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

Wang W, Xie E, Li X, et al.

摘要

尽管卷积神经网络 (CNN) 在计算机视觉领域取得了巨大成功，但这项工作研究了一种更简单、无卷积的骨干网络，可用于许多密集预测任务。与最近提出的专为图像分类设计的 Vision Transformer (ViT)[4] 不同的是，我们引入了 Pyramid Vision Transformer (PVT)，它克服了将 Transformer 移植到各种密集预测任务的困难。与当前的技术水平相比，PVT 有几个优点：(1) 与通常产生低分辨率输出并导致高计算和内存成本的 ViT 不同，PVT 不仅可以在图像的密集分区上进行训练以获得高输出分辨率，这对于密集预测很重要，而且还使用渐进收缩金字塔减少大型特征图的计算。(2) PVT 继承了 CNN 和 Transformer[3]的优点，使其成为无需卷积的各种视觉任务的统一主干，可以直接替代 CNN 主干。(3) 我们通过大量实验验证了 PVT，表明它提高了许多下游任务的性能，包括对象检测、实例分割和语义分割。例如，在参数数量相当的情况下，PVT+RetinaNet 在 COCO 数据集上实现了 40.4 AP，超过 ResNet50+RetinNet (36.3 AP) 4.1 绝对 AP（见图 2）。我们希望 PVT 可以作为像素级预测的替代和有用的支柱，并促进未来的研究。

关键词：Computer Vision；Transformer；Vision Transformer；

1 引言

Transformer[3]自 2017 年诞生之后，迅速在 NLP 领域攻城略地，在极短的时间内晋升成为 NLP 领域绝对的霸主。Transformer 模型来源于论文《Attention is all you need》[3]，论文提出了自注意力机制，能在 sequence to sequence 问题中很好地学到上下文信息进行特征学习。Transformer 进军 CV 领域早在 2018 年就开始了，但是进程缓慢，直到 2020 年谷歌提出 Vision Transformer(ViT)[4]，Transformer 在 cv 领域的应用才正式被引爆。ViT 正是利用 transformer[3]理解上下文语义这一特点，将图像分割成 patch 块并转成 embedding，通过 Transformer Encoder 进行特征学习，相较于传统 CNN 网络，Transformer 的架构关注的不仅仅是局部特征，更是全局特征。

2 相关工作

已有的视觉主干网络主要有 CNN 架构和 Transformer 架构，CNN 架构主要包括 VGG[1]、ResNet[2]等，Transformer 架构主要为 ViT（Vision Transformer）[4]。CNN 以较轻量的计算与适应较多的下游任务而著称，通过卷积核参数的训练提取有效的特征，是一种普遍有效的方案，但 CNN 存在一个缺陷，那就是容易关注于局部的特征而没有考虑到全局特征，使计算的结果具有偏局部性，丧失了一定程度的上下文语义。Vision Transformer 发挥主要作用的部分是 Encoder，用于结合上下文语义提取关键特征。虽然 ViT 能较好地学习到图像的上下文信息，但存在一个缺点：模型的计算成本和内存成本与所计算图像的分辨率成正比，当计算高分辨率图像时将十分占用资源。而且，原始版本的 ViT 只适合做分类任务，而不能完成其他下游任务。

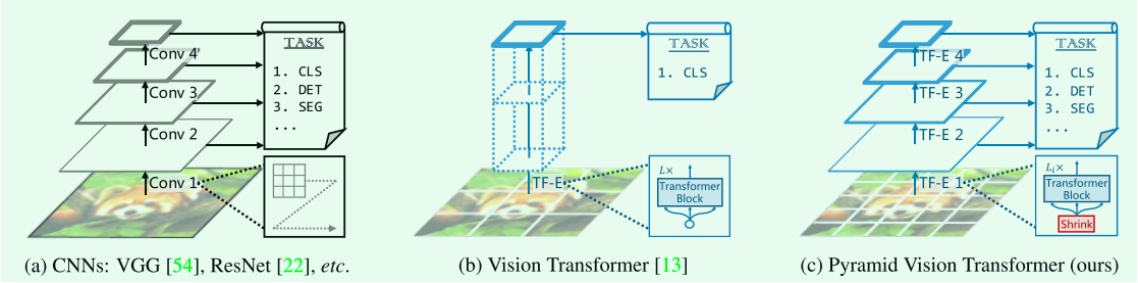
2.1 方法提出

这两种主要的架构各有优缺点，倘若有一种结构能将二者优点结合起来，那将会是一个不错的方案。Wenhai Wang 等人于 2020 年发表的《Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions》即为二者结合的新的 backbone 架构。

3 本文方法

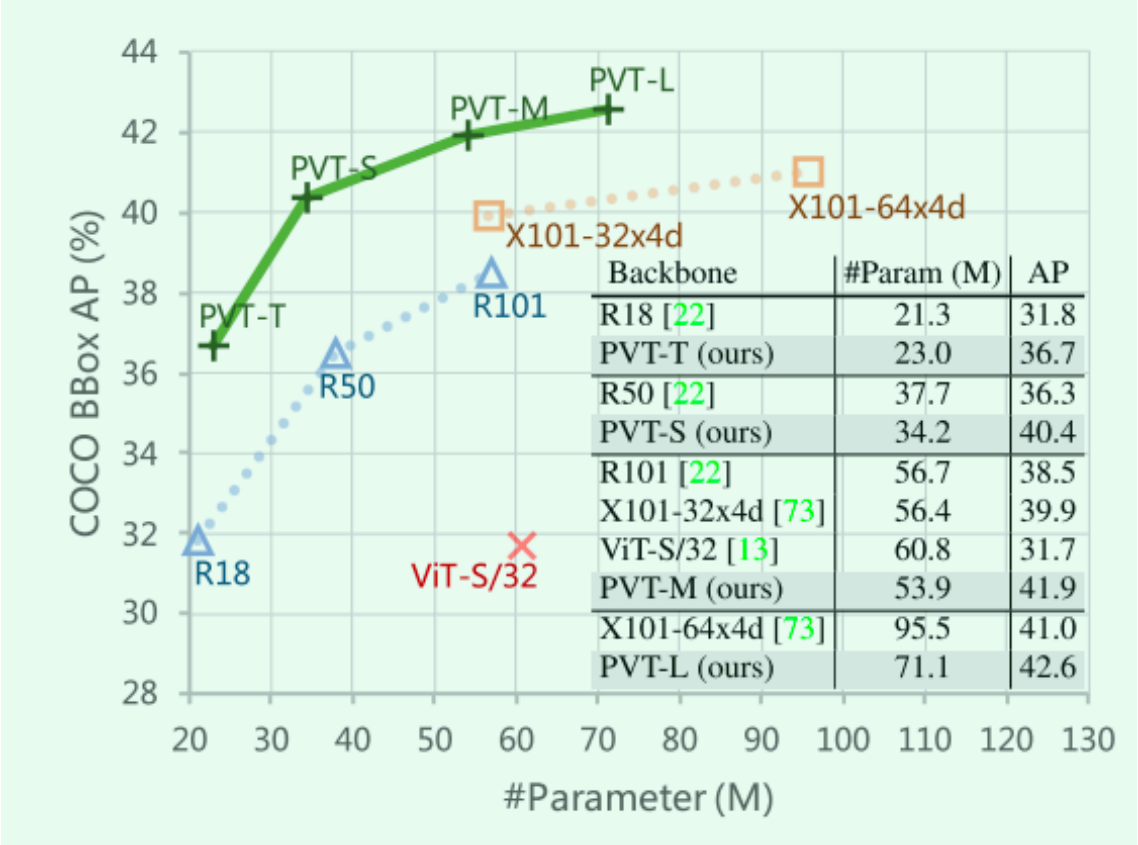
3.1 本文方法概述

本文提出的 PVT 作为基于 Transformer 架构的模型，研究了一种更简单、无卷积的骨干网络，可用于密集预测任务。PVT 不仅可以在图像的密集分区上进行训练以获得高输出分辨率，这对于密集预测很重要，而且还使用了渐进收缩金字塔减少大型特征图的计算。同时，PVT 继承了 CNN 和 Transformer 的优点，使其成为无需卷积的各种视觉任务的统一主干，可以直接替代 CNN 主干，完成更多下游任务。CNN、ViT 与 PVT 对比如图 1所示。由图 2可以看出，PVT 相比于其他 backbone，在参数量相等的境况下，可以取得更高的准确率。



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

图 1: 三种主干网络结构图，从左到右分别为 CNN、ViT、以及本文提出的 PVT



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

图 2: PVT 与 CNN、ViT 在 COCO 数据集上的参数量-准确率对比图

3.2 主要模块

PVT 模型图如 3所示，主要分为 PatchEmbeding 和 Transformer Encoder 两部分，Transformer Encoder 又可分为 Spatial-reduction Attention（SRA）两部分，各模块作用如下。

PatchEmbedding：将每个 patch 拉成一维可训练的 embedding，同时嵌入 pos_embd 和 cls_token。

Transformer Encoder：特征提取主要部分，包含 SRA 和 MLP。

SRA：负责将 R 个 embedding 聚合得到 K、V，再与 Q 相乘得到相应的加权后的 embedding。

MLP：多层感知机，映射一组输入向量到一组输出向量。

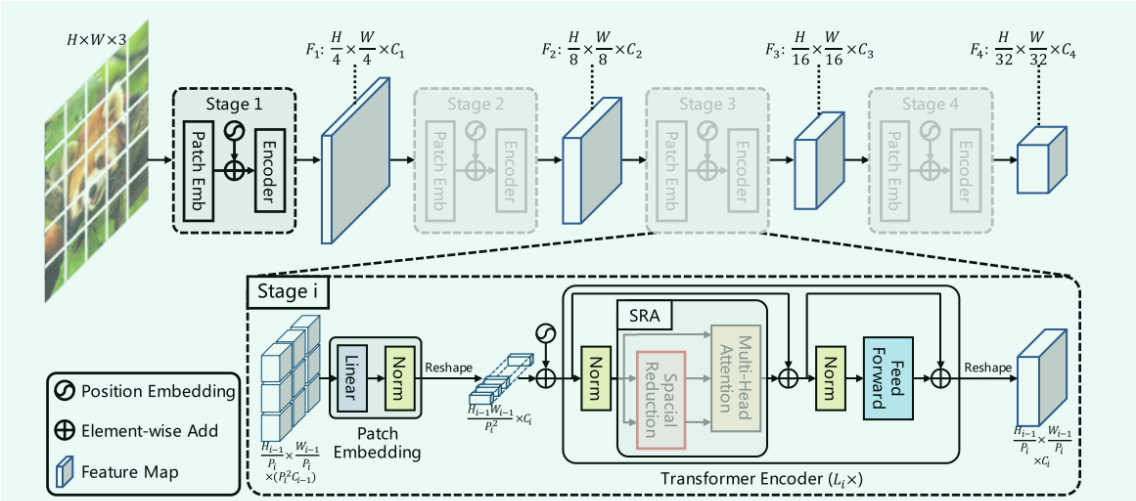


Figure 3: **Overall architecture of Pyramid Vision Transformer (PVT).** The entire model is divided into four stages, each of which is comprised of a patch embedding layer and a L_i-layer Transformer encoder. Following a pyramid structure, the output resolution of the four stages progressively shrinks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

图 3: PVT 模型结构图

3.3 损失函数定义

本人使用 Label Smoothing Cross Entropy 作为本次复现的损失函数。Label Smoothing 如公式 1 所示。

$$y_k^{LS} = y_k(1-\ ) + /K \tag{1}$$

其中 K 为类别数, α 为 label smoothing 引入的超参数，y_k 在 k 为正确类别时为 1，其余为 0，也就是 $y_k \in \{0,1\}$ 。Label Smoothing Cross Entropy 损失函数为公式 2 所示。

$$H(y,p) = \sum_{n=1}^{K} -y_k log(p_k) = -(1-\alpha+\alpha/K)logp_t - \alpha/K \sum_{i \neq t} logp_i \tag{2}$$

其中 p_t 为正确类别对应的输出概率，p_i 为错误类别的输出概率。

4 复现细节

4.1 与已有开源代码对比

本次复现使用的是 Jittor 框架，在代码编写上，本人更多地根据原论文给出的描述和公式进行代码编写，所实现的代码于源代码编写思路有很大的差异，各模块内容尽可能提炼、整合，最后完成相较于源代码可读性更高的代码。伪代码如下：

Procedure 1 PVT 网络伪代码

Input: 批图像 X
Output: 各类别概率 P
for *i* ***in*** *num_stages* **do**
    x = PathchEmbed(i, x)
    for *j* ***in*** *num_layers* **do**
        x = TransformerEncoder(i, x)
    end
    x = MLP(i, x)
end

4.2 实验环境搭建

硬件：

CPU：Intel I7-8700

RAM：32GB

GPU：RTX 3080Ti

软件：

OS：Windows11

Environment：Anaconda3 python3.8

IDE：PyCharm

框架：Jittor（计图）

4.3 SRA 模块解析

SRA 是本文提出的模块，主要作用为实现聚合运算以减少计算量，SRA 模块与传统 Multi-Head Attention 对比如图 4所示。

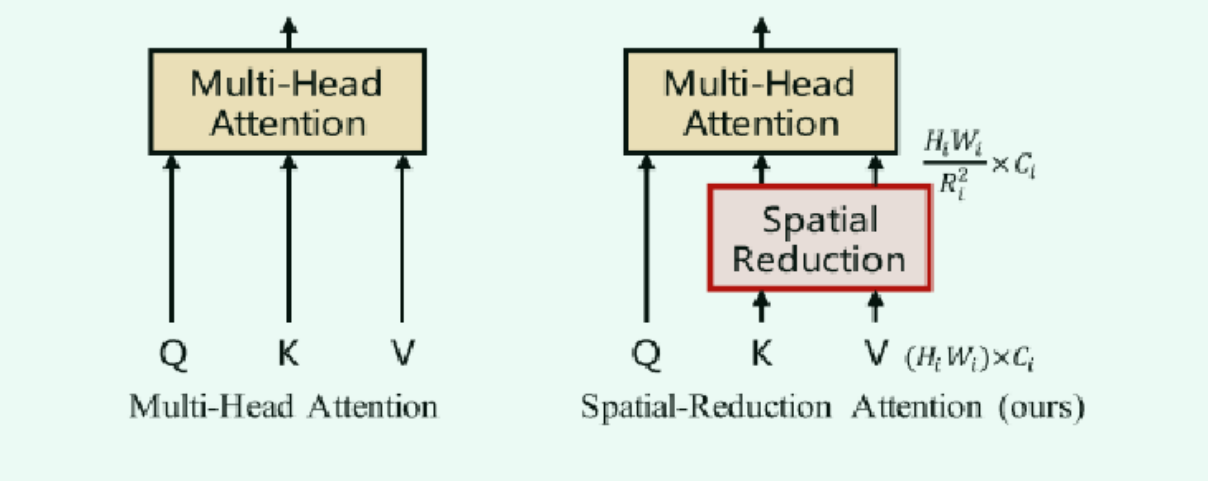


图 4: Multi-Head Attention(左) 与 Spatial-Reduction Attention(右) 结构对比图

SRA 计算公式如下所示：

$$SRA(Q, K, V) = Concat(head_0, ..., head_{N_i})W^O$$

$$head_j = Attention(QW_j^Q, SR(K)W_j^K, SR(V)W_j^V)$$

SR 计算公式如下：

$$SR(x) = Norm(Reshape(x, R_i W^S))$$

其中，w^S 为聚合的权重矩阵，将 R_i 个 K、V 进行聚合计算。SRA 模块的核心思想为：将 R 个点聚合成一个，计算 K,V，这样减少了 R^2 倍的计算量。

5 实验结果分析

复现结果如下，实验训练的 PVT 网络为 PVT-Tiny，训练时的准确率与 loss 如图 5所示，测试时的准确率与 loss 如图 6所示，其中灰线代表未使用数据增强的结果，粉红线代表使用数据增强的结果。对 cifar10 数据测试结果如图 7所示。

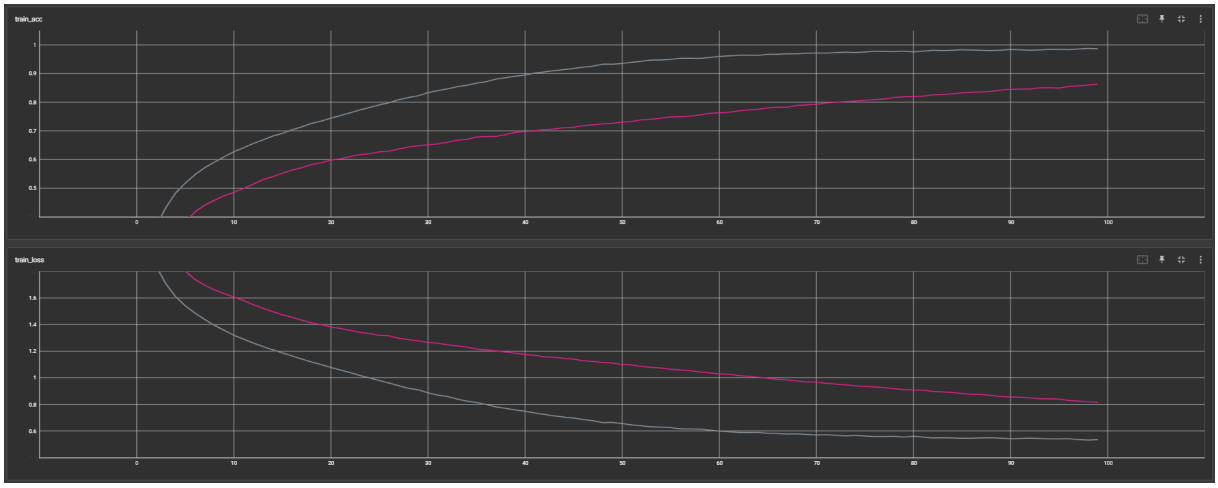


图 5: 训练时准确率与 loss 变化图

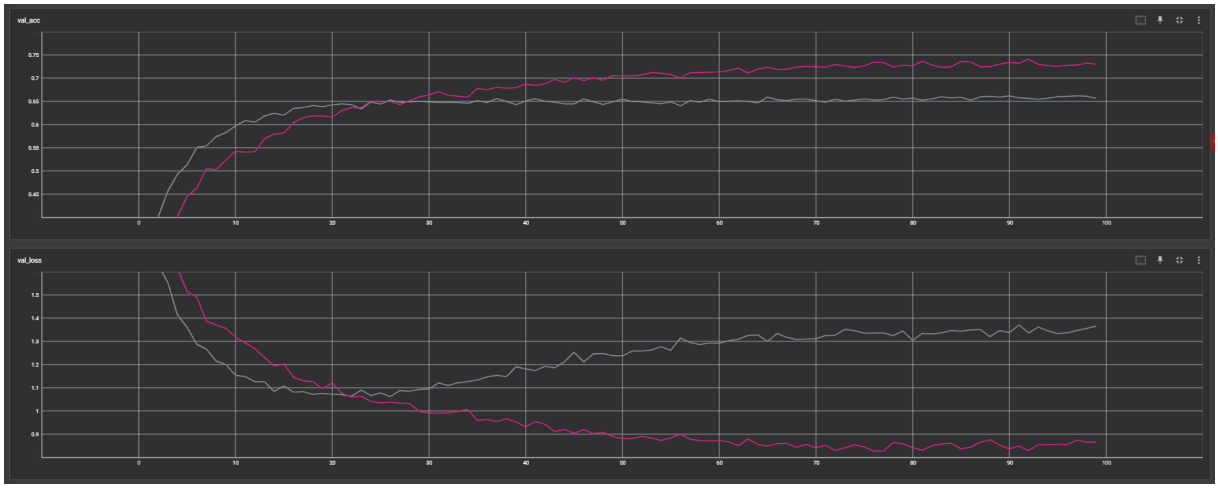
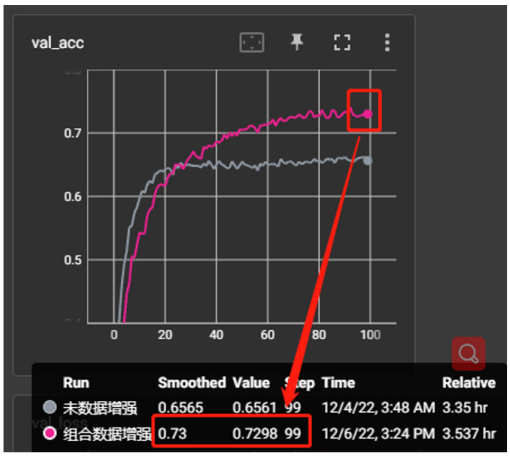


图 6: 测试时准确率与 loss 变化图

图 7: 数据测试结果图

所训练的 PVT-Tiny 取得与原论文相近的准确率（如图 8所示），最终测试的准确率约为 73%，与原论文取得的 75.1% 仅存在极小的差距，其中的原因可能来自于本人使用的数据增强手段及损失函数

与原作所使用的不一致，但这并不影响对复现结果的评判。



Method	#Param (M)	GFLOPs	Top-1 Err (%)
ResNet18* [22]	11.7	1.8	30.2
ResNet18 [22]	11.7	1.8	31.5
DeiT-Tiny/16 [63]	5.7	1.3	27.8
PVT-Tiny (ours)	13.2	1.9	24.9
ResNet50* [22]	25.6	4.1	23.9
ResNet50 [22]	25.6	4.1	21.5
ResNeXt50-32x4d* [73]	25.0	4.3	22.4
ResNeXt50-32x4d [73]	25.0	4.3	20.5

图 8: 复现结果与原论文准确率比较图

6 总结与展望

通过本次论文复现，加深了对 transformer 的理解，并且深刻掌握新的 backbone 思路，为接下来项目应用、网络结构的改进提供思路和启发。同时，本次复现上手新框架，培养了查阅文档的能力，扩展计算机视觉领域的视野。

参考文献

[1] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[2] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[3] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in neural information processing systems, 2017, 30.

[4] Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.