SSD

深度学习相关的目标检测方法也可以大致分为两派： 需要region proposal的，如R-CNN、SPP-net、Fast R-CNN、Faster R-CNN、R-FCN； 端到端（End-to-End）无需区域提名的，如YOLO、SSD。

检测与分类的关系

SSD是在精度和效率方面都较为不错的检测算法。 SSD中检测问题是转换为分类问题来处理的。分类问题往往是预测出图片中物体的标签值；检测问题不仅仅要预测出物体的标签值还要找出物体的bounding box。 分类问题中物体往往占据图片的很大比例，如图1所示；而检测问题往往是一幅图片中有多个不同大小的物体，不仅需要预测出各个物体的标签值还需要找出各个物体的bounding box，如图2所示。 简而言之检测问题需要给出物体的bounding box和label，因此检测网络的输出需要有：

1. 类别概率；
2. bounding box坐标，可以用bounding box的中心点和宽高来表示，(cx,cy,h,w)。
   需要注意的是在检测问题中类别需要加入背景类别，因为图片中很多区域是没有物体的。为说明问题方便假设数据集只有猫狗，因此类别标签总共有三种，使用one-hot编码表示，[1 0 0]代表猫，[0 1 0]代表狗，[0 0 1]代表背景。

图1

图2

检测转换为分类问题

将检测问题转换为分类问题的一个简单方式是训练一个分类网络，该分类网络输出包含三个概率，分别代表是猫、狗、背景的概率值。训练分类网络之前需要正确地给图片打标签。
比如，对于图3我们该如何给该训练图片打标签用来训练分类网络呢？


图3 Input image for object detection

为了说明如何给图3打标签我们需要对图3进行多次剪切操作，剪切出的图如图4所示：

图4

图4中Patch2精确的包含了猫，因此其标签是"cat"即[1 0 0]。对于Patch1、Patch3没有包含任何物体，因此它们的标签是"background"即[0 0 1]。

滑动窗口

分类网络训练好之后可以用滑动窗口的方式来检测图片。首先需要一些确定大小的窗口，如图5蓝色矩形框所示，使用滑动窗口遍历整个图片。

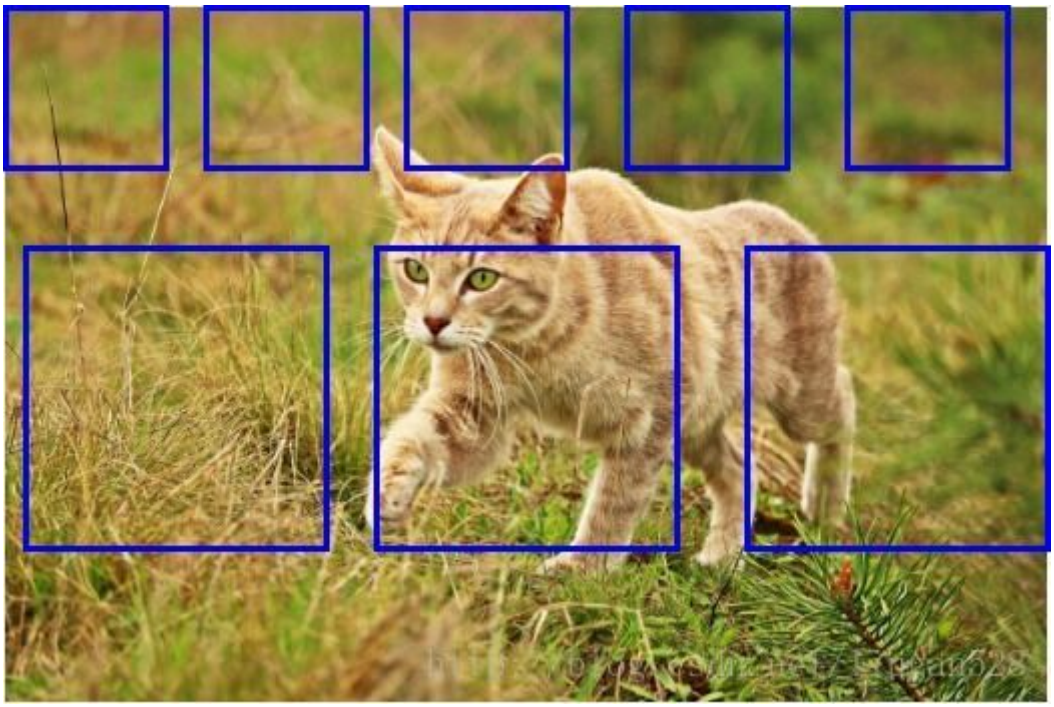

图5

然后将滑动窗口所在区域的图片crop下来并resize到分类网络所需输入的大小就可以进行分类了。我们使用更小的滑动窗口重复操作就可以捕获到更小尺寸的目标。也就是说如果一个目标在图片中我们需要一个合适大小的滑动窗口才能准确的crop到该目标并给出其分类标签。图6的gif动图演示了滑动窗口是如何遍历图片的：

图6

但是究竟需要进行多少次crop才能覆盖到所有的目标呢？我们不仅仅要考虑滑动窗口crop的patch的位置还要考虑滑动窗口的尺度，因为被检测的目标可能是任意的尺度。这样就会导致有上千个patch分别送入网络来对一张图片中的目标进行预测。 下面让我们来看一下如何减少计算时间。 ## 减少冗余计算以减少计算时间 考虑图7crop的两个patch：

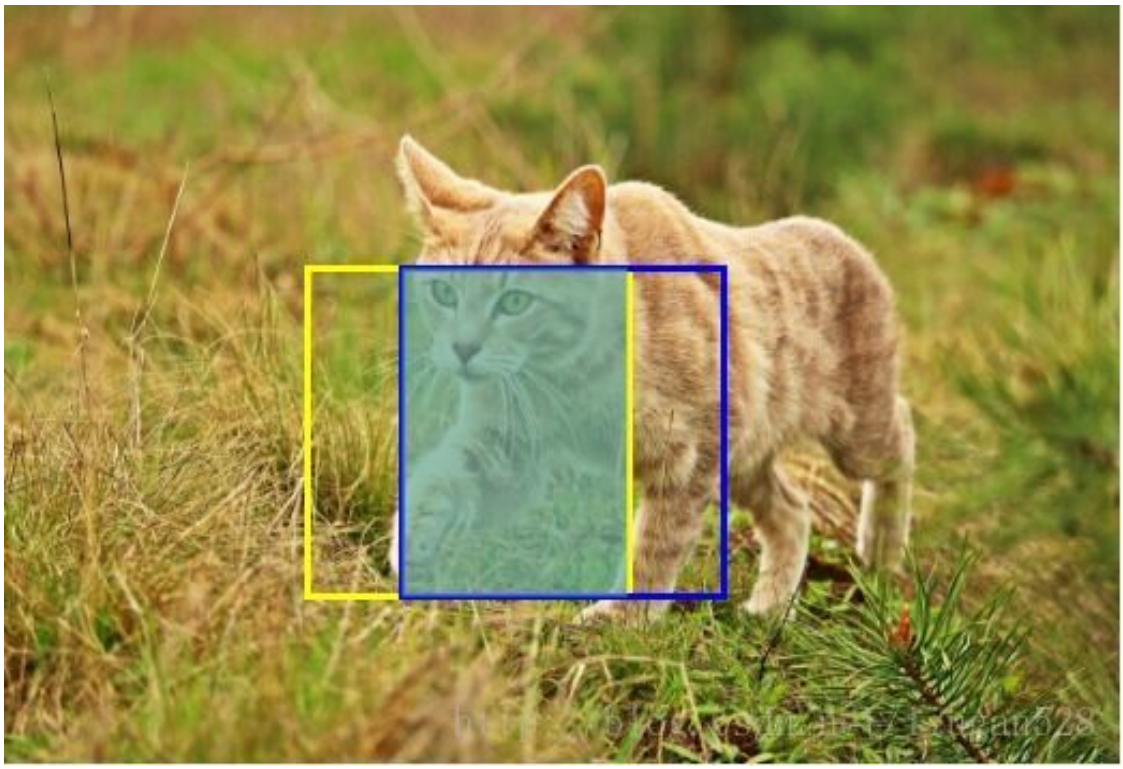

图7

我们可以看到两个patch（黄、蓝）有很多重叠的地方。这就意味着当分别把这两个patch送入网络的时候重叠的地方是需要重复计算的。在SPP-Net中引入，在Fast RCNN中发扬的一种技术可以很好的解决这个问题。 下面我们通过一个小网络来理解这个技术细节。

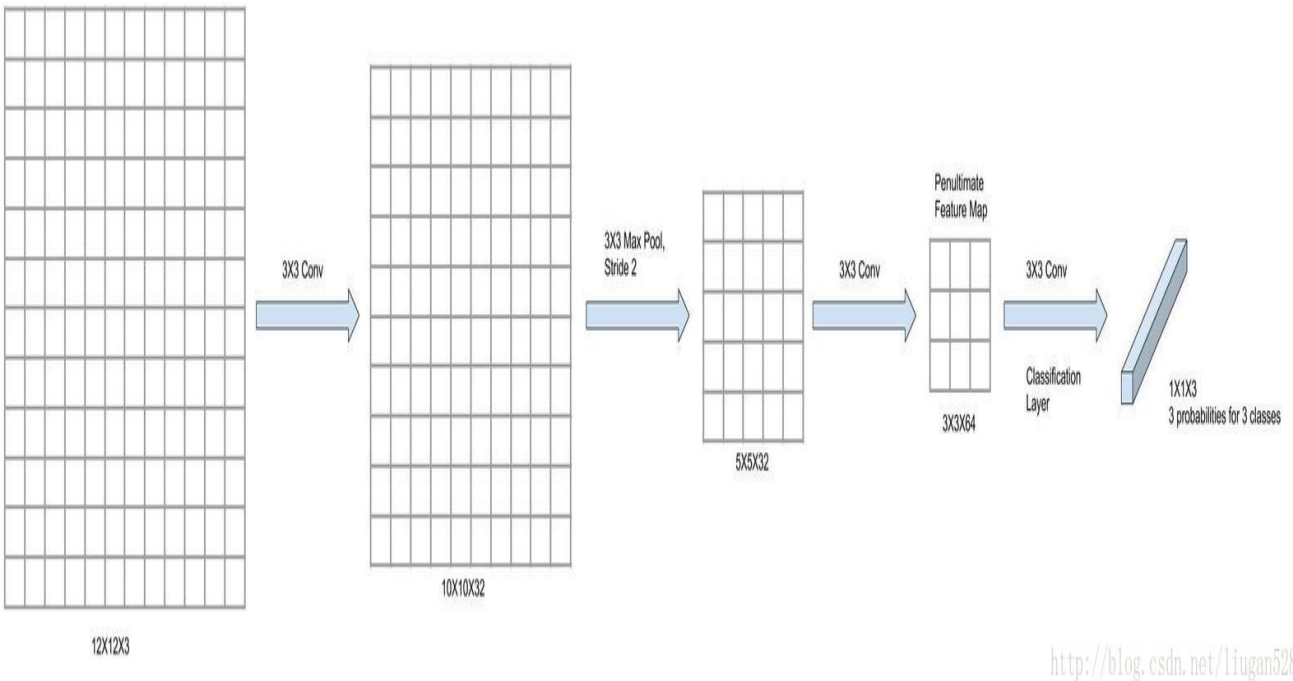

图8

图8展示了一个12X12大小的图像经过3个卷积层，每个卷积层滤波器大小为3X3。经过三个卷积之后得到了倒数第二个特征图（penultimate feature map, PFM）其大小为3X3X64。在该3X3的特征图上再使用3个3X3的卷积核得到1X1X3的最终特征图（final feature map, FFM），对应着一个含有三个概率值的向量，分别对应着预测为猫、狗、背景的概率。 现在我们来看一个稍大的图，如图9所示，原始图像大小为14X14。先看左上角12X12大小的patch（中心点是(6,6)），经过三个3X3的卷积之后得到了PFM中的一个3X3的小patch，再经过一个3X3的卷积之后得到FFM上的一个得分。该过程可由图9中贯穿始终的蓝色区域形象化的描述。对于第二个12X12patch（右上角红色框区域，中心点为(8,6)，stride为2）同样会在最红的特征图上产生一个得分，其过程可由图9中贯穿始终的红色框形象化的描述。 由图9可看出每个12X12的patch都可以在PFM上产生一个3X3的patch，在FFM上产生一个得分。

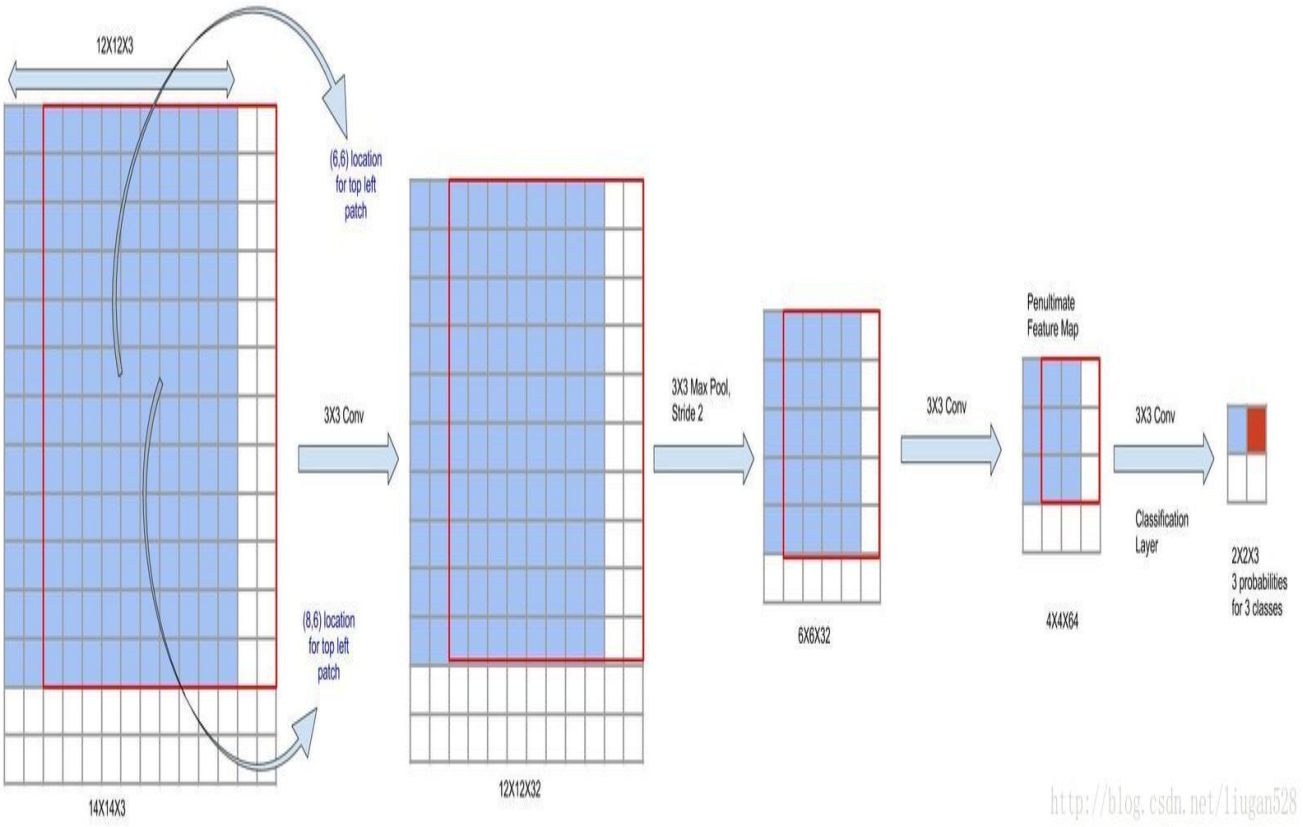

图9

在计算特征图的过程如果每个patch都分别计算那么计算量是巨大的，非常耗时。但是利用该技术之后我们就可以避免对不同patch包含的相同部分的重复计算。下面我们对整个图像仅仅计算一次整幅图的特征图。我们知道在PFM的每个patch对应着和原图中不同的patch。此时我们可以在PFM上直接使用预测权重，相当于在PFM上进行卷积而不是在原图上进行卷积，因此会省去很多计算。 总结一下，我们将整幅图像送入网络得到PFM，然后使用3X3的滑动窗口在PFM上进行滑动分别得到每个patch的得分。 现在还存在一个小问题，并不是原图上所有的patch都呈现在输出中。在上面例子中patch的中心点是(6,6)、(8,6)等。但是中心点是(7,6)的patch被跳过了。后面我们会阐述如何处理如何处理这类patch。

默认Boxes（Default Boxes/Anchor Boxes）

在分类输出中能够直接表示的Boxes被称为Default Boxes或Anchor Boxes。比如在上面的例子中中心点在(6,6)、(8,6)的box（亦即patch）就是Anchor Boxes，其大小为12X12。

网络训练方法

下面我们再以一个例子说明如何训练网络。我们用一个24X24的图片来进行展示，如图10所示。对该图进行和上述相同的卷积操作可以得到FFM，大小为7X7。

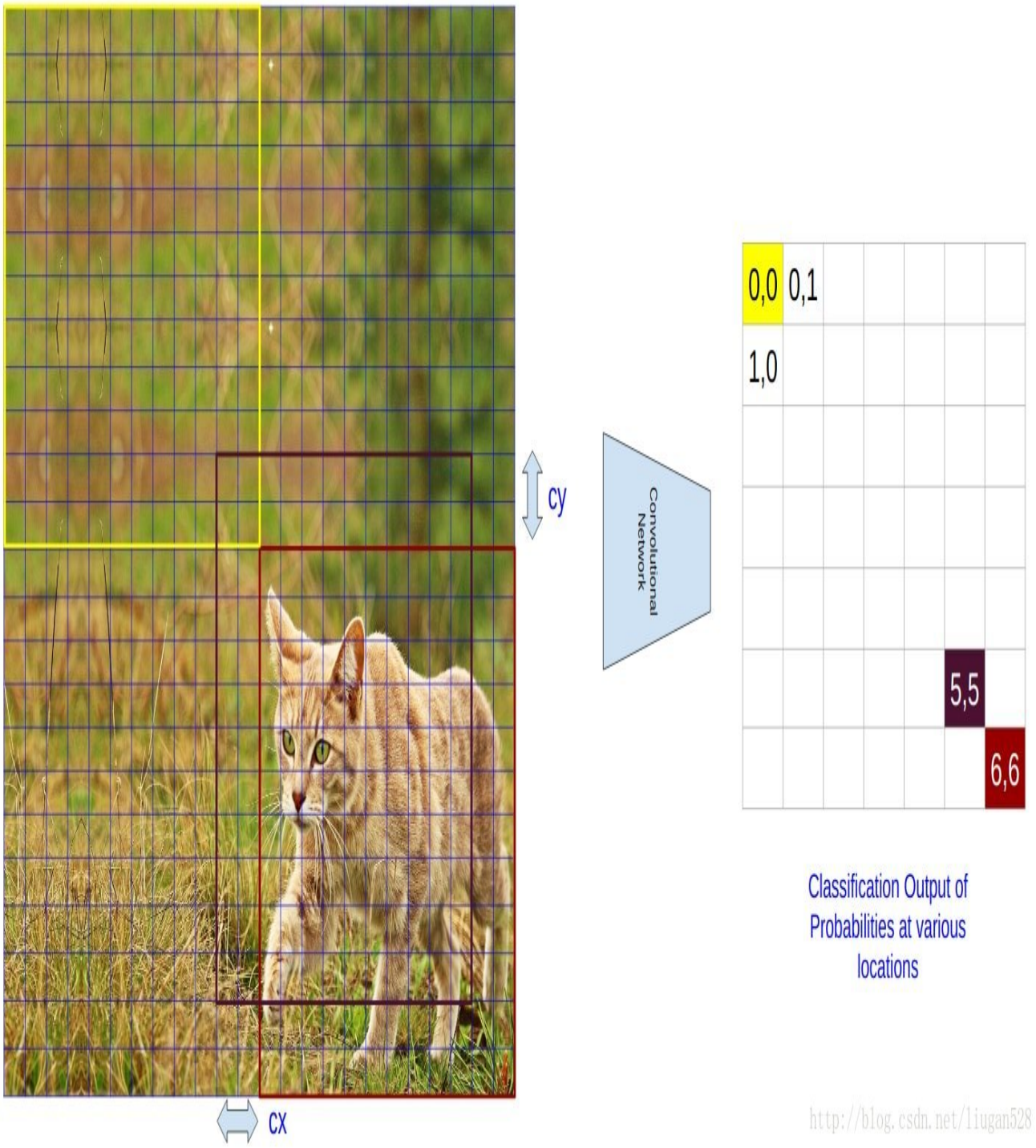

图10

在准备训练集的时候首要做的就是给FFM打ground truth的标签。我们给7X7大小的FFM建立索引(i,j)。每个Anchor Box都对应着FFM的一个位置。图中(6,6)位置精确的对应着猫因此其标签是[1 0 0]，而(0,0)、(0,1)、(1,0)等位置没有任何目标因此它们的标签是[0 0 1]。 对于仅含有部分猫的patch该如何打标签呢？ 例如对于FFM中(5,5)这个位置对应着原图中品红矩形框的patch，可以看出矩形框相对于猫有一定的偏移，没有精确包住猫，但有很高的重叠。 对于这种问题我们有两种选择：要么打标签为背景要么打标签为猫。如果box精确的包含了猫可以打标签为猫，其它打标签为背景。但是这样会产生两个问题：1. 训练样本非常不均衡；2.如果没有box能精确的包含猫则没法打标签。 因此对于品红patch我们应该打标签为猫，但是我们要把偏移量考虑进去。记cx、cy为品红patch中心点距红色框（精确包含了猫）中心点的偏移量。我们需要设计一种方法使得网络可以预测偏移量，这样就可以找出目标的精确坐标了。

因此对于每个位置我们不仅需要输出类别标签还需要输出中心点的偏移量。为了预测cx、cy我们可以加入回归损失函数。

处理多尺度问题

处理好了不同位置的目标，现在还需要处理不同尺度的目标。尺度有默认的Anchor Boxes的尺度、明显小于Anchor Boxes的尺度、明显大于Anchor Boxes的尺度。

接近Anchor Boxes的尺度

对于和Anchor Boxes大小差不多的目标，可以用类似偏移量的预测方法。记预测的目标的宽高分别为w、h。依然是利用回归损失函数来预测宽、高。

明显小于Anchor Boxes的尺度

方便起见我们这里只讨论目标的尺度远小于Anchor Boxes的情况。
该问题的解决方案是，对于卷积过程中产生的每个特征图都做预测，如图11所示。这是SSD算法中介绍的核心内容。

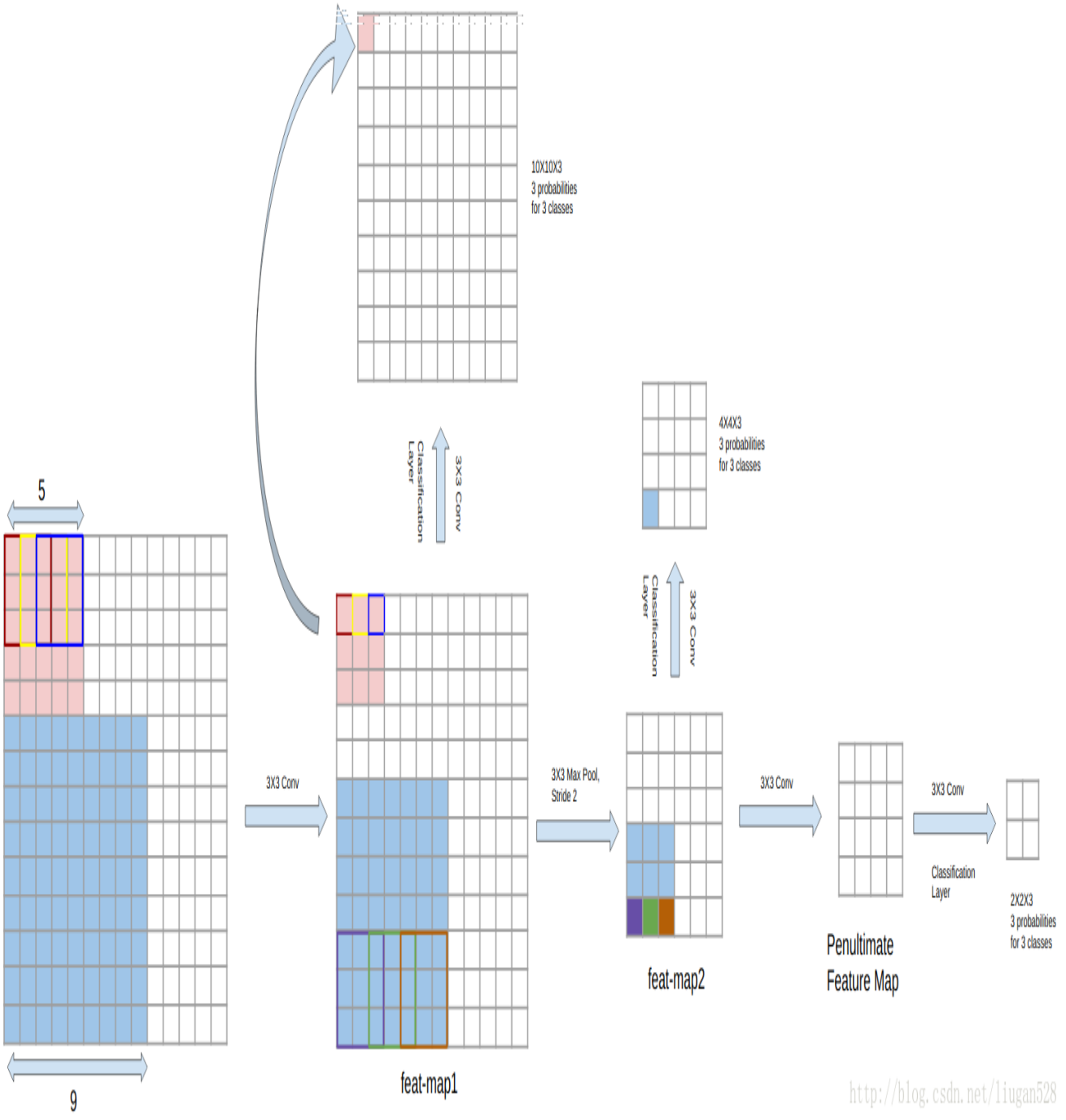

图11

用所有的特征图做预测

前面所述的方法中我们只是利用一个3X3的卷积核在PFM上进行卷积来得到输出（包括类别box的概率、中心点偏移量、宽、高）。这里我们对所有的特征图利用3X3的卷积核来进行预测。我们知道更低层次的预测（即靠近原始图片的卷积得到的特征图）处理的是更小的目标。卷积特征图对应着原图的一部分，叫做感受野。卷积的层次越深则每个神经元对应的感受野越大，越浅层的感受野越小。

在图11的例子中第一个特征图（feat-map1）左上角3X3的box对应着原图5X5的大小，第二个特征图（feat-map2）对应着原图9X9的大小。由此可见网络层次越深感受野越大。

也就是说处理与Anchor Boxes尺度相差较大的目标，我们可以在感受野与Anchor Boxes差不多的特征图上进行预测。例如，对于图11的原图如果目标的大小是6X6那么我们应该在feat-

map2上进行预测。
对于尺度明显大于Anchor Boxes的目标也可以用类似的方法来处理。

总结

本文从理论的角度介绍了SSD的核心技术，还有许多细节没有介绍，比如处理宽、高不相同的patch等。更多细节请参考SSD: Single Shot MultiBox Detector。

参考文献

1. http://cv-tricks.com/object-detection/single-shot-multibox-detector-ssd/
2. SSD: Single Shot MultiBox Detector