P2T：一种用于视觉场景理解的金字塔池化 Transformer

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

Abstract—近期，Transformer 在各项视觉任务已经成功地取得了新的进展。其中一个具有挑战性的 Transformer 应用难点就是由图像词符（Token）序列导致的高计算开支（二次方的复杂度）。目前，普遍采用的解决方式是利用单层池化进行采样，从而减少特征词符的序列长度。但是由单层池化计算得到的池化特征看起来并没有十分强大，所以本文将在池化策略上考虑如何改进已有的视觉 Transformer，从而解决这个问题。为了实现这一点，我们观察到金字塔池化计算方式由于其出色的上下文提取能力，从而在各项视觉任务中取得了很好的效果。然而，金字塔池化在骨干网络的设计中至今并没有被探索过。因此，我们提出将金字塔池化引入到视觉 Transformer 中的多头自注意力模块（Multi-Head Self-Attention，MHSA）里面，既减少了图像词符序列的长度，同时提取到更好的语境特征。基于我们提出的金字塔池化多头注意力，我们提出金字塔池化 Transformer（Pyramid Pooling Transformer，P2T）骨干网络。通过详尽的实验，我们证明当采用 P2T 模块作为骨干网络时，比起传统的基于神经卷积或者 Transformer 的网络，它在图像分类、语义分割、物体检测和实例分割等视觉任务上都体现出了显著的优越性。本文代码发布于 https://github.com/yuhuan-wu/P2T。

Index Terms—Transformer，骨干网络，高效自注意力机制，金字塔池化，场景理解

1 介绍

近十多年来，卷积神经网络（Convolutional Neural Networks，CNNs）已经统治了机器视觉的各个领域并且取得了非常优秀的进展 [4], [10]–[17]。卷积神经网络在许多大型数据集上的视觉任务上都取得了最先进的结果。在另一个平行的领域里面，如自然语言处理（Natural Language Processing，NLP）上，一种流行的技术是 Transformer。Transformer 全部依赖于其自注意力机制来获取大范围的全局相关性，取得了非常耀眼的成功。考虑到全局信息对于视觉任务来说也非常重要，Transformer 也可以被用来解决卷积神经网络的局限性，即卷积神经网络通常只能利用堆叠的方法来增大它的感受野。

许多研究者在如何将 Transformer 用于视觉任务的问题上做了很多的努力 [18]。早期的研究者尝试用 Transformer 来进一步处理由卷积神经网络提取到的深层次特征 [4], [11]，从而能够拟合目标问题 [19]–[21]。Dosovitskiy 等人 [22] 在单纯利用 Transformer 来解决图像分类这个问题上取得了巨大的突破。他们将一张图片给分割成多个小块，并且将每个小块图片当作自然语言处理中的单词（Word）或词符（Token），

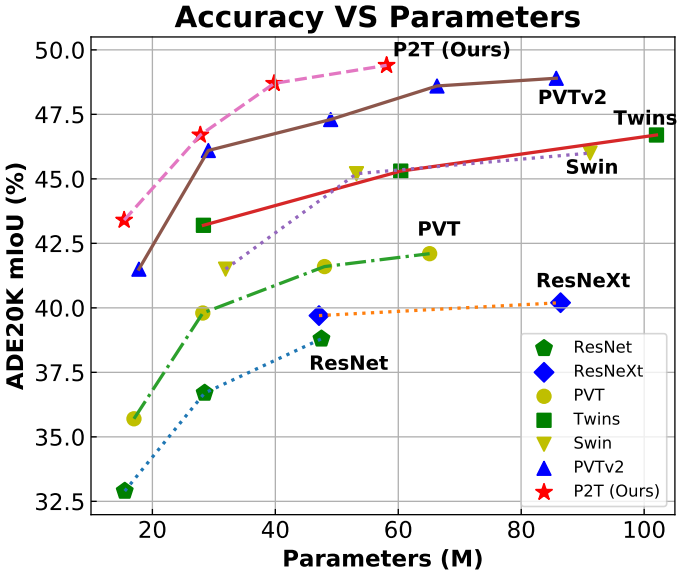


图 1. 在 ADE20K 数据集上语义分割的实验结果 [2]。我们采用 Semantic FPN [3] 作为基本方法，并使用了包括 ResNet [4]、ResNeXt [5]、PVT [6]、Twins [7]、Swin Transformer [8]、PVTv2 [9] 以及我们提出的 P2T 等不同的骨干网络。

因此 Transformer 能够直接地用于对图像的处理。这种简单的方法在 ImageNet [23] 数据集上得到了非常具有竞争力的结果。因此，计算机视觉领域出现了一个新的概念，即视觉 Transformer。在视觉 Transformer 概念出现的短时间内，出现了数量巨大的工作来改进视觉 Transformer [22]，同时也带来了比卷积神经网络更加优秀的性能 [6], [8], [24]–[27]。

- 吴宇寰（Yu-Huan Wu）和程明明（Ming-Ming Cheng）隶属于南开大学计算机学院的媒体计算实验室。
- 刘云（Yun Liu）隶属于新加坡科技研究局（A*STAR）。
- 占新（Xin Zhan）隶属于阿里巴巴达摩院。
- 前面两位作者对本文做了同等的贡献。
- 通讯作者：程明明。(E-mail: cmm@nankai.edu.cn)

- 这篇文章是吴宇寰在阿里巴巴达摩院实习期间完成的。
- 本文是 [1] 的中译版。有关翻译的问题欢迎与译者负责人吴宇寰（wuyuhuan@mail.nankai.edu.cn）进行讨论。

尽管如此，视觉 Transformer 仍然存在一个具有挑战性的问题，即数据序列的长度。当将图像块视为自然语言处理中的词符时，编码得到的序列长度仍然比自然语言处理中要长得多。例如，在自然语言处理中，著名的 WMT2014 英德数据集 [28] 包含了 5000 万个英语单词和 200 万个句子，其平均序列长度为 25。相比之下，在计算机视觉领域，我们通常使用 224×224 图像分辨率的 ImageNet 数据集 [23] 来进行图像分类任务。如果我们使用普遍采用的 4×4 图像块，那么编码的得到的序列长度将为 3136。由于 Transformer 中的多头自注意力模块（Multi-Head Self-Attention, MHSA）的计算复杂度是图像大小的二次方（而不是卷积神经网络中的线性），直接将 Transformer 应用于视觉任务对计算资源的要求很高。为了实现一个可用于图像分类的纯 Transformer 网络，ViT [22] 使用较大的图像块来减少序列长度，例如 16×16 或者 32×32，并在图像分类上取得了巨大的成功。后来，许多 Transformer 工作 [6]–[8], [24]–[26], [29], [30] 通过引入金字塔结构来提高 ViT [22] 的性能。其中，输入层首先使用 4×4 小图像块和然后通过合并相邻图像块来逐渐减小序列的长度。

为了进一步降低多头自注意力模块的计算开支，PVT [6] 和 MViT [25] 在多头自注意力模块的计算中使用了单个池化操作对特征图进行下采样。通过使用池化后的特征，他们模拟的是词符到区域（Token-to-Region）的关系，而不是经常使用的词符对词符（Token-to-Token）的关系。Swin Transformer [8] 提出在建模相互关系的时候，选择只计算小窗口内的多头自注意力模块而不是在整个输入窗口上进行计算。跟卷积神经网络类似，它通过使用移动窗口（Window Shift）的策略以及堆叠更多的层数来逐渐扩大网络的感受野 [31]。尽管如此，视觉 Transformer 的一个基本特征是它的直接全局关系建模，这也是众多研究者从卷积神经网络转移到 Transformer 的原因。

在这篇文章里面，我们考虑如何改进 PVT [6] 和 MViT [25]，因为他们使用单层池化操作提取到的池化特征似乎都不太强大。如果我们能够在压缩特征序列长度的同时获取更强的特征表征，那么我们可能会在各类任务下取得更好的性能。为此，我们注意到金字塔池化 [32]–[35] 是一种具有较长历史的计算机视觉技术，它通过提取上下文信息并利用具有不同感受域的多层池化操作在输入特征上进行多尺度运算。这种简单的技术已在各种下游视觉任务中被证明有效，例如语义分割 [35] 和物体检测 [34]。然而，最近的金字塔池化方法高度依赖于预训练的卷积神经网络骨干，因此它们仅限于一些特定的视觉任务。换言之，金字塔池化技术在具有广泛应用的骨干网络设计中**尚未被探索**。为了弥补这一个差距，我们将金字塔池化应用于视觉 Transformer 模块中，从而减少序列长度并且同时学习到强大的上下文表征。金字塔池化的计算效率也非常高，它给视觉 Transformer 带来的计算开支几乎可以忽略不计。

我们通过提出一个新的 Transformer 骨干网络来实现这一目标，即**金字塔池化** Transformer（Pyramid Pooling Transformer，P2T）。我们将金字塔池化的想法应用于视觉 Transformer 的多头自注意力模块中，不但减少了该模块的计算开支，同时也获取到了丰富的上下文信息。通过将基于新池化的多头自注意力模块用于 Transformer 中，P2T 在特征学习和视觉识别方面的表现都比其他基于单层池化操作的 PVT [6] 和 MViT [25] 更加强大。我们利用各种典型的视觉任务，例如图像分类、语义分割、物体检测和实例分割等基础视觉任等来评估 P2T 的性能。大量实验表明，对于这些基本视觉任务，P2T 的性能优于所有以前基于卷积神经网络和 Transformer 的骨干网络（有关语义分割的比较，请参见图 1）。

总而言之，我们的主要贡献包括：

- 我们将金字塔池化封装到多头自注意力模块中，不但减少了图像特征序列的长度，同时也提取了强大的上下文特征。
- 我们将基于金字塔池化的多头自注意力模块引入到视觉 Transformer 中来构建一种灵活并且对视觉任务有效的新骨干网络，我们称之为金字塔池化 Transformer（Pyramid Pooling Transformer，P2T）。
- 充分的实验证明，当将 P2T 用作各种场景理解任务的骨干网络时，P2T 的性能明显优于以前那些基于卷积神经网络或者 Transformer 的网络。

2 相关工作

2.1 卷积神经网络

自 AlexNet [10] 在 ILSVRC-2012 竞赛 [23] 中获得冠军以来，研究者已经发明了许多先进的技术来改进卷积神经网络，这些努力在计算机视觉领域中已经变成了许多令人印象深刻的模型架构。VGG [11] 和 GoogleNet [12] 首先尝试加深卷积神经网络从而获得更好的图像识别效果。然后，ResNets [4] 在残差连接的帮助下成功构建了非常深的卷积神经网络。ResNeXts [5] 和 Res2Nets [17] 通过探索其基数操作（Cardinal Operation）来改进 ResNets [4]。DenseNets [13] 通过引入了密集连接将每一层连接到其后所有的网络层来实现更好的梯度优化。MobileNets [36], [37] 将普通卷积分解为 1×1 卷积和深度可分离卷积，从而来构建可用于移动和嵌入式视觉应用的轻量级卷积神经网络。ShuffleNets [38], [39] 通过将 1×1 卷积替换为分组的 1×1 卷积和交换通道操作（Channel Shuffle Operation）来进一步减少 MobileNets [36], [37] 的延迟。EfficientNet [15] 和 MnasNet [40] 通过采用神经架构搜索（Neural Architecture Search，NAS）来获取到最优的网络架构。由于我们的工作重点是 Transformer [18]，对卷积神经网络的全面调查超出了本文的范围。如果想了解更广泛的调查，请参考 [41] 和 [42] 的工作。

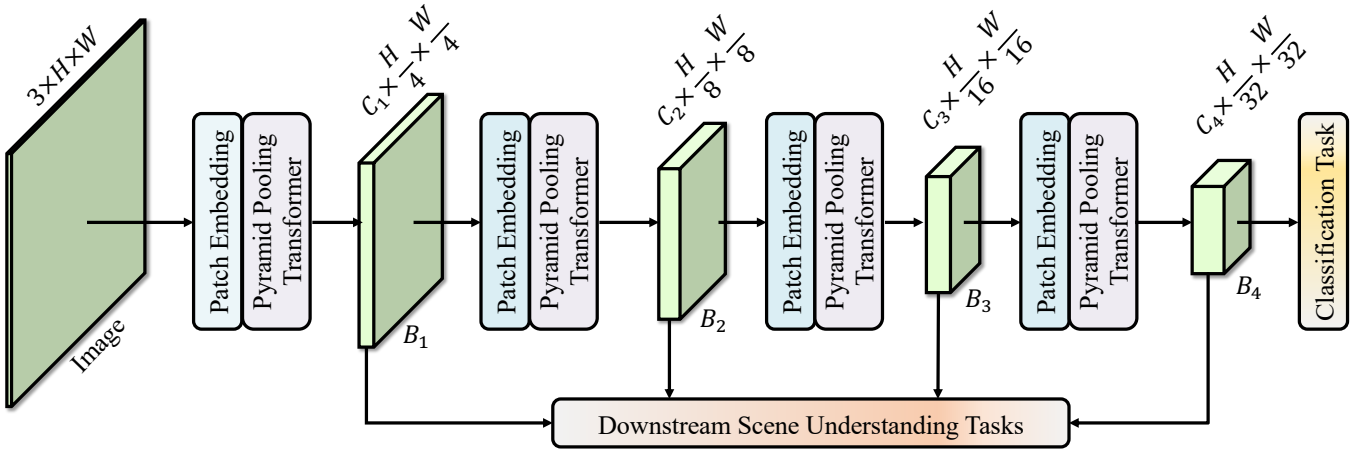


图 2. 本文提出的 P2T 网络架构。我们用本文提出的基于池化的多头自注意力模块代替传统的模块。特征 $\{B_1, B_2, B_3, B_4\}$ 可以用于下游场景理解任务。

2.2 视觉 Transformer

Transformer 的提出最初是用于自然语言处理领域的机器翻译任务 [18]。通过多头自注意力模块，Transformer 完全依赖于自注意力机制对词符和词符之间的依赖关系进行全局建模。考虑到计算机视觉任务对全局关系的要求也很高，因此采用 Transformer 用于改善视觉任务是一个很自然的想法。但是，Transformer 的提出原本是用于处理序列数据，因此无法直接用来处理图像信息。为此，一些研究人员使用卷积神经网络来提取图像二维特征然后扁平化送入 Transformer [19], [20], [43]–[45]。其中，DETR [19] 是这个方向的里程碑式工作。

跟前者依赖于卷积神经网络骨干进行特征提取不同的是，Dosovitskiy 等人 [22] 提出了第一个视觉 Transformer（ViT）。他们将图像分割成一个个小图像块，并把每个图像块视为自然语言处理研究中的单词或词符。因此，只要配合标签词符（Class Token），纯 Transformer 网络可以直接用来处理图像分类任务。他们的方法在 ImageNet 数据集上取得了非常具有竞争力的结果 [23]。然后，DeiT [46] 通过知识蒸馏技术来减轻训练 ViT [22] 所需的资源。T2T-ViT [47] 建议图像分割的时候保留一些重叠部分从而更好地保留图像局部结构。CvT [29] 引入了深度卷积，用于在多头自注意力模块中的查询（Query）、索引（key）和值（Value）的计算。CPVT [48] 建议通过深度卷积将绝对位置编码替换为条件位置编码。一些研究人员提出给视觉 Transformer 构造金字塔结构池化 [6], [8], [24], [25]。其中，PVT [6] 和 MViT [25] 率先采用单层池化操作来减少计算多头自注意力模块计算时的词符数量。但是通过这样方式，他们实际上是进行了“词符-区域”的关系建模，而不是预期的“词符-词符”之间的建模。由于本文还通过池化解决了长序列的难点，因此我们采用了 PVT [6] 和 MViT [25] 作为本文中用于对比的基线方法。Swin Transformer [8] 通过在小窗口内计算多头自注意力模块来减少了所需的计算量。然而，Swin Transformer 通过窗口移位建模来逐渐实现全局相互关系，有点像卷积神经网络通过堆叠更多网络层 [31] 来扩大感受野。我们认为 Swin Transformer [8] 牺牲了视觉 Transformer 的一个基本特征，即*直接全局关系建模*。

PVT [6] 和 MViT [25] 利用单层池化操作提取的池化特征看起来不那么强大。跟他们不一样的是，我们将金字塔池化的想法应用于视觉 Transformer 来减少了序列长度的同时，也学习到了更加有效的上下文特征。有了更加有效的上下文特征，金字塔池化可能比单层池化更好地计算多头自注意力模块中的自注意力关系。金字塔池化计算效率非常高，因此引入带来的计算开支可以忽略不计。实验表明，本文所提出的 P2T 比以前的基于卷积神经网络和基于 Transformer 的网络的性能要好得多。此外，我们的设计还兼容其他 Transformer 技术，比如图像块字典学习（Patch Embedding）[49]、位置编码 [48] 和前馈网络（Feed-forward Network）[26], [50], [51]。

2.3 金字塔池化

在计算机视觉中，金字塔池化是一种具有较长历史且广为人知的提取特征表示的技术。在深度卷积神经网络 [10] 复兴之前，出现了一些利用金字塔池化来识别自然场景的著名工作 [32], [33]。受到 [32], [33] 这些工作的启发，何恺明等人 [34] 将金字塔池化引入深度卷积神经网络里面进行图像分类和物体检测。他们采用了几种池化操作将卷积神经网络骨干网络的最终卷积特征图池化为几个固定大小的特征图。然后将这些得到的特征图扁平化并拼接成固定长度的特征表示，从而实现了具有鲁棒性的视觉识别。之后，赵恒爽等人 [35] 应用金字塔池化于语义分割任务。他们没有采用 [34] 中的扁平化操作，而是将池化后的固定大小特征图上采样为原始大小，并将上采样后的特征图拼接起来进行后续预测。他们的成功表明金字塔池化在网络预测中的有效性。在此之后，金字塔池化已经被广泛应用于语义分割 [35], [52]–[55] 和物体检测 [34], [56]–[58] 等各种视觉任务。

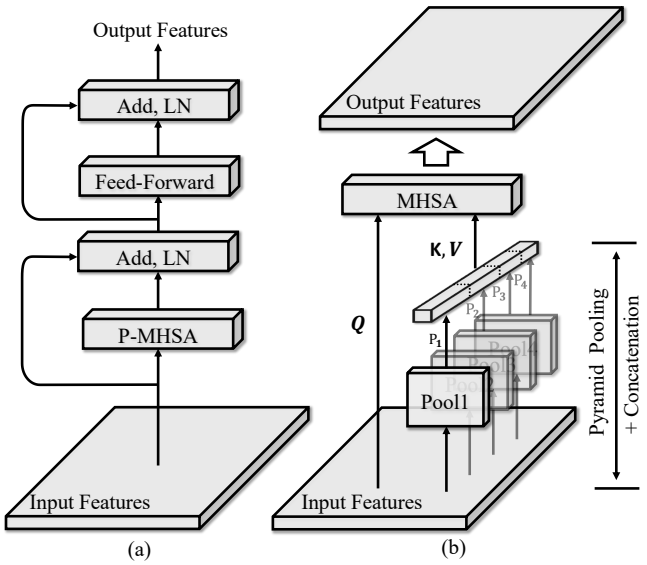


图 3. 金字塔池化 Transformer 基本单元示意图。 (a) 金字塔池化 Transformer 的简单结构示意图。(b) 基于池化的多头自注意力模块的详细结构。

与在特定任务下探索卷积神经网络中金字塔池化的现有文献不同，我们建议将金字塔池化的概念引入到视觉 Transformer 骨干网络里面。有了这个想法，我们首先将金字塔池化嵌入到 P2T 骨干网络的普通注意力模块中，这可以减少词符序列的长度，同时学习到强大的上下文特征表示。P2T 可以很容易地被各种视觉任务用于特征表示的学习，而之前关于金字塔池化的工作仅限于特定的视觉任务。本文在图像分类、语义分割、对象检测和实例分割方面的大量实验证明了 P2T 与现有的基于卷积神经网络或 Transformer 的网络相比的优越性。因此，这项工作的独特性将有利于后续对各种视觉任务的研究。

3 方法

在这个小节，我们首先在 §3.1 介绍我们整个 P2T 网络结构的大体思路。然后，我们在 §3.2 展示基于池化多头自注意力模块的 P2T 的网络架构。最后，我们在 §3.3 介绍一些网络的实现细节。

3.1 概述

P2T 的整体架构在 图 2 中可以详细看到。以自然彩色图像作为输入，P2T 首先将其拆分为 $\frac{H}{4} \times \frac{W}{4}$ 个块，每个块都被展平为 48 （$4 \times 4 \times 3$）个元素。跟 [6] 一样，我们将这些扁平化后的图像块输入到一个图像块编码模块中；它由一个线性投影层组成；然后加上可学习的位置编码。图像块编码模块将 48 的特征维度扩展到 C_1。然后，我们堆叠将在 §3.2 中介绍的金字塔池化 Transformer 模块。整个网络可以分为四个阶段，分别具有特征维度为 C_i ($i = \{1, 2, 3, 4\}$)。在每两个阶段之间，每个 2×2 图像块组被拼接起来，并从 $4 \times C_i$ 线性投影到 C_{i+1} 维度 ($i = \{1, 2, 3\}$)。这样，四个阶段的尺度就变成了 $\frac{H}{4} \times \frac{W}{4}$，$\frac{H}{8} \times \frac{W}{8}$，$\frac{H}{16} \times \frac{W}{16}$，和 $\frac{H}{32} \times \frac{W}{32}$。从四个阶段，我们可以分别推导出四个特征表示 $\{\mathrm{B}_1, \mathrm{B}_2, \mathrm{B}_3, \mathrm{B}_4\}$。其中只有 B_4 将用于图像分类的最终预测，但是所有金字塔特征都可以用于下游场景理解任务。

3.2 金字塔池化 Transformer

金字塔池化已被广泛用于许多与卷积神经网络协作的场景理解任务中 [34], [35], [59]–[66]。然而，现有文献通常基于已设计好的骨干网络，并在它们基础之上利用金字塔池化额外设计一些模块，用于提取特定任务的全局上下文信息。相比之下，本文首次探索了 Transformer 和骨干网络之中的金字塔池化，目的是为了普遍改善各种场景理解任务。为此，我们将金字塔池化的想法与 Transformer 结合，同时减少多头自注意力模块的计算负荷并捕获丰富的上下文信息。

P2T 的基本单元结构在图 3 (a) 中进行了展示。输入特征首先通过基于金字塔池化的多头自注意力模块，其输出与短接的自身相加，然后是 LayerNorm [67]。如同传统的 Transformer 模块 [6], [22], [46]，接下来是一个前馈网络（FFN）用于特征投影。一个残差连接和 LayerNorm [67] 被再次应用。上述过程可以被表述为

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

其中 X，X_{att} 和 X_{out} 分别为多头自注意力模块的输入、输出和 Transformer 块的输出。P-**型多头自注意力模块**（在文中或实验中也会采用 P-MHSA 缩写指代）是基于池化的多头自注意力模块的代称。

3.2.1 基于池化的多头自注意力模块（P-MHSA）

在这里，我们介绍了我们基于池化的多头自注意力模块的设计。其结构如图 3（b）所示。首先，输入的 X 被重塑为二维空间。然后，我们在重塑的 X 上应用不同比例的多个平均池层，用以生成金字塔特征图，具体如下所示：

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

其中 $\mathrm{P}_1, \mathrm{P}_2, ..., \mathrm{P}_n$ 表示生成的金字塔特征图，n 是池化层的数量。接下来，我们将金字塔特征图送入深度卷积进行相对位置编码：

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

其中 $\mathrm{DWConv}(\cdot)$ 表示深度卷积，核大小为 3×3，P_i^{enc} 为相对位置编码的 P_i。由于 P_i 是池化特征，所以在 式 (3) 中的

表 1. 本文 P2T 的具体配置。每个模块的参数以及堆叠的 P2T 基础模块的数量都会显示在括号中。对于第一阶段，我们应用一个 7×7 卷积与 C 输出通道和 S 的步幅用于图像块编码。每个 IRB 使用 E 的扩展比率。为简单起见，我们省略了图像块嵌入操作，即在第 t（$t = \{2, 3, 4\}$）阶段之后的 3×3 卷积，步长为 $S = 2$ 。

阶段序号	输入大小	运算操作	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	$\begin{bmatrix} C = 48 \\ E = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 64 \\ E = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 64 \\ E = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} C = 64 \\ E = 8 \end{bmatrix} \times 3$
3	28×28	P-MHSA IRB	$\begin{bmatrix} C = 96 \\ E = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 128 \\ E = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 128 \\ E = 8 \end{bmatrix} \times 4$	$\begin{bmatrix} C = 128 \\ E = 8 \end{bmatrix} \times 8$
4	14×14	P-MHSA IRB	$\begin{bmatrix} C = 240 \\ E = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C = 320 \\ E = 4 \end{bmatrix} \times 9$	$\begin{bmatrix} C = 320 \\ E = 4 \end{bmatrix} \times 18$	$\begin{bmatrix} C = 320 \\ E = 4 \end{bmatrix} \times 27$
5	7×7	P-MHSA IRB	$\begin{bmatrix} C = 384 \\ E = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C = 512 \\ E = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C = 512 \\ E = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C = 640 \\ E = 4 \end{bmatrix} \times 3$
	1×1	-	全局平均池化，维度为 1000 的全连接层，Softmax 层			
参数量			11.6M	24.1M	36.1M	54.5M
输入大小为 224×224 的计算开支（FLOPS）			1.8G	3.7G	6.5G	9.8G

操作只需要一点计算开支。之后，我们对这些金字塔特征图进行扁平化和串联:

$$\mathrm{P} = \mathrm{LayerNorm}(\mathrm{Concat}(\mathrm{P}_1^{enc}, \mathrm{P}_2^{enc}, ..., \mathrm{P}_n^{enc})). \tag{4}$$

公式中为了简单起见，省略了扁平化操作。这样，如果池化率足够大，P 可以是一个比输入 X 更短的序列。此外，P 包含了输入 X 的上下文抽象，因此在计算多头自注意力模块时可以作为输入 X 的有力替代。

假设多头自注意力模块 [22] 中的查询、键和值张量分别为 Q、K 和 V。与其采用传统的如下方法:

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

我们提出采用

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

其中 W^q, W^k 和 W^v 表示表示生成查询、键和值的线性变换的权重矩阵。

然后，Q, K, V 被送入注意力模块，以计算注意力特征 A。它可以被表述为如下形式:

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

其中 d_K 是 K 的通道尺寸。和 $\sqrt{d_K}$ 可以作为一个近似的标准化。Softmax 函数是沿着矩阵的行来应用的。为了简介，式 (7) 省略了多头的概念 [18], [22]。

由于 K 和 V 的长度比 X 小，所提出的 P-型多头自注意力模块比传统的多头自注意力模块更有效率。此外，由于 K 和 V 包含高度抽象的多尺度信息，所提出的 P-型多头自注意力模块在全局上下文依赖性建模方面有更强的能力，这对场景理解很有帮助 [34], [35], [60], [63]–[66]。从不同的角度来看，金字塔池化通常被用作连接在已有骨干网络基础之上的有效技术；相比之下，本文首先通过 Transformer 在骨干网络内利用金字塔池化，从而为场景理解提供强大的特征表示学习。通过上述分析，P-型多头自注意力模块有望比传统的多头自注意力模块更高效、更有效 [18], [22]。

复杂度分析： 正如在式 (2) 中所描述的，所提出的基于金字塔池化的注意力利用多个池化操作来生成金字塔特征图。金字塔池化操作只有可忽略的 $O(NC)$ 计算复杂度，其中 N 和 C 分别代表序列长度和特征尺寸。因此，计算多头自我注意力的计算复杂度可以表述为:

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

其中 M 是所有池化特征的总和序列长度。对于默认的金字塔池化比例 {12, 16, 20, 24}，我们有 $M \approx \frac{N}{66.3} \approx \frac{N}{8^2}$，这与 PVT 中多头自注意力模块的计算开支相当 [6]。

3.2.2 前馈网络

前馈网络（Feed-Forward Network，FFN）是用于特征增强的 Transformer 的一个重要组成部分 [18], [68]。以前的 Transformer 通常应用全连接网络作为前馈网络 [6], [18], [22]，完全依靠注意力来捕捉像素间的依赖关系。虽然有效，但这种架构并不擅长学习二维近邻关系，而二维近邻关系在场景理解中起着关键作用。为此，我们按照 [50], [51] 将深度卷积插入到前馈网络中，这样得到的 Transformer 可以继承 Transformer（长距离依赖性建模）和卷积神经网络（二维近邻关系）的优点。具体来说，我们采用 MobileNetV2 中提出的倒置瓶颈模块（Inverted Bottleneck Block，IRB）作为前馈网络。

为了使倒置瓶颈模块适用于视觉 Transformer，我们首先将输入序列 X_{att} 变换为二维特征图 X_{att}^I:

$$\mathrm{X}_{att}^I = \mathrm{Seq2Image}(\mathrm{X}_{att}), \tag{9}$$

其中 Seq2Image(·) 是将一维序列重塑为二维特征图。考虑到输入的 X^I_{att}，IRB 可以直接应用，比如说：

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

其中 $W^1_{\text{IRB}}, W^2_{\text{IRB}}$ 表示 1×1 卷积的权重矩阵，Act 表示非线性激活函数，$X^{\text{out}}_{\text{IRB}}$ 是 IRB 的输出。由于 $X^{\text{out}}_{\text{IRB}}$ 是一个二维特征图，我们最后将其转化为一维序列：

$$X^S_{\text{IRB}} = \text{Image2Seq}(X^{\text{out}}_{\text{IRB}}), \tag{11}$$

其中 Image2Seq(·) 是将二维特征图重塑为一维序列的操作，X^S_{IRB} 是前馈网络的输出，其形状与 X_{att} 相同。

3.3 实现细节

不同深度的 P2T 设置： 遵循之前的骨干架构 [4]–[6], [8], [26]，我们通过在每个阶段堆叠不同数量的金字塔池 Transformer 来构建不同深度的 P2T。通过这种方式，我们提出了四个版本的 P2T，即 P2T-Tiny、P2T-Small、P2T-Base 和 P2T-Large，其参数数量分别与 ResNet-18 [4]、ResNet-50 [4]、ResNet-101 [4] 和 PVT-Large [6] 相当。除了 P2T-Tiny 中每个头有 48 个特征通道，P-型多头自注意力模块的每个头有 64 个特征通道。不同版本的 P2T 的其他配置显示在表 1。

金字塔池化设置： 我们根据经验将 P-型多头自注意力模块中并行池化操作的数量设置为 4。在不同的阶段，金字塔池化的池化比率 Transformer 是不同的。第一阶段的池化比率根据经验设定为 $\{12, 16, 20, 24\}$。除了在最后阶段，接下来每个阶段的池化比率都除以 2。在最后阶段，它们被设定为 $1, 2, 3, 4$。在每个 Transformer 块中，P-型多头自注意力模块的所有深度卷积（在式 (3)）都有相同的参数。

其他设置： 虽然深度卷积（在式 (3)）的内核大小较大（如 5×5）可以带来更好的性能，但为了提高效率，所有深度卷积的内核大小被设置为 3×3。我们选择 Hardswish [69] 作为非线性激活函数，因为它比 GELU [70] 节省了很多内存。除此之外，Hardswish [69] 在实际使用上也很有效。和 PVTv2 [9] 一样，我们采用了重叠图像块编码。也就是说，我们使用 3×3 的卷积，跨度为 2，用于从第二阶段到最后阶段的图像块编码，而我们应用 7×7 的卷积，跨度为 4，用于第一阶段的图像块编码。

4 实验

我们首先介绍了在 §4.1 中进行的图像分类实验。然后，我们在 §4.2、§4.3 和 §4.4 中分别验证了 P2T 在几个场景理解任务中的有效性，即语义分割、物体检测和实例分割。最后，我们在 §4.5 中进行了消融实验，以更好地理解我们的方法。

表 2. 数据集 ImageNet-1K [23] 上的图像分类结果。“Top-1”表示 top-1 的准确率。“*”表示用知识蒸馏的结果 [46]。“P#(M)”表示参数的数量（单位为百万）。除了 ViT-B [22] 的速度是在输入大小为 384×384 的情况下测试的，每个网络的计算开支（GFlops）和运行速度（每秒帧数，FPS）都是以默认的 224×224 的输入大小报告。FPS 是在单个 RTX 2070 GPU 上测试的。提出的 P2T 的结果以**粗体**标记。

方法	#P (M) ↓	GFlops ↓	Top-1 (%) ↑	FPS ↑
ResNet-18 [4]	11.7	1.8	68.5	1410
DeiT-Tiny/16* [46]	5.7	1.3	72.2	1212
ViL-Tiny [71]	6.7	1.3	76.7	441
PVT-Tiny [6]	13.2	1.9	75.1	608
PVTv2-B1 [9]	13.1	2.1	78.7	502
P2T-Tiny (**本文**)	11.6	1.8	79.8	473
ResNet-50 [4]	25.6	4.1	78.5	483
ResNeXt-50-32x4d [5]	25.0	4.3	79.5	407
Res2Net-50 [17]	25.7	4.5	80.3	430
DeiT-Small/16* [46]	22.1	4.6	79.9	489
PVT-Small [6]	24.5	3.8	79.8	336
T2T-ViT$_t$-14 [47]	21.5	5.2	80.7	305
Swin-T [8]	29.0	4.5	81.3	349
Twins-SVT-S [7]	24.0	2.9	81.7	439
ViL-Small [71]	25.0	4.9	82.4	187
PVTv2-B2 [9]	25.4	4.0	82.0	284
P2T-Small (**本文**)	24.1	3.7	82.4	284
ResNet-101 [4]	44.7	7.9	79.8	288
ResNeXt-101-32x4d [5]	44.2	8.0	80.6	228
Res2Net-101 [17]	45.2	8.3	81.2	265
PVT-Medium [6]	44.2	6.7	81.2	216
T2T-ViT$_t$-19 [47]	39.2	8.4	81.4	202
Swin-S [8]	50.0	8.7	83.0	207
ViL-Medium [71]	40.4	8.7	83.5	114
MViT-B-16 [25]	37.0	7.8	83.1	222
PVTv2-B3 [9]	45.2	6.9	83.2	189
P2T-Base (**本文**)	36.1	6.5	83.5	182
ResNeXt-101-64x4d [5]	83.5	15.6	81.5	147
MViT-B-24 [25]	53.5	10.9	83.0	151
ViL-Base [71]	57.0	13.4	83.7	67
PVT-Large [6]	61.4	9.8	81.7	152
DeiT-Base/16* [46]	86.6	17.6	81.8	161
ViT-Base/16 [22]	86.6	17.6	77.9	49
Swin-B [8]	88.0	15.4	83.3	140
Twins-SVT-L [7]	99.2	14.8	83.3	143
PVTv2-B4 [9]	62.6	10.1	83.6	133
PVTv2-B5 [9]	82.0	11.8	83.8	120
P2T-Large (**本文**)	54.5	9.8	83.9	128

4.1 图像分类

图像分类是评估骨干网络性能最常见的任务。它的目的是为每个自然图像输入分配一个类别标签。许多任务通过应用建立在图像分类之上的分类网络作为特征提取的骨干。

实验设计： 如 §3.1 中所述，这里只利用了最后阶段的输出特征 B_4。按照常规的卷积神经网络网络 [4], [13], [17] ，我们

表 3. ADE20K 验证集的语义分割实验结果。我们用各种网络架构替换了 Semantic FPN [3] 的骨干部分。GFlops 的数量是以 512 × 512 的输入大小计算的。FPS 是在单张 RTX 2070 GPU 上测试的。P2T 骨干网络的结果以**粗体**标记。

骨干网络	Semantic FPN [3]			
	参数量（M）↓	计算量（G）↓	mIoU (%) ↑	每秒帧数 ↑
ResNet-18 [4]	15.5	31.9	32.9	68
PVT-Tiny [6]	17.0	32.1	35.7	36
PVTv2-B1 [9]	17.8	33.1	41.5	30
P2T-Tiny (**本文**)	15.4	31.6	43.4	31
ResNet-50 [4]	28.5	45.4	36.7	35
PVT-Small [6]	28.2	42.9	39.8	26
Swin-T [8]	31.9	46	41.5	26
Twins-SVT-S [7]	28.3	37	43.2	27
PVTv2-B2 [9]	29.1	44.1	46.1	21
P2T-Small (**本文**)	27.8	42.7	46.7	24
ResNet-101 [4]	47.5	64.8	38.8	26
ResNeXt-101-32x4d [5]	47.1	64.6	39.7	20
PVT-Medium [6]	48.0	59.4	41.6	19
Swin-S [8]	53.2	70	45.2	18
Twins-SVT-B [7]	60.4	67	45.3	17
PVTv2-B3 [9]	49.0	60.7	47.3	15
P2T-Base (**本文**)	39.8	58.5	48.7	16
ResNeXt-101-64x4d [5]	86.4	104.2	40.2	15
PVT-Large [6]	65.1	78.0	42.1	15
Swin-B [8]	91.2	107	46.0	13
Twins-SVT-L [7]	102	103.7	46.7	13
PVTv2-B4 [9]	66.3	79.6	48.6	11
PVTv2-B5 [9]	85.7	89.4	48.9	10
P2T-Large (**本文**)	58.1	77.7	49.4	12

在 B_4 上面附加了一个全局平均池层和一个全连接层，以获得最终的分类分数。我们在 ImageNet-1K 数据集 [23] 上训练我们的网络，它有 128 万张训练图像和 5 万张验证图像。为了公平比较，我们按照 PVT [6] 采用与 DeiT [46] 相同的训练协议（没有知识蒸馏），这是训练视觉 Transformer 的标准选择。具体来说，我们使用 AdamW [75] 作为优化器，初始学习率为 10^{-3}，权重衰减为 0.05，每个小批次为 1024 张图像。我们用余弦学习率衰减策略训练 P2T 300 个迭代单位。用于训练和测试的图像大小被调整为 224 × 224。模型在前五个迭代单位中进行预热。数据扩增也与 [6], [46] 相同。

实验结果： 定量的比较总结在 表 2。除了我们按照 ViT [22] 官方以 384 × 384 的输入大小进行训练和评估，其他所有的模型都是以 224 × 224 的输入大小进行训练和评估的。P2T 在很大程度上超过了 ResNets [4] 和 ResNeXts [5] 等常规卷积神经网络模型。例如，尽管 P2T-Tiny/Small/Base/Large 的运行时间是 ResNet-18/50/101 [4] 和 ResNeXt-101-64x4d [5] 的 2.98/1.70/1.58/1.15 倍，但 P2T-Tiny/Small/Base/Large 的最高准确率分别比 ResNet-18/50/101 [4] 和 ResNeXt-101-64x4d [5] 高 11.3%/3.9%/3.7%/2.4%。另一方面可以看出，与最近最先进的 Transformer 模型相比，我们的 P2T 也取得了优异的成绩。例如，P2T-Small/Base/Large 比 Swin Transformer [8] 好 1.1%/0.5%/0.6%，网络参数少，同时计算开支还低。尽管 PVTv2 [9] 比 PVT [6] 有很大的改进，但我们的 P2T-Tiny/Small/Base/Large 仍然比 PVTv2-B1/B2/B3/B4 [9] 有 1.1%/0.4%/0.3%/0.3% 的优势，参数更少，同时计算开支更低。P2T 在计算自注意力时应用了四个并行池化操作，仍然取得了与 PVTv2 [9] 相当的速度。虽然 ViL [71] 实现了与 P2T 相当的性能，然而 ViL [71] 的速度比我们的 P2T 慢得多，而且 ViL [71] 的计算开支也比 P2T 大很多。在参数量少的情况下，P2T 在很大程度上也优于 ViT [22] 和 DeiT [46]，这意味着 P2T 在没有大量训练数据和知识蒸馏的情况下就能达到更好的性能。因此，P2T 是非常适合用于图像分类任务的。

4.2 语义分割

给定一个自然图像输入，语义分割的目的是为每个像素分配一个语义标签。它是计算机视觉中最基本的稠密预测任务之一。

实验设置： 我们在 ADE20K [2] 数据集上评估了 P2T 及其竞争对手。ADE20K 数据集是一个具有挑战性的场景理解数据集，有 150 个细粒度的语义类别。这个数据集有 20000 张训练图片、2000 张验证图片和 3302 张测试图片。和 [6], [7] 一样，Semantic FPN [3] 被选为基本方法，用来进行公平比较。我们将 Semantic FPN [3] 的骨干网替换成各种网络架构。所有 Sematic FPN 的方法的骨干网络都在 ImageNet-1K [23] 数据集上进行了预训练，其他层则使用 Xavier 方法初始化 [76]。所有的网络都训练了 80000 迭代次数。我们应用 AdamW [75] 作为网络优化器，初始学习率为 10^{-4}，权重衰减为 10^{-4}。采用 $\gamma = 0.9$ 的 poly 学习率计划。每个小批次有 16 张图像；用于训练的图像被调整大小并随机裁剪为 512 × 512。还启用了跨 GPU 的同步批次规范化。在测试过程中，图像短边被调整到 512，长边按比例进行调整。多尺度测试和翻转功能被禁用。与 [6] 一致，我们同样使用 MMSegmentation 工具箱 [77] 来实现上述实验。

实验结果： 定量比较结果显示在表 3。我们将我们提出的 P2T 与 ResNets [4]、ResNeXts [5]、PVT [6]、Swin Transformers [8]、Twins [7] 和 PVTv2 [9] 进行比较。每个网络的结果都来自于官方论文或使用官方训练配置重新训练的结果。受益于金字塔池化技术，采用我们的 P2T 骨干的 Semantic FPN [3] 的结果要比其他卷积神经网络和 Transformer 的竞争对手好得多。在参数量和计算量更小的前提下，P2T-Tiny/Small/Base/Large 的 mIoU 性能分别比 ResNet-18/50/101 [4] 和 ResNeXt-10-64x4d [5] 好了 10.5%/10.0%/9.9%/9.2%。与 Swin Transformer [8] 相比，P2T-Small/Base/Large 分别比 Swin-T/S/B [8] 实现了 5.2%/3.5%/3.4% 的提升，表明全局关系建模对于视觉识别具有重要意义。Twins [7] 结合了 Swin Transformers [8] 的局部自我注意力和 PVT [6] 的整体自我注意力。可以看出，Twins [7] 比 Swin Transformers [8] 表现

表 4. 在 MS-COCO val2017 数据集 [72] 上，用 RetinaNet [73] 进行物体检测的结果和用 Mask R-CNN [74] 进行实例分割的结果。 “R” 和 “X” 分别代表 ResNet [4] 和 ResNeXt [5]。Flops 的数量是以 800 × 1280 的输入大小计算的。FPS 是在单个 RTX 2070 GPU 上测试的。P2T 骨干网的结果以**粗体**标记。

骨干网络	物体检测									实例分割								
	参数量	计算量	每秒帧数	RetinaNet [73]						参数量	计算量	每秒帧数	Mask R-CNN [74]					
	(M) ↓	(G)↓	↑	AP ↑	AP_{50}	AP_{75}	AP_S ↑	AP_M	AP_L	(M) ↓	(G) ↓	↑	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 [71]	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 [6]	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 [9]	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 (**本文**)	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 [6]	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 [8]	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 [71]	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 [7]	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 [9]	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 (**本文**)	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 [5]	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 [6]	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 [8]	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 [9]	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 (**本文**)	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 [5]	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 [6]	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 [7]	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 [9]	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 [9]	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 (**本文**)	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

更好，这说明全局自我注意力很重要。与 Twins [7] 不同的是，我们通过金字塔池化来应用纯粹的全局自我注意力，学习更丰富的场景信息。P2T-Small/Base/Large 的 mIoU 性能相对 Twins-SVT-S/B/L [7] 提升了 3.5%/3.4%/2.7%。PVTv2 [9] 是 PVT [6] 的改进版，作为我们 P2T 的最强竞争对手。P2T-Tiny/Small/Base/Large 分别比 PVTv2-B1/B2/B3/B4 [9] 要强 1.9%/0.6%/1.4%/0.8%。此外，P2T-Tiny/Small/Base/Large 总是比相应的 PVTv2-B1/B2/B3/B4 [9] 有更少的参数，更少的计算开支，以及更快的速度。最后，我们发现 P2T-Tiny 的性能相对 ResNeXt-101-64x4d [5] 提升了 3.2%，速度快一倍。基于上述观察，我们可以得出结论，P2T 非常能胜任语义分割任务。

4.3 物体检测

物体检测也是计算机视觉领域几十年来最基本和最具挑战性的任务之一。它的目的是检测和识别自然图像中某些类别的语义对象的实例。在这里，我们在 MS-COCO [72] 数据集上评估了 P2T 及其竞争对手。

实验设置： MS-COCO [72] 是一个大规模的比赛数据集，用于物体检测、实例分割和关键点检测。MS-COCO train2017（118k 张图像）和 val2017（5k 张图像）集在我们的实验中分别用于训练和验证。RetinaNet [73] 被应用为基本框架，因为它已经被这个社区广泛认可 [6], [8]。每个小批次有 16 张图像，初始学习率为 10^{-4}。按照流行的 MMDetection 检测工具箱 [78]，我们对每个网络进行 12 个迭代单位的训练，在 8 和 11 个迭代单位之后，学习率被除以 10。网络优化器是 AdamW [75]，一个用于训练 Transformer 的流行优化器。权重衰减被设置为 10^{-4}。在训练和测试过程中，输入图像的短边被调整为 800 像素，长边将保持图像的比例在 1333 像素以内。在训练阶段，只有随机水平翻转被用于数据增强。我们使用标准的 COCO API 进行评估，使用 AP、AP_{50}、AP_{75}、AP_S、AP_M 和 AP_L 等指标报告结果。AP_S、AP_M 和 AP_L 分别是指小型、中型和大型对象的 AP 得分，其具体定义在 [72]。AP 通常被看作是主要指标。对于每个指标来说，更高的分数代表更好的性能。除此之外，我们还报告了每个方法的参数量、计算开支、速度以供参考。

实验结果： 对 MS-COCO 数据集的评估结果在 表 4 的左边部分进行了总结。其他网络的结果来自官方论文或使用官方配置重新实现。下面的讨论如果没有说明，则是指 AP 的度量。我们可以观察到，我们的 P2T 在所有微小/小/大的复杂性设置下都取得了最好的性能。例如，P2T-Small 比 Swin-T [8]、Twins-SVT-S [7] 和 PVTv2-B2 [9] 分别实现了 2.9%、1.4% 和

表 5. 对多个金字塔池化比率的消融研究。 “Top-1” 表示 ImageNet-1K 验证集 [23] 上的 top-1 分类准确率，“mIoU” 表示 ADE20K 数据集 [2] 上的语义分割结果。

实验序号	池化系数	下采样比率 ↑	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

表 6. 在不同阶段用多个池化操作取代单一池化操作的消融实验。 由于我们的网络的第 2 阶段和第 3 阶段都只有两个基本块，我们把它们合并为一个选择。“Top-1” 表示 ImageNet-1K 验证集 [23] 上的 top-1 分类准确率，“mIoU” 是 ADE20K 数据 [2] 上语义分割的结果。

实验序号	网络的阶段序号			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

0.6% 的 AP 提升。P2T-Tiny 比 PVTv2 [9] 好 1.1%。与 ViL [71] 相比，P2T-Tiny/Small 分别比 ViL-Tiny/Small [71] 好 0.5% 和 0.2%。请注意，ViL [71] 的运行速度比 P2T 慢得多，如表 4所示。在基本复杂度设置下，P2T-Base 比 Swin-S [9]）高出 1.0%，比最佳竞争对手 PVTv2-B3 高出 0.2%。在大复杂度设置下，P2T-Large 比 PVTv2-B4 [9] 和 Twins-SVT-B [7] 分别取得了 1.1% 和 1.9% 的更好 AP。在所有的复杂程度上，P2T 总是优于 PVTv2 [9]，网络参数更少，计算开支更低，速度更快。P2T-Tiny/Small/Base/Large 分别比 ResNet-18/50/101 [4] 和 ResNeXt-101-64x4d [5] 好 9.5%/8.1%/7.0%/6.2%。综上所述，P2T 在物体检测方面有很强的能力。

4.4 实例分割

实例分割是另一项基本的视觉任务。它可以被看作是物体检测的高级案例。相对物体检测来说，它输出的是细粒度的物体掩码区域而不是物体检测中的边界框。

实验设置： 我们在著名的 MS-COCO 数据集 [72] 上评估实例分割的性能。MS-COCO train2017 和 val2017 验证集在我们的实验中被用于训练和验证。Mask R-CNN [74] 被用作基本框架，使用不同的骨干网络。训练设置与我们在 §4.3 中用于物体检测的设置相同。我们以 AP^b、AP^b_{50}、AP^b_{75}、AP^m、AP^m_{50} 和 AP^m_{75} 指标报告对象检测和实例分割的评估结果，其中”b” 和”m” 分别表示边界框（Bounding Box）和掩码指标（Mask Metric）。AP^b 和 AP^m 被设定为主要评价指标。

表 7. 关于池化操作选择的消融研究。我们可以看到，其他选择的效果比平均池化差。“Top-1” 表示 ImageNet-1K 数据集上的 top-1 准确率 [23] 的图像分类。“mIoU” 是 ADE20K 数据集 [2] 上用于语义分割的平均 IoU 率。

池化类型	Top-1 (%) ↑	mIoU (%) ↑
平均值池化	74.7	35.7
最大值池化	73.0	33.2
卷积	73.8	35.5

实验结果： P2T 与其对比方法的比较显示在 表 4 的右边部分。与现有的卷积神经网络和 Transformer 骨干网络相比，P2T 在所有的网络复杂层级上实现了最佳性能。在边界框指标 AP^b 上，P2T-Small/Base 比 Swin-T/S [8] 好 3.3%/2.4%，P2T-Small/Large 比 Twins-SVT-S/B [7] 好 2.1%/3.1%。P2T-Tiny/Small/Base/Large 分别比 PVTv2-B1/B2/B3/B4 [9] 好 1.5%/0.2%/0.2%/0.8%，同时参数少，计算开支低，速度快。在掩码指标 AP^m 方面，我们也观察到与使用边界框指标类似的提升。与基于 ResNet 的骨干网相比，P2T 在所有复杂程度上都明显优于 ResNets [4] 和 ResNeXts [5]。同样令人惊讶的是，我们最轻的 P2T-Tiny 比 ResNeXt-101-64x4d [5] 在边界框和掩码指标方面分别好 0.5% 和 1.2%。综上所述，P2T 对于实例分割来说是非常有能力的。

4.5 消融实验

实验设置： 在本节中，我们进行了消融实验，以分析每个设计选择在 P2T 中的作用。我们评估了各种实验设置在语义分割和图像分类上的表现。由于计算资源有限，我们只在 ImageNet 数据集 [23] 上对每组设置训练 100 个迭代单位，而其他训练设置保持与 §4.1 中相同。然后，我们在 ADE20K 数据集 [2] 上微调 ImageNet 预训练模型，训练设置与 §4.2 相同。

探究多种金字塔池化比率： 为了验证使用多种池化比率的意义，我们进行了实验，以评估 P2T 与一个/两个/四个平行池化操作的性能。基线是没有相对位置编码、IRB 和重叠图像块编码的 P2T-Small。结果显示在 表 5。可以看出，具有大池子比率的单一池子操作（例如 16, 24）对序列长度有很大的下采样比率。尽管如此，它在图像分类和语义分割方面的性能都非常差。然而，当单一池化操作的池化率为 12 时，如果我们进一步降低池化率，性能将达到饱和。当我们采用两个平行的池化操作时，即使有很高的下采样比率，对图像分类和语义分割来说，性能仍然变得更好。当我们有四个并行的池化操作时，我们的下采样率与池化率为 8（PVT 中的设置）的下采样率相同，且达到了最佳性能。

探究金字塔池化对不同阶段的意义： 我们对 P2T 的金字塔池化设计进行了不同阶段的消融研究。由于第 1 阶段只包含

表 8. 对固定池化大小的消融实验。GFlops 是以语义分割模型 512 × 512 的输入大小计算的，即 Semantic FPN [3]。“Mem” 表示 Semantic FPN [3] 的训练 GPU 内存使用量，批量大小为 2。“Top-1” 和 “mIoU” 分别表示 ImageNet-1K [23] 上的 top-1 分类精度和 ADE20K [2] 的分割 mIoU。

池化操作	GFlops ↓	Mem (GB) ↓	Top-1 (%) ↑	mIoU (%) ↑
固定池化比率	41.6	3.3	74.7	35.7
固定池化大小	38.9	2.9	74.4	33.3

表 9. 相对位置编码（RPE）、IRB 和重叠图像块编码（OPE）的消融实验。“Top-1” 和 “mIoU” 分别表示 ImageNet-1K [23] 上的 top-1 分类精度和 ADE20K [2] 上的分割 mIoU。

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

下采样的卷积，我们不在第 1 阶段进行这种消融研究。基线与上次消融研究相同。单个池化操作的池化率被设置为 8，以确保相同的下采样率。结果显示在表 6。我们可以看到金字塔池化可以提高所有阶段的性能。当更多的阶段应用多个池化操作时，性能变得更高。从结果来看，在第 4 阶段（表 6 的第 3 号）应用多个池化操作的改进比其他阶段（表 6 的第 2、4 号）更大，因为第 4 阶段比第 [2, 3] 阶段和第 5 阶段有更多的基本模块数。

探究池化操作的选择： 我们对不同的池化操作进行了实验，如表 7 中所示。有三种典型的选择，即最大池化、深度卷积和默认的平均池化。深度卷积的核大小与最大/平均池化相同，以保持相同的下采样率。很明显，不同的池化类型并不影响计算的复杂性，它们只影响下采样核的参数数量。关于 ImageNet 分类准确率的结果 [23] 和 ADE20K 分割 mIoU 的结果 [2]，平均池化要比其他两种选择好得多。因此，我们应用平均池化作为默认的池化选择。

探究固定的池化大小： 当使用固定的池化比率时，池化特征图的尺寸会随着输入特征图的变化而变化。在这里，我们试图将所有阶段的池化大小固定为：1, 2, 3, 6。同时在所有阶段，都使用自适应平均池化。结果显示在表 8。与我们的默认设置相比，固定的池化大小大约节省了 10% 的内存用量和 12% 的计算开支。然而，top-1 的分类精度下降了 0.3%。而语义分割的性能则降低了 2.4%。因此，我们选择使用固定的池化比率，而不是固定的池化大小。

探究激活函数的选择： 我们使用 Hardswish 函数 [69] 进行非线性激活函数，以减少训练阶段的 GPU 内存使用。通常情况下，当我们在 ImageNet [23] 上训练 P2T-Small、批次大小为 64 的情况下时，GELU [70] 的 GPU 内存使用量为 10.5 GB，比 Hardswish [69] 多 3.6 GB（+52%）。我们还发现，如果我们采用 Hardswish [69]，准确率没有明显下降。

探究其他设计： 为了验证其他设计选择的有效性，如相对位置编码、IRB 和重叠图像块编码，我们在基线上逐一添加这些组件。实验结果显示在表 9。可以看出，相对位置编码对图像分类和语义分割都有明显的改善。在大的池化比率下，池化后的特征会有较小的尺度，所以相对位置编码只需要可忽略的计算开支（对于 224 × 224 的输入大小，仅需 5M Flops）。前馈网络中额外的深度卷积，即 IRB，也显示出明显的性能提升，证明了捕捉二维近邻关系的重要性。我们进一步按照 [9] 增加重叠图像块编码，对于图像分类和语义分割，分别观察到 0.2%/1.4% 的精度提升。

5 结论

本文将金字塔池化引入多头自注意力模块，以减轻多头自注意力模块在视觉 Transformer 中的高计算开支。与多头自注意力模块中应用单一池化操作的策略相比，我们基于金字塔池化的多头自注意力模块不仅减少了序列长度，而且通过金字塔池化同时学习了强大的上下文表征学习。通过基于金字塔池化的多头自注意力模块，我们构建了一个新的骨干网络，称为金字塔池化 Transformer，简称 P2T。为了证明 P2T 的有效性，我们在几个基础视觉任务上进行了广泛的实验，包括图像分类、语义分割、物体检测和实例分割。实验结果表明，P2T 明显优于以前基于卷积神经网络和 Transformer 的骨干网络。

鸣谢

这项工作得到了新一代人工智能重大项目的部分支持，项目编号为 2018AAA0100400；得到了国家自然科学基金委员会的部分支持，项目编号为 61922046；得到了 A*STAR 的 AME 计划基金支持（编号为 A1892b0026 和 A19E3b0099）。

References

[1] Y.-H. Wu, Y. Liu, X. Zhan, and M.-M. Cheng, “P2T: Pyramid pooling transformer for scene understanding,” IEEE Trans. Pattern Anal. Mach. Intell., 2022.

[2] 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.

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

[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] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

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

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

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

[61] 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.

[62] 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.

[63] 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.

[64] 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.

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

[66] 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.

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

[68] 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.

[69] 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.

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

[71] 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.

[72] 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.

[73] 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.

[74] 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.

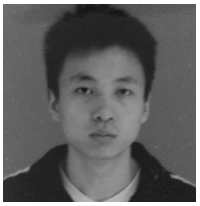
[75] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in Int. Conf. Learn. Represent., 2017.

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

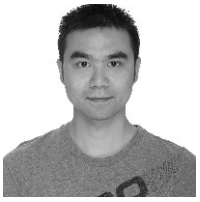
[77] M. Contributors, "MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark," https://github.com/open-mmlab/mmsegmentation, 2020.

[78] 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.

吴宇寰 目前是南开大学计算机学院的四年级博士生，导师是程明明教授。他于 2018 年在西安电子科技大学获得学士学位。他在 IEEE TPAMI/TIP/CVPR/ICCV 等顶级期刊会议上发表了 10+ 篇论文。他的研究兴趣包括计算机视觉、医学图像处理、自动驾驶。



刘云 于 2016 年和 2020 年分别获得南开大学的学士学位和博士学位。他的博士生导师是程明明教授。随后，他在苏黎世联邦理工学院计算机视觉实验室与 Luc Van Gool 教授合作了一年半的时间，担任博士后学者。目前，他是 A*STAR 信息通信研究所 (I2R) 的一名研究科学家。他的研究兴趣包括计算机视觉和机器学习。



占新 于 2010 年和 2015 年分别获得中国科学技术大学的学士和博士学位。目前，他是阿里巴巴达摩院自动驾驶实验室的研究员。他的研究兴趣包括自动驾驶的感知。

程明明 于 2012 年在清华大学获得博士学位。之后，他在牛津大学跟随 Philip Torr 教授做了两年的研究工作。他现在是南开大学的教授，领导媒体计算实验室。他的研究兴趣包括计算机图形学、计算机视觉和图像处理。他获得的研究奖项包括 ACM 中国新星奖、IBM 全球 SUR 奖和 CCF-Intel 青年教师研究计划。他是 IEEE TPAMI 和 TIP 的编辑委员会成员。