空间金字塔池化

空间金字塔池化层简介：

在对图片进行卷积操作的时候，卷积核的大小是不会发生变化的额，反向调节的权重仅仅是数值会发生变化。但是，但是，但是，输入的图片的大小你是否可以控制呢？哈哈，我们的输入图片大小是会发生变化的，这里图片大小的变化并不会在卷积操作和 polling 操作产生影响，但是会对全连接层的链接产生影响。这篇文章的核心就是解决如何对于不同的输入图片（主要是针对大小不同）都可以直接运用到已经训练好的网络中去。

为什么要引入空间金字塔池化：

首先说一下为什么要有这个层：我们处理图片的大小不一，都有自己不同的像素值，但是同一批数据，如果非要进过一定的裁剪把他处理成为相同大小的图像，例如，我们可以先把图片的四个角裁剪下来，在加上一个中心区域的图片，这是五个变形的图片，然后再把图片水平翻转之后依然是相同的操作会得到五个图片，总合计是 10 个大小相同的图片，这是一种方法，当然还有其他的方法，例如在 overFeat 那篇

论文中也提到一种方法，等等。这些裁剪技术都会达到不错的结果，但是依然会存在一些问题，例如，有些区域裁剪的时候都会有重复，无形之中加大了该区域的权重。所以，这篇 paper 就提出了金字塔池化来解决输入图片大小不一的情况。

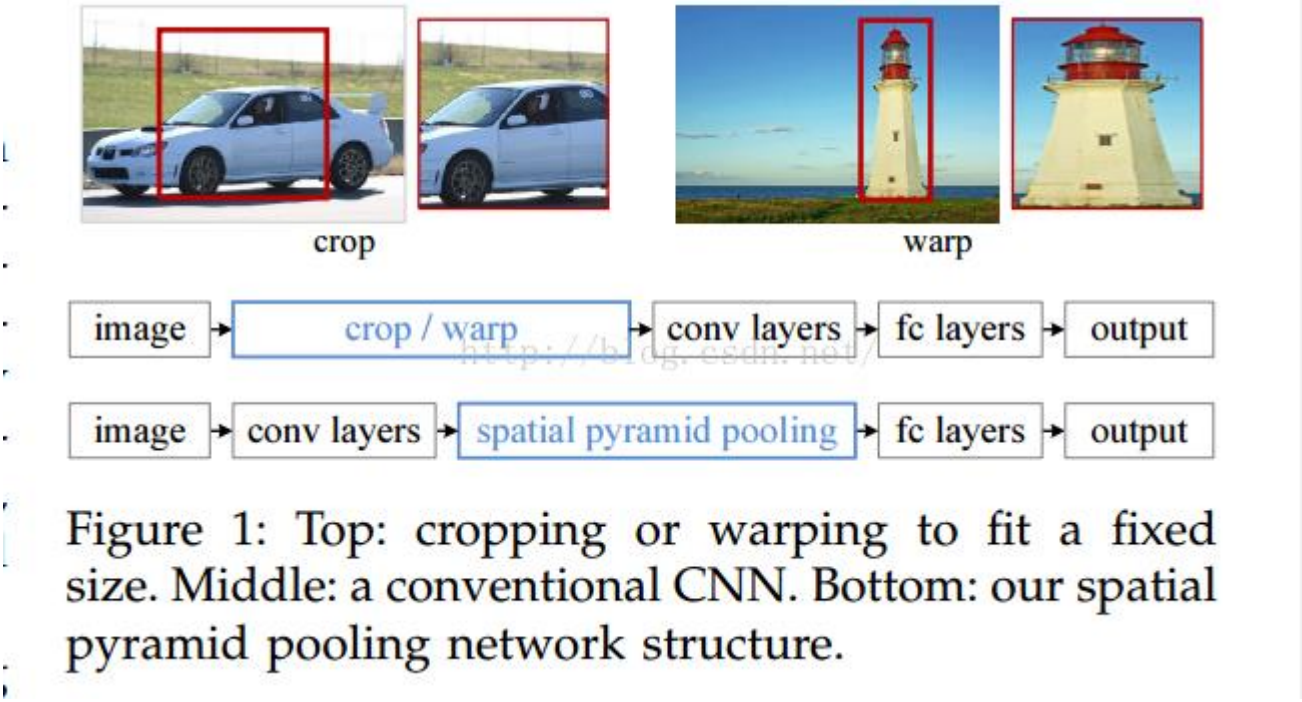


Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

观察图一，最上面代表的是把原来的图片都进行适当的 crop 和 wrap 之后得到适当的像素值，中间的就是对应的网络模型。最下面的是这篇论文的模型，在最后一个卷积层后面紧随种恶一个 spatital pyramid polling 层，紧跟着的是全连接层。这样就可以解决不管输入图片的大小是多大，都可以用当前的网络进行测试的问题。

但是我们需要明白的是为什么这个层放到了最后一个圈基层的后面呢？也就是说为什么 convolution 和 polling 都对如数图片的大小不敏感，而全连接层却敏感呢？我们来看看。假设输入图片的大小是 100*100 经过 5 个卷积核 3*3 之后会差生 5*98*98 的 feature maps，就算你的输入图片的大小变成 102*102，那我的 feature maps 就是 5*100*100。这里的 feature maps 经过 2*2 的 polling 之后得到的是 25*25 和 26*26。没什么影响，这里的卷积核的大小是固定的，可以去卷积任何大小的图片。但是全连接层就不同了。假设最后一个卷积层有 50 个输出，下一层的全链接有 1000 个输入，那么这个链接矩阵就是 50*1000，哈哈，你想想，如果这里每次的输入图片大小都不一样，到这里如何进行链接呢？因为不同的图片大小经过最后一个卷积层的输入到输出之后压根就不可能都是 50 啊。这就是我们为什么要在全连接层这人进行操作的原因。

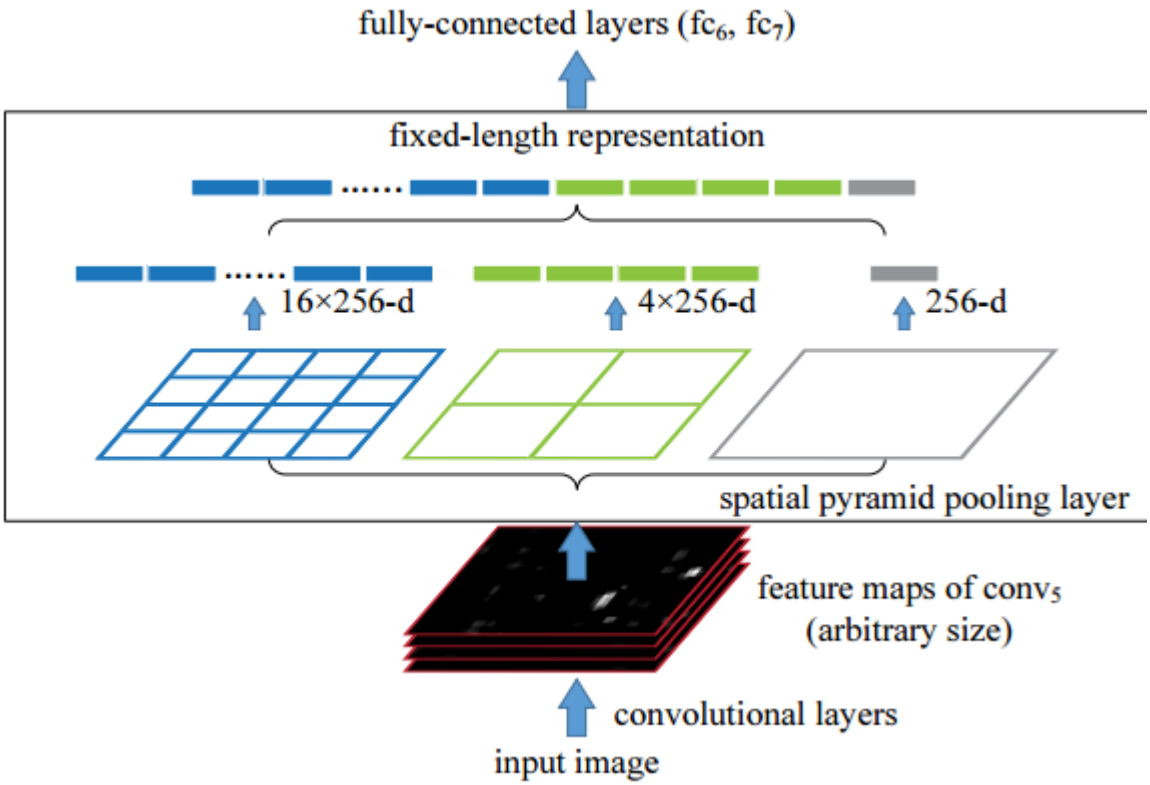
空间金字塔池化层的特点

当然，这篇 paper 的特点可不是仅仅有一个。金字塔池化层有如下的三个优点，第一：他可以解决输入图片大小不一造成的缺陷。

第二：由于把一个 feature map 从不同的角度进行特征提取，再聚合的特点，显示了算法的 robust 的特性。第三：同时也在 object recongtion 增加了精度。其实，你也可以这样想，最牛掰的地方是因为在卷积层的后面对每一张图片都进行了多方面的特征提取，他就可以提高任务的精度。好比是不同大小的图片在不同的网络中进行训练一样，大大提高了模型的精度。SPP 在现有的各种网络模型上都得到了 state of the art 的高度，例如 R-CNN 上面。不仅仅如此，R-CNN 需要对不同大小的边框内的图像 feed into 不同的网络模型，整个过程是特别的耗时，SPP-Net 刚好就可以决绝这个问题，大大的减少了时间。

什么是金字塔池化层

哈哈，唧唧歪歪的说了半天，都没有说到重点上去，现在我们步入正题，说说什么是金字塔池化层。



如上图所示，从下往上看，这是一个传统的网络架构模型，5 层卷积层，这里的卷积层叫做 convolution 和 pooling 层的联合体，统一叫做卷积层，后面跟随全连接层。我们这里需要处理的就是在网络的全连接层前面加一层金字塔 pooling 层解决输入图片大小不一的情况。我们可以看到这里的 spatital pyramid pooling layer 就是把前一卷积层的 feature maps 的每一个图片上进行了 3 个卷积操作。最右边的就是原图像，中间的是把图像分成大小是 4 的特征图，最右边的就是把图像分成大小是 16 的特征图。那么每一个 feature map 就会变成

16+4+1=21 个 feature maps。这不就解决了特征图大小不一的状况了吗？

```
[pool3x3]          [pool2x2]          [pool1x1]
type=pool          type=pool          type=pool
pool=max           pool=max           pool=max
inputs=conv5       inputs=conv5       inputs=conv5
sizeX=5            sizeX=7            sizeX=13
stride=4           stride=6           stride=13

[fc6]
type=fc
outputs=4096
inputs=pool3x3,pool2x2,pool1x1
```

那么具体是如何操作的呢？我们来看看：图片的输入大小相等的时候，我们假设图片经过第五层的卷积核之后的输出大小是 a*a （例如，13*13），我们 bins 的大小是 n*n,那么每一个窗口 win=cell(a/n), 然而 stride 的大小是 stride=floor(a/n)，前面一个是向上取整，后面的是向下取整，最终会形成如上图所示的三个 pooling 操作。这三个的

本质都是最大化池，只不过使用了不同的窗口大小和移动的步长而已。fc6 代表的是全链接层。实验表明，多层次的卷积行为可以使得实验的最终精度提升。而不同大小的图片操作都是一样的方法。

其他的都是一些实验相关的数据，很简单，自己看把。