第七章 图像分割

8.1 传统的基于 CNN 的分割方法缺点？

传统的基于 CNN 的分割方法：为了对一个像素分类，使用该像素周围的一个图像块作为 CNN 的输入，用于训练与预测，这种方法主要有几个缺点：

1）存储开销大，例如，对每个像素使用 15*15 的图像块，然后不断滑动窗口，将图像块输入到 CNN 中进行类别判断，因此，需要的存储空间随滑动窗口的次数和大小急剧上升；

2）效率低下，相邻像素块基本上是重复的，针对每个像素块逐个计算卷积，这种计算有很大程度上的重复；

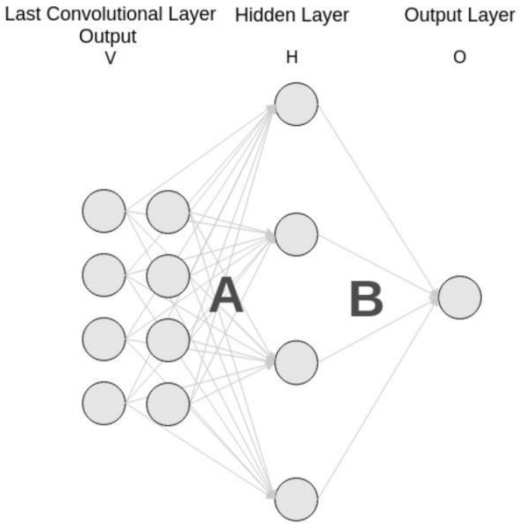
3）像素块的大小限制了感受区域的大小，通常像素块的大小比整幅图像的大小小很多，只能提取一些局部特征，从而导致分类性能受到限制。

而全卷积网络(FCN)则是从抽象的特征中恢复出每个像素所属的类别。即从图像级别的分类进一步延伸到像素级别的分类。

8.1 FCN

8.1.1 FCN 改变了什么？

对于一般的分类 CNN 网络，如 VGG 和 Resnet，都会在网络的最后加入一些全连接层，经过 softmax 后就可以获得类别概率信息。但是这个概率信息是 1 维的，即只能标识整个图片的类别，不能标识每个像素点的类别，所以这种全连接方法不适用于图像分割。



而 FCN 提出可以把后面几个全连接都换成卷积，这样就可以获得一张 2 维的 feature map，后接 softmax 获得每个像素点的分类信息，从而解决了分割问题，如图 4。

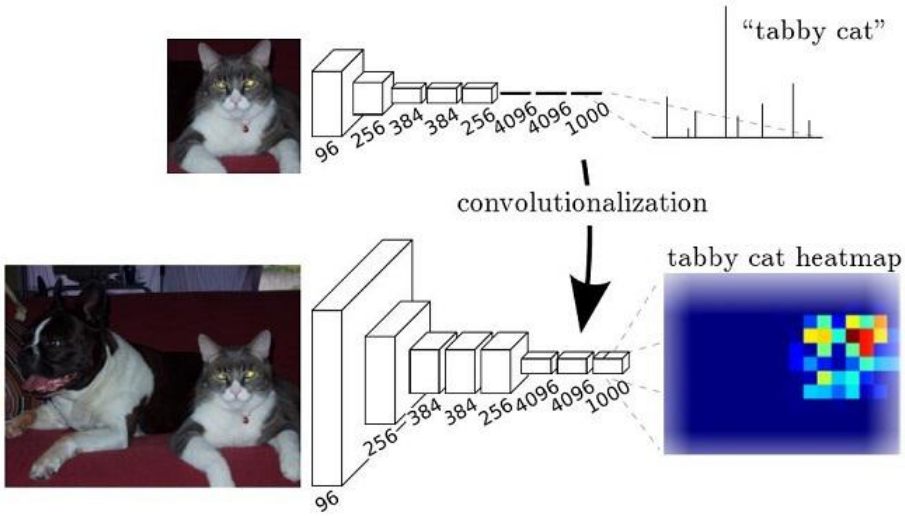
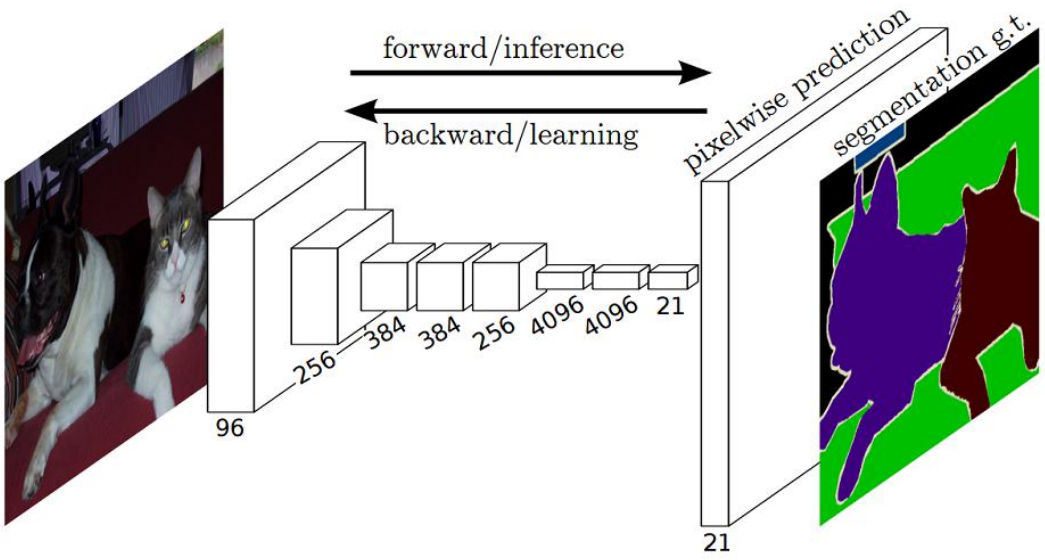


图 4

8.1.2 FCN 网络结构？

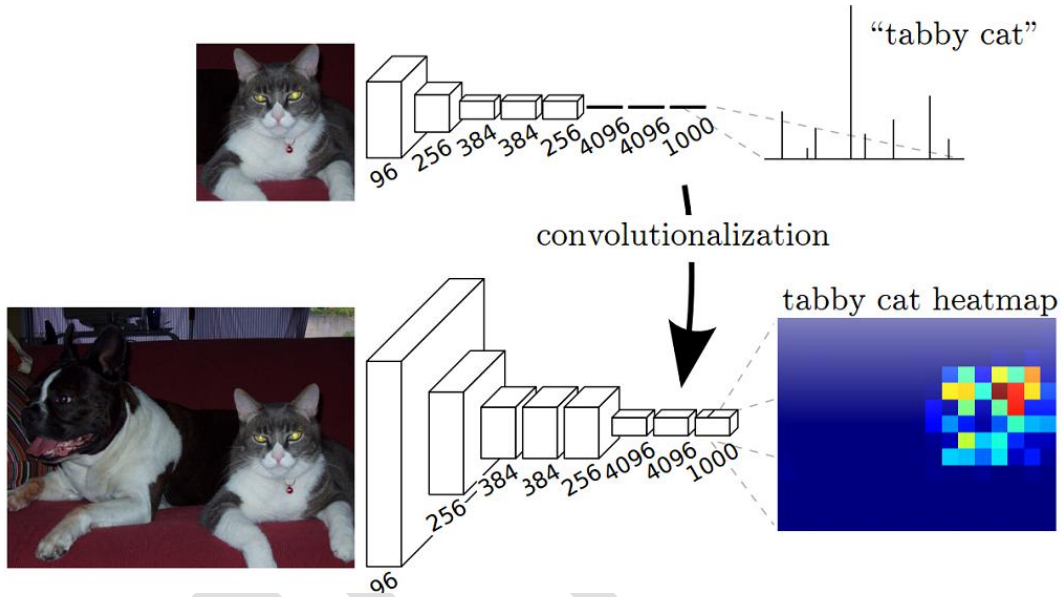
FCN 对图像进行像素级的分类，从而解决了语义级别的图像分割（semantic segmentation）问题。与经典的 CNN 在卷积层之后使用全连接层得到固定长度的特征向量进行分类（全联接层＋softmax 输出）不同，FCN 可以接受任意尺寸的输入图像，采用反卷积层对最后一个卷积层的 feature map 进行上采样，使它恢复到输入图像相同的尺寸，从而可以对每个像素都产生了一个预测，同时保留了原始输入图像中的空间信息，最后在上采样的特征图上进行逐像素分类。

下图是语义分割所采用的全卷积网络(FCN)的结构示意图：



8.1.3 全卷积网络举例？

通常 CNN 网络在卷积层之后会接上若干个全连接层, 将卷积层产生的特征图(feature map)映射成一个固定长度的特征向量。以 AlexNet 为代表的经典 CNN 结构适合于图像级的分类和回归任务，因为它们最后都得到整个输入图像的一个概率向量，比如 AlexNet 的 ImageNet 模型输出一个 1000 维的向量表示输入图像属于每一类的概率(softmax 归一化)。



如图所示：

（1）在 CNN 中, 猫的图片输入到 AlexNet, 得到一个长为 1000 的输出向量, 表示输入图像属于每一类的概率, 其中在“tabby cat”这一类统计概率最高, 用来做分类任务。

（2）FCN 与 CNN 的区别在于把于 CNN 最后的全连接层转换成卷积层，输出的是一张已经 Label 好的图片, 而这个图片就可以做语义分割

（3）CNN 的强大之处在于它的多层结构能自动学习特征，并且可以学习到多个层次的特征：较浅的卷积层感知域较小，学习到一些局部区域的特征 较深的卷积层具有较大的感知域，能够学习到更加抽象一些的特征。高层的抽象特征对物体的大小、位置和方向等敏感性更低，从而有助于识别性能的提高, 所以我们常常可以将卷积层看作是特征提取器。

8.1.4 为什么 CNN 对像素级别的分类很难？

（1）存储开销很大。例如对每个像素使用的图像块的大小为 15x15，然后不断滑动窗口，每次滑动的窗口给 CNN 进行判别分类，因此则所需的存储空间根据滑动窗口的次数和大小急剧上升。

（2）计算效率低下。相邻的像素块基本上是重复的，针对每个像素块逐个计算卷积，这

种计算也有很大程度上的重复。

（3）像素块大小的限制了感知区域的大小。通常像素块的大小比整幅图像的大小小很多，只能提取一些局部的特征，从而导致分类的性能受到限制。

8.1.5 全连接层和卷积层如何相互转化？

两者相互转换的可能性：

全连接层和卷积层之间唯一的不同就是卷积层中的神经元只与输入数据中的一个局部区域连接，并且在卷积列中的神经元共享参数。然而在两类层中，神经元都是计算点积，所以它们的函数形式是一样的。因此，将此两者相互转化是可能的：

（1）对于任一个卷积层，都存在一个能实现和它一样的前向传播函数的全连接层。权重矩阵是一个巨大的矩阵，除了某些特定块，其余部分都是零。而在其中大部分块中，元素都是相等的。

（2）任何全连接层都可以被转化为卷积层。比如 VGG16 中第一个全连接层是 25088*4096 的数据尺寸，将它转化为 512*7*7*4096 的数据尺寸，即一个 K=4096 的全连接层，输入数据体的尺寸是 7*7*512，这个全连接层可以被等效地看做一个 F=7,P=0,S=1,K=4096 的卷积层。换句话说，就是将滤波器的尺寸设置为和输入数据体的尺寸一致 7*7, 这样输出就变为 1*1*4096, 本质上和全连接层的输出是一样的。

输出激活数据体深度是由卷积核的数目决定的(K=4096)。

在两种变换中，将全连接层转化为卷积层在实际运用中更加有用。假设一个卷积神经网络的输入是 227x227x3 的图像，一系列的卷积层和下采样层将图像数据变为尺寸为 7x7x512 的激活数据体, AlexNet 的处理方式为使用了两个尺寸为 4096 的全连接层，最后一个有 1000 个神经元的全连接层用于计算分类评分。我们可以将这 3 个全连接层中的任意一个转化为卷积层：

（1）第一个连接区域是[7x7x512]的全连接层，令其滤波器尺寸为 F=7,K=4096，这样输出数据体就为[1x1x4096]。

（2）第二个全连接层，令其滤波器尺寸为 F=1,K=4096，这样输出数据体为[1x1x4096]。

（3）最后一个全连接层也做类似的，令其 F=1,K=1000，最终输出为[1x1x1000]。

8.1.6 FCN 的输入图片为什么可以是任意大小？

对于 CNN，一幅输入图片在经过卷积和 pooling 层时，这些层是不关心图片大小的。比如对于一个卷积层，outputsize = (inputsize - kernelsize) / stride + 1，它并不关心 inputsize 多大，对于一个 inputsize 大小的输入 feature map，滑窗卷积，输出 outputsize 大小的 feature map 即可。pooling 层同理。但是在进入全连接层时，feature map（假设大小为 n×n）要拉成一条向量，

而向量中每个元素（共 n×n 个）作为一个结点都要与下一个层的所有结点（假设 4096 个）全连接，这里的权值个数是 4096×n×n，而我们知道神经网络结构一旦确定，它的权值个数都是固定的，所以这个 n 不能变化，n 是 conv5 的 outputsize，所以层层向回看，每个 outputsize 都要固定，那每个 inputsize 都要固定，因此输入图片大小要固定。

8.1.7 把全连接层的权重 W 重塑成卷积层的滤波器有什么好处？

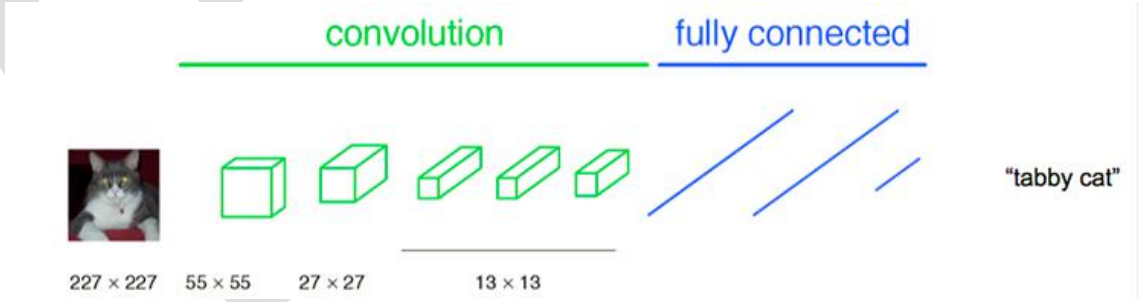
这样的转化可以在单个向前传播的过程中，使得卷积网络在一张更大的输入图片上滑动，从而得到多个输出(可以理解为一个 label map)。

比如：我们想让 224×224 尺寸的浮窗，以步长为 32 在 384×384 的图片上滑动，把每个经停的位置都带入卷积网络，最后得到 6×6 个位置的类别得分，那么通过将全连接层转化为卷积层之后的运算过程为:

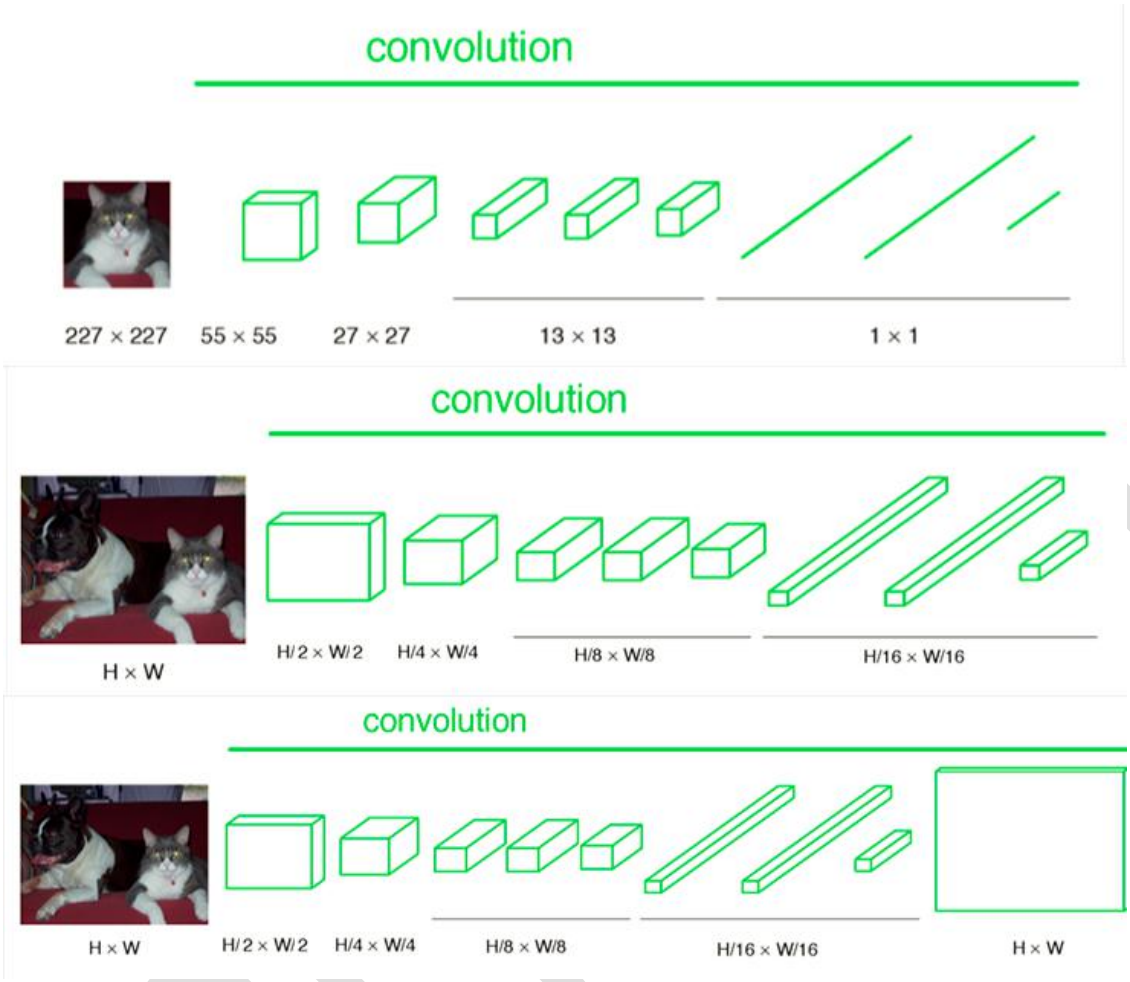
如果 224×224 的输入图片经过卷积层和下采样层之后得到了[7x7x512]的数组，那么，384×384 的大图片直接经过同样的卷积层和下采样层之后会得到[12x12x512]的数组，然后再经过上面由 3 个全连接层转化得到的 3 个卷积层，最终得到[6x6x1000]的输出((12 − 7)/1 + 1 = 6)，这个结果正是浮窗在原图经停的 6×6 个位置的得分。

一个确定的 CNN 网络结构之所以要固定输入图片大小，是因为全连接层权值数固定，而该权值数和 feature map 大小有关，但是 FCN 在 CNN 的基础上把 1000 个结点的全连接层改为含有 1000 个 1×1 卷积核的卷积层，经过这一层，还是得到二维的 feature map，同样我们也不关心这个 feature map 大小，所以对于输入图片的 size 并没有限制。

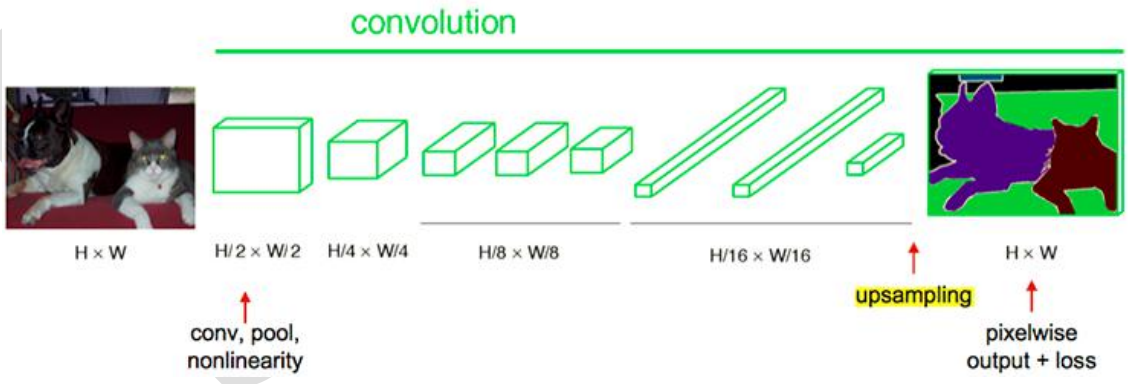
如下图所示，FCN 将传统 CNN 中的全连接层转化成卷积层，对应 CNN 网络 FCN 把最后三层全连接层转换成为三层卷积层:



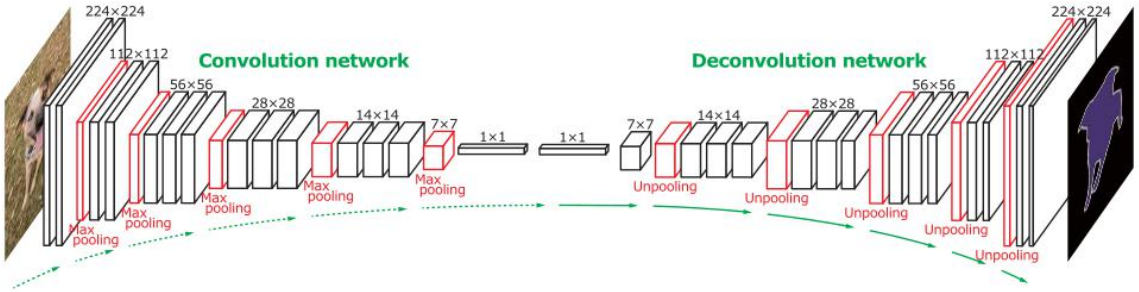
一个分类网络



变为全卷积网络



End-to-end,pixels-to pixels 网络



（1）全连接层转化为全卷积层 ：在传统的 CNN 结构中，前 5 层是卷积层，第 6 层和第 7 层分别是一个长度为 4096 的一维向量，第 8 层是长度为 1000 的一维向量，分别对应 1000 个不同类别的概率。FCN 将这 3 层表示为卷积层，卷积核的大小 (通道数，宽，高) 分别为 (4096,1,1)、(4096,1,1)、(1000,1,1)。看上去数字上并没有什么差别，但是卷积跟全连接是不一样的概念和计算过程，使用的是之前 CNN 已经训练好的权值和偏置，但是不一样的在于权值和偏置是有自己的范围，属于自己的一个卷积核。

（2）CNN 中输入的图像大小是统一固定成 227x227 大小的图像，第一层 pooling 后为 55x55，第二层 pooling 后图像大小为 27x27，第五层 pooling 后的图像大小为 13x13，而 FCN 输入的图像是 H*W 大小，第一层 pooling 后变为原图大小的 1/2，第二层变为原图大小的 1/4，第五层变为原图大小的 1/8，第八层变为原图大小的 1/16。

（3）经过多次卷积和 pooling 以后，得到的图像越来越小，分辨率越来越低。其中图像到 H/32*W/32 的时候图片是最小的一层时，所产生图叫做 heatmap 热图，热图就是我们最重要的高维特征图，得到高维特征的 heatmap 之后就是最重要的一步也是最后的一步对原图像进行 upsampling，把图像进行放大几次到原图像的大小。

相较于使用被转化前的原始卷积神经网络对所有 36 个位置进行迭代计算优化模型，然后再对 36 个位置做预测，使用转化后的卷积神经网络进行一次前向传播计算要高效得多，因为 36 次计算都在共享计算资源。这一技巧在实践中经常使用，通常将一张图像尺寸变得更大，然后使用变换后的卷积神经网络来对空间上很多不同位置进行评价得到分类评分，然后在求这些分值的平均值。

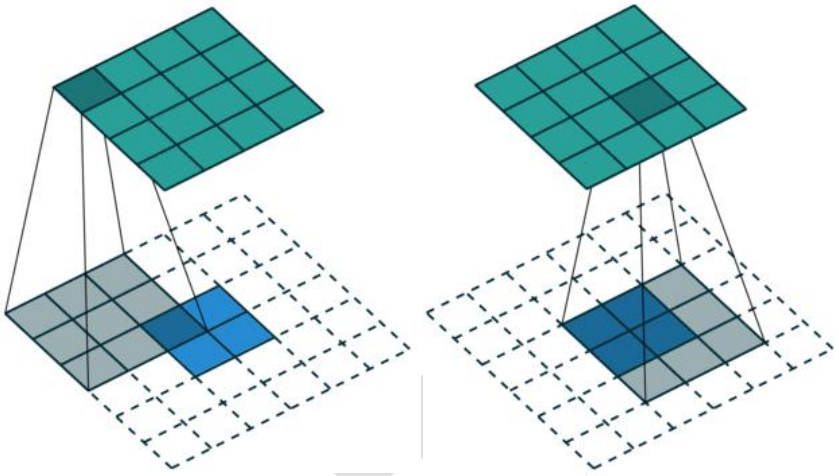
8.1.8 反卷积层理解

Upsampling 的操作可以看成是反卷积(deconvolutional)，卷积运算的参数和 CNN 的参数一样是在训练 FCN 模型的过程中通过 bp 算法学习得到。反卷积层也是卷积层，不关心 input 大小，滑窗卷积后输出 output。deconv 并不是真正的 deconvolution（卷积的逆变换），最近比较公认的叫法应该是 transposed convolution，deconv 的前向传播就是 conv 的反向传播。

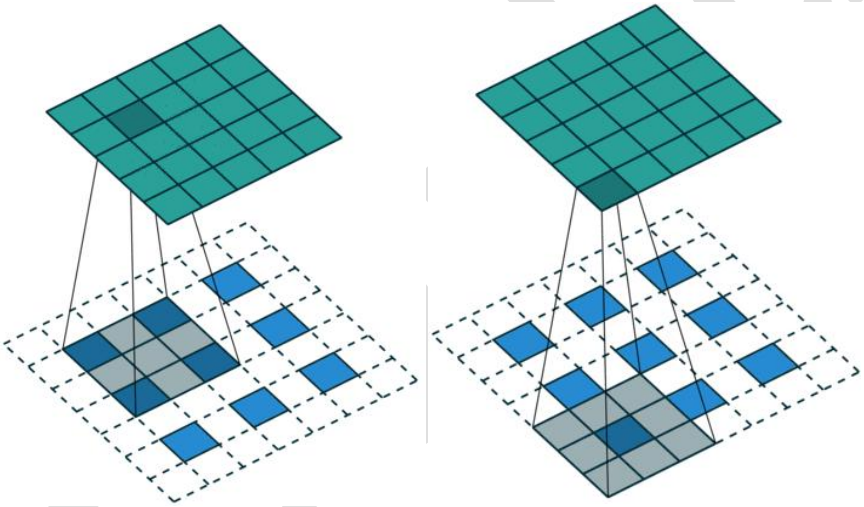
反卷积参数: 利用卷积过程 filter 的转置（实际上就是水平和竖直方向上翻转 filter）作为计算卷积前的特征图。

反卷积的运算如下所示:

蓝色是反卷积层的 input，绿色是反卷积层的 outputFull padding, transposed Full padding, transposed。



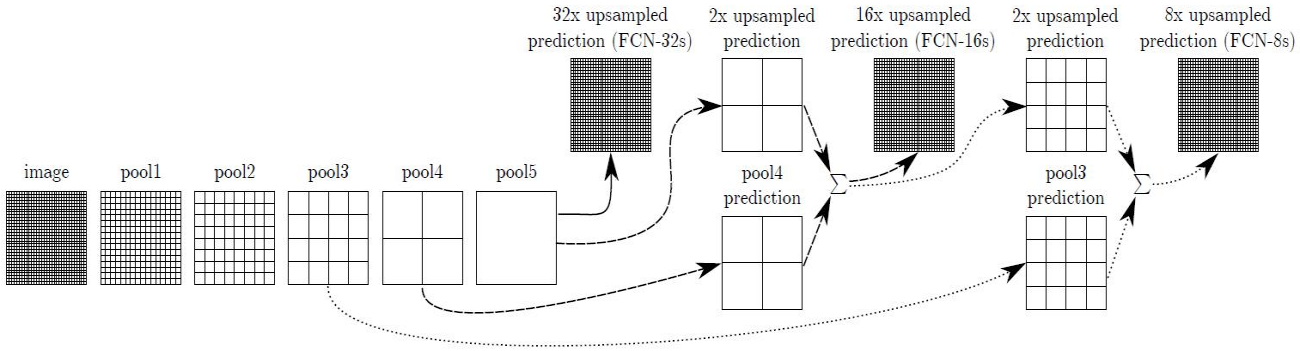
上图中的反卷积，input 是 2×2, output 是 4×4。Zero padding, non-unit strides, transposed。



上图中的反卷积，input feature map 是 3×3，转化后是 5×5, output 是 5×5

8.1.9 跳级(skip)结构

对 CNN 的结果做处理，得到了 dense prediction，而作者在试验中发现，得到的分割结果比较粗糙，所以考虑加入更多前层的细节信息，也就是把倒数第几层的输出和最后的输出做一个 fusion，实际上也就是加和：



实验表明，这样的分割结果更细致更准确。在逐层 fusion 的过程中，做到第三行再往下，结果又会变差，所以作者做到这里就停了。

8.1.10 模型训练

（1）用 AlexNet，VGG16 或者 GoogleNet 训练好的模型做初始化，在这个基础上做 fine-tuning，全部都 fine-tuning，只需在末尾加上 upsampling，参数的学习还是利用 CNN 本身的反向传播原理。

（2）采用 whole image 做训练，不进行 patchwise sampling。实验证明直接用全图已经很 effective and efficient。

（3）对 class score 的卷积层做全零初始化。随机初始化在性能和收敛上没有优势。

举例：

FCN 例子: 输入可为任意尺寸图像彩色图像；输出与输入尺寸相同，深度为：20 类目标+背景=21，模型基于 AlexNet。

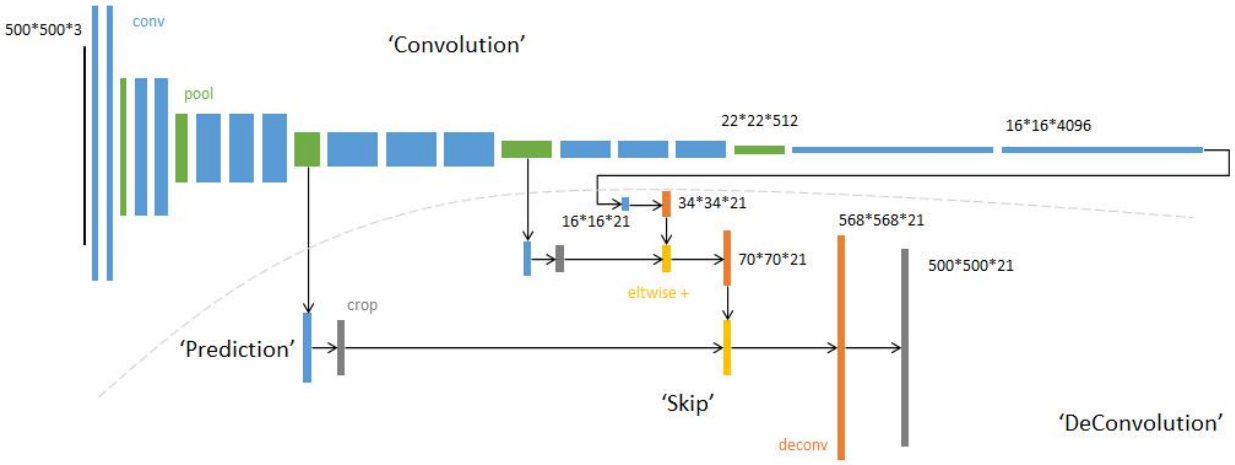
蓝色：卷积层。

绿色：Max Pooling 层。

黄色: 求和运算, 使用逐数据相加，把三个不同深度的预测结果进行融合：较浅的结果更为精细，较深的结果更为鲁棒。

灰色: 裁剪, 在融合之前，使用裁剪层统一两者大小, 最后裁剪成和输入相同尺寸输出。

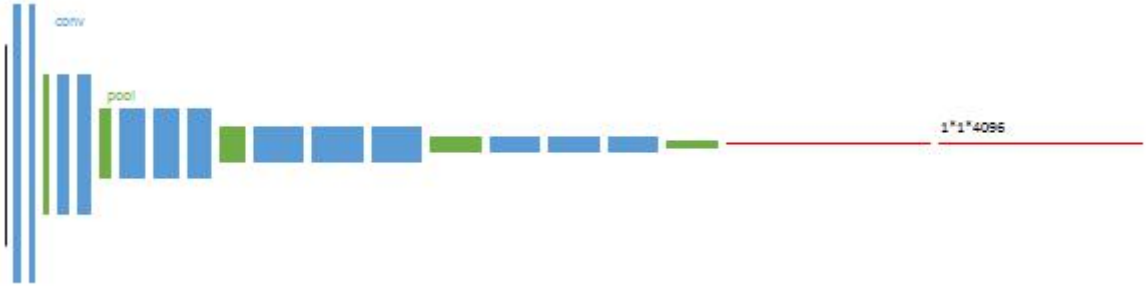
对于不同尺寸的输入图像，各层数据的尺寸（height，width）相应变化，深度（channel）不变。



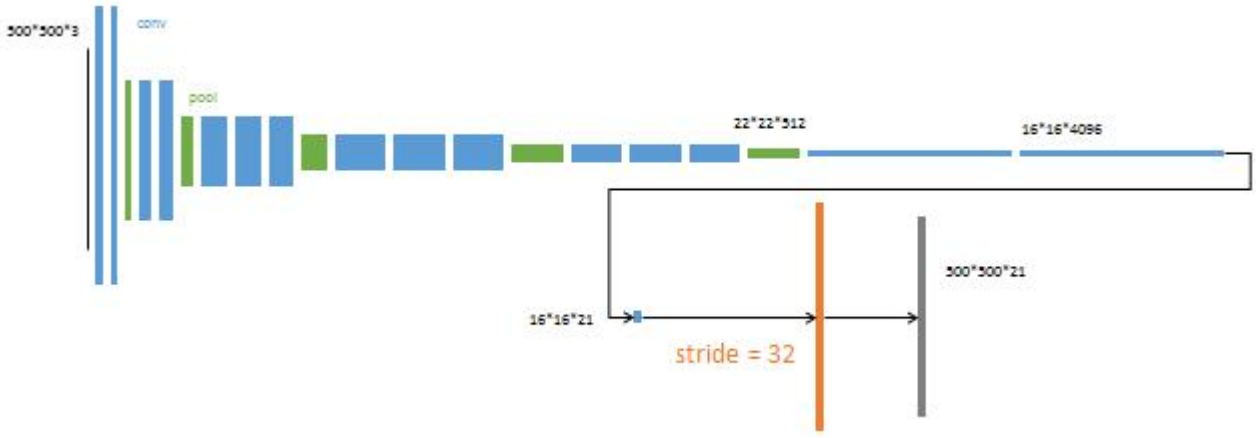
（1）全卷积层部分进行特征提取，提取卷积层（3 个蓝色层）的输出来作为预测 21 个类别的特征

（2）图中虚线内是反卷积层的运算，反卷积层（3 个橙色层）可以把输入数据尺寸放大。和卷积层一样，升采样的具体参数经过训练确定。

- 以经典的 AlexNet 分类网络为初始化。最后两级是全连接（红色），参数弃去不用。

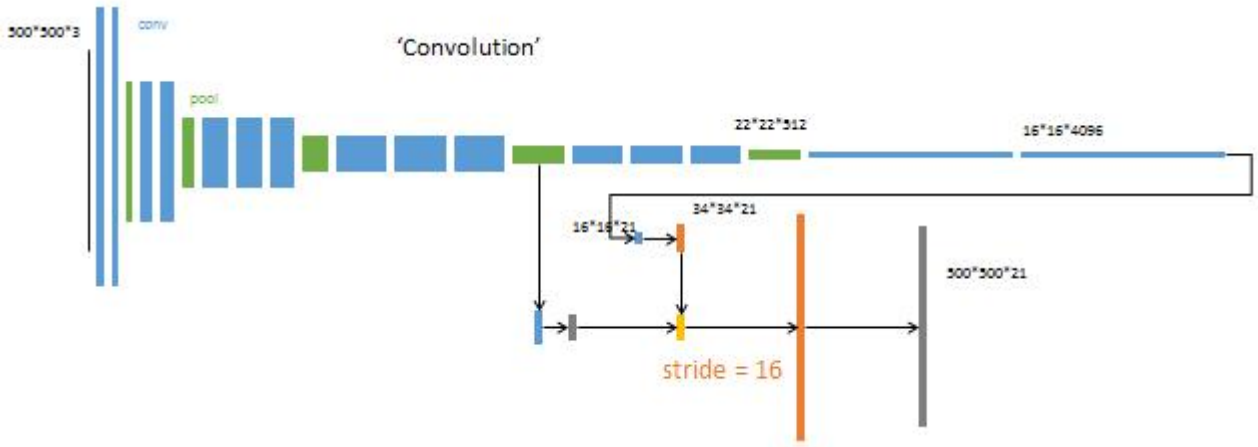


- 从特征小图（）预测分割小图（），之后直接升采样为大图。



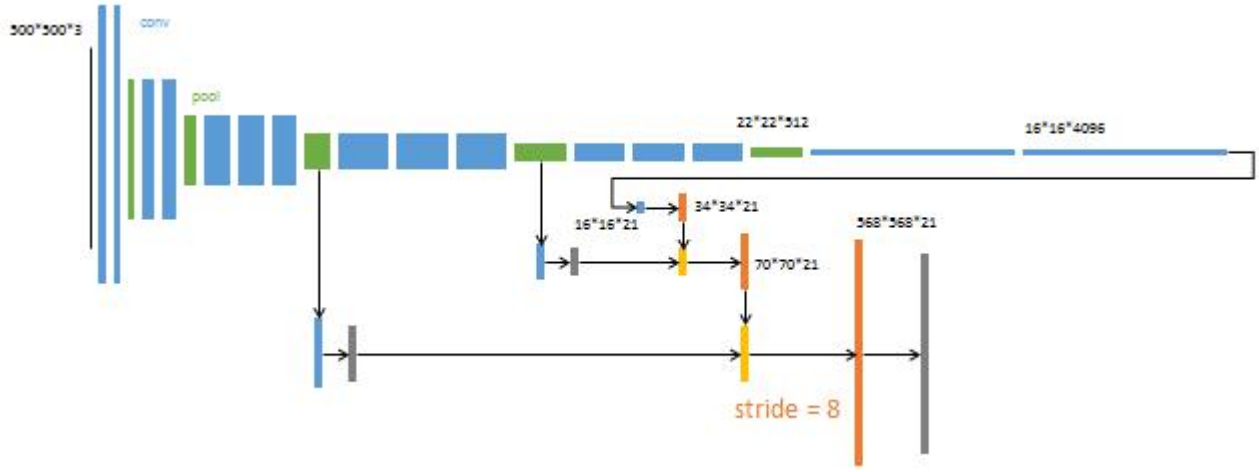
反卷积（橙色）的步长为 32，这个网络称为 FCN-32s

- 升采样分为两次完成（橙色×2），在第二次升采样前，把第 4 个 pooling 层（绿色）的预测结果（蓝色）融合进来。使用跳级结构提升精确性。



第二次反卷积步长为 16，这个网络称为 FCN-16s

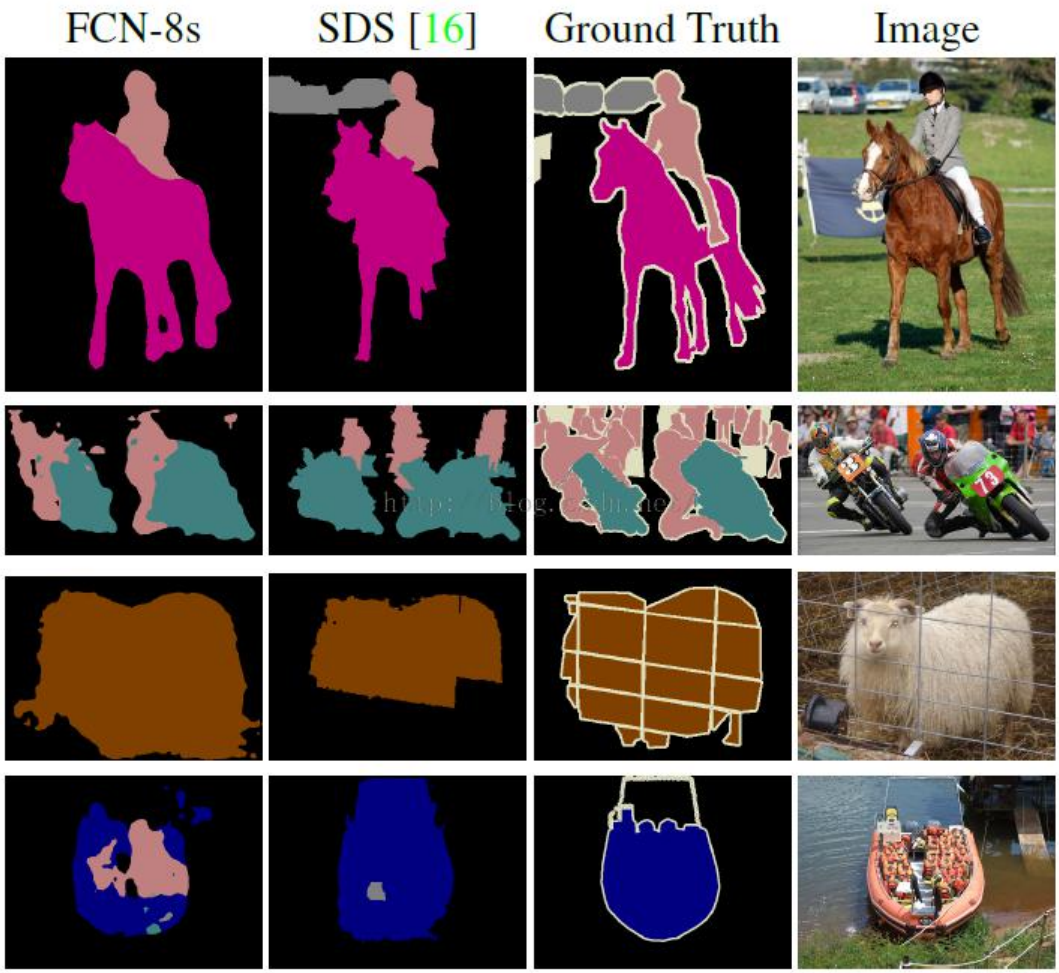
- 升采样分为三次完成（橙色×3），进一步融合了第 3 个 pooling 层的预测结果。



第三次反卷积步长为 8，记为 FCN-8s。

其他参数：

- minibatch：20 张图片。
- learning rate：0.001。
- 初始化：分类网络之外的卷积层参数初始化为 0。
- 反卷积参数初始化为 bilinear 插值。
- 最后一层反卷积固定位 bilinear 插值不做学习。



8.1.11 FCN 缺点

（1）得到的结果还是不够精细。进行 8 倍上采样虽然比 32 倍的效果好了很多，但是上采样的结果还是比较模糊和平滑，对图像中的细节不敏感。

（2）对各个像素进行分类，没有充分考虑像素与像素之间的关系。忽略了在通常的基于像素分类的分割方法中使用的空间规整（spatial regularization）步骤，缺乏空间一致性。

8.2 U-Net

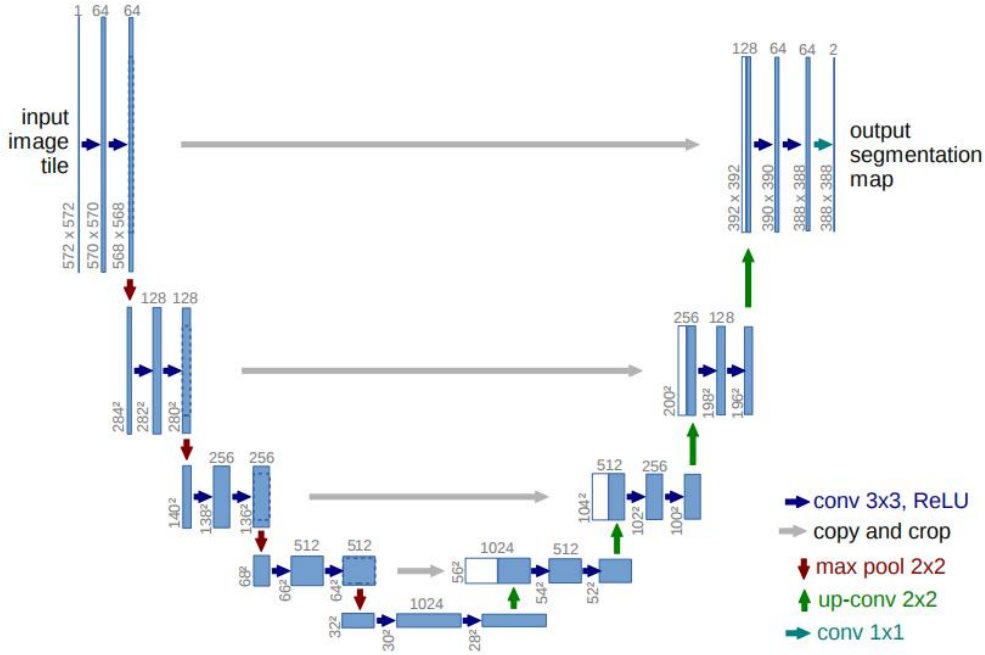
卷积网络的典型利用是在分类任务，输出任务的单个类标签。然而，在许多视觉任务，尤其是生物医学图像处理，目标输出应该包括定位等，每个像素都应该有类标签。另外，大量的训练图片往往超过生物医学图像的任务要求。所以，Ciresan 等训练了一个神经网络，用滑动窗口来预测每个像素的类标签，提供像素的周围区域（patch）作为输入。首先，这个网络可以定位。第二，输入的是 patches，这样训练数据就比图片数据多很多。

明显的，Ciresan 的方法有两个缺点。

第一，它很慢，因为这个网络必须训练每个 patch，并且因为 patch 间的重叠有很多的冗余(冗余会造成什么影响呢？卷积核里面的 W，就是提取特征的权重，两个块如果重叠的部分太多，这个权重会被同一些特征训练两次，造成资源的浪费，减慢训练时间和效率，虽然说会有一些冗余，训练集大了，准确率不就高了吗？可是你这个是相同 的图片啊，重叠的东西都是相同的，举个例子，我用一张相同的图片训练 20 次，按照这个意思也是增大了训练集啊，可是会出现什么结果呢，很显然，会导致过拟合，也就是对你这个图片识别很准，别的图片就不一定了)。

第二，定位准确性和上下文间不可兼得。大的 patches 需要更多的 max-pooling 层这样减小了定位准确性(为什么？因为你是对以这个像素为中心的点进行分类，如果 patch 太大，最后经过全连接层的前一层大小肯定是不变的，如果你 patch 大就需要更多的 pooling 达到这个大小)，因为 Pooling 层会降低分辨率，丢失信息)，而小的 patches 只能看到很小的局部信息，包含的背景信息不够。许多现在的方法使用不同层的特征来同时兼容定位和利用 context。

在这篇论文，我们建立了一个更好全卷积方法。我们定义和扩展了这个方法它使用更少的训练图片但产生更精确的分割。



(1) 使用全卷积神经网络。(全卷积神经网络就是卷积取代了全连接层，全连接层必须固定图像大小而卷积不用，所以这个策略使得，你可以输入任意尺寸的图片，而且输出也是图片，所以这是一个端到端的网络。)

(2) 左边的网络 contracting path：使用卷积和 maxpooling。

(3) 右边的网络 expansive path:使用上采样与左侧 contracting path ,pooling 层的 featuremap 相结合，然后逐层上采样到 392X392 的大小 heatmap。(pooling 层会丢失图像信息和降低图像分辨率且是不可逆的操作，对图像分割任务有一些影响，对图像分类任务的影响不大，为什么

要做上采样？

因为上采样可以补足一些图片的信息，但是信息补充的肯 定不完全，所以还需要与左边的分辨率比较高的图片相连接起来（直接复制过来再裁剪到与上采样图片一样大小），这就相当于在高分辨率和更抽象特征当中做一个折衷，因为随着卷积次数增多，提取的特征也更加有效，更加抽象，上采样的图片是经历多次卷积后的图片，肯定是比较高效和抽象的图片，然后把它与左边不怎么抽象但更高分辨率的特征图片进行连接）。

(4) 最后再经过两次卷积，达到最后的 heatmap，再用一个 1X1 的卷积做分类，这里是分成两类，所以用的是两个神经元做卷积，得到最后的两张 heatmap,例如第一张表示的是第一类的得分（即每个像素点对应第一类都有一个得分），第二张表示第二类的得分 heatmap,然后作为 softmax 函数的输入，算出概率比较大的 softmax 类，选择它作为输入给交叉熵进行反向传播训练。

8.3 SegNet

可训练的图像分割引擎，包含一个 encoder 网络，一个对应的 decoder 网络，衔接像素级分类层，解码网络与 VGG16 的 13 层卷积层相同。解码网络是将低分辨率的编码特征图映射到全分辨率的特征图。解码网络使用最大池化层的池化索引进行非线性上采样，上采样过程就不需要学习。上采样得到的稀疏图与可训练的滤波器卷积得到致密的特征图。

使用池化层索引进行上采样的优势：

1）提升边缘刻画度；

2）减少训练的参数；

3）这种上采样模式可以包含到任何编码-解码网络中。

SegNet 网络的结构如下图所示：

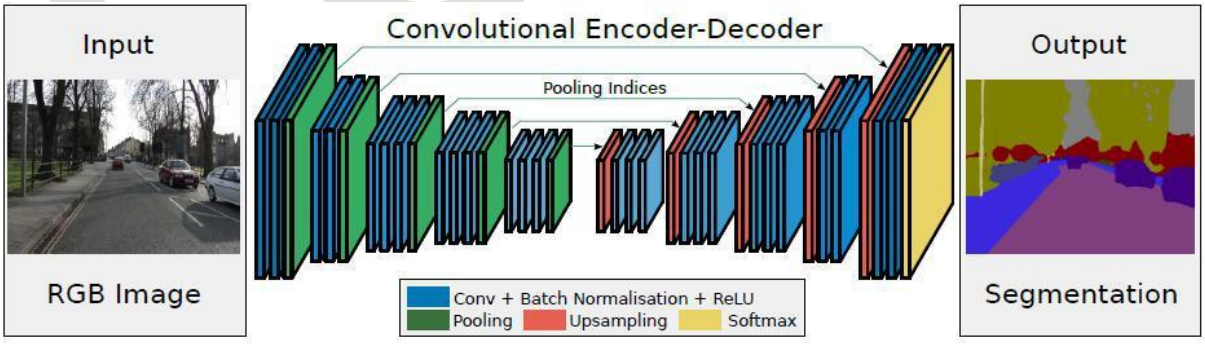


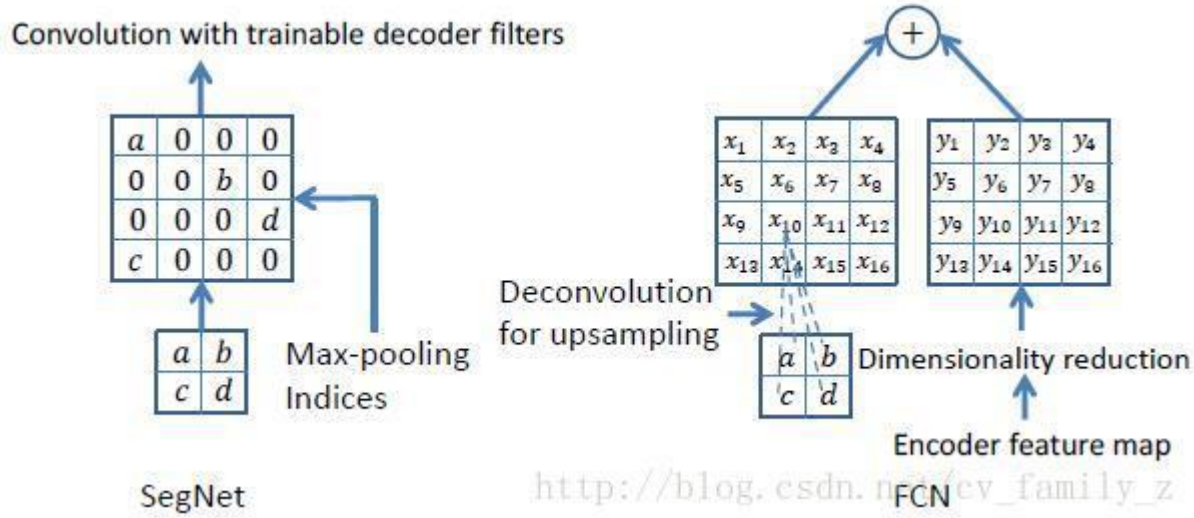
Fig. 2. An illustration of the SegNet architecture. There are no fully connected layers and hence it is only convolutional. A decoder upsamples its input using the transferred pool indices from its encoder to produce a sparse feature map(s). It then performs convolution with a trainable filter bank to densify the feature map. The final decoder output feature maps are fed to a soft-max classifier for pixel-wise classification.

SegNet 网络结构如图 1 所示，Input 为输入图片，Output 为输出分割的图像，不同颜色代表不同的分类。语义分割的重要性就在于不仅告诉你图片中某个东西是什么，而且告知你他在图片的位置。我们可以看到是一个对称网络，由中间绿色 pooling 层与红色 upsampling 层作

为分割，左边是卷积提取高维特征，并通过 pooling 使图片变小，SegNet 作者称为 Encoder，右边是反卷积（在这里反卷积与卷积没有区别）与 upsampling，通过反卷积使得图像分类后特征得以重现，upsampling 使图像变大，SegNet 作者称为 Decoder，最后通过 Softmax，输出不同分类的最大值。这就是大致的 SegNet 过程，下面对这个过程里面使用到的方法进行介绍。

编码网络与滤波器族卷积得到特征图，进行 BN，ReLU，最大池化。最大池化是为了获得空间小位移的平移不变。最大池化和下采样损失了边缘细节，因此，在编码过程中保存边缘信息很重要。考虑到内存原因，只保存最大池化索引，如最大特征值的位置。

SegNet 解码技术如下图所示：



解码网络使用保存的最大池化索引上采样，得到稀疏的特征图，将特征图与可训练的解码滤波器族卷积得到致密的特征图。之后进行 BN。高维的特征图输入 soft-max 层，对每个像素进行分类，得到每个像素属于 K 类的概率。 图 3 中右边是 FCN 的解码技术，FCN 对编码的特征图进行降维，降维后输入到解码网络，解码网络中，上采样使用反卷积实现，上采样的特征图与降维的编码图进行 element-wise add 得到最终的解码特征图。FCN 解码模型需要存储编码特征图，在嵌入式设备中内存紧张。

SegNet 的 Encoder 过程中，卷积的作用是提取特征，SegNet 使用的卷积为 same 卷积（详见卷积神经网络 CNN（1）），即卷积后不改变图片大小；在 Decoder 过程中，同样使用 same 卷积，不过卷积的作用是为 upsampling 变大的图像丰富信息，使得在 Pooling 过程丢失的信息可以通过学习在 Decoder 得到。SegNet 中的卷积与传统 CNN 的卷积并没有区别。

8.4 空洞卷积(Dilated Convolutions)

在图像分割领域，图像输入到 CNN（典型的网络比如 FCN[3]）中，FCN 先像传统的 CNN 那样对图像做卷积再 pooling，降低图像尺寸的同时增大感受野，但是由于图像分割预测是 pixel-wise 的输出，所以要将 pooling 后较小的图像尺寸 upsampling 到原始的图像尺寸进行预测（upsampling 一般采用 deconv 反卷积操作，deconv 可参见知乎答案如何理解深度学习中的 deconvolution networks？），之前的 pooling 操作使得每个 pixel 预测都能看到较大感受野信息。因此图像分割 FCN 中有两个关键，一个是 pooling 减小图像尺寸增大感受野，另一个是 upsampling 扩大图像尺寸。在先减小再增大尺寸的过程中，肯定有一些信息损失掉了，那么能

不能设计一种新的操作，不通过 pooling 也能有较大的感受野看到更多的信息呢？答案就是 dilated conv。

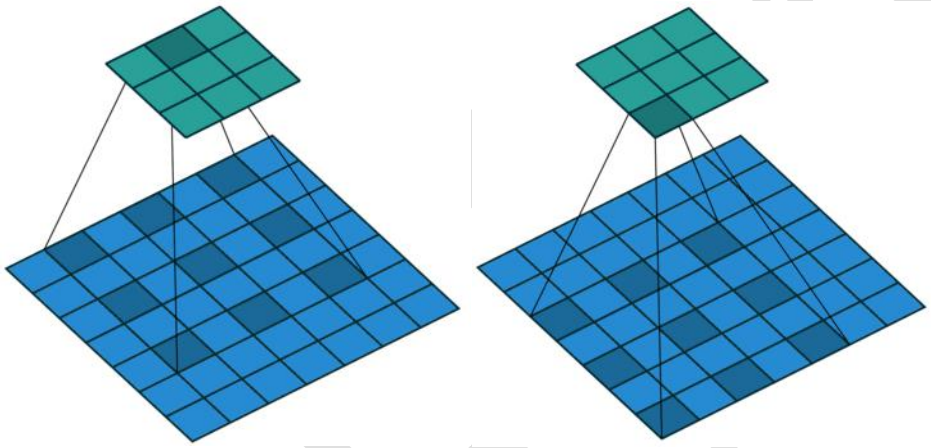
以前的 CNN 主要问题总结：

（1）Up-sampling / pooling layer

（2）内部数据结构丢失；空间层级化信息丢失。

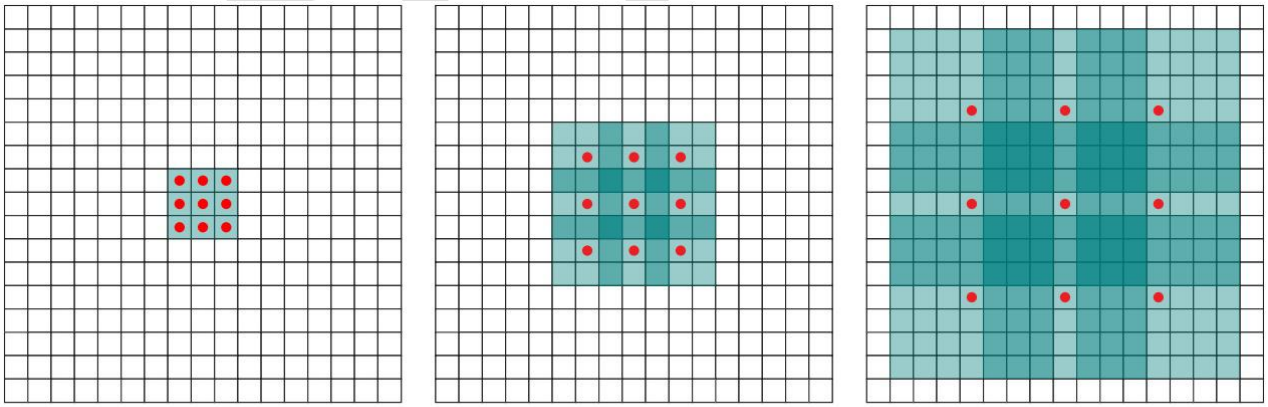
（3）小物体信息无法重建 (假设有四个 pooling layer 则 任何小于 2^4 = 16 pixel 的物体信息将理论上无法重建。)

举例如下：



Dilated Convolution with a 3 x 3 kernel and dilation rate 2

下面看一下 dilated conv 原始论文[4]中的示意图



(a) (b) (c)

(a) 图对应 3x3 的 1-dilated conv，和普通的卷积操作一样，(b)图对应 3x3 的 2-dilated conv，实际的卷积 kernel size 还是 3x3，但是空洞为 1，也就是对于一个 7x7 的图像 patch，只有 9 个红色的点和 3x3 的 kernel 发生卷积操作，其余的点略过。也可以理解为 kernel 的 size 为 7x7，但是只有图中的 9 个点的权重不为 0，其余都为 0。 可以看到虽然 kernel size 只有 3x3，但是这个卷积的感受野已经增大到了 7x7（如果考虑到这个 2-dilated conv 的前一层是一个 1-dilated conv 的话，那么每个红点就是 1-dilated 的卷积输出，所以感受野为 3x3，所以 1-dilated 和 2-dilated 合起来就能达到 7x7 的 conv），(c)图是 4-dilated conv 操作，同理跟在两个 1-dilated 和 2-dilated

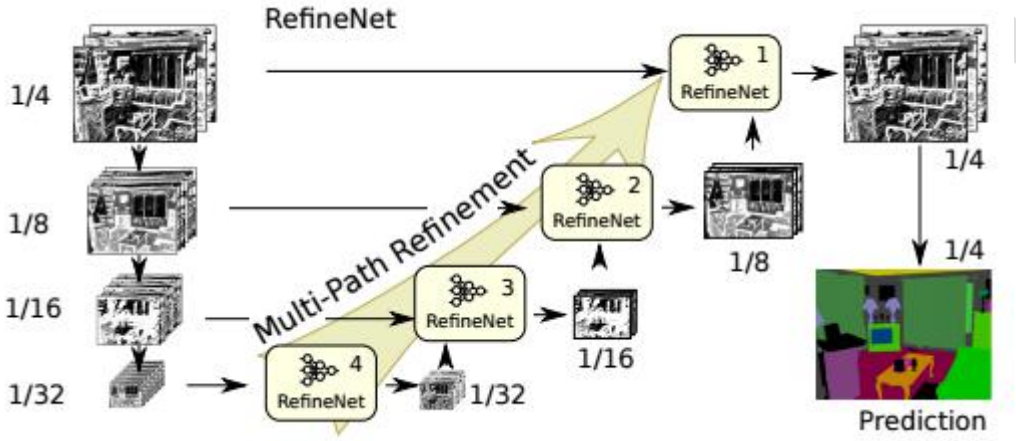
conv 的后面，能达到 15x15 的感受野。对比传统的 conv 操作，3 层 3x3 的卷积加起来，stride 为 1 的话，只能达到(kernel-1)*layer+1=7 的感受野，也就是和层数 layer 成线性关系，而 dilated conv 的感受野是指数级的增长。

dilated 的好处是不做 pooling 损失信息的情况下，加大了感受野，让每个卷积输出都包含较大范围的信息。在图像需要全局信息或者语音文本需要较长的 sequence 信息依赖的问题中，都能很好的应用 dilated conv，比如图像分割、语音合成 WaveNet、机器翻译 ByteNet 中。

8.4 RefineNet

网络结构：

RefineNet block 的作用就是把不同 resolution level 的 feature map 进行融合。网络结构如下：

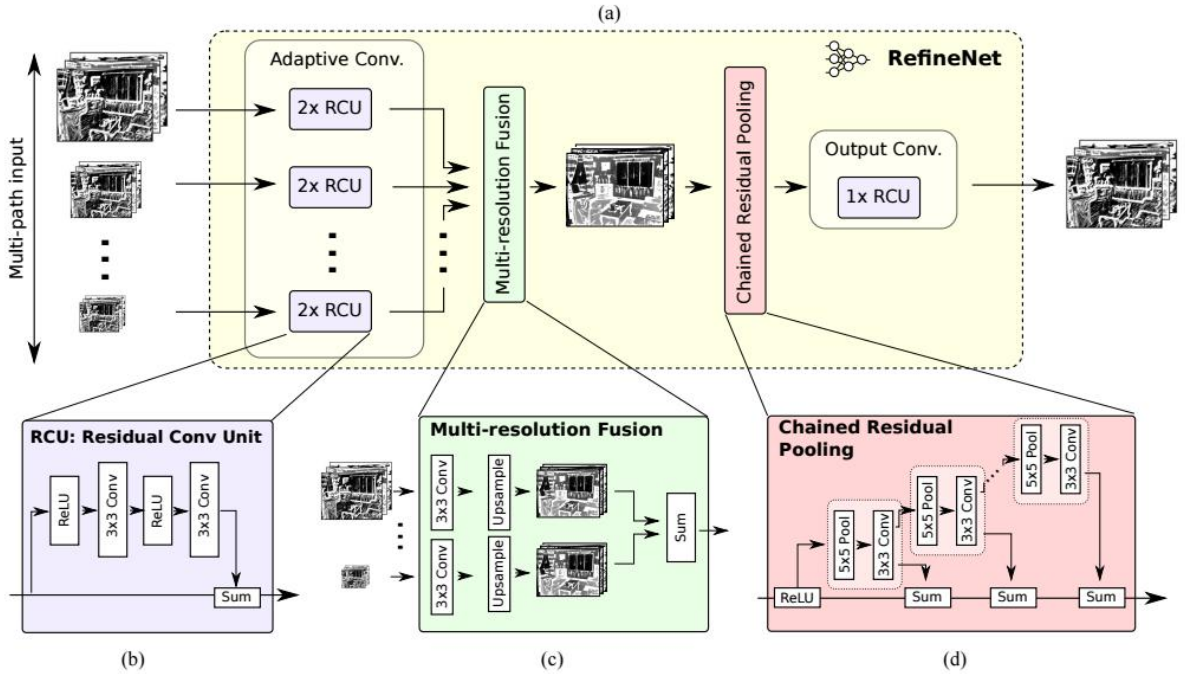


最左边一栏就是 FCN 的 encoder 部分(文中是用的 ResNet)，先把 pretrained ResNet 按 feature map 的分辨率分成四个 ResNet blocks，然后向右把四个 blocks 分别作为 4 个 path 通过 RefineNet block 进行融合 refine，最后得到一个 refined feature map(接 softmax 再双线性插值输出)。

注意除了 RefineNet-4，所有的 RefineNet block 都是二输入的，用于融合不同 level 做 refine，而单输入的 RefineNet-4 可以看作是先对 ResNet 的一个 task adaptation。

RefineNet Block

接下来仔细看一下 RefineNet block，可以看到主要组成部分是 Residual convolution unit, Multi-resolution fusion, Chained residual pooling, Output convolutions. 切记这个 block 作用是融合多个 level 的 feature map 输出单个 level 的 feature map，但具体的实现应该是和输入个数、shape 无关的。



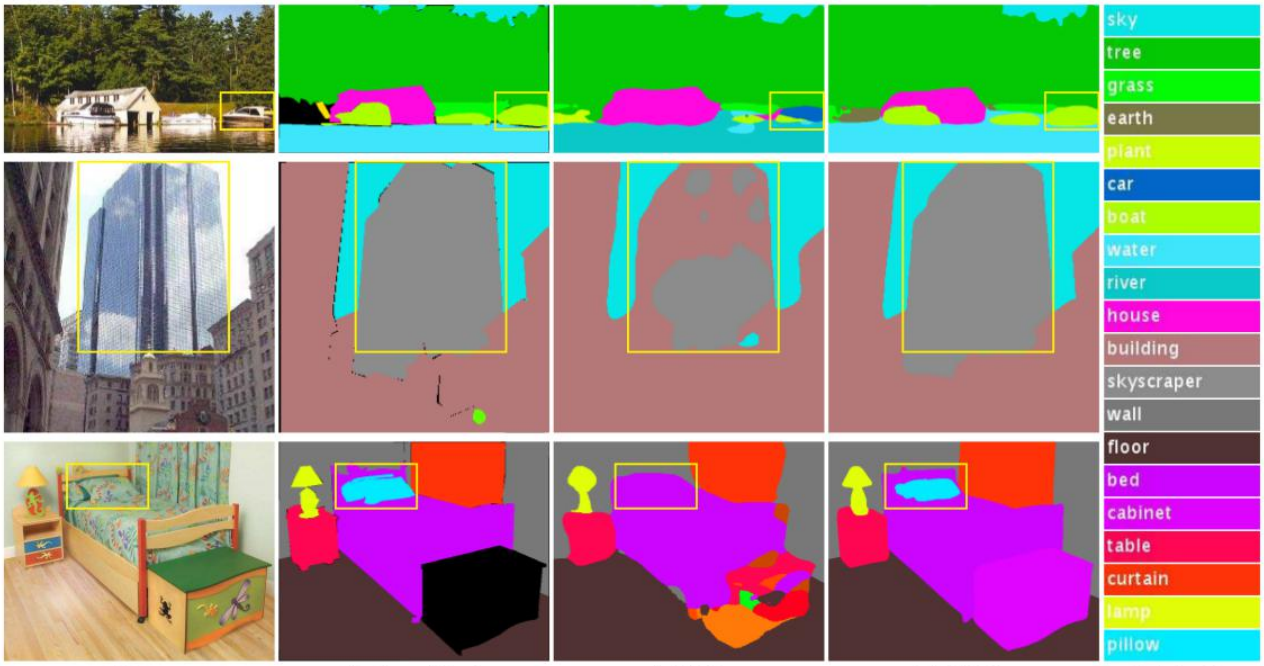
- Residual convolution unit 就是普通的去除了 BN 的 residual unit；
- Multi-resolution fusion 是先对多输入的 feature map 都用一个卷积层进行 adaptation(都化到最小的 feature map 的 shape)，再上采样再做 element-wise 的相加。注意如果是像 RefineNet-4 那样的单输入 block 这一部分就直接 pass 了；
- Chained residual pooling 没太看懂怎么残差相加的(其中的 ReLU 非常重要)，之后再修改；
- Output convolutions 就是输出前再加一个 RCU。

8.5 PSPNet

场景解析对于无限制的开放词汇和不同场景来说是具有挑战性的.本文使用文中的 pyramid pooling module 实现基于不同区域的上下文集成，提出了 PSPNet，实现利用上下文信息的能力进行场景解析.

作者认为，FCN 存在的主要问题是没有采取合适的策略来用全局的信息，本文的做法就是借鉴 SPPNet 来设计了 PSPNet 解决这个问题.

很多 State-of-the-art 的场景解析框架都是基于 FCN 的.基于 CNN 的方法能够增强动态物体的理解，但是在无限制词汇和不同场景中仍然面临挑战.举个例子，如下图.



(a) Image (b) Ground Truth (c) FCN (d) PSPNet (e) ColorMap

FCN 认为右侧框中是汽车，但是实际上是船，如果参考上下文的先验知识，就会发现左边是一个船屋，进而推断是框中是船.FCN 存在的主要问题就是不能利用好全局的场景线索.

对于尤其复杂的场景理解，之前都是采用空间金字塔池化来做的，和之前方法不同（为什么不同，需要参考一下经典的金字塔算法），本文提出了 pyramid scene parsing network(PSPNet).

本文的主要贡献如下:

(1) 提出了 PSPNet 在基于 FCN 的框架中集成困难的上下文特征

(2) 通过基于深度监督误差开发了针对 ResNet 的高效优化策略

(3) 构建了一个用于 state-of-the-art 的场景解析和语义分割的实践系统（具体是什么？）

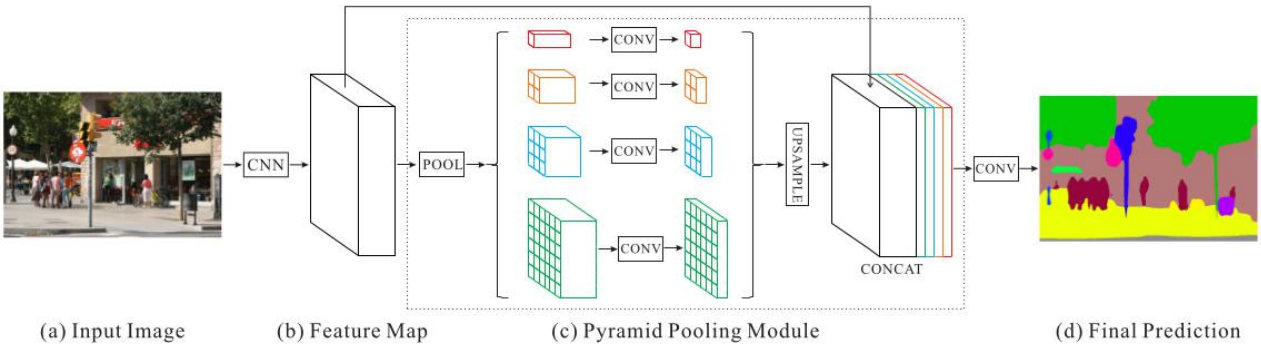
通过观察 FCN 的结果，发现了如下问题：

(1) 关系不匹配（Mismatched Relationship）

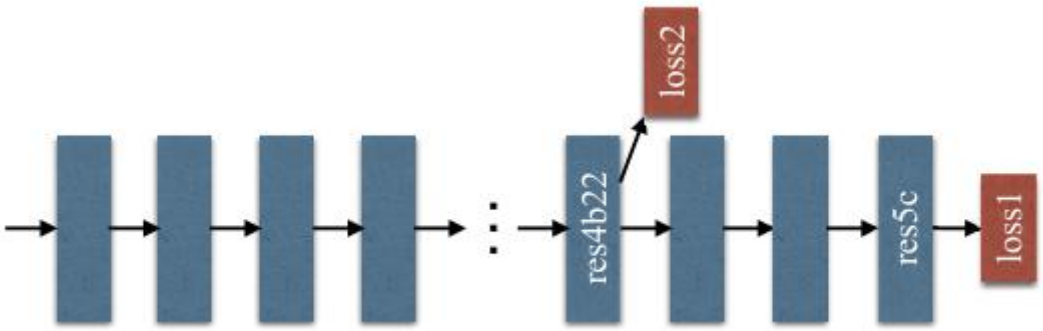
(2) 易混淆的类别（Confusion Categories）

(3) 不显眼的类别（Inconspicuous Classes）

总结以上结果发现，以上问题部分或者全部与上下文关系和全局信息有关系，因此本文提出了 PSPNet.框架如下:



并且加入额外的深度监督 Loss



8.6 DeepLab 系列

8.6.1 DeepLabv1

DeepLab 是结合了深度卷积神经网络（DCNNs）和概率图模型（DenseCRFs）的方法。

在实验中发现 DCNNs 做语义分割时精准度不够的问题，根本原因是 DCNNs 的高级特征的平移不变性，即高层次特征映射，根源于重复的池化和下采样。

针对信号下采样或池化降低分辨率，DeepLab 是采用的 atrous（带孔）算法扩展感受野，获取更多的上下文信息。

分类器获取以对象中心的决策是需要空间变换的不变性，这天然地限制了 DCNN 的定位精度，DeepLab 采用完全连接的条件随机场（CRF）提高模型捕获细节的能力。

除空洞卷积和 CRFs 之外，论文使用的 tricks 还有 Multi-Scale features。其实就是 U-Net 和 FPN 的思想，在输入图像和前四个最大池化层的输出上附加了两层的 MLP，第一层是 128 个 3×3 卷积，第二层是 128 个 1×1 卷积。最终输出的特征与主干网的最后一层特征图融合，特征图增加 5×128=640 个通道。

实验表示多尺度有助于提升预测结果，但是效果不如 CRF 明显。

论文模型基于 VGG16，在 Titan GPU 上运行速度达到了 8FPS，全连接 CRF 平均推断需要 0.5s ，在 PASCAL VOC-2012 达到 71.6% IOU accuracy。

8.6.2 DeepLabv2

DeepLabv2 是相对于 DeepLabv1 基础上的优化。DeepLabv1 在三个方向努力解决，但是问题依然存在：特征分辨率的降低、物体存在多尺度，DCNN 的平移不变性。

因 DCNN 连续池化和下采样造成分辨率降低，DeepLabv2 在最后几个最大池化层中去除下采样，取而代之的是使用空洞卷积，以更高的采样密度计算特征映射。

物体存在多尺度的问题，DeepLabv1 中是用多个 MLP 结合多尺度特征解决，虽然可以提供系统的性能，但是增加特征计算量和存储空间。

论文受到 Spatial Pyramid Pooling (SPP) 的启发，提出了一个类似的结构，在给定的输入上以不同采样率的空洞卷积并行采样，相当于以多个比例捕捉图像的上下文，称为 ASPP (atrous spatial pyramid pooling) 模块。

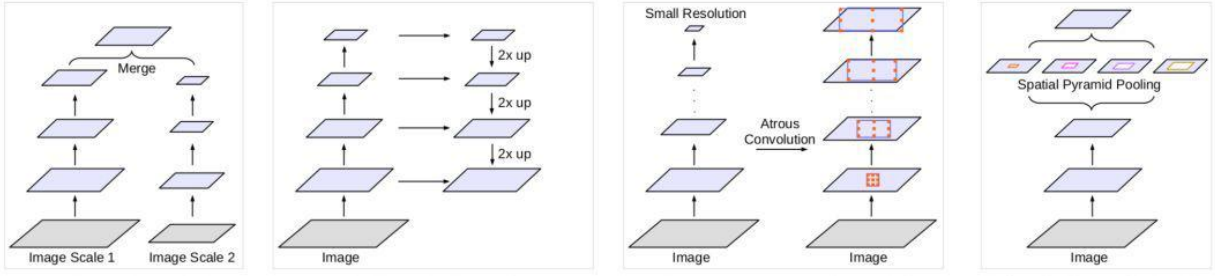
DCNN 的分类不变形影响空间精度。DeepLabv2 是采样全连接的 CRF 在增强模型捕捉细节的能力。

论文模型基于 ResNet，在 NVidia Titan X GPU 上运行速度达到了 8FPS，全连接 CRF 平均推断需要 0.5s ，在耗时方面和 DeepLabv1 无差异，但在 PASCAL VOC-2012 达到 79.7 mIOU。

8.6.3 DeepLabv3

好的论文不止说明怎么做，还告诉为什么。DeepLab 延续到 DeepLabv3 系列，依然是在空洞卷积做文章，但是探讨不同结构的方向。

DeepLabv3 论文比较了多种捕获多尺度信息的方式：



(a) Image Pyramid (b) Encoder-Decoder (c) Deeper w. Atrous Convolution (d) Spatial Pyramid Pooling

Figure 2. Alternative architectures to capture multi-scale context.

1. **Image Pyramid**：将输入图片放缩成不同比例，分别应用在 DCNN 上，将预测结果融合得到最终输出。

2. **Encoder-Decoder**：利用 Encoder 阶段的多尺度特征，运用到 Decoder 阶段上恢复空

间分辨率，代表工作有 FCN、SegNet、PSPNet 等工。

3. **Deeper w. Atrous Convolution**：在原始模型的顶端增加额外的模块，例如 DenseCRF，捕捉像素间长距离信息。

4. **Spatial Pyramid Pooling**：空间金字塔池化具有不同采样率和多种视野的卷积核，能够以多尺度捕捉对象。

DeepLabv1-v2 都是使用带孔卷积提取密集特征来进行语义分割。但是为了解决分割对象的多尺度问题，DeepLabv3 设计采用多比例的带孔卷积级联或并行来捕获多尺度背景。

此外，DeepLabv3 将修改之前提出的带孔空间金字塔池化模块，该模块用于探索多尺度卷积特征，将全局背景基于图像层次进行编码获得特征，取得 state-of-art 性能，在 PASCAL VOC-2012 达到 86.9 mIOU。

8.6.4 DeepLabv3+

语义分割关注的问题:

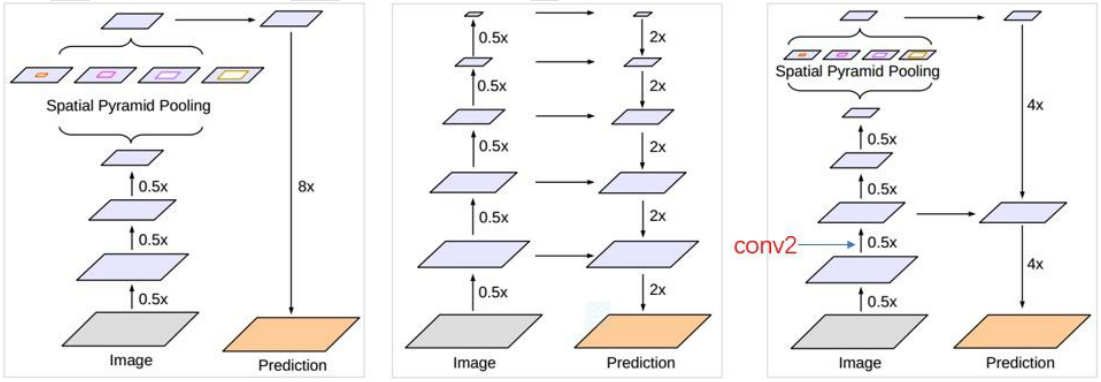
1、 实例对象多尺度问题。

2、 因为深度网络存在 stride=2 的层，会导致 feature 分辨率下降，从而导致预测精度降低，而造成的边界信息丢失问题。

deeplab V3 新设计的 aspp 结构解决了问题 1，deeplab v3+主要目的在于解决问题 2。

问题 2 可以使用空洞卷积替代更多的 pooling 层来获取分辨率更高的 feature。但是 feature 分辨率更高会极大增加运算量。 以 deeplab v3 使用的 resnet101 为例， stride=16 将造成后面 9 层 feature 变大，后面 9 层的计算量变为原来的 2*2=4 倍大。stride=8 则更为恐怖，后面 78 层的计算量都会变大很多，

解决方案：1、编解码器结构。2 Modified Aligned Xception

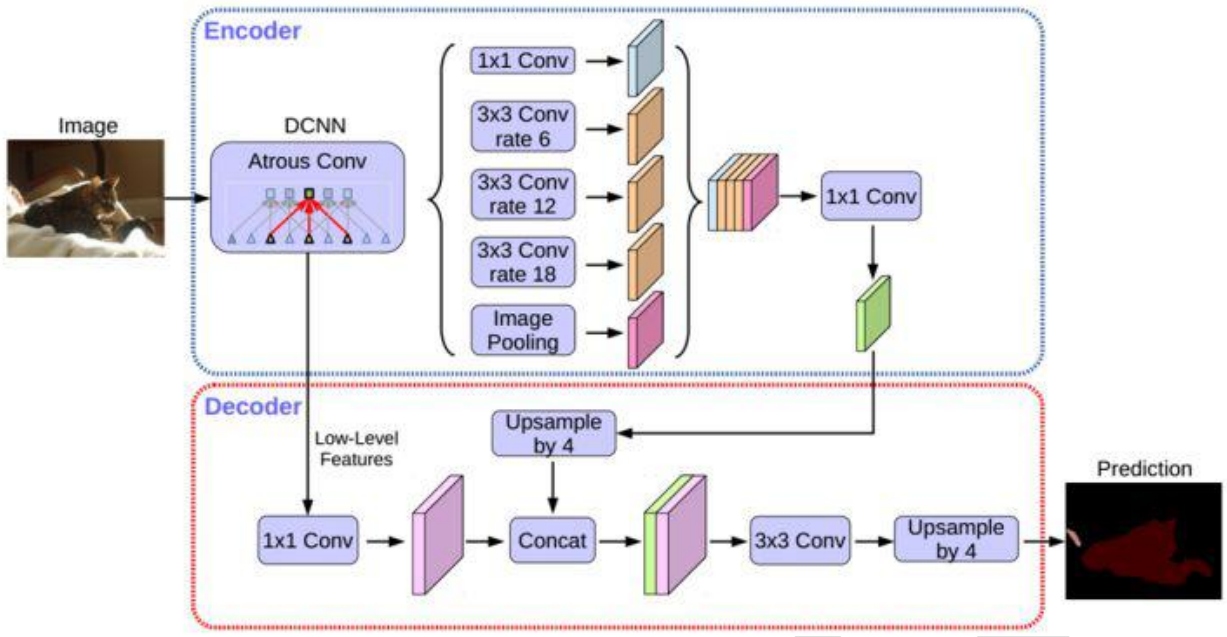


(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv.

在 deeplabv3 基础上 加入解码器。 A 是 aspp 结构。 A 的 8*的上采样可以看做是一个 naïve 的解码器。 B 是编解码结构。集合了高层和底层的 feature。 C 就是本文采取的结构。Conv2（图中红色）的提取到结果和最后提取出的 feature 上采样 4 后融合。

方法：

（1）Encoder-Decoder with Atrous Convolution



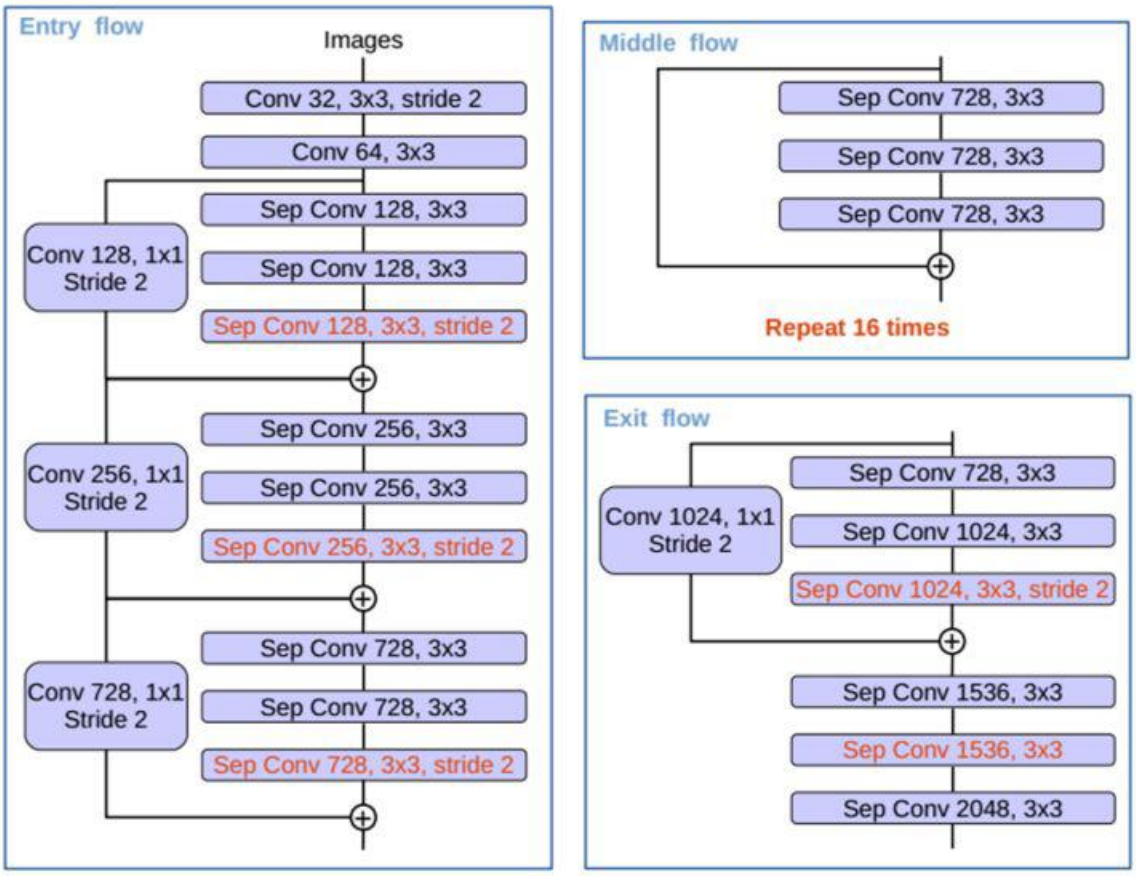
编码器采用 deeplabv3。

解码器部分：先从低层级选一个 feature，将低层级的 feature 用 1*1 的卷积进行通道压缩（原本为 256 通道，或者 512 通道），目的在于减少低层级的比重。作者认为编码器得到的 feature 具有更丰富的信息，所以编码器的 feature 应该有更高的比重。 这样做有利于训练。

再将编码器的输出上采样，使其分辨率与低层级 feature 一致。 举个例子，如果采用 resnet conv2 输出的 feature，则这里要*4 上采样。 将两种 feature 连接后，再进行一次 3*3 的卷积（细化作用），然后再次上采样就得到了像素级的预测。 后面的实验结果表明这种结构在 stride=16 时既有很高的精度速度又很快。stride=8 相对来说只获得了一点点精度的提升，但增加了很多的计算量。

（2）**Modified Aligned Xception**

Xception 主要采用了 deepwish seperable convolution 来替换原来的卷积层。简单的说就是这种结构能在更少参数更少计算量的情况下学到同样的信息。这边则是考虑将原来的 resnet-101 骨架网换成 xception。

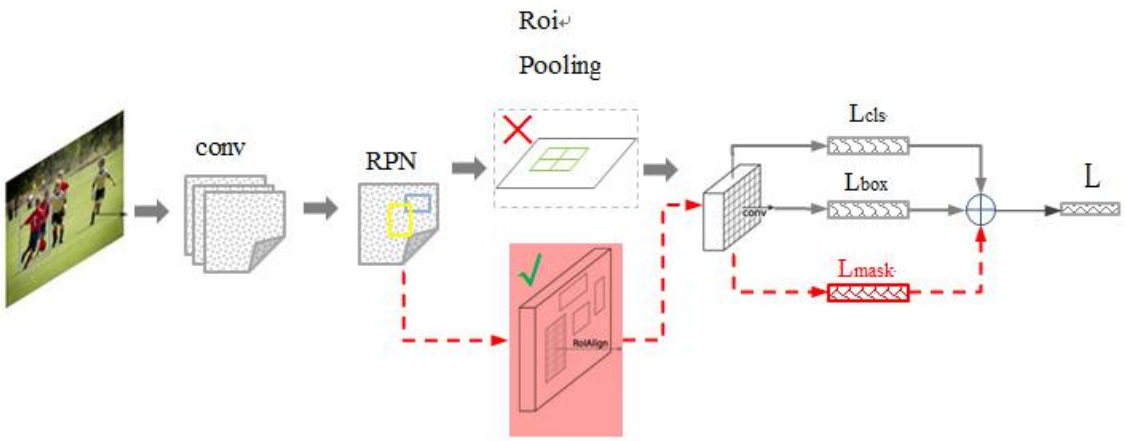


红色部分为修改

- 更多层：重复 8 次改为 16 次（基于 MSRA 目标检测的工作）。
- 将原来简单的 pool 层改成了 stride 为 2 的 deepwish seperable convolution。
- 额外的 RELU 层和归一化操作添加在每个 3×3 depthwise convolution 之后（原来只在 1*1 卷积之后）

8.7 Mask-R-CNN

8.7.1 Mask-RCNN 的网络结构示意图



其中 黑色部分为原来的 Faster-RCNN，红色部分为在 Faster 网络上的修改：

1）将 Roi Pooling 层替换成了 RoiAlign；

2）添加并列的 FCN 层（mask 层）；

先来概述一下 Mask-RCNN 的几个特点（来自于 Paper 的 Abstract）：

1）在边框识别的基础上添加分支网络，用于 语义 Mask 识别；

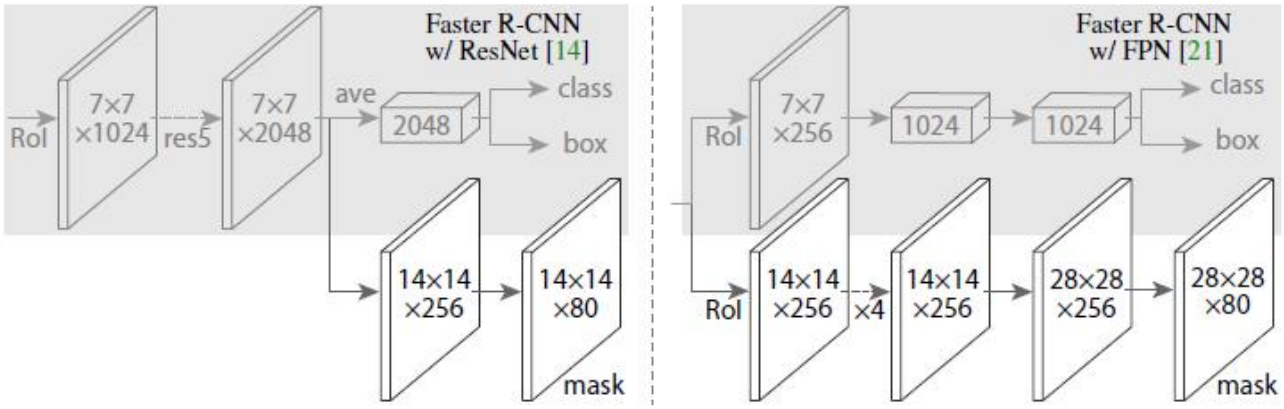
2）训练简单，相对于 Faster 仅增加一个小的 Overhead，可以跑到 5FPS；

3）可以方便的扩展到其他任务，比如人的姿态估计 等；

4）不借助 Trick，在每个任务上，效果优于目前所有的 single-model entries；包括 COCO 2016 的 Winners。

8.7.2 RCNN 行人检测框架

来看下后面两种 RCNN 方法与 Mask 结合的示意图:



图中灰色部分是 原来的 RCNN 结合 ResNet or FPN 的网络，下面黑色部分为新添加的并联 Mask 层，这个图本身与上面的图也没有什么区别，旨在说明作者所提出的 Mask RCNN 方法的泛化适应能力 - 可以和多种 RCNN 框架结合，表现都不错。

8.7.3 Mask-RCNN 技术要点

- 技术要点 1 - 强化的基础网络

  通过 ResNeXt-101+FPN 用作特征提取网络，达到 state-of-the-art 的效果。

- 技术要点 2 - ROIAlign

  采用 ROIAlign 替代 RoiPooling（改进池化操作）。引入了一个插值过程，先通过双线性插值到 14*14，再 pooling 到 7*7，很大程度上解决了仅通过 Pooling 直接采样带来的 Misalignment 对齐问题。

  PS： 虽然 Misalignment 在分类问题上影响并不大，但在 Pixel 级别的 Mask 上会存在较大误差。

  后面我们把结果对比贴出来（Table2 c & d），能够看到 ROIAlign 带来较大的改进，可以看到，Stride 越大改进越明显。

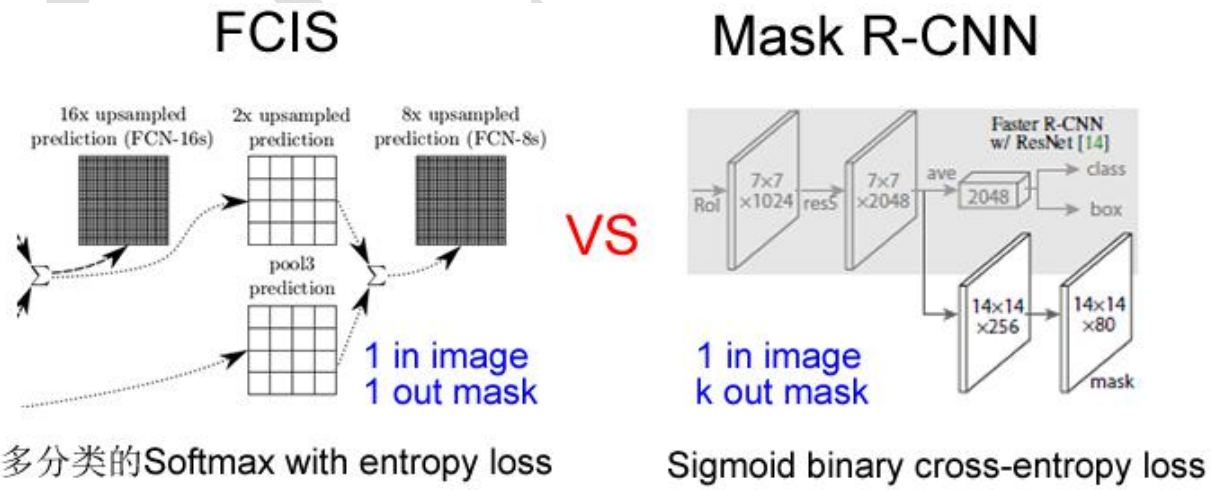
- 技术要点 3 - Loss Function

  每个 ROIAlign 对应 K * m^2 维度的输出。K 对应类别个数，即输出 K 个 mask，m 对应池化分辨率（7*7）。Loss 函数定义：

  $$Lmask(Cls\_k) = Sigmoid\ (Cls\_k),$$

  平均二值交叉熵 （average binary cross-entropy）Loss，通过逐像素的 Sigmoid 计算得到。

  Why K 个 mask？通过对每个 Class 对应一个 Mask 可以有效避免类间竞争（其他 Class 不贡献 Loss ）。



通过结果对比来看（Table2 b），也就是作者所说的 Decouple 解耦，要比多分类 的 Softmax 效果好很多。

另外，作者给出了很多实验分割效果，就不都列了，只贴一张 和 FCIS 的对比图（FCIS 出

现了 Overlap 的问题）

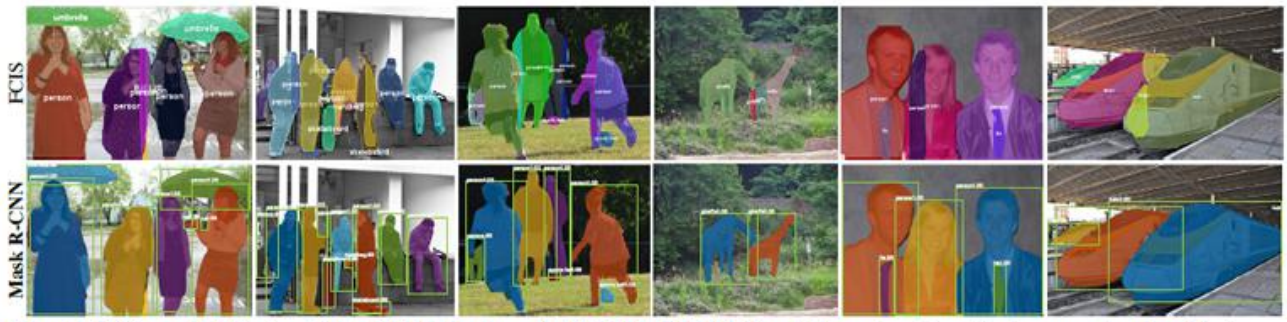


Figure 5. FCIS+++ [20] (top) *vs*. Mask R-CNN (bottom, ResNet-101-FPN). FCIS exhibits systematic artifacts on overlapping objects.

8.8 CNN 在基于弱监督学习的图像分割中的应用

https://zhuanlan.zhihu.com/p/23811946

最近基于深度学习的图像分割技术一般依赖于卷积神经网络 CNN 的训练，训练过程中需要非常大量的标记图像，即一般要求训练图像中都要有精确的分割结果。

对于图像分割而言，要得到大量的完整标记过的图像非常困难，比如在 ImageNet 数据集上，有 1400 万张图有类别标记，有 50 万张图给出了 bounding box,但是只有 4460 张图像有像素级别的分割结果。对训练图像中的每个像素做标记非常耗时，特别是对医学图像而言，完成对一个三维的 CT 或者 MRI 图像中各组织的标记过程需要数小时。

如果学习算法能通过对一些初略标记过的数据集的学习就能完成好的分割结果，那么对训练数据的标记过程就很简单，这可以大大降低花在训练数据标记上的时间。这些初略标记可以是：

1， 只给出一张图像里面包含哪些物体，

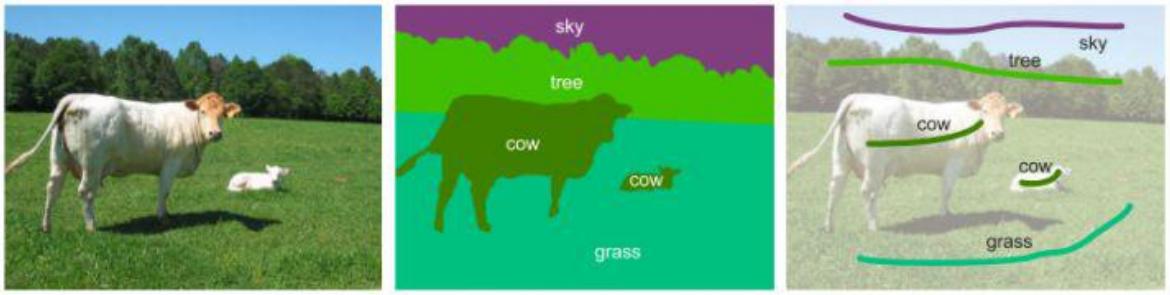
2， 给出某个物体的边界框，

3， 对图像中的物体区域做部分像素的标记，例如画一些线条、涂鸦等（scribbles)。

8.8.1 Scribble 标记

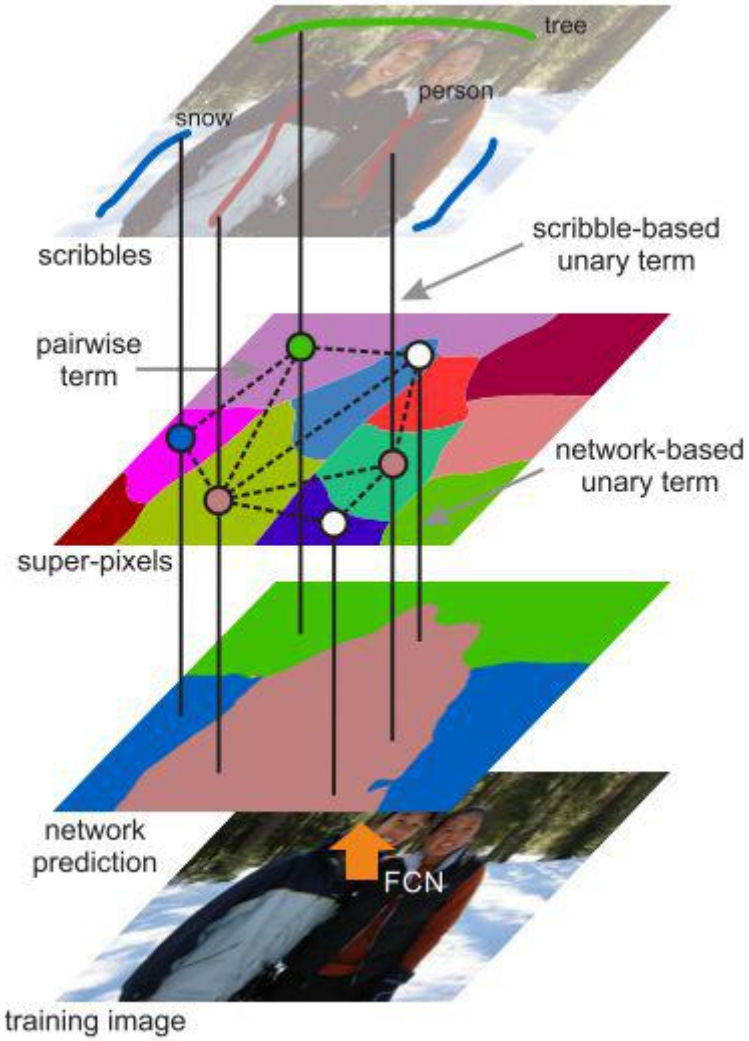
ScribbleSup: Scribble-Supervised Convolutional Networks for Semantic Segmentation (CVPR 2016)

香港中文大学的 Di Lin 提出了一个基于 Scribble 标记的弱监督学习方法。Scribble 是一个很方便使用的标记方法，因此被用得比较广泛。如下图，只需要画五条线就能完成对一副图像的标记工作



(a) image (b) mask annotation (c) scribble annotation

ScribbleSup 分为两步，第一步将像素的类别信息从 scribbles 传播到其他未标记的像素，自动完成所有的训练图像的标记工作； 第二步使用这些标记图像训练 CNN。在第一步中，该方法先生成 super-pxels, 然后基于 graph cut 的方法对所有的 super-pixel 进行标记。



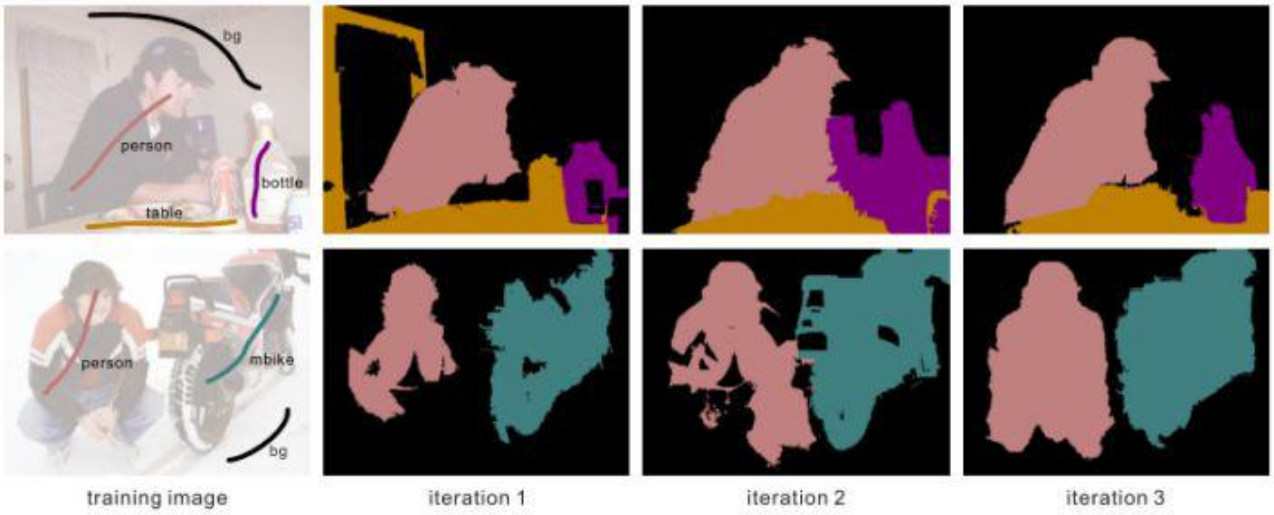
Graph cut 的能量函数为：

$$\sum_i \psi_i(y_i|X,S) + \sum_{i,j} \psi_{ij}(y_i, y_j, X)$$

在这个 graph 中，每个 super-pixel 是 graph 中的一个节点，相接壤的 super-pixel 之间有一条连接的边。这个能量函数中的一元项包括两种情况，一个是来自于 scribble 的，一个是来自 CNN 对该 super-pixel 预测的概率。整个最优化过程实际上是求 graph cut 能量函数和 CNN 参数联合最优值的过程：

$$\sum_i \psi_i^{scr}(y_i|X,S) + \sum_i -logP(y_i|X,\Theta) + \sum_{i,j} \psi_{ij}(y_i,y_j|X)$$

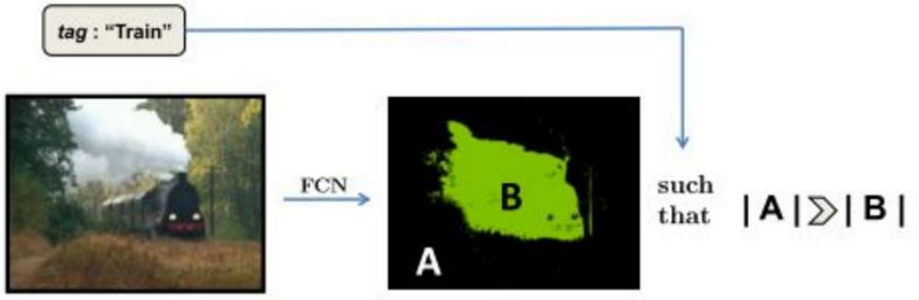
上式的最优化是通过交替求 Y 和 Θ 的最优值来实现的。文章中发现通过三次迭代就能得到比较好的结果。



8.8.2 图像级别标记

Constrained Convolutional Neural Networks for Weakly Supervised Segmentation（ICCV 2015）

UC Berkeley 的 Deepak Pathak 使用了一个具有图像级别标记的训练数据来做弱监督学习。训练数据中只给出图像中包含某种物体，但是没有其位置信息和所包含的像素信息。该文章的方法将 image tags 转化为对 CNN 输出的 label 分布的限制条件，因此称为 Constrained convolutional neural network (CCNN).



该方法把训练过程看作是有线性限制条件的最优化过程：

$$\underset{\theta,P}{\text{minimize}} \quad D\left(P(X)\|Q(X|\theta)\right)$$
$$\text{subject to} \quad A\vec{P} \geq \vec{b}, \qquad \sum_X P(X) = 1.$$

其中 $P(X)$ 是一个隐含的类别分布，$Q(X)$ 是 CNN 预测的类别分布。目标函数是 KL-divergence 最小化。其中的线性限制条件来自于训练数据上的标记，例如一幅图像中前景类别像素个数期望值的上界或者下界（物体大小）、某个类别的像素个数在某图像中为 0，或者至少为 1 等。该目标函数可以转化为为一个 loss function，然后通过 SGD 进行训练。

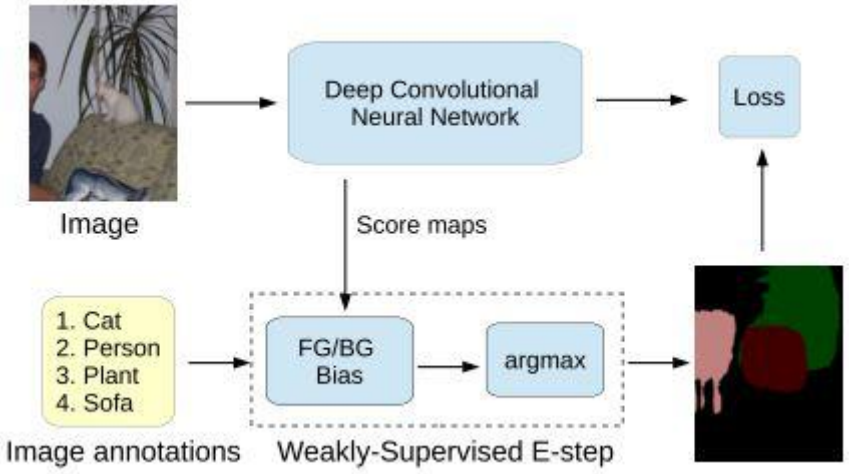
(a) Original image (b) Ground truth (c) Image tags (d) Image tags + size

实验中发现单纯使用 Image tags 作为限制条件得到的分割结果还比较差，在 PASCAL VOC 2012 test 数据集上得到的 mIoU 为 35.6%，加上物体大小的限制条件后能达到 45.1%， 如果再使用 bounding box 做限制，可以达到 54%。FCN-8s 可以达到 62.2%，可见弱监督学习要取得好的结果还是比较难。

8.8.3 DeepLab+bounding box+image-level labels

Weakly- and Semi-Supervised Learning of a DCNN for Semantic Image Segmentation

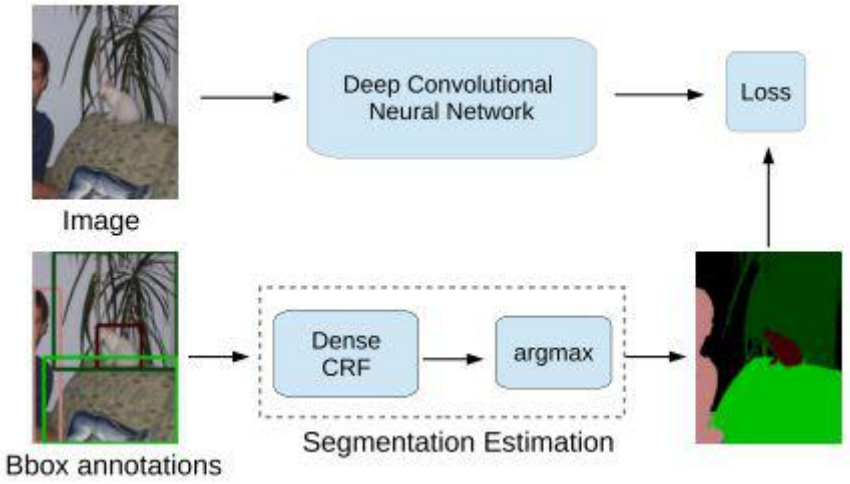
Google 的 George Papandreou 和 UCLA 的 Liang-Chieh Chen 等在 DeepLab 的基础上进一步研究了使用 bounding box 和 image-level labels 作为标记的训练数据。使用了期望值最大化算法（EM）来估计未标记的像素的类别和 CNN 的参数。



对于 image-level 标记的数据，我们可以观测到图像的像素值 x 和图像级别的标记 z ,但是不知道每个像素的标号 y ,因此把 y 当做隐变量。使用如下的概率图模式：

$$P(x,y,z;\theta)=P(x)(\prod_{m=1}^{M}P(y_m|x;\theta))P(z|y)$$

使用 EM 算法估计 θ 和 y 。E 步骤是固定 θ 求 y 的期望值，M 步骤是固定 y 使用 SGD 计算 θ.



对于给出 bounding box 标记的训练图像，该方法先使用 CRF 对该训练图像做自动分割，然后在分割的基础上做全监督学习。通过实验发现，单纯使用图像级别的标记得到的分割效果较差，但是使用 bounding box 的训练数据可以得到较好的结果，在 VOC2012 test 数据集上得到 mIoU 62.2%。另外如果使用少量的全标记图像和大量的弱标记图像进行结合，可以得到与全监督学习(70.3%)接近的分割结果(69.0%)。

8.8.4 统一的框架

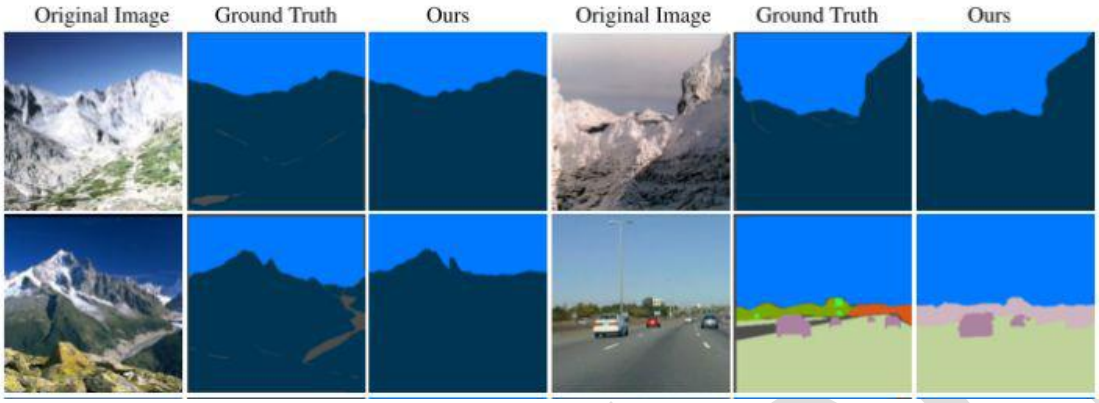
Learning to Segment Under Various Forms of Weak Supervision (CVPR 2015)

Wisconsin-Madison 大学的 Jia Xu 提出了一个统一的框架来处理各种不同类型的弱标记：图像级别的标记、bounding box 和部分像素标记如 scribbles。该方法把所有的训练图像分成共计 n 个 super-pixel，对每个 super-pixel 提取一个 d 维特征向量。因为不知道每个 super-pixel 所属的类别，相当于无监督学习，因此该方法对所有的 super-pixel 做聚类，使用的是最大间隔聚类方法(max-margin clustering, MMC),该过程的最优化目标函数是：

$$\frac{1}{2}tr(W^TW)+\lambda\sum_{p=1}^{n}\sum_{c=1}^{C}\xi(w_c;x_p;h_p^c)$$

其中 W 是一个特征矩阵，每列代表了对于的类别的聚类特征。 ξ 是将第 p 个 super-pixel

划分到第c类的代价。在这个目标函数的基础上，根据不同的弱标记方式，可以给出不同的限制条件，因此该方法就是在相应的限制条件下求最大间隔聚类。



该方法在 Siftflow 数据集上得到了比较好的结果，比 state-of-the-art 的结果提高了 10%以上。

小结：在弱标记的数据集上训练图像分割算法可以减少对大量全标记数据的依赖，在大多数应用中会更加贴合实际情况。弱标记可以是图像级别的标记、边框和部分像素的标记等。训练的方法一般看做是限制条件下的最优化方法。另外 EM 算法可以用于 CNN 参数和像素类别的联合求优。