语义分割之 deeplab v3+

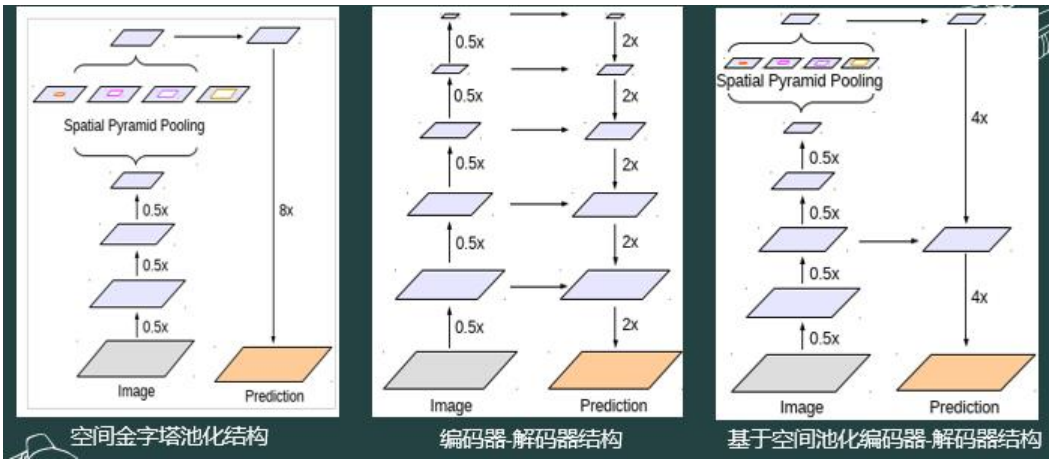
作者：vcjmhg
原文链接：https://ld246.com/article/1596017654281
来源网站：链滴

概述

deeplab v3+是deeplab系列中最新内容，也是当前最流行的语义分割算法，本篇文章主要记录的是人在学习deeplab v3+过程中的一些收获以及个人对该算法的理解。

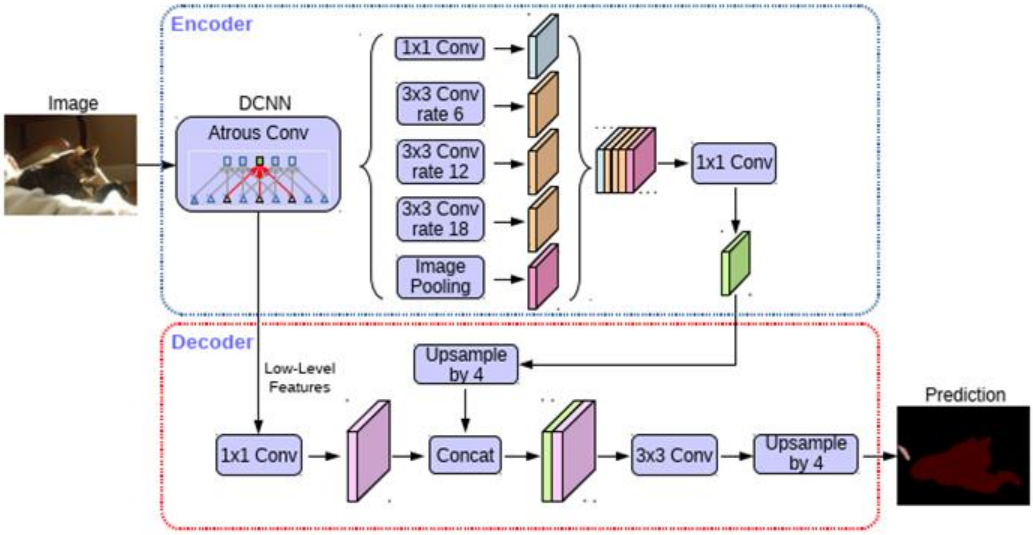
首先我们先简单回顾下deeplap v3 相关的创新点以及不足。在上一讲的时候我们讲到v3相比v2创新主要有四个方面，首先它提出了更加通用的框架，其次重新设计了空洞卷积，将空洞卷积和级联模块合起来使用，而不再单独使用。第三点它改进了ASPP，在ASPP的最后一层使用了BN层。最后一点去掉了CRF。并且我们在最后，也说了deeplab v3的不足点一就是v3获取处理结果的时候直接按照8者16进行上采样来获取最终结果，这样处理很粗糙。因为，直接按照8或16的采样率进行上采样的操并不能充分恢复降采样的过程中损失的细节信息，会导致分割不精确的情况发生。因此v3+的创新点一就是在在v3的基础上加入编码器和解码器结构来恢复原始分辨率的分割结果，使得边缘细节信息能较好的保留。同时另一个创新点是将v3的基础网络ResNet101换成了Xception网络，使用深度分割积进一步提高分割算法的速度和精度。

算法



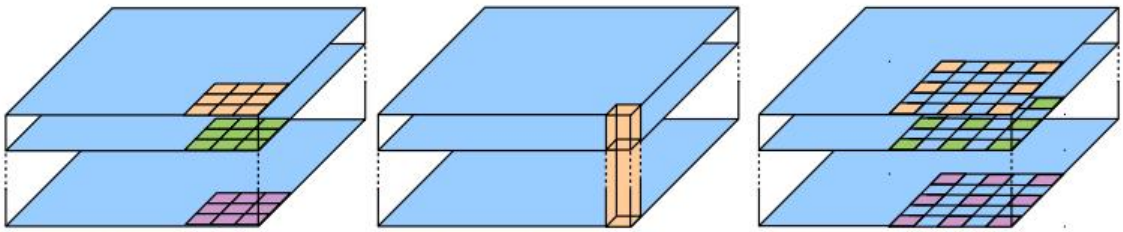
原文链接：语义分割之 deeplab v3+

首先我们看到左边这张图是是我们之前常见的空间金字塔池化结构，也就是我们之前v2和v3中用的结，它通过在不同分辨率上的池化操作来捕获丰富的上下文信息。而中间这张图片是FCN算法中用的网结构，我们称为编码器-解码器结构。在这个结构中编码器一般采用图像分类预训练得到的网络，采不断的池化和跨步卷积（strided convolution）能够获得长范围的语境信息，从而得到更好的分类结。然而在此过程中特征分辨率不断降低，图像细节信息丢失，这对于分割任务而言影响很大。因此在码器之后需要利用解码器进行图像分辨率的恢复。值得注意的一点：一般解码器常常具有类似skip的构将编码器得到的细节信息加入后续解码器的信息恢复中。而右边这幅图，展示的就是deeplab v3+所使用的新的结构，它是将前面两者结合起来，在空间池化的基础上使用了编码器和解码器架构，从能够有效的解决v3中使用上仅仅通过采样恢复特征图时导致细节信息丢失的问题。



我们详细讲下这个deeplab v3+提出的新结构。整个网络的编码器应该就不用再讲了，其实就是和deplab v3一样的，依然采用了ASPP的设计思想。因此我们主要看下，解码器部分的内容，这里和Deep3+最大的不同点就是v3+采用的是级联解码器，不是一步到位的。而v3只是在则是在双线性插值之后接按照原采样率进行上采样一步得出了结果。首先我们看解码器最左边，这里采用11的卷积核主要是了对编码过程中提取到low-level（低层次）的特征信息进行压缩（一般压缩到48）。这里压缩目的是么那？论文没有说，但应该是为了将那些低层次的特征信息压缩到和编码器输出的特征图4倍上采样后的空间分辨率保持一致，这样才能进行融合。然后将融合后的结果通过一个33的卷积来细化特征，后经过一个四倍上采样之后来输出最终结果。

接下来我们讲下v3+对网络架构的优化—将原来的ResNet101换成了Xception。在正式讲xception容之前我们先讲下深度可分离卷积。



深度可分离卷积首先对输入特征图的每一个通道分别单独做卷积操作，然后再用点卷积（1x1卷积）所有结果混合得到输出。这种方式相比普通卷积可以大幅减少需要的计算量，最后，再用空洞卷积代对每一个通道做的卷积操作，整个过程我们称为深度可分离卷积。

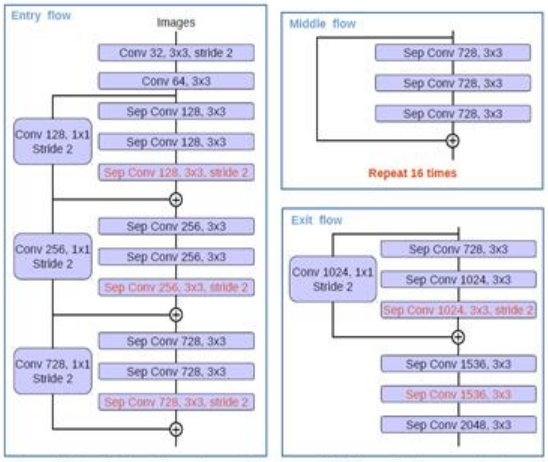


Figure 3. The Xception model is modified as follows: (1) more layers (same as MSRA's modification except the changes in Entry flow), (2) all the max pooling operations are replaced by depthwise separable convolutions with striding, and (3) extra batch normalization and ReLU are added after each 3 × 3 depthwise convolution, similar to MobileNet.

接下来我们看下对xception模型的修改，因为作者不是在v3+上直接就是用xception网络，而是在原的xception网络上进行修改，他修改的地方主要有三个方面，首先作者是用了更深的xception网络，和原网络的不同点在于，它不修改入口流网络结构从而实现计算效率和内存使用效率的提升。第二点所有最大值池化操作被深度可分离卷积替换，这使我们能够应用深度可分离卷积以任意分辨率提取。后一点是在每个3×3的深度卷积后增加BN层和ReLU。

实验

Channels	8	16	32	48	64
mIOU	77.61%	77.92%	78.16%	**78.21%**	77.94%

作者为了验证deeplab v3+的效果针对算法结构中的三个关键点，设计了一系列的实验进行验证。上这张图是不同通道下的mIOU图，我们可以看到当通道数是48的时候，效果是最好的。为了获得更好3卷积，作者设计了下边的实验，我们从实验结果中可以看到当使用两个3卷积进行串联，并连接co2特征模块的时候，模型效果最好。

Features Conv2	Features Conv3	3 × 3 Conv Structure	mIOU
✓		$[3 \times 3, 256]$	78.21%
✓		$[3 \times 3, 256] \times 2$	**78.85%**
✓		$[3 \times 3, 256] \times 3$	78.02%
✓		$[3 \times 3, 128]$	77.25%
✓		$[1 \times 1, 256]$	78.07%
✓	✓	$[3 \times 3, 256]$	78.61%

上边这张是使用ResNet101来作为主干网络的结果图，从图中我们可以看到：

- 左表的第二组采用的encoder，平均多增加了20B的计算消耗

● 测试了使用output_stride=32，这样计算速度更快。但是相对于output_stride=16 准确率下降了12%左右。

	Encoder train OS	Encoder eval OS	Decoder	MS	Flip	SC	COCO	JFT	mIOU	Multiply-Adds
baseline	16	16							79.17%	68.00B
	16	16		✓					80.57%	601.74B
	16	16		✓	✓				80.79%	1203.34B
	16	8							79.64%	240.85B
	16	8		✓					81.15%	2149.91B
	16	8		✓	✓				81.34%	4299.68B
添加decoder	16	16	✓						79.93%	89.76B
	16	16	✓	✓					81.38%	790.12B
	16	16	✓	✓	✓				81.44%	1580.10B
	16	8	✓						80.22%	262.59B
	16	8	✓	✓					81.60%	2338.15B
	16	8	✓	✓	✓				81.63%	4676.16B
使用深度分离卷积	16	16	✓			✓			79.79%	54.17B
	16	16	✓	✓	✓	✓			81.21%	928.81B
	16	8	✓			✓			80.02%	177.10B
	16	8	✓	✓	✓	✓			81.39%	3055.35B
	16	16	✓			✓	✓		82.20%	54.17B
	16	16	✓	✓	✓	✓	✓		83.34%	928.81B
	16	8	✓			✓	✓		82.45%	177.10B
	16	8	✓	✓	✓	✓	✓		83.58%	3055.35B
	16	16	✓			✓	✓	✓	83.03%	54.17B
	16	16	✓	✓	✓	✓	✓	✓	84.22%	928.81B
	16	8	✓			✓	✓	✓	83.39%	177.10B
	16	8	✓	✓	✓	✓	✓	✓	84.56%	3055.35B

Table 5. Inference strategy on the PASCAL VOC 2012 *val* set when using modified *Xception*. **train OS**: The *output stride* used during training. **eval OS**: The *output stride* used during evaluation. **Decoder**: Employing the proposed decoder structure. **MS**: Multi-scale inputs during evaluation. **Flip**: Adding left-right flipped inputs. **SC**: Adopting depthwise separable convolution for both ASPP and decoder modules. **COCO**: Models pretrained on MS-COCO. **JFT**: Models pretrained on JFT.

上边这张图是以xception为主干网络后的实验结果图，注意看跨红框的内容，我们和使用ResNet101主干网络的结果图进行对比，我们发现使用xception之后运算量变得更小，结果也更加准确。

Method	mIOU
Deep Layer Cascade (LC) [82]	82.7
TuSimple [77]	83.1
Large_Kernel_Matters [60]	83.6
Multipath-RefineNet [58]	84.2
ResNet-38_MS_COCO [83]	84.9
PSPNet [24]	85.4
IDW-CNN [84]	86.3
CASIA_IVA_SDN [63]	86.6
DIS [85]	86.8
DeepLabv3 [23]	85.7
DeepLabv3-JFT [23]	86.9
DeepLabv3+ (Xception)	87.8
DeepLabv3+ (Xception-JFT)	89.0

Method	Coarse	mIOU
ResNet-38 [83]	✓	80.6
PSPNet [24]	✓	81.2
Mapillary [86]	✓	82.0
DeepLabv3	✓	81.3
DeepLabv3+	✓	82.1

最后两张图是deeplab v3+在PASCAl VOC 2012数据集合和cityspaces数据集上与其他算法的效果比图，我们可以看到deeplab v3+算法取得了最好的结果。

结论

最后作者得出结论：实验结果表明，deeplab v3所提出的模型在PASCAL VOC 2012和Cityscapes数集上取得了最好的性能表现。