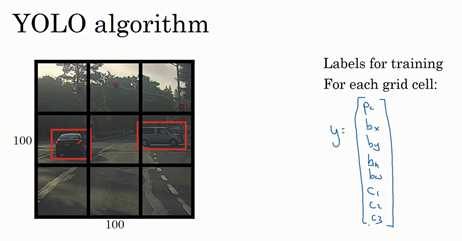
### 3.5 Bounding Box预测（Bounding box predictions）

在上一个视频中，你们学到了滑动窗口法的卷积实现，这个算法效率更高，但仍然存在问题，不能输出最精准的边界框。在这个视频中，我们看看如何得到更精准的边界框。

图片包含 树, 照片, 不同, 建筑物

描述已自动生成

在滑动窗口法中，你取这些离散的位置集合，然后在它们上运行分类器，在这种情况下，这些边界框没有一个能完美匹配汽车位置，也许这个框（编号1）是最匹配的了。还有看起来这个真实值，最完美的边界框甚至不是方形，稍微有点长方形（红色方框所示），长宽比有点向水平方向延伸，有没有办法让这个算法输出更精准的边界框呢？



其中一个能得到更精准边界框的算法是**YOLO算法**，**YOLO**(**You only look once**)意思是你只看一次，这是由**Joseph Redmon**，**Santosh Divvala**，**Ross Girshick**和**Ali Farhadi**提出的算法。

是这么做的，比如你的输入图像是100×100的，然后在图像上放一个网格。为了介绍起来简单一些，我用3×3网格，实际实现时会用更精细的网格，可能是19×19。基本思路是使用图像分类和定位算法，前几个视频介绍过的，然后将算法应用到9个格子上。（基本思路是，采用图像分类和定位算法，本周第一个视频中介绍过的，逐一应用在图像的9个格子中。）更具体一点，你需要这样定义训练标签，所以对于9个格子中的每一个指定一个标签，是8维的，和你之前看到的一样，，等于0或1取决于这个绿色格子中是否有图像。然后、、和作用就是，如果那个格子里有对象，那么就给出边界框坐标。然后、和就是你想要识别的三个类别，背景类别不算，所以你尝试在背景类别中识别行人、汽车和摩托车，那么、和可以是行人、汽车和摩托车类别。这张图里有9个格子，所以对于每个格子都有这么一个向量。

图片包含 文字

描述已自动生成

我们看看左上方格子，这里这个（编号1），里面什么也没有，所以左上格子的标签向量是。然后这个格子（编号2）的输出标签也是一样，这个格子（编号3），还有其他什么也没有的格子都一样。

现在这个格子呢？讲的更具体一点，这张图有两个对象，**YOLO算法做的就是，取两个对象的中点，然后将这个对象分配给包含对象中点的格子。所以左边的汽车就分配到这个格子上（编号4），然后这辆Condor（车型：神鹰）中点在这里，分配给这个格子（编号6）。**所以即使中心格子（编号5）同时有两辆车的一部分，我们就假装中心格子没有任何我们感兴趣的对象，所以对于中心格子，分类标签和这个向量类似，和这个没有对象的向量类似，即。而对于这个格子，这个用绿色框起来的格子（编号4），目标标签就是这样的，这里有一个对象，，然后你写出、、和来指定边界框位置，然后还有类别1是行人，那么，类别2是汽车，所以，类别3是摩托车，则数值，即。右边这个格子（编号6）也是类似的，因为这里确实有一个对象，它的向量应该是这个样子的，作为目标向量对应右边的格子。

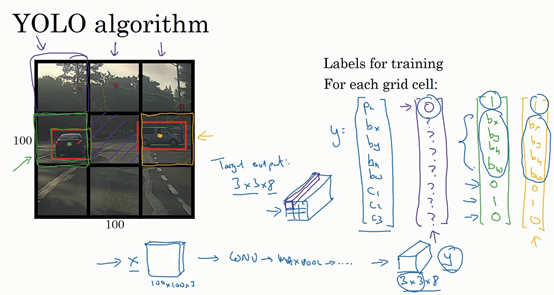
所以对于这里9个格子中任何一个，你都会得到一个8维输出向量，因为这里是3×3的网格，所以有9个格子，总的输出尺寸是3×3×8，所以**目标输出是3×3×8**。图片包含 文字, 地图

描述已自动生成

对于这个例子中，左上格子是1×1×8，对应的是9个格子中左上格子的输出向量。所以对于这3×3中每一个位置而言，对于这9个格子，每个都对应一个8维输出目标向量，其中一些值可以是**dont care-s**（即？），如果这里没有对象的话。所以总的目标输出，这个图片的输出标签尺寸就是3×3×8。



如果你现在要训练一个输入为100×100×3的神经网络，现在这是输入图像，然后你有一个普通的卷积网络，卷积层，最大池化层等等，最后你会有这个，选择卷积层和最大池化层，这样最后就映射到一个3×3×8输出尺寸。所以你要做的是，有一个输入，就是这样的输入图像，然后你有这些3×3×8的目标标签。当你用反向传播训练神经网络时，将任意输入映射到这类输出向量。



所以这个算法的优点在于神经网络可以输出精确的边界框，所以测试的时候，你做的是喂入输入图像，然后跑正向传播，直到你得到这个输出。然后对于这里3×3位置对应的9个输出，我们在输出中展示过的，你就可以读出1或0（编号1位置），你就知道9个位置之一有个对象。如果那里有个对象，那个对象是什么（编号3位置），还有格子中这个对象的边界框是什么（编号2位置）。只要每个格子中对象数目没有超过1个，这个算法应该是没问题的。一个格子中存在多个对象的问题，我们稍后再讨论。但实践中，我们这里用的是比较小的3×3网格，实践中你可能会使用更精细的19×19网格，所以输出就是19×19×8。这样的网格精细得多，那么多个对象分配到同一个格子得概率就小得多。

重申一下，把对象分配到一个格子的过程是，你观察对象的中点，然后将这个对象分配到其中点所在的格子，所以即使对象可以横跨多个格子，也只会被分配到9个格子其中之一，就是3×3网络的其中一个格子，或者19×19网络的其中一个格子。在19×19网格中，两个对象的中点（图中蓝色点所示）处于同一个格子的概率就会更低。

图片包含 文字

描述已自动生成

所以要注意，首先这和图像分类和定位算法非常像，我们在本周第一节课讲过的，就是它显式地输出边界框坐标，所以这能让神经网络输出边界框，可以具有任意宽高比，并且能输出更精确的坐标，不会受到滑动窗口分类器的步长大小限制。其次，这是一个卷积实现，你并没有在3×3网格上跑9次算法，或者，如果你用的是19×19的网格，19平方是361次，所以你不需要让同一个算法跑361次。相反，这是单次卷积实现，但你使用了一个卷积网络，有很多共享计算步骤，在处理这3×3计算中很多计算步骤是共享的，或者你的19×19的网格，所以这个算法效率很高。

图片包含 文字, 白板

描述已自动生成

事实上**YOLO**算法有一个好处，也是它受欢迎的原因，因为这是一个卷积实现，实际上它的运行速度非常快，可以达到**实时识别**。在结束之前我还想给你们分享一个小细节，如何编码这些边界框、、和，我们在下一张幻灯片上讨论。

这里有两辆车，我们有个3×3网格，我们以右边的车为例（编号1），红色格子里有个对象，所以目标标签就是，，然后、、和，然后，，，即。你怎么指定这个边界框呢？

**Specify the bounding boxes**：

图片包含 天空

描述已自动生成

在**YOLO**算法中，对于这个方框（编号1所示），我们约定左上这个点是，然后右下这个点是,要指定橙色中点的位置，大概是0.4，因为它的位置大概是水平长度的0.4，然后大概是0.3，然后边界框的高度用格子总体宽度的比例表示，所以这个红框的宽度可能是蓝线（编号2所示的蓝线）的90%，所以是0.9，它的高度也许是格子总体高度的一半，这样的话就是0.5。换句话说，、、和单位是相对于格子尺寸的比例，所以和必须在0和1之间，因为从定义上看，橙色点位于对象分配到格子的范围内，如果它不在0和1之间，如果它在方块外，那么这个对象就应该分配到另一个格子上。这个值（和）可能会大于1，特别是如果有一辆汽车的边界框是这样的（编号3所示），那么边界框的宽度和高度有可能大于1。

指定边界框的方式有很多，但这种约定是比较合理的，如果你去读**YOLO**的研究论文，**YOLO**的研究工作有其他参数化的方式，可能效果会更好，我这里就只给出了一个合理的约定，用起来应该没问题。不过还有其他更复杂的参数化方式，涉及到**sigmoid**函数，确保这个值（和）介于0和1之间，然后使用指数参数化来确保这些（和）都是非负数，因为0.9和0.5，这个必须大于等于0。还有其他更高级的参数化方式，可能效果要更好一点，但我这里讲的办法应该是管用的。

这就是**YOLO**算法，你只看一次算法，在接下来的几个视频中，我会告诉你一些其他的思路可以让这个算法做的更好。在此期间，如果你感兴趣，也可以看看**YOLO**的论文，在前几张幻灯片底部引用的**YOLO**论文。

**Redmon, Joseph, et al. "You Only Look Once: Unified, Real-Time Object Detection." (2015):779-788.**

不过看这些论文之前，先给你们提个醒，**YOLO**论文是相对难度较高的论文之一，我记得我第一次读这篇论文的时候，我真的很难搞清楚到底是怎么实现的，我最后问了一些我认识的研究员，看看他们能不能给我讲清楚，即使是他们，也很难理解这篇论文的一些细节。所以如果你看论文的时候，发现看不懂，这是没问题的，我希望这种场合出现的概率要更低才好，但实际上，即使是资深研究员也有读不懂研究论文的时候，必须去读源代码，或者联系作者之类的才能弄清楚这些算法的细节。但你们不要被我吓到，你们可以自己看看这些论文，如果你们感兴趣的话，但这篇论文相对较难。现在你们了解了**YOLO**算法的基础，我们继续讨论别的让这个算法效果更好的研究。