**图像分割**

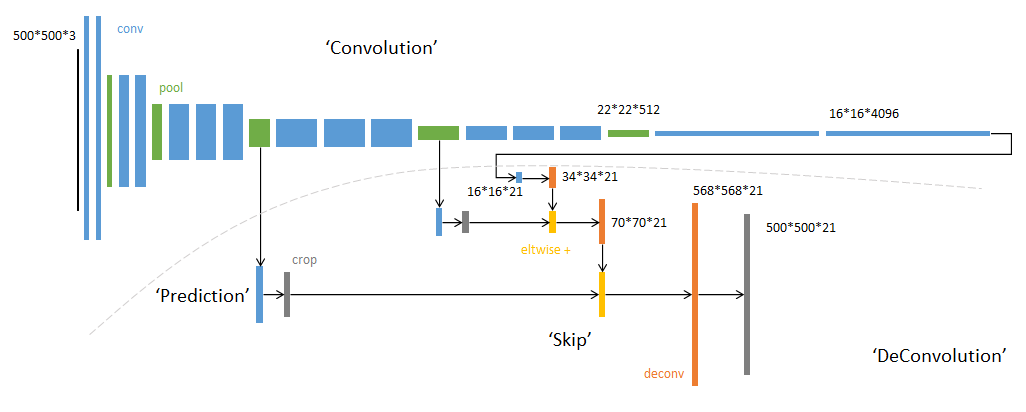
CNN图像语义分割基本上是这个套路：

* 下采样+上采样：Convlution + Deconvlution／Resize
* 多尺度特征融合：特征逐点相加／特征channel维度拼接
* 获得像素级别的segement map：对每一个像素点进行判断类别

1. **Fully Convolutional Networks (FCN)**

第一篇论文是Fully Convolutional Networks for Semantic Segmentation，简称FCN。这篇论文是第一篇成功使用深度学习做图像语义分割的论文。论文的主要贡献有两点：

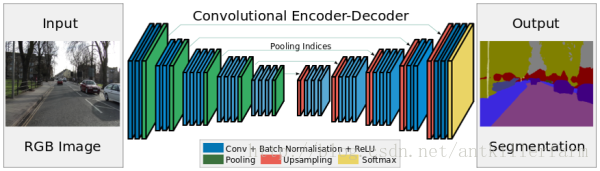
* 提出了全卷积网络。将全连接网络替换成了卷积网络，使得网络可以接受任意大小的图片，并输出和原图一样大小的分割图。只有这样，才能为每个像素做分类。
* 使用了反卷积层（Deconvolution）。分类神经网络的特征图一般只有原图的几分之一大小。想要映射回原图大小必须对特征图进行上采样，这就是反卷积层的作用。虽然名字叫反卷积层，但其实它并不是卷积的逆操作，更合适的名字叫做转置卷积（Transposed Convolution），作用是从小的特征图卷出大的特征图。



一些重点：

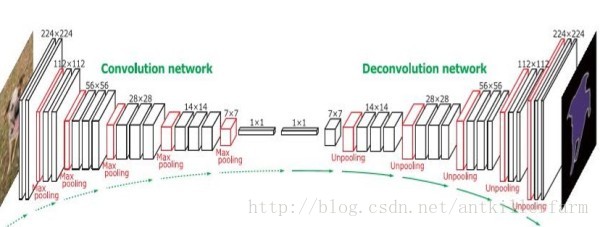
* 损失函数是在最后一层的 spatial map上的 pixel 的 loss 和，在每一个 pixel 使用 softmax loss
* 使用 skip 结构融合多层（3层）输出，底层网络应该可以预测更多的位置信息，因为他的感受野小可以看到小的 pixels
* 上采样 lower-resolution layers 时，如果采样后的图因为 padding 等原因和前面的图大小不同，使用 crop ，当裁剪成大小相同的，spatially aligned ，使用 concat 操作融合两个层

1. **SegNet**



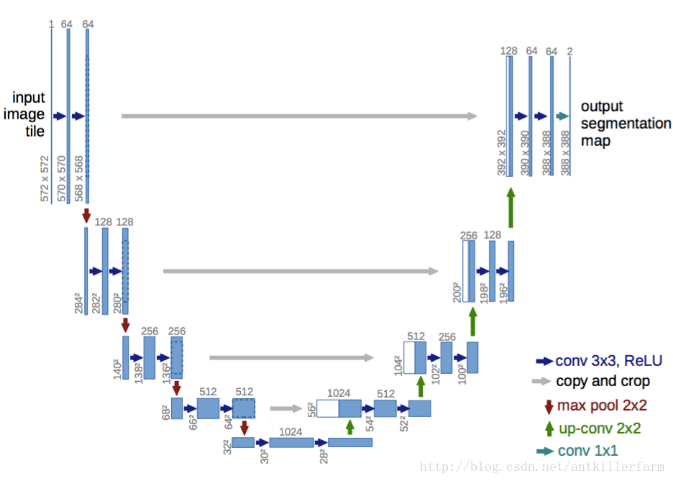
相比于CNN下采样阶段的结构规整，FCN上采样时的结构就显得凌乱了。因此，SegNet采用了几乎和下采样对称的上采样结构。

1. **DeconvNet**



从上图可见，DeconvNet和SegNet的结构非常类似，只不过DeconvNet在encoder和decoder之间使用了FC层作为中继。

1. **U-Net**



1. **DeepLab**

**CRF(条件随机场)**的二元势函数一般是描述像素点与像素点之间的关系，鼓励相似像素分配相同的标签，而相差较大的像素分配不同标签，而这个“距离”的定义与颜色值和实际相对距离有关，这样CRF能够使图像尽量在边界处分割。全连接CRF模型的不同就在于其二元势函数描述的是每一个像素与其他所有像素的关系，使用该模型在图像中的所有像素对上建立点对势能从而实现极大地细化和分割。

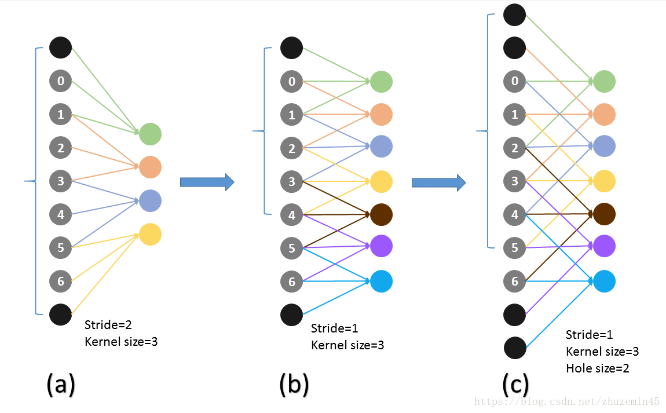
**Deeplab v1**

 DeeplabV1方法分为两步走，第一步仍然采用了FCN得到 coarse score map并插值到原图像大小，然后第二步借用fully connected CRF对从FCN得到的分割结果进行细节上的refine。

在第一步中，deeplab仍然采用了FCN来得到score map,并且也是在VGG网络上进行fine-tuning。但是在得到score map的处理方式上，要比原FCN处理细腻很多。 CVPR 2015的FCN中输入是一张500x500的图像，在第一个卷积层上conv1\_1来了一个100的大padding。最终在fc7层勉强得到一个16x16的score map。虽然处理上相对粗糙，但确是第一次将图像分割在CNN上完成end-to-end，并且在当时performance是state-of-the-art，也很理解。 DeeplabV1摒弃了这种做法，取而代之的是对VGG的网络结构上做了小改动：将VGG网络的pool4和pool5层的stride由原来的2改为了1。就是这样一个改动，使得vgg网络总的stride由原来的32变成8，进而使得在输入图像为514x514，正常的padding时，fc7能得到67x67的score map, 要比FCN确实要dense很多很多。但是这种改变网络结果的做法也带来了一个问题： stride改变以后，如果想继续利用vgg model进行fine tuning，会导致后面filter作用的区域发生改变，换句话说就是感受野发生变化。

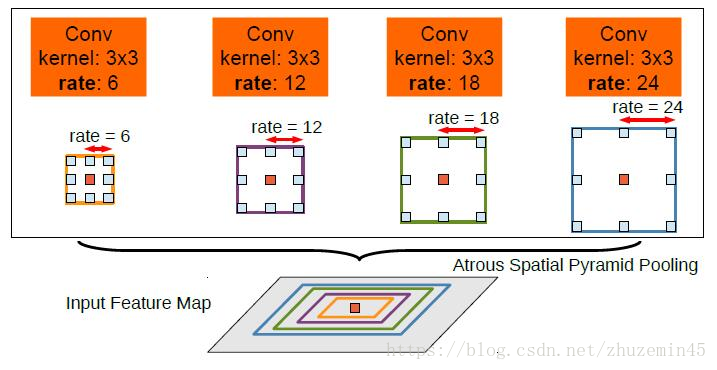
**Astrous conv**

作者想出了一招，来解决两个看似有点矛盾的问题： 既想利用已经训练好的模型进行fine-tuning，又想改变网络结构得到更加dense的score map. 这个解决办法就是采用Hole算法。如下图(a) (b)所示，在以往的卷积或者pooling中，一个filter中相邻的权重作用在feature map上的位置都是物理上连续的。如下图(c)所示，为了保证感受野不发生变化，某一层的stride由2变为1以后，后面的层需要采用hole算法，具体来讲就是将连续的连接关系是根据hole size大小变成skip连接的（图(c)为了显示方便直接画在本层上了）。注意虽然(c)中的padding为2，但是padding的两个像素不会连在同一个filter上。 pool4的stride由2变为1，则紧接着的conv5\_1, conv5\_2和conv5\_3中hole size为2。接着pool5由2变为1, 则后面的fc6中hole size为4。



**DeepLab V2**

Multi-scale对performance提升很大，而我们知道，receptive field，视野域（或者感受野），是指feature map上一个点能看到的原图的区域，那么如果有多个receptive field，是不是相当于一种Multi-scale？出于这个思路，V2版本在V1的基础上增加了一个多视野域。具体看图可以很直观的理解。



rate也就是hole size，这个结构作者称之为ASPP（atrous spatial pyramid pooling），基于洞的空间金字塔此外，DeepLab V2有两个基础网络结构，一个是基于vgg16，另外一个是基于resnet101的。

**Deeplab v3**

v3的创新点一是改进了ASPP模块；二是参考了图森组的[Understanding Convolution for Semantic Segmentation](https://arxiv.org/abs/1702.08502" \t "_blank)中HDC的思想。其实就是对应纵横两种结构。**backbone还是resnet 101.**

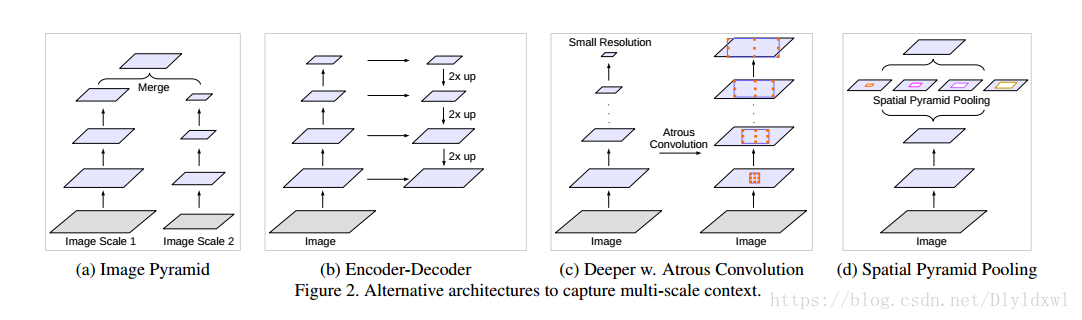
论文中Fig2画了几种常见的捕获multi-scale context的方法。

（a）图像金字塔。输入图像进行尺度变换得到不同分辨率input，然后将所有尺度的图像放入CNN中得到不同尺度的分割结果，最后将不同分辨率的分割结果融合得到原始分辨率的分割结果，类似的方法为DeepMedic；

（b）编码-解码。FCN和UNet等结构；

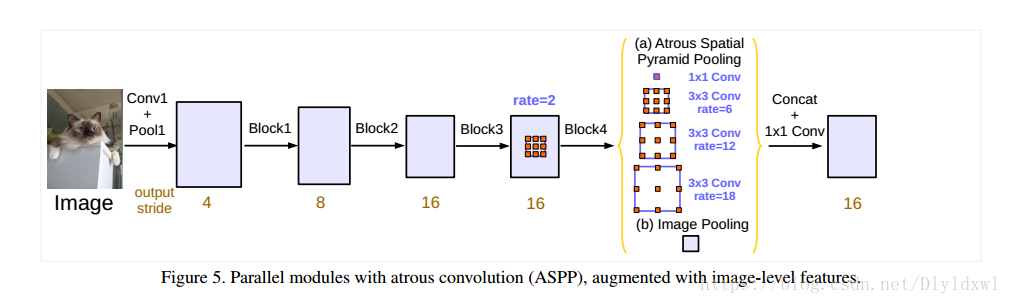
（c）本文提出的串联结构。

（d）本文提出的Deeplab v3结构。最后两个结构右边其实还需要8×/16×的upsample，在deeplab v3+中有所体现。当然论文的Sec 4.1也有提到，下采样GT容易在反向传播中丢失细节，因此上采样feature map效果更好。



**ASPP的改进**

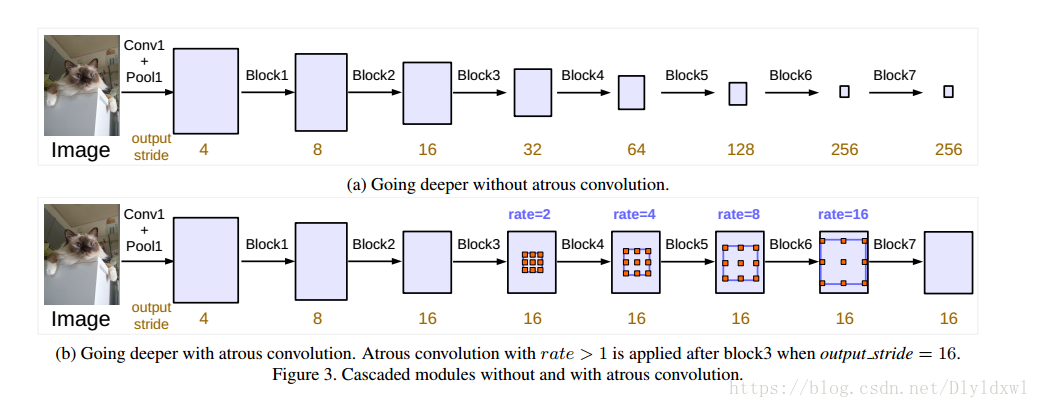
改进后的aspp长下图那个样子，**多了个1\*1的conv和global avg pool**。关于1\*1卷积，论文中3.3第一段解释的有点意思，当rate=feature map size时，dilation conv就变成了1\*1 conv，所以这个1\*1conv相当于rate很大的空洞卷积。此外引入了全局池化这个branch，这个思想是来源于PSPnet（参考博客）,简言之就是spp在分割上的应用，多尺度pooling。根据代码实现来看，每个branch后面都没有relu，其实有没有BN，个人觉得不是很要紧，毕竟BN是线性操作，可以合并到conv里面，论文的Sec 4.1 说明了V3的所有层是用了BN的，BN可以加速训练还有弱正则，所以一般都会用。针对ASPP，作者设计了一种“纵式”的结构，如下图fig5。



deeplab v3中说到了需要8×/16×的upsample 最终的feature map，很明显这是一个很粗糙的做法。

**"串联"结构**

如下图所示，复制conv4的结构3次，后面的每个block都有一个基准dilation Rate，在每一个block里面参考HDC的思想，又设置了[1,2,1]的rate，所以每个conv的rate = Rate\*rate.在论文4.2的Multi-grid部分详细进行了解释对比。



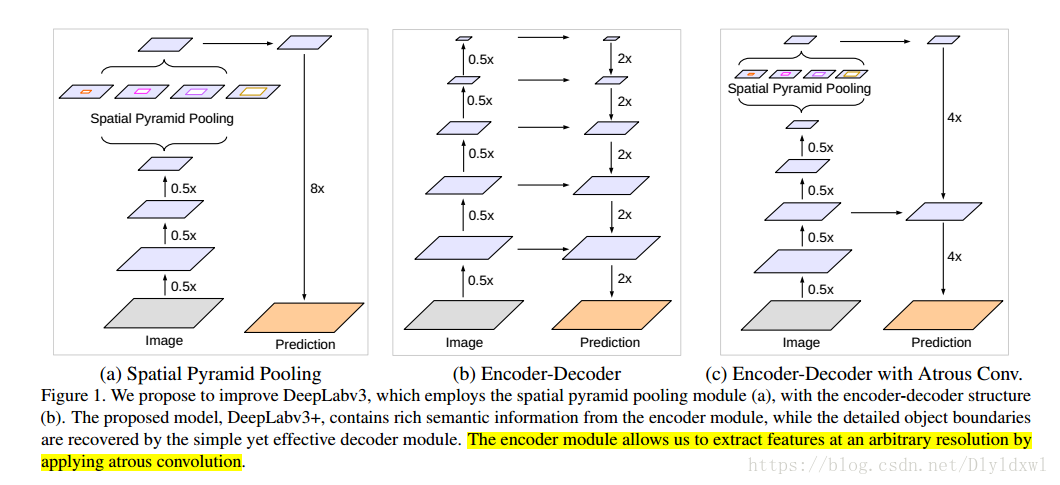
两种方法的结构合并并不会带来提升，相比较来说，aspp的纵式结构要好一点。所以deeplab v3一般也是指aspp的结构。

**Deeplab v3+**

在deeplab v3中说到了需要8×/16×的upsample 最终的feature map，很明显这是一个很粗糙的做法。

**v3+的创新点一是设计基于v3的decode module，二是用modify xception作为backbone。**

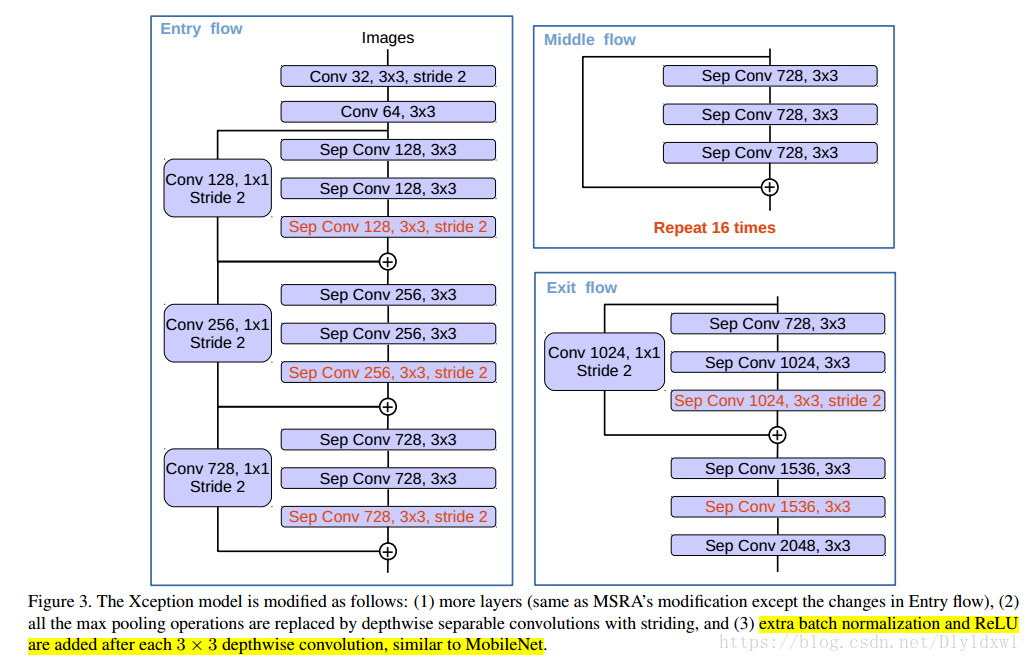
论文中同样给出了一幅对比图，（a）是v3的纵式结构，（b）是常见的编码—解码结构，（c）是本文提出的基于deeplab v3的encode-decode结构。



论文中介绍了两种backbone，一是Resnet101，二是改进后的xception。xception效果好于resnet，所以我只关注了xception，毕竟v3+主打也是xception backbone。下面从backbone和decode来简要概括v3+的结构。

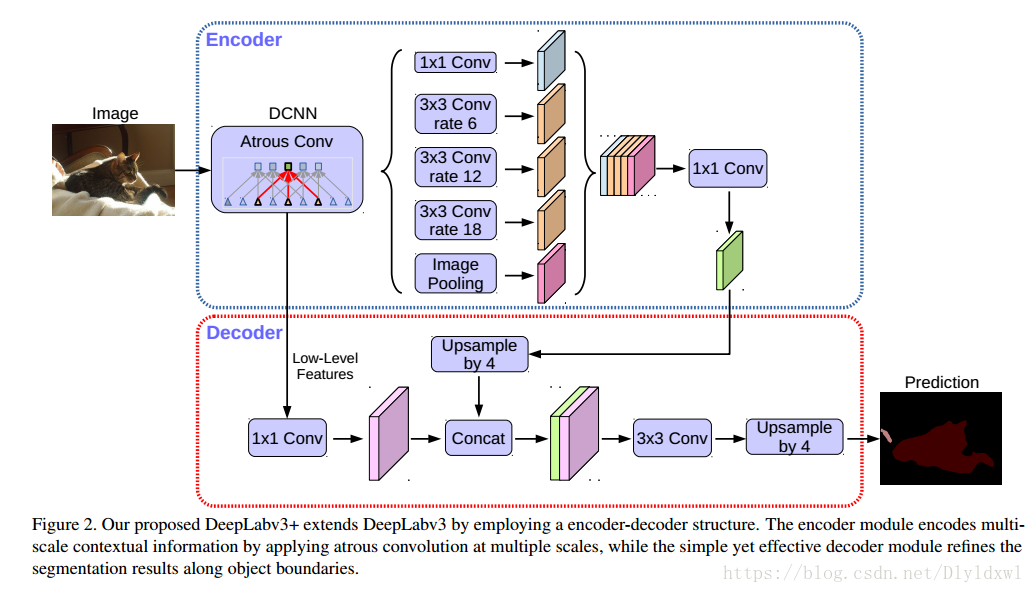
**Modify Xception**

具体结构参照下图。和原来的xception不一样的地方有三，一是Middle flow重复了16次，ori xception是8次，也就是用了more layers；二是pooling均换为了dw+pw+s=2，本人在很多网络也实验过，把pool换成conv或者合并到现有的卷积层，均能提高一定的map。在Entry flow中stride =16，所以训练的时候需要把Exit flow的stride=2换为rate=2的dilation conv（Middle不改变resolution）（train\_os=16），因此后面的层也都是rate=2，这又导致The Gridding Effect，然而论文4.3中说明了这个地方用HDC思想并不会带来提升，因此保留了一系列的rate=2；三是在所有的dw层后面加上了BN和relu，加BN无可厚非，线性操作而已，但是重新加relu是真的玄学啊。



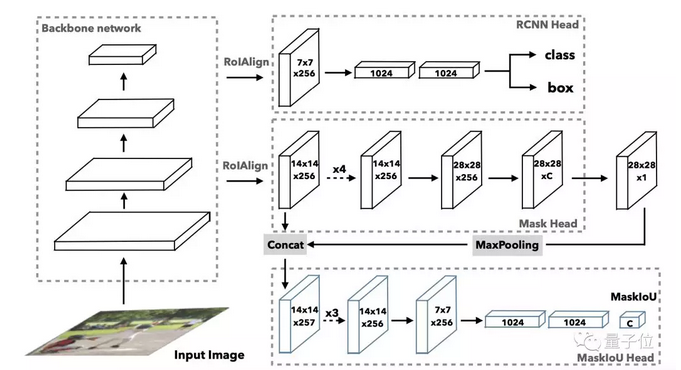
**Decoded module**

xception的输出2048维特征接到ASPP上得到256维multi-scale context feature map（一般s=16），再4×上采样，和backbone上的同分辨率的low-level feature map concat（一般是entry flow的第一个shortcut block的输出，刚好s=4）。这个时候要让low-level feature map在concat后的总特征图中占比小，因为它的语义信息太少了，所以接了1\*1的低维conv，这个地方可不是为了降低计算量，关于这个conv的channel选取，论文给出了对比试验Table 1。concat后再接3\*3 conv block，它的channel和block个数，论文中也进行了实验验证Table 2.最后再进行4×上采样，达到原图的分辨率。此外，作者还实验了将aspp和decode中的卷积替换为depthwise conv，mIOU没有明显降低，flops大大降低了。



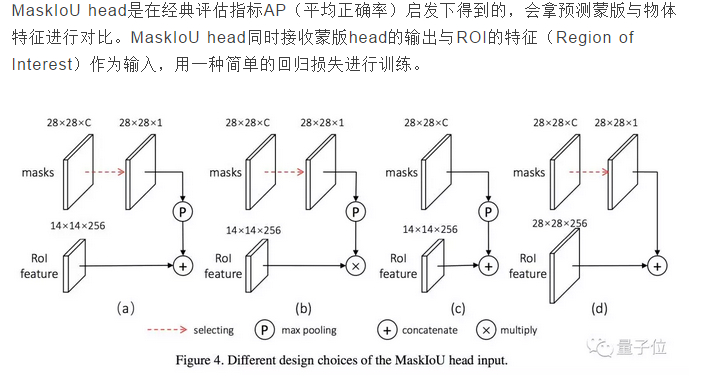
1. **Mask Scoring R-CNN(cvpr2019)**

这篇CVPR 2019论文就提出了一种新的打分方法：**给蒙版打分**，他们称之为**蒙版得分**（mask score）



Mask Scoring R-CNN中提出的计分方式很简单：不仅仅直接依靠检测得到的分类算分，而且还让模型单独学一个针对蒙版的得分规则：**MaskIoU head**。

MaskIoU head是在经典评估指标AP（平均正确率）启发下得到的，会拿预测蒙版与物体特征进行对比。MaskIoU head同时接收蒙版head的输出与ROI的特征（Region of Interest）作为输入，用一种简单的回归损失进行训练。



最后，**同时考虑分类得分与蒙版的质量得分**，就可以去评估算法质量了。

评测方法公平公正，实例分割模型性能自然也上去了。

实验证明，在挑战COCO benchmark时，在用MS R-CNN的蒙版得分评估时，在不同基干网路上，AP始终提升近1.5%。

**注意：原mask rcnn**

**损失函数：分类误差+检测误差+分割误差，即L=Lcls + Lbox + Lmask**

