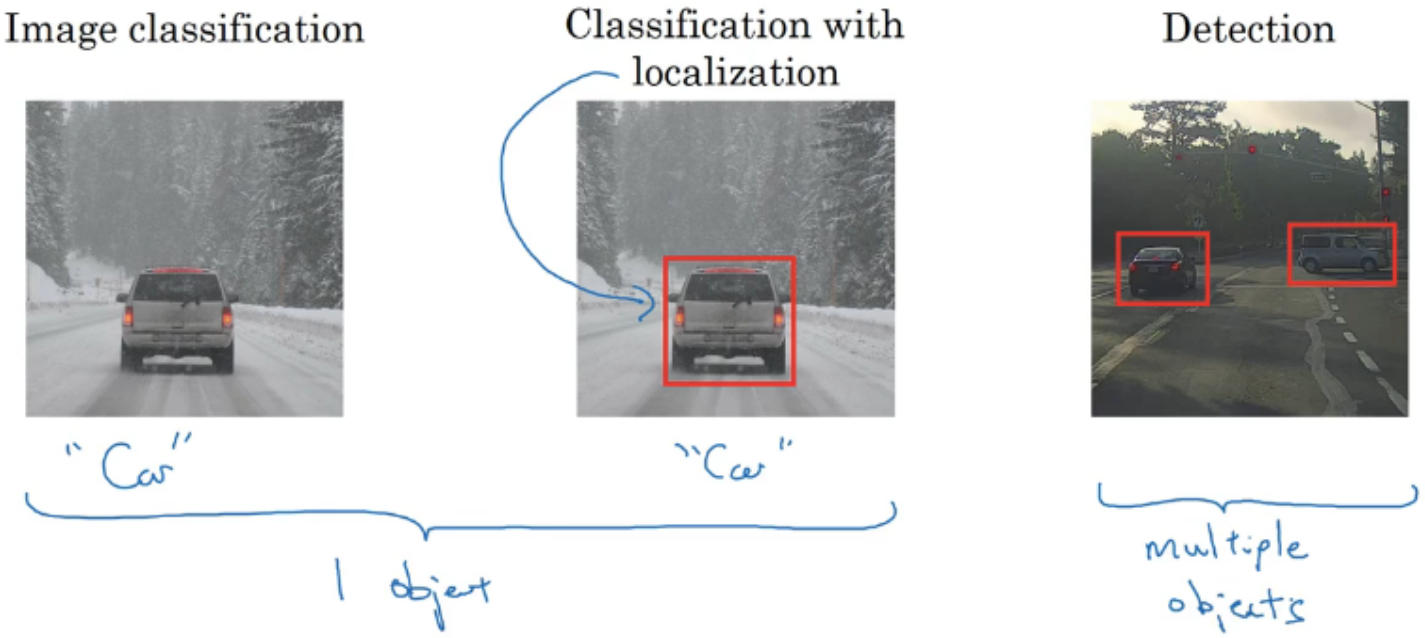
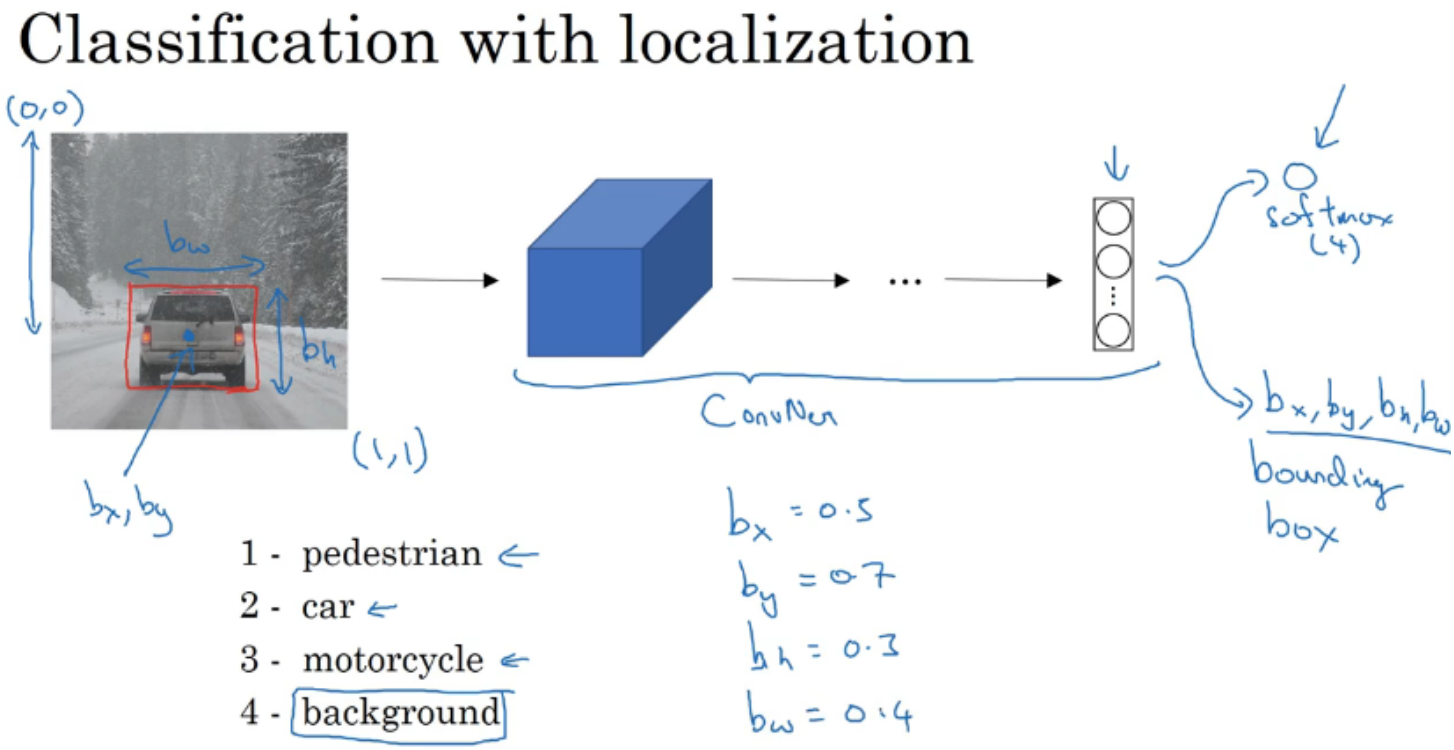
1. 目标定位

1.1 图像检测问题

- 图像分类：是否为汽车；
- 定位分类：是否为汽车，汽车在哪；
- 目标检测：检测不同汽车并定位。



1.2 定位分类



定义标签 y，它有如下四个类别：

- 1 - pedestrian
- 2 - car
- 3 - motorcycle
- 4 - background

神经网络的输出包括对象的位置信息(b_x, b_y, b_h, b_w)和对象的标签信息。

最终的目标标签为：

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

其中$c_1|c_2|c_3$代表是否存在对应的目标，1为存在，0为不存在。

- $P_c = 1$时表示图像中存在目标物体；
- $P_c = 0$时表示图像中不存在目标物体，此时y的其他值没有意义。

采用平方误差形式的loss函数为：

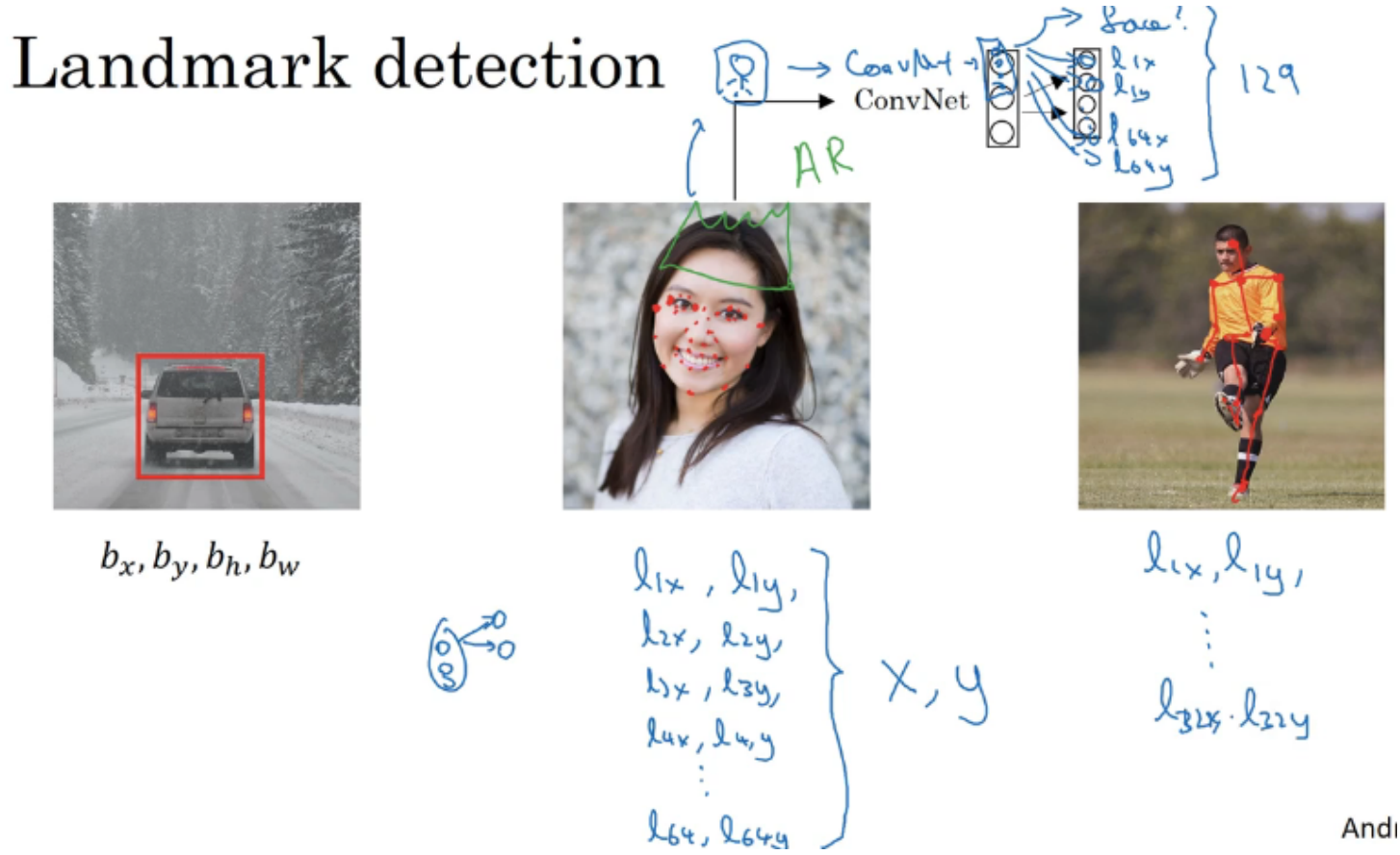
- $P_c = 1$时，$L(\hat{y}, y) = (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \cdots + (\hat{y}_8 - y_8)^2$
- $P_c = 0$时，需要关注神将网络输出的准确度，$L(\hat{y}, y) = (\hat{y}_1 - y_1)^2$。此时相当于只考虑神经网络对$P_c$的准确度。

在实际应用中，可以使用更好的方式：

- 对$c_1|c_2|c_3$和softmax使用log likelihood loss；
- 对$b_x|b_y|b_h|b_w$使用平方误差；
- 对P_c应用logistic regression误差或者平方误差。

2. 特征点检测

神经网络可以通过输出图片上的特征点的坐标(x, y)来实现目标特征的识别。

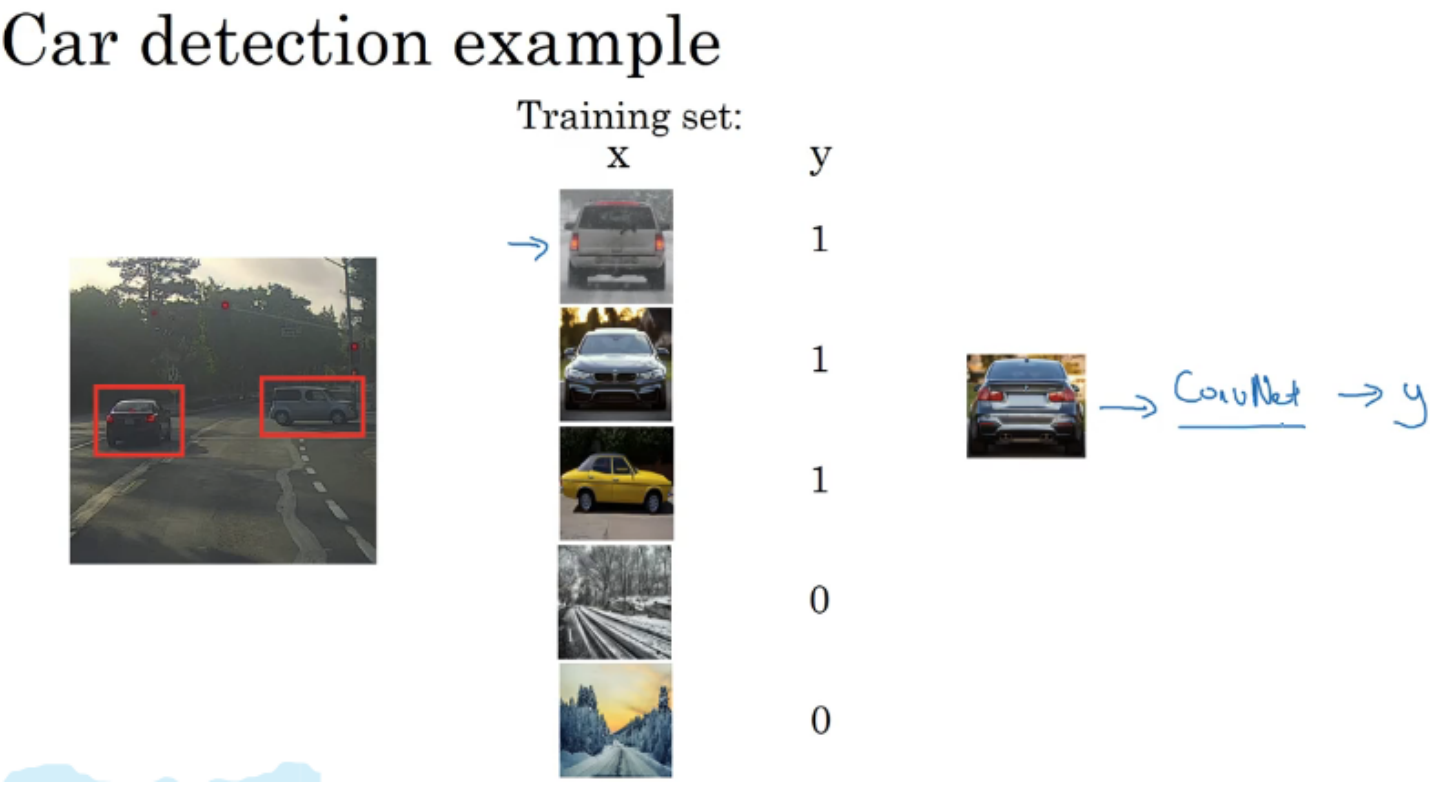


- 人脸表情识别的问题中，可以标定数据集中特征点的位置信息来对人脸进行不同位置不同特征的定位和标记。
- 人体姿态检测中，可以通过对人体不同特征位置关键点的标注来记录人体的姿态。

3. 目标检测

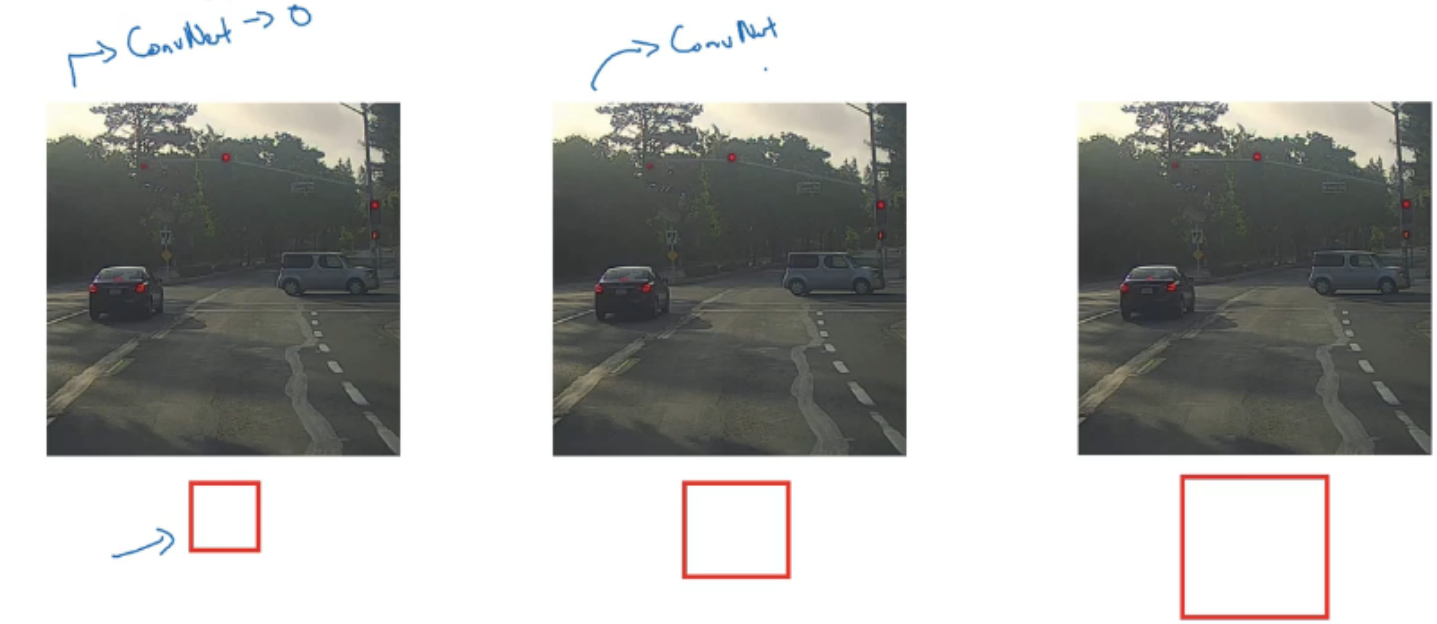
目标检测采用基于滑动窗口的检测算法。

3.1 训练模型



- X_train: 对图像进行裁剪，仅保留汽车的部分或者没有汽车。
- y_train: 针对X_train，有汽车的标注为1，没有汽车的标注为0。

3.2 滑动窗口检测

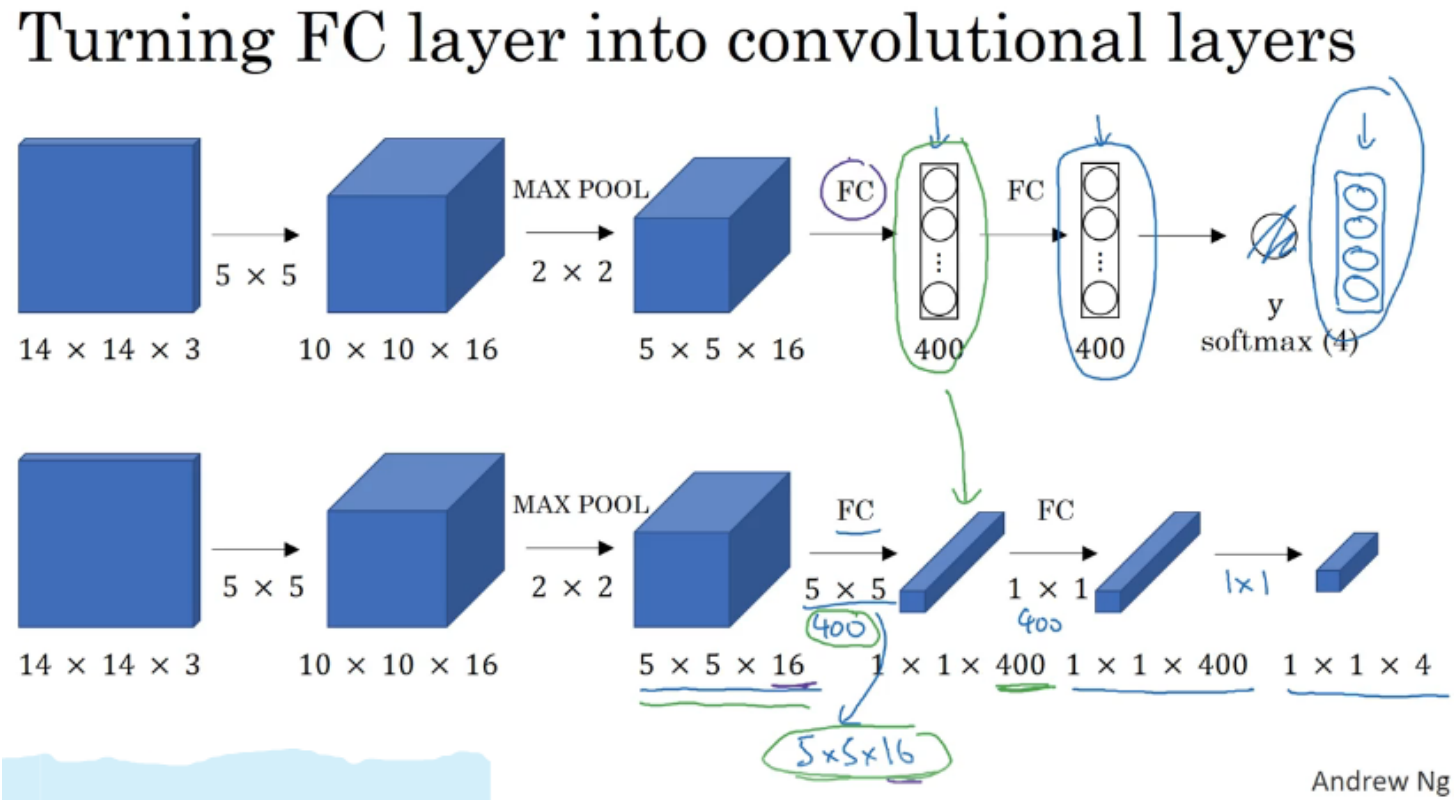


1. 选定一个特定大小的窗口，将窗口的图片输入到卷积神经网络中，判断是否有汽车；
2. 固定步幅移动窗口，遍历图像的每个区域，不断地进行检测；
3. 选择一个更大的窗口，再次遍历图像，不断地进行检测。

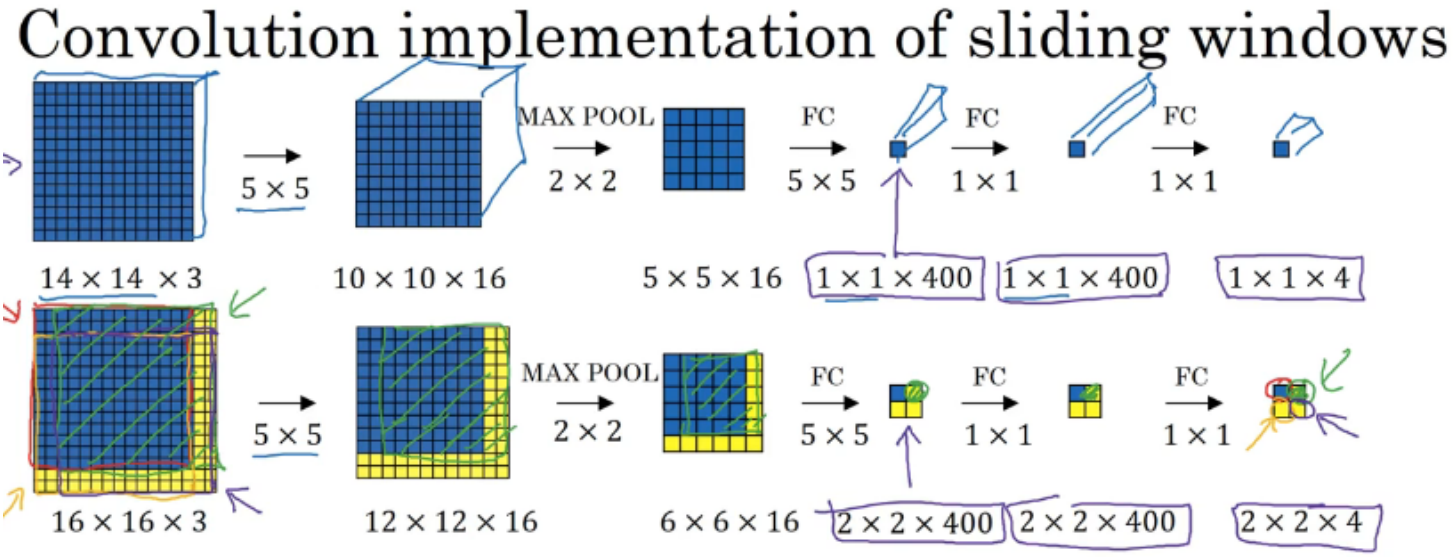
缺点：计算成本大，每个窗口的小图都需要进行卷积运算。

3.3 卷积层代替全连接层

卷积神经网络中的全连接层被卷积核为1×1大小的卷积层代替。1×1的卷积核相当于在三维图像的切片上使用了一个全连接层。

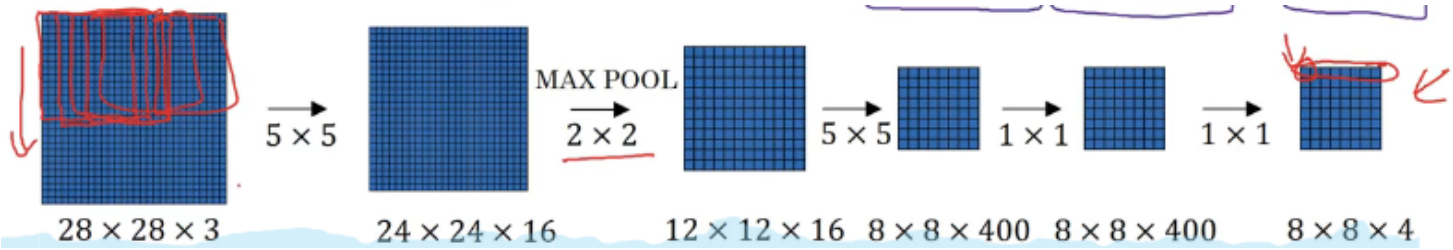


3.4 通过卷积实现滑动窗口检测算法



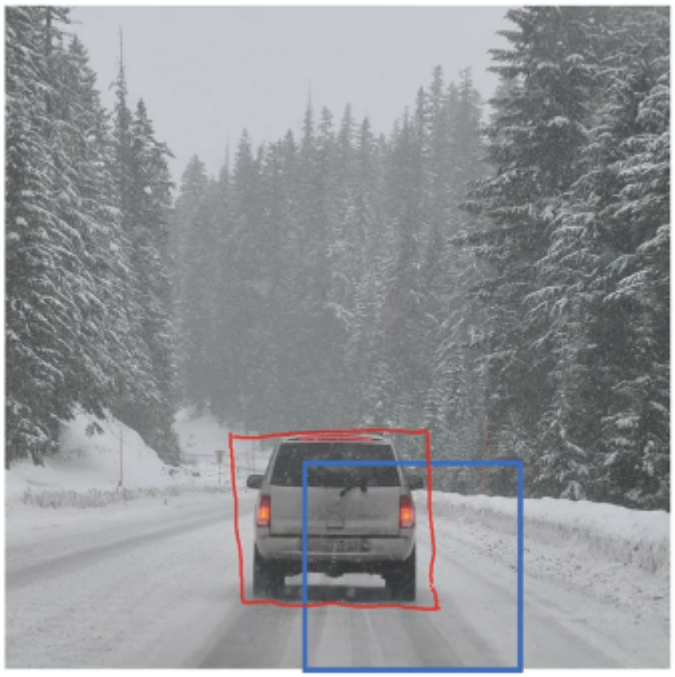
蓝色区域代表滑动窗口，滑动的步长为2，输入一个大小为$16 \times 16 \times 3$的图片，则会进行4次卷积操作，输出4个标签。但是在滑动的过程中，有大量的重复计算。最终在输出的四个子方块中，蓝色部分的图像位于左上角。如果是对绿色区域的窗口进行卷积运算，最后的输出图像位于右上角。因此我们并不需要把输入图片分割成4个子集分别进行传播，而是把他们作为一张图片进行计算。

对于一个更大的图片，也可以利用1×1卷积使得只需要一次向前传播，便能够同时得到所有窗口图片的预测值。

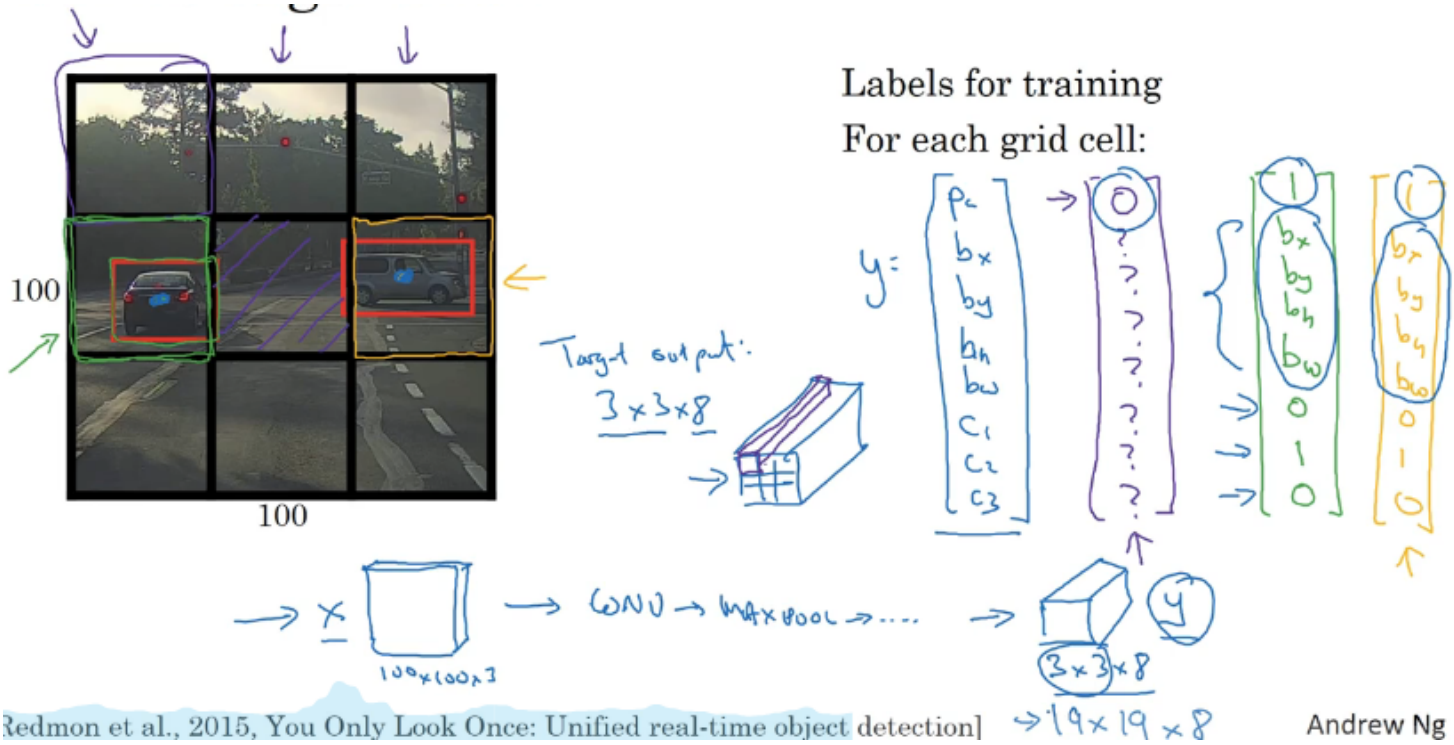


4. Bounding box predictions

利用卷积实现的滑动窗口法存在一个问题：不能输出精准的边界框（Bounding Box）。



4.1 YOLO algorithm

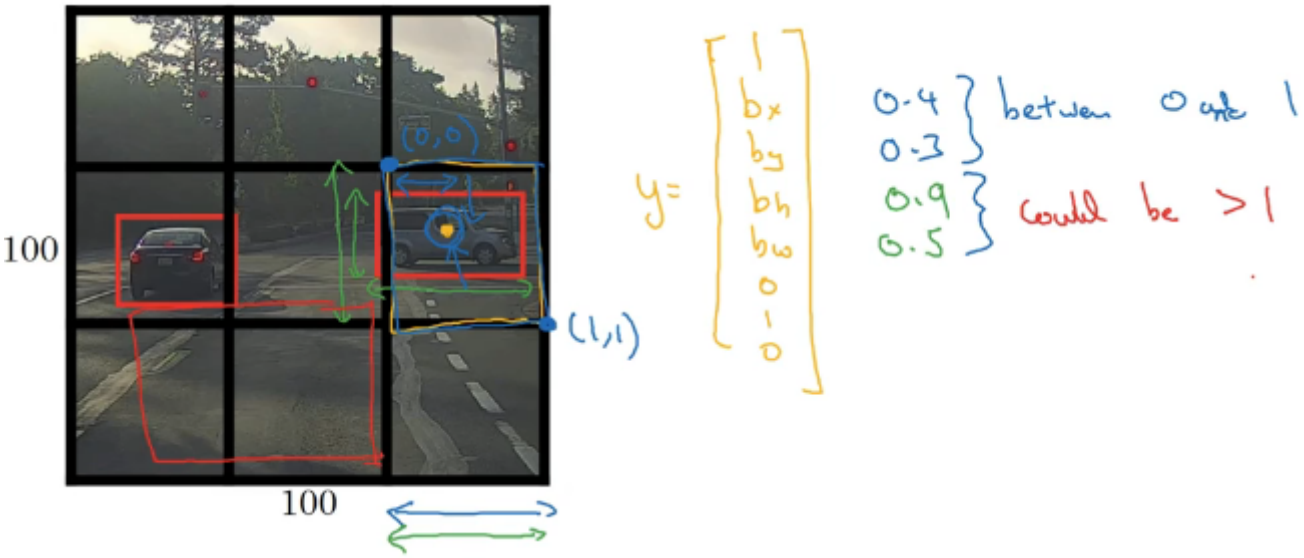


- 将图像分为精细的$n \times n$网格，比如上图的3×3；
- 每一个格子都使用图像分类和定位算法，每个格子的标签为：$y = [P_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3]$
- 将每一个格子的标签输出合并，大小为$n \times n \times 8$。

YOLO算法的优点是神经网络可以输出更为精准的边界框。

- YOLO算法将对象分配到其中点所在的格子中，即使这个对象横跨多个格子；
- YOLO算法显示地输出边界框，可以具有任意的宽高比，能够输出更精确的坐标，不受滑动窗口算法步长大小的限制；
- YOLO算法不是在$n \times n$网格上进行n^2次运算，只使用了单次的卷积便能实现，算法效率高，可以达到实时识别。

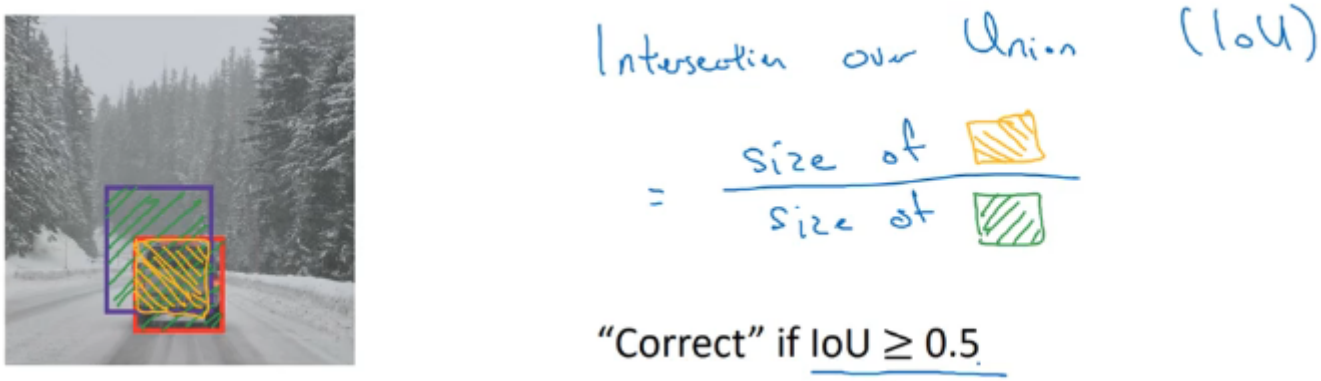
4.2 如何编码边界框(b_x, b_y, b_h, b_w)



- 每个网格的左上角为$(0,0)$，右下角为$(1,1)$；
- (b_x,b_y)表示中点的坐标，他们的值在$[0,1]$区间内；
- (b_h,b_w)表示边界框的大小，可能是跨网格的，因此他们的值可能会>1。

5. 交并比(Intersection over union)

交并比函数用于判断目标检测算法是否运作良好。



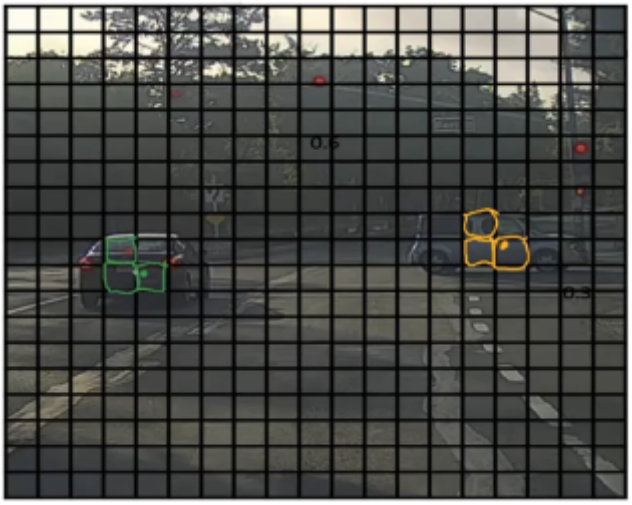
理想的边界框和实际的边界框，IoU函数用于计算边界框的交集和并集之比：

$$IoU = \frac{Area(intersection)}{Area(union)}$$

只要$IoU \geq 0.5$，那么识别的结果是可以接受的。IoU越高，对目标算法的检测越严格。

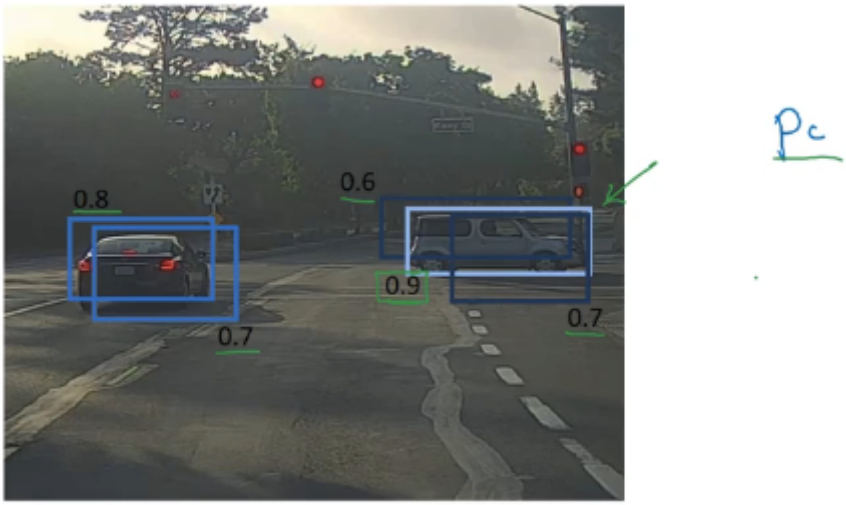
6. 非最大值抑制(Non-max suppression)

前面提到的目标检测算法可能会对同一个对象做出多次的检测，而非最大值抑制可以确保算法只对一个对象进行一次检测。



如果将图片分为精细的格子，那么会有很多格子输出同一个对象。

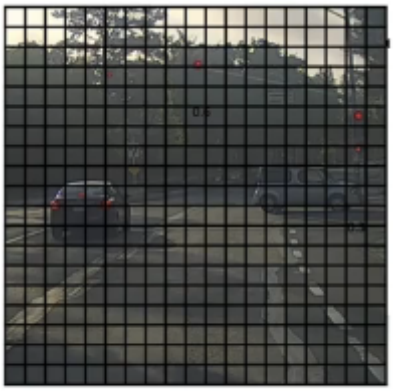
6.1 基本思想



- 每个网格使用目标检测算法后会输出一个范围在$[0,1]$之间的P_c，表示存在车的概率；
- 对于一个对象，可能会输出多个重叠的边界框，但是这些边界框的P_c大小不同；
- 选择具有最大P_c值的边界框，其他与该边界框存在高IoU的边界框将会被抑制；
- 逐一审视剩余的边界框，寻找具有最高P_c值的边界框，重复上述步骤。

非最大值抑制相当于输出概率最大的分类结果。

6.2 算法步骤



19× 19

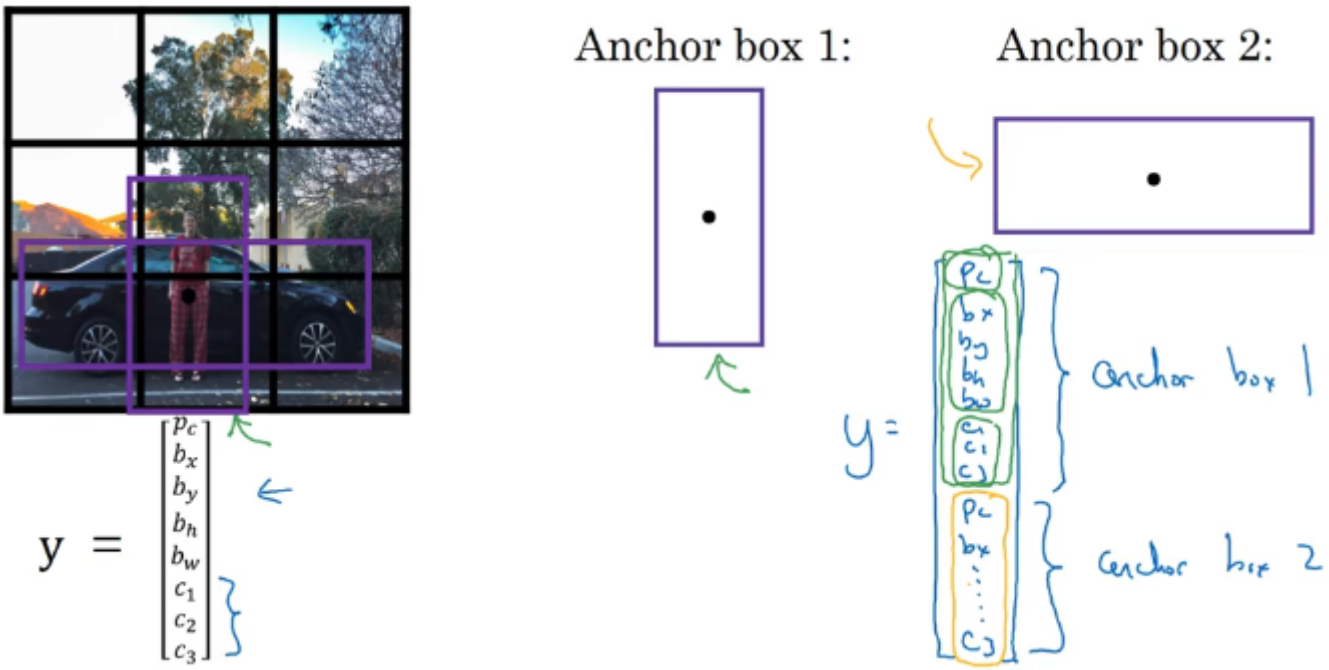
- 对于每个网格，其输出的标签为：$y = [P_c, b_x, b_y, b_h, b_w]$，$P_c$表示格子中有对象的概率；
- 抛弃所有$P_c < 0.6$的边界框，即抛弃所有低概率的边界框；
- 对于剩余的边界框：
  - 选择具有最大P_c值的边界框；
  - 抛弃剩余的边界框中与具有最大P_c值的边界框的交并比$IoU \geq 0.5$的边界框；

对于多对象的目标检测，输出标签中就会有多个分量，应该对每个输出类别分别做一次独立的非最大值抑制。

7. Anchor box

使用上面的方法，一个网格只能检测出一个对象。如果想让一个格子能够检测出多个对象，则可以使用**Anchor box**方法。

7.1 Overlapping objects

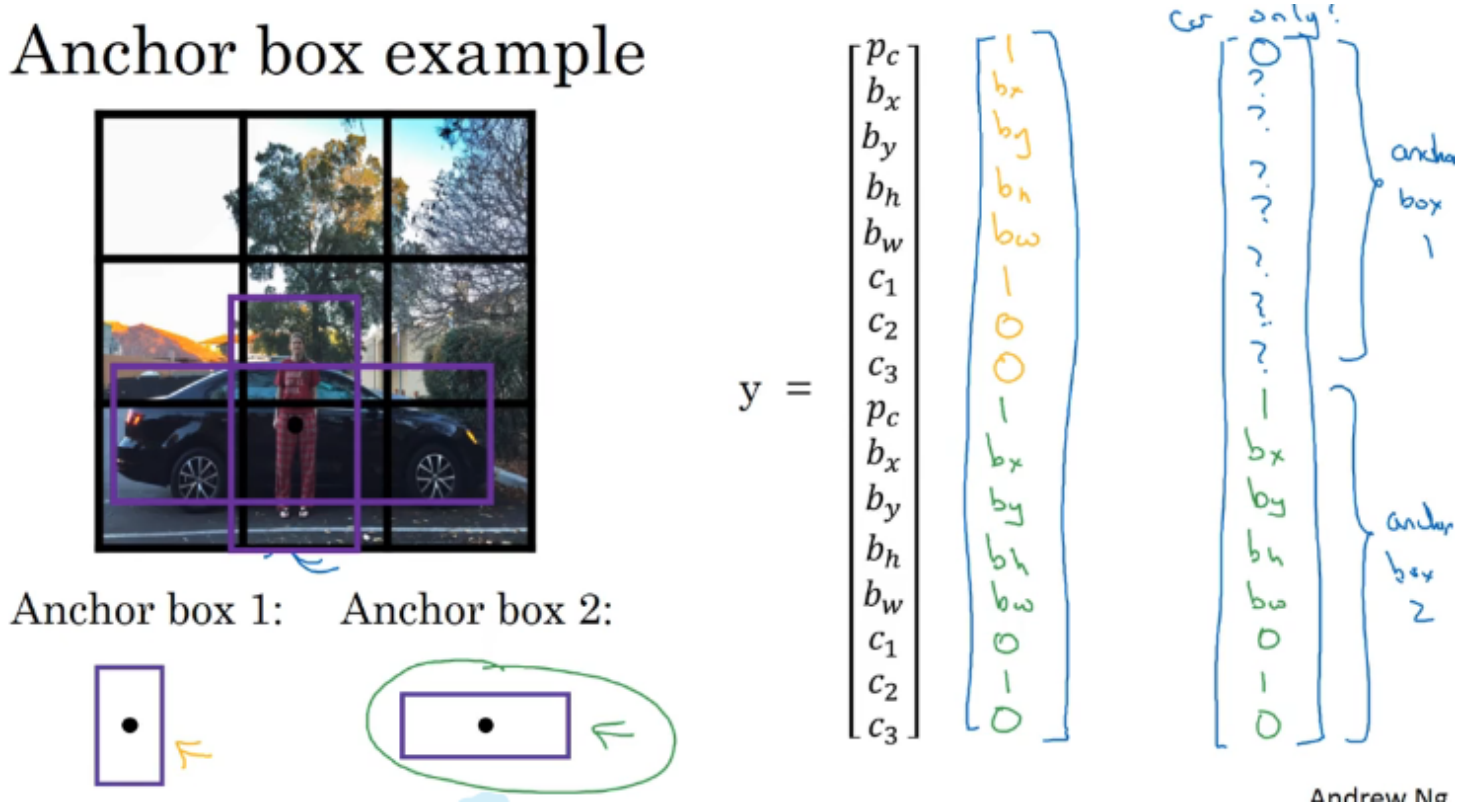


对于重叠的目标，如果只利用我们之前定义的标签$y = [P_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3]$，只能得到一个目标的输出。

如果预先定义多个不同形状的Anchor box，则将不同预测目标的Anchor box连接起来，得到新的标签$y = [P_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3, P_c, b_x, b_y, b_h, b_w, c_1, c_2, c_3, \cdots]$

- 不使用Anchor box：输出的大小为$n \times n \times 8$；
- 使用Anchor box：输出的大小为$n \times n \times 16$。

7.2 Anchor box example



设计两个Anchor box，行人的形状更像Anchor box 1，汽车的形状更像Anchor box 2，将行人和汽车分配到不同位置。

如果方格中只有汽车，这时的目标向量为$y = [0, ?, ?, ?, ?, ?, ?, ?, 1, b_x, b_y, b_h, b_w, 0, 1, 0]$。这时不用关心"?"的参数。

7.3 一些难点

存在以下两种问题，但出现的可能性不是很大，对目标检测算法不会带来很大影响。

- 如果存在三个对象，只能用一些额外的手段来处理；
- 如果存量Anchor box相同的两个对象，也必须引入一些专门的处理手段。

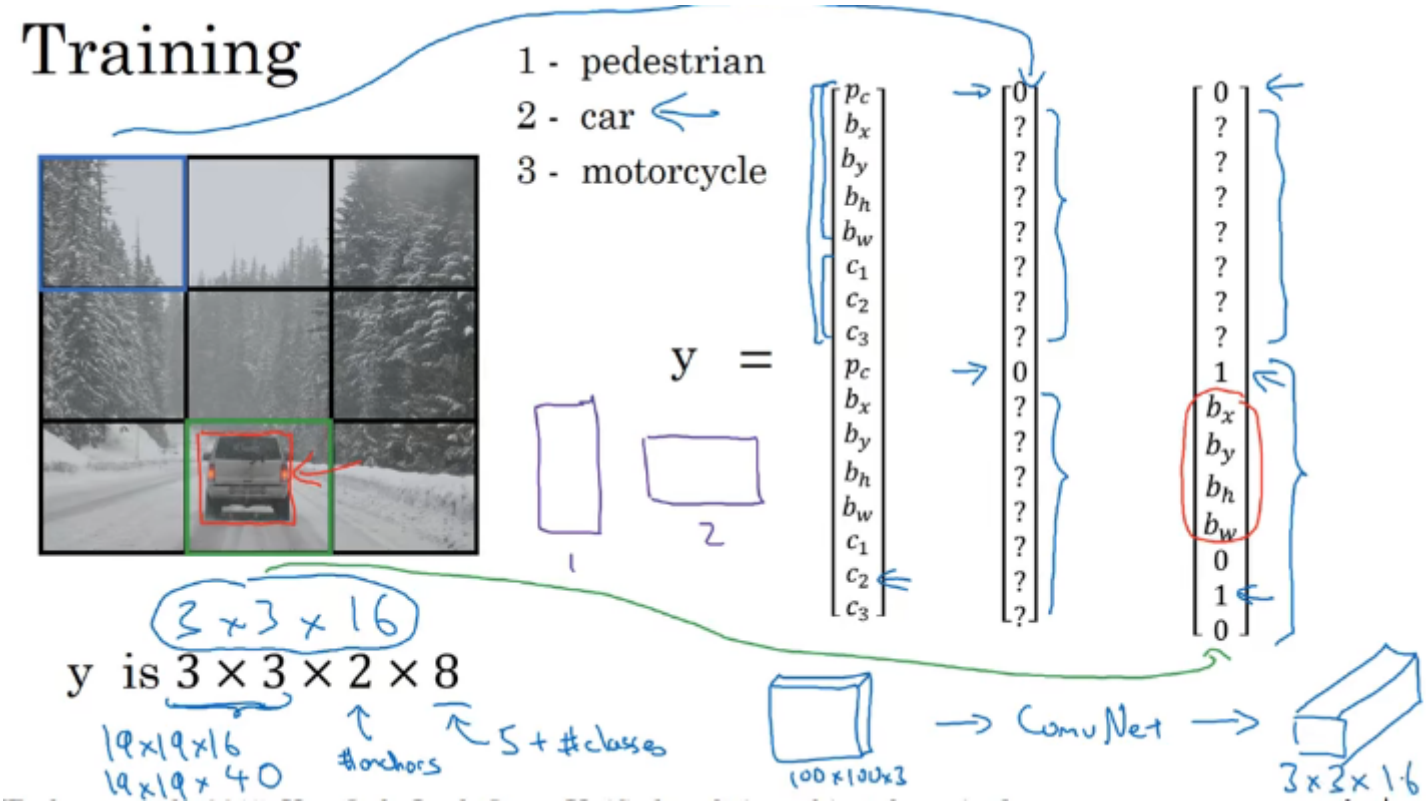
7.4 Anchor box的选择

- 手动指定Anchor box的形状：可以选择5到10个Anchor box覆盖多种形状；
- 使用K-means算法等高级的方法：将不同对象的形状进行聚类，选择最具代表性的Anchor box来代表检测对象的形状。

8. YOLO算法

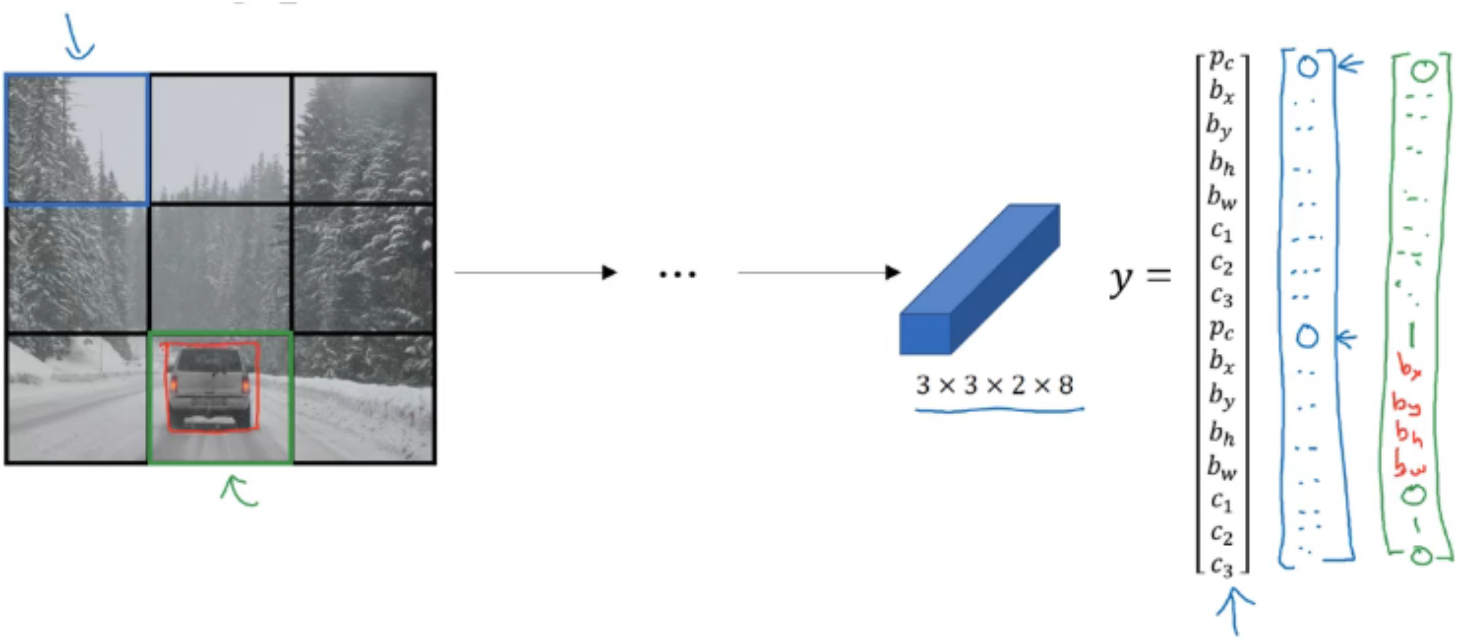
8.1 Training

- 检测目标：行人、汽车、摩托车。
- Input X：图片。
- Output y：使用3×3的网格划分，输出大小为$3 \times 3 \times 16$。



8.2 Predictions

每个格子都会有一个输出的结果。



8.3 Non-max supression

- 如果使用了2个Anchor box，每个格子都会有两个边界框，其中一个边界框的P_c值很低；

- 舍弃P_c值低的边界框；

- 对每个检测对象分别使用NMS得到最终的结果；

9. 候选区域

9.1 R-CNN

R-CNN (Regions with convolutional networks)会在图片中选择候选区域，只在少数的窗口上运行卷积算法，避免了传统的滑动窗口算法在一些无对象区域的无用运算。

具体做法是运行图像分割算法，将图片分割成不同的色块，在这个色块上运行分类器。

9.2 Fast algorithms

- R-CNN：使用某种算法得到候选区域，在每个候选区域上运行分类器，每个区域会输出一个label和bounding box，这样就能在确实存在对象的区域得到一个精确的边界框，但是速度较慢。
- Fast R-CNN：得到候选区域，使用卷积实现的滑动窗口去分类所有的候选区域，但是得到候选区域的聚类步骤仍然非常缓慢。
- Faster R-CNN：使用卷积网络得到候选区域。