL [73] is much slower than our P2T, and the computational cost of ViL [73] is also much larger than P2T. P2T also largely outperforms ViT [19] and DeiT [48] with much fewer parameters, implying that P2T achieves better performance without a large amount of training data and knowledge distillation. Therefore, P2T is very capable for image classification.

4.2 Semantic Segmentation

Given a natural image input, semantic segmentation aims at assigning a semantic label to each pixel. It is one of the most fundamental dense prediction tasks in computer vision.

Experimental setup. We evaluate P2T and its competitors on the ADE20K [14] dataset. The ADE20K dataset is a challenging scene understanding dataset with 150 fine-grained semantic classes. This dataset has 20000, 2000, and 3302 images for training, validation, and testing, respectively. Following [21], [28], Semantic FPN [26] is chosen as the basic method for a fair comparison. We replace the backbone of Semantic FPN [26] with various network architectures. All backbones of semantic FPN have been pretrained on the ImageNet-1K [10] dataset, and other layers are initialized

using the Xavier method [77]. All networks are trained for 80k iterations. We apply AdamW [76] as the network optimizer, with the initial learning rate of 1e-4 and weight decay of 1e-4. The *poly* learning rate schedule with $\gamma = 0.9$ is adopted. Each mini-batch has 16 images. Images are resized and randomly cropped to 512×512 for training. Synchronized batch normalization across GPUs is also enabled. During testing, images are resized to the shorter side of 512 pixels. Multi-scale testing and flipping are disabled. Following [21], we use the MMSegmentation toolbox [78] to implement the above experiments.

Experimental results. Quantitative comparison results are shown in Table 3. We compare our proposed P2T with ResNets [4], ResNeXts [27], PVT [21], Swin Transformers [22], Twins [28], and PVTv2 [29]. The results of each network are from the official papers or re-implemented using the official training configurations. Benefiting from the pyramid pooling technique, the results of Semantic FPN [26] with our P2T backbone are much better than other CNN and transformer competitors. Typically, P2T-Tiny/Small/Base/Large are 10.5%/10.0%/9.9%/9.2% better than ResNet-18/50/101 [4] and ResNeXt-10-64x4d [27] with fewer parameters and GFlops, respectively. Compared with Swin Transformer [22] which introduces local self-attention with transformers, P2T-Small/Base/Large achieve 5.2%/3.5%/3.4% improvement over Swin-T/S/B [22], respectively, showing that global relationship modeling is significant for visual recognition. Twins [28] combine the local self-attention from Swin Transformers [22] and the global self-attention from PVT [21]. As can be observed, Twins [28] perform better than Swin Transformers [22], suggesting that global self-attention is significant again. Unlike Twins [28], we apply pure global self-attention via pyramid pooling, learning richer contexts. P2T-Small/Base/Large are 3.5%/3.4%/2.7% better than Twins-SVT-S/B/L [28], respectively. PVTv2 [29] is the improved version of PVT [21], serving as the strongest competitor for our P2T. P2T-Tiny/Small/Base/Large are 1.9%/0.6%/1.4%/0.8% better than PVTv2-B1/B2/B3/B4 [29], respectively. Besides, P2T-Tiny/Small/Base/Large always have fewer parameters, less computational cost, and faster speed than the corresponding PVTv2-B1/B2/B3/B4 [29]. At last, we found that P2T-Tiny is 3.2% better than ResNeXt-101-64x4d [27] with twice the speed. Based on the above observations, we can conclude that P2T is very capable for semantic segmentation.

4.3 Object Detection

Object detection is also one of the most fundamental and challenging tasks for decades in computer vision. It aims to detect and recognize instances of semantic objects of certain classes in natural images. Here, we evaluate P2T and its competitors on the MS-COCO [11] dataset.

Experimental setup. MS-COCO [11] is a large-scale challenging dataset for object detection, instance segmentation, and keypoint detection. MS-COCO `train2017` (118k images) and `val2017` (5k images) sets are used for training and validation in our experiments, respectively. RetinaNet [74] is applied as the basic framework because it has been widely acknowledged by this community [21], [22]. Each

TABLE 4
Object detection results with RetinaNet [74] and instance segmentation results with Mask R-CNN [75] on the MS-COCO `val2017` set
[11]. "R" and "X" represent the ResNet [4] and ResNeXt [27], respectively. The number of Flops is computed with the input size of 800×1280.
FPS is tested on a single RTX 2070 GPU. The results of P2T backbones are marked in **bold**.

Backbone	Object Detection			RetinaNet [74]						Instance Segmentation			Mask R-CNN [75]					
	#Param (M)↓	#Flops (G)↓	#FPS ↑	AP↑	AP$_{50}$	AP$_{75}$	AP$_S$↑	AP$_M$	AP$_L$	#Param (M)↓	#Flops (G)↓	#FPS ↑	AP^b↑	AP$^b_{50}$	AP$^b_{75}$	AP^m↑	AP$^m_{50}$	AP$^m_{75}$
R-18 [4]	21.3	190	19.3	31.8	49.6	33.6	16.3	34.3	43.2	31.2	209	17.3	34.0	54.0	36.7	31.2	51.0	32.7
ViL-Tiny [73]	16.6	204	4.2	40.8	61.3	43.6	26.7	44.9	53.6	26.9	223	3.9	41.4	63.5	45.0	38.1	60.3	40.8
PVT-Tiny [21]	23.0	205	10.7	36.7	56.9	38.9	22.6	38.8	50.0	32.9	223	10.0	36.7	59.2	39.3	35.1	56.7	37.3
PVTv2-B1 [29]	23.8	209	8.5	40.2	60.7	42.4	22.8	43.3	54.0	33.7	227	8.0	41.8	64.3	45.9	38.8	61.2	41.6
P2T-Tiny (Ours)	**21.1**	**206**	**9.3**	**41.3**	**62.0**	**44.1**	**24.6**	**44.8**	**56.0**	**31.3**	**225**	**8.8**	**43.3**	**65.7**	**47.3**	**39.6**	**62.5**	**42.3**
R-50 [4]	37.7	239	13.0	36.3	55.3	38.6	19.3	40.0	48.8	44.2	260	11.5	38.0	58.6	41.4	34.4	55.1	36.7
PVT-Small [21]	34.2	261	7.7	40.4	61.3	43.0	25.0	42.9	55.7	44.1	280	7.0	40.4	62.9	43.8	37.8	60.1	40.3
Swin-T [22]	38.5	248	9.7	41.5	62.1	44.2	25.1	44.9	55.5	47.8	264	8.8	42.2	64.6	46.2	39.1	61.6	42.0
ViL-Small [73]	35.7	292	3.4	44.2	65.2	47.6	28.8	48.0	57.8	45.0	310	3.2	44.9	67.1	49.3	41.0	64,2	44.1
Twins-SVT-S [28]	34.3	236	8.5	43.0	64.2	46.3	28.0	46.4	57.5	44.0	254	7.7	43.4	66.0	47.3	40.3	63.2	43.4
PVTv2-B2 [29]	35.1	266	5.8	43.8	64.8	46.8	26.0	47.6	59.2	45.0	285	5.4	45.3	67.1	49.6	41.2	64.2	44.4
P2T-Small (Ours)	**33.8**	**260**	**7.4**	**44.4**	**65.3**	**47.6**	**27.0**	**48.3**	**59.4**	**43.7**	**279**	**6.7**	**45.5**	**67.7**	**49.8**	**41.4**	**64.6**	**44.5**
R-101 [4]	56.7	315	9.8	38.5	57.8	41.2	21.4	42.6	51.1	63.2	336	9.1	40.4	61.1	44.2	36.4	57.7	38.8
X-101-32x4d [27]	56.4	319	8.5	39.9	59.6	42.7	22.3	44.2	52.5	62.8	340	7.9	41.9	62.5	45.9	37.5	59.4	40.2
PVT-Medium [21]	53.9	349	5.7	41.9	63.1	44.3	25.0	44.9	57.6	63.9	367	5.3	42.0	64.4	45.6	39.0	61.6	42.1
Swin-S [22]	59.8	336	7.1	44.5	65.7	47.5	27.4	48.0	59.9	69.1	354	6.6	44.8	66.6	48.9	40.9	63.4	44.2
PVTv2-B3 [29]	55.0	354	4.5	45.9	66.8	49.3	28.6	49.8	61.4	64.9	372	4.2	47.0	68.1	51.7	42.5	65.7	45.7
P2T-Base (Ours)	**45.8**	**344**	**5.0**	**46.1**	**67.5**	**49.6**	**30.2**	**50.6**	**60.9**	**55.7**	**363**	**4.7**	**47.2**	**69.3**	**51.6**	**42.7**	**66.1**	**45.9**
X-101-64x4d [27]	95.5	473	6.2	41.0	60.9	44.0	23.9	45.2	54.0	101.9	493	5.7	42.8	63.8	47.3	38.4	60.6	41.3
PVT-Large [21]	71.1	450	4.4	42.6	63.7	45.4	25.8	46.0	58.4	81.0	469	4.1	42.9	65.0	46.6	39.5	61.9	42.5
Twins-SVT-B [28]	67.0	376	5.1	45.3	66.7	48.1	28.5	48.9	60.6	76.3	395	4.6	45.2	67.6	49.3	41.5	64.5	44.8
PVTv2-B4 [29]	72.3	457	3.4	46.1	66.9	49.2	28.4	50.0	62.2	82.2	475	3.2	47.5	68.7	52.0	42.7	66.1	46.1
PVTv2-B5 [29]	91.7	514	3.2	46.2	67.1	49.5	28.5	50.0	62.5	101.6	532	3.0	47.4	68.6	51.9	42.5	65.7	46.0
P2T-Large (Ours)	**64.4**	**449**	**3.8**	**47.2**	**68.4**	**50.9**	**32.4**	**51.6**	**62.2**	**74.0**	**467**	**3.5**	**48.3**	**70.2**	**53.3**	**43.5**	**67.3**	**46.9**

mini-batch has 16 images with an initial learning rate of 1e-4. Following the popular MMDetection toolbox [79], we train each network for 12 epochs, and the learning rate is divided by 10 after 8 and 11 epochs. The network optimizer is AdamW [76], a popular optimizer for training transformers. The weight decay is set as 1e-4. During training and testing, the shorter side of input images is resized to 800 pixels. The longer side will keep the ratio of the images within 1333 pixels. In the training stage, only random horizontal flipping is used for data augmentation. Standard COCO API is utilized for evaluation, and we report results in terms of AP, AP$_{50}$, AP$_{75}$, AP$_S$, AP$_M$, and AP$_L$ metrics. AP$_S$, AP$_M$, and AP$_L$ mean AP scores for small, medium, and large objects defined in [11], respectively. AP is usually viewed as the primary metric. For each metric, larger scores indicate better performance. We also report the number of parameters and the computational cost for reference.

Experimental results. The evaluation results on the MS-COCO dataset are summarized in the left part of Table 4. The results of other networks are from the official papers or re-implemented using the official configurations. The below discussion refers to the metric of AP if not stated. We can observe that our P2T achieves the best performance under all tiny/small/large complexity settings. For example, P2T-Small achieves 2.9%, 1.4%, and 0.6% higher AP over Swin-T [22], Twins-SVT-S [28], and PVTv2-B2 [29], respectively. P2T-Tiny is 1.1% better than PVTv2 [29]. Compared with ViL [73], P2T-Tiny/Small are 0.5% and 0.2% better than ViL-Tiny/Small [73], respectively. Note that ViL [73] runs at a much slower speed than P2T, as shown in Table 4. With the base complexity setting, P2T-Base outperforms Swin-S

[29] by 1.0% and is 0.2% better than the best competitor PVTv2-B3. With the large complexity setting, P2T-Large achieves 1.1% and 1.9% better AP than PVTv2-B4 [29] and Twins-SVT-B [28], respectively. At all complexity levels, P2T always outperforms PVTv2 [29] with fewer network parameters, less computational cost, and faster speed. P2T-Tiny/Small/Base/Large are 9.5%/8.1%/7.0%/6.2% better than ResNet-18/50/101 [4] and ResNeXt-101-64x4d [27], respectively. Therefore, P2T is very capable for object detection.

4.4 Instance Segmentation

Instance segmentation is another fundamental vision task. It can be regarded as an advanced case of object detection by outputting fine-grained object masks instead of bounding boxes in object detection.

Experimental setup. We evaluate the performance of instance segmentation on the well-known MS-COCO dataset [11]. MS-COCO `train2017` and `val2017` sets are used for training and validation in our experiments. Mask R-CNN [75] is applied as the basic framework using different backbone networks. The training settings are the same as what we use for object detection in §4.3. We report evaluation results for object detection and instance segmentation in terms of AP^b, AP$^b_{50}$, AP$^b_{75}$, AP^m, AP$^m_{50}$, and AP$^m_{75}$ metrics, where "b" and "m" indicate bounding box and mask metrics, respectively. AP^b and AP^m are set as the primary evaluation metrics.

Experimental results. The comparisons between P2T and its competitors are displayed in the right part of

TABLE 5
Ablation studies on multiple pyramid pooling ratios. The "D ratio" indicates the downsampling ratio between the initial sequence length and the downsampled sequence length. "Top-1" denotes the top-1 classification accuracy rate on the ImageNet-1K validation set [10], and "mIoU" indicates the results for semantic segmentation on the ADE20K dataset [14].

No.	Pooling Ratio(s)	D Ratio ↑	Top-1 (%) ↑	mIoU (%) ↑
1	24	576	70.6	27.5
2	16	256	72.5	33.0
3	12	144	73.9	34.3
4	8	64	73.9	34.4
5	12, 24	115	74.4	34.8
6	12, 16, 20, 24	66	74.7	35.7

TABLE 6
Ablation studies for replacing single pooling operation with multiple pooling operations at different stages. Since both stage 2 and 3 of our network only have two basic blocks, we merge them into one choice. "Top-1" denotes the top-1 classification accuracy rate on the ImageNet-1K validation set [10], and "mIoU" is the results for semantic segmentation on the ADE20K dataset [14].

No.	Stage # of Network			Top-1 (%) ↑	mIoU ↑
	[2, 3]	4	5		
1				73.9	34.4
2	✔			74.1	34.9
3	✔	✔		74.5	35.5
4	✔	✔	✔	74.7	35.7

TABLE 7
Ablation studies on the choices of pooling operations. We can see that other choices work worse than average pooling. "Top-1" indicates the top-1 accuracy rate on the ImageNet-1K dataset [10] for image classification. "mIoU" is the mean IoU rate on the ADE20K dataset [14] for semantic segmentation.

Pooling Type	Top-1 (%) ↑	mIoU (%) ↑
Average Pooling	74.7	35.7
Max Pooling	73.0	33.2
Convolution	73.8	35.5

TABLE 8
Ablation study on the fixed pooled size. GFlops is computed with an input size of 512×512 for the semantic segmentation model, *i.e.*, Semantic FPN [26]. Memory (Mem) denotes the training GPU memory usage for Semantic FPN [26] with a batch size of 2. "Top-1" and "mIoU" indicate the top-1 classification accuracy on ImageNet-1K [10] and segmentation mIoU on ADE20K [14], respectively.

Pooling Operation	GFlops ↓	Mem (GB) ↓	Top-1 (%) ↑	mIoU (%) ↑
Fixed Pooled ratios	41.6	3.3	74.7	35.7
Fixed Pooled Sizes	38.9	2.9	74.4	33.3

4.5 Ablation Studies

Experimental setup. In this section, we perform ablation studies to analyze the efficacy of each design choice in P2T. We evaluate the performance of model variants on semantic segmentation and image classification. Due to the limited computational resources, we only train each model variant on the ImageNet dataset [10] for 100 epochs, while other training settings keep the same as in §4.1. Then, we fine-tune the ImageNet-pretrained model on the ADE20K dataset [14] with the same training settings as in §4.2.

Multiple pyramid pooling ratios. To validate the significance of using multiple pooling ratios, we conduct experiments to evaluate the performance of P2T with one/two/four parallel pooling operations. The baseline is P2T-Small without relative positional encoding, IRB, and overlapping patch embedding. The results are shown in Table 5. As can be seen, the single pooling operation with a large pooling ratio (*e.g.*, 16, 24) has a large squeezed ratio for the sequence length. Still, it results in very poor performance for both image classification and semantic segmentation. However, when the single pooling operation is with a pooling ratio ≤ 12, the performance will be saturated if we further decrease the pooling ratio. When we adopt two parallel pooling operations, even with a high squeezed ratio, the performance still becomes better for both image classification and semantic segmentation. When we have four parallel pooling operations, we can derive the best performance with the comparable squeezed ratio for the pooling ratio of 8 (the setting in PVT [21]).

Significance of pyramid pooling for different stages. We perform ablation studies of the pyramid pooling design of P2T. for different stages. Since stage 1 only contains convolutions for downsampling, we do not perform such ablation study at stage 1. The baseline is same with the last ablation studies. The pooling ratio of single pooling operation is set to 8 for ensuring comparable downsampling ratios. Results are shown in Table 6. We can observe pyramid pooling can improve the performance at all stages. The performance becomes higher when more stages are applied with multiple pooling operations. From the results, the improvement on applying multiple pooling operations on stage 4 (No. 3 of Table 6) is larger than that on other stages (No. 2, 4 of Table 6), because stage 4 has more basic blocks than the summation of stage [2,3] and stage 5.

Pooling operation choices. We conduct experiments for different pooling operations, as shown in Table 7. There are

Table 4. P2T achieves the best performance consistently compared to existing CNN and transformer backbone networks. Compared with transformer-based backbones, P2T achieves the best performance at all complexity levels. Regarding the bounding box metric AP^b, P2T-Small/Base are 3.3%/2.4% better than Swin-T/S [22], and P2T-Small/Large are 2.1%/3.1% better than Twins-SVT-S/B [28]. P2T-Tiny/Small/Base/Large are 1.5%/0.2%/0.2%/0.8% better than PVTv2-B1/B2/B3/B4 [29] with fewer parameters, less computational cost, and faster speed, respectively. In terms of the mask metric AP^m, we also observe similar improvements as observed using bounding box metrics. Compared with ResNet-based backbones, P2T significantly outperforms ResNets [4] and ResNeXts [27] at all complexity levels. It is also surprising that our lightest P2T-Tiny is 0.5% and 1.2% better than ResNeXt-101-64x4d [27] in terms of bounding box and mask metrics, respectively. Therefore, P2T is very capable for instance segmentation.

TABLE 9
Ablation study on the relative positional encoding (RPE), IRB, and overlapping patch embedding (OPE). "Top-1" and "mIoU" indicate the top-1 classification accuracy on ImageNet-1K [10] and segmentation mIoU on ADE20K [14], respectively.

RPE	IRB	OPE	Top-1 (%) ↑	mIoU (%) ↑
			74.7	35.7
✔			76.4	37.4
✔	✔		79.5	42.7
✔	✔	✔	79.7	44.1

three typical choices, *i.e.*, max pooling, depthwise convolution, and the default average pooling. The kernel size of depthwise convolution is the same as max/average pooling to keep the same downsampling rate. It is obvious that different pooling types do not affect the computational complexity and they only affect the number of the parameters of the downsampling kernel. Regarding the results of ImageNet classification accuracy [10] and ADE20K segmentation mIoU [14], average pooling is much better than the other two choices. Thus, we apply average pooling as the default pooling choice.

Fixed pooled sizes. When using fixed pooling ratios, the size of the pooled feature map will vary with the input feature map. Here, we try to fix the pooled sizes as {1, 2, 3, 6} for all stages, using adaptive average pooling. The results are shown in Table 8. Compared with our default setting, about 10% of memory usage and 12% of computational cost are saved. However, the top-1 classification accuracy drops by 0.3%, and the semantic segmentation performance is 2.4% lower. Hence, we choose to use fixed pooling ratios rather than fixed pooled sizes.

Selection of activation functions. We use the Hardswish function [71] for nonlinear activation to reduce GPU memory usage in the training phase. Typically, when we train P2T-Small on ImageNet [10] with a batch size of 64, the GPU memory usage of GELU [72] is 10.5GB, which is 3.6GB (+52%) more than that of Hardswish [71]. We also find that there is no significant accuracy decrease if we employ Hardswish [71].

Other design choices. To validate the effectiveness of other design choices like relative positional encoding, IRB, and overlapping patch embedding, we add these components one by one to the baseline. Experimental results are shown in Table 9. As can be seen, relative positional encoding has significant improvement for both image classification and semantic segmentation. With large pooling ratios, the pooled features would have small scales, so relative positional encoding only needs negligible computational cost (5M Flops for the input size of 224×224). An extra depthwise convolution in the feed-forward network, *i.e.*, IRB, also shows significant improvement, demonstrating the necessity of the 2D locality enhancement. We further follow [29] to add overlapping patch embedding, and 0.2%/1.4% improvement is observed for image classification and semantic segmentation, respectively.

5 CONCLUSION

This paper introduces pyramid pooling into MHSA for alleviating the high computational cost of MHSA in the vision transformer. Compared with the strategy of applying a single pooling operation in MHSA [21], [23], our pooling-based MHSA not only reduces the sequence length but also learns powerful contextual representations simultaneously via pyramid pooling. Equipped with the pooling-based MHSA, we construct a new backbone network, called Pyramid Pooling Transformer (P2T). To demonstrate the effectiveness of P2T, we conduct extensive experiments on several fundamental vision tasks, including image classification, semantic segmentation, object detection, and instance segmentation. Experimental results suggest that P2T significantly outperforms previous CNN- and transformer-based backbone networks.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Adv. Neural Inform. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. Learn. Represent.*, 2015.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1–9.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778.

[5] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger, "Convolutional networks with dense connectivity," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[6] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-excitation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, 2020.

[7] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[8] Y.-H. Wu, Y. Liu, J. Xu, J.-W. Bian, Y. Gu, and M.-M. Cheng, "MobileSal: Extremely efficient RGB-D salient object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.

[9] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. H. Torr, "Res2Net: A new multi-scale backbone architecture," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 652–662, 2021.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[13] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 3213–3223.

[14] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20K dataset," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 633–641.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Adv. Neural Inform. Process. Syst.*, 2017, pp. 6000–6010.

[16] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Eur. Conf. Comput. Vis.*, 2020, pp. 213–229.

[17] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[18] R. Hu and A. Singh, "Transformer is all you need: Multimodal multitask learning with a unified transformer," *arXiv preprint arXiv:2102.10772*, 2021.

[19] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Int. Conf. Learn. Represent.*, 2021.

[20] B. Heo, S. Yun, D. Han, S. Chun, J. Choe, and S. J. Oh, "Rethinking spatial dimensions of vision transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 11 936–11 945.

[21] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid Vision Transformer: A versatile backbone for dense prediction without convolutions," in *Int. Conf. Comput. Vis.*, 2021.

[22] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical vision transformer using shifted windows," in *Int. Conf. Comput. Vis.*, 2021, pp. 10 012–10 022.

[23] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, "Multiscale vision transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 6824–6835.

[24] Y. Liu, Y.-H. Wu, G. Sun, L. Zhang, A. Chhatkuli, and L. Van Gool, "Vision transformers with hierarchical attention," *arXiv preprint arXiv:2106.03180*, 2021.

[25] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte, "SwinIR: Image restoration using Swin Transformer," in *Int. Conf. Comput. Vis.*, 2021, pp. 1833–1844.

[26] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 6399–6408.

[27] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 1492–1500.

[28] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, "Twins: Revisiting the design of spatial attention in vision transformers," in *Adv. Neural Inform. Process. Syst.*, 2021, pp. 9355–9366.

[29] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "PVTv2: Improved baselines with pyramid vision transformer," *arXiv preprint arXiv:2106.13797*, 2021.

[30] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand *et al.*, "Findings of the 2014 workshop on statistical machine translation," in *The Workshop on Statistical Machine Translation*, 2014, pp. 12–58.

[31] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "CvT: Introducing convolutions to vision transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 22–31.

[32] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze, "LeViT: a vision transformer in ConvNet's clothing for faster inference," in *Int. Conf. Comput. Vis.*, 2021, pp. 12 259–12 269.

[33] Q. Han, Z. Fan, Q. Dai, L. Sun, M.-M. Cheng, J. Liu, and J. Wang, "Demystifying local vision transformer: Sparse connectivity, weight sharing, and dynamic weight," *arXiv preprint arXiv:2106.04263*, 2021.

[34] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Int. Conf. Comput. Vis.*, vol. 2. IEEE, 2005, pp. 1458–1465.

[35] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene cate-gories," in *IEEE Conf. Comput. Vis. Pattern Recog.*, vol. 2. IEEE, 2006, pp. 2169–2178.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015.

[37] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 2881–2890.

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[39] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 4510–4520.

[40] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet v2: Practical guidelines for efficient CNN architecture design," in *Eur. Conf. Comput. Vis.*, 2018, pp. 116–131.

[41] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 6848–6856.

[42] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 2820–2828.

[43] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[44] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[45] H. Wang, Y. Zhu, H. Adam, A. Yuille, and L.-C. Chen, "MaX-DeepLab: End-to-end panoptic segmentation with mask transformers," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021, pp. 5463–5474.

[46] R. Liu, Z. Yuan, T. Liu, and Z. Xiong, "End-to-end lane shape prediction with transformers," in *IEEE Winter Conf. Appl. Comput. Vis.*, 2021, pp. 3694–3702.

[47] J. Hu, L. Cao, L. Yao, S. Zhang, Y. Wang, K. Li, F. Huang, R. Ji, and L. Shao, "ISTR: End-to-end instance segmentation with transformers," *arXiv preprint arXiv:2105.00637*, 2021.

[48] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Int. Conf. Mach. Learn.*, 2021, pp. 10 347–10 357.

[49] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token ViT: Training vision transformers from scratch on ImageNet," in *Int. Conf. Comput. Vis.*, 2021, pp. 558–567.

[50] X. Chu, Z. Tian, B. Zhang, X. Wang, X. Wei, H. Xia, and C. Shen, "Conditional positional encodings for vision transformers," *arXiv preprint arXiv:2102.10882*, 2021.

[51] Z. Jiang, Q. Hou, L. Yuan, D. Zhou, X. Jin, A. Wang, and J. Feng, "Token labeling: Training a 85.5% top-1 accuracy vision transformer with 56M parameters on ImageNet," *arXiv preprint arXiv:2104.10858*, 2021.

[52] Y. Li, K. Zhang, J. Cao, R. Timofte, and L. Van Gool, "LocalViT: Bringing locality to vision transformers," *arXiv preprint arXiv:2104.05707*, 2021.

[53] K. Yuan, S. Guo, Z. Liu, A. Zhou, F. Yu, and W. Wu, "Incorporating convolution designs into visual transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 579–588.

[54] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, 2017.

[55] M. M. K. Sarker, H. A. Rashwan, F. Akram, S. F. Banu, A. Saleh, V. K. Singh, F. U. Chowdhury, S. Abdulwahab, S. Romani, P. Radeva *et al.*, "Slsdeep: Skin lesion segmentation based on dilated residual and pyramid pooling networks," in *Med. Image. Comput. Comput. Assist. Interv.* Springer, 2018, pp. 21–29.

[56] Y. Yuan, L. Huang, J. Guo, C. Zhang, X. Chen, and J. Wang, "Ocnet: Object context for semantic segmentation," *Int. J. Comput. Vis.*, pp. 1–24, 2021.

[57] X. Lian, Y. Pang, J. Han, and J. Pan, "Cascaded hierarchical atrous spatial pyramid pooling module for semantic segmentation," *Pattern Recognition*, vol. 110, p. 107622, 2021.

[58] D. Yoo, S. Park, J.-Y. Lee, and I. So Kweon, "Multi-scale pyramid pooling for deep convolutional representation," in *Int. Conf. Comput. Vis. Worksh.*, 2015, pp. 71–80.

[59] S.-W. Kim, H.-K. Kook, J.-Y. Sun, M.-C. Kang, and S.-J. Ko, "Parallel feature pyramid network for object detection," in *Eur. Conf. Comput. Vis.*, 2018, pp. 234–250.

[60] Z. Huang, J. Wang, X. Fu, T. Yu, Y. Guo, and R. Wang, "Dc-spp-yolo: Dense connection and spatial pyramid pooling based yolo for object detection," *Information Sciences*, vol. 522, pp. 241–258, 2020.

[61] P. Zhang, D. Wang, H. Lu, H. Wang, and X. Ruan, "Amulet: Aggregating multi-level convolutional features for salient object detection," in *Int. Conf. Comput. Vis.*, 2017, pp. 202–211.

[62] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 5410–5418.

[63] J.-J. Liu, Q. Hou, Z.-A. Liu, and M.-M. Cheng, "Poolnet+: Exploring the potential of pooling for salient object detection," *IEEE TPAMI*, 2022.

[64] Y.-H. Wu, Y. Liu, L. Zhang, W. Gao, and M.-M. Cheng, "Regularized densely-connected pyramid network for salient instance segmentation," *IEEE Trans. Image Process.*, vol. 30, pp. 3897–3907, 2021.

[65] D. Park, K. Kim, and S. Young Chun, "Efficient module based single image super resolution for multiple problems," in *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2018, pp. 882–890.

[66] H. Zhang, V. Sindagi, and V. M. Patel, "Image de-raining using a conditional generative adversarial network," *IEEE Trans. Circ. Syst. Video Technol.*, vol. 30, no. 11, pp. 3943–3956, 2019.

[67] H. Zhang and V. M. Patel, "Densely connected pyramid dehazing network," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 3194–3203.

[68] T. Wang, A. Borji, L. Zhang, P. Zhang, and H. Lu, "A stagewise refinement model for detecting salient objects in images," in *Int. Conf. Comput. Vis.*, 2017, pp. 4019–4028.

[69] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[70] Y. Dong, J.-B. Cordonnier, and A. Loukas, "Attention is not all you need: Pure attention loses rank doubly exponentially with depth," *arXiv preprint arXiv:2103.03404*, 2021.

[71] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for MobileNetV3," in *Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.

[72] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016.

[73] P. Zhang, X. Dai, J. Yang, B. Xiao, L. Yuan, L. Zhang, and J. Gao, "Multi-scale vision Longformer: A new vision transformer for high-resolution image encoding," in *Int. Conf. Comput. Vis.*, 2021, pp. 2998–3008.

[74] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[75] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, 2020.

[76] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Int. Conf. Learn. Represent.*, 2019.

[77] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Int. Conf. Artif. Intell. Stat.*, 2010, pp. 249–256.

[78] M. Contributors, "MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark," https://github.com/open-mmlab/mmsegmentation, 2020.

[79] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu *et al.*, "MMDetection: Open MMLab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.
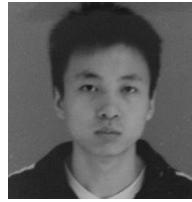
Yun Liu received his bachelor's degree and his doctoral degree from Nankai University in 2016 and 2020, respectively. His Ph.D. supervisor was Prof. Ming-Ming Cheng. Then, he worked with Prof. Luc Van Gool for one and a half years as a postdoctoral scholar at Computer Vision Lab, ETH Zurich. Currently, he is a scientist at Institute for Infocomm Research (I2R), A*STAR. His research interests include computer vision and machine learning.
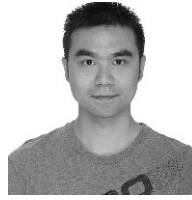
Xin Zhan received his bachelor's and doctoral degrees from USTC in 2010 and 2015, respectively. Currently, he works as a researcher of Alibaba DAMO Academy. His research interests include perception for autonomous driving.

Ming-Ming Cheng received his PhD degree from Tsinghua University in 2012. Then he did two years research fellow with Prof. Philip Torr in Oxford. He is now a professor at Nankai University, leading the Media Computing Lab. His research interests include computer graphics, computer vision, and image processing. He received research awards, including ACM China Rising Star Award, IBM Global SUR Award, and CCF-Intel Young Faculty Researcher Program. He is on the editorial boards of IEEE TPAMI/TIP.

Yu-Huan Wu is currently a Ph.D. candidate with the TMCC, College of Computer Science at Nankai University, supervised by Prof. Ming-Ming Cheng. He received his bachelor's degree from Xidian University in 2018. His research interests include computer vision and machine learning.