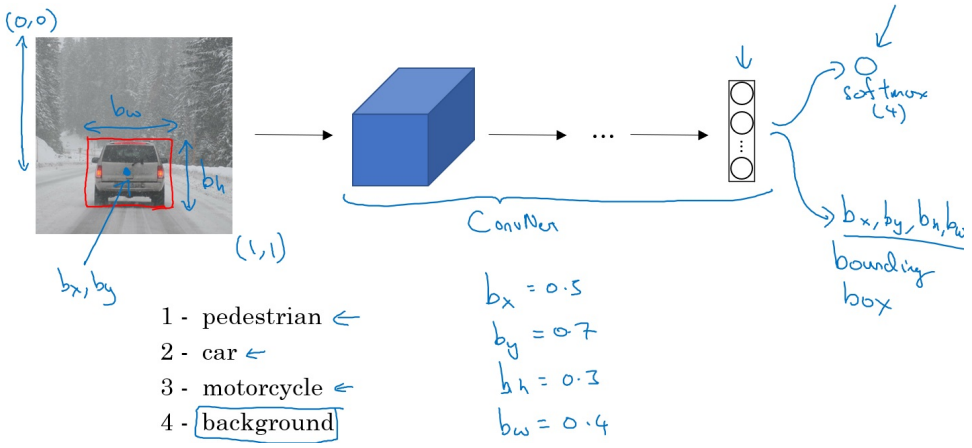
# 1 目标检测

目标检测是计算机视觉领域中一个新兴的应用方向，其任务是对输入图像进行分类的同时，检测图像中是否包含某些目标，并对他们准确定位并标识。

## 1.1 目标定位

图像分类问题一般都采用 Softmax 回归来解决，最后输出的结果是一个多维列向量，且向量的维数与假定的分类类别数一致。在此基础上希望检测其中的包含的各种目标并对它们进行定位，这里对这个监督学习任务的标签表示形式作出定义。




定位分类问题不仅要求判断出图片中物体的种类，还要在图片中标记出它的具体位置，用**边框（Bounding Box，或者称包围盒）**把物体圈起来。一般来说，定位分类问题通常只有一个较大的对象位于图片中间位置；而在目标检测问题中，图片可以含有多个对象，甚至单张图片中会有多个不同分类的对象。如上图所示，分类器将输入的图片分成行人、汽车、摩托车、背景四类，最后输出的就会是一个四维列向量，四个值分别代表四种类别存在的概率。

为了定位图片中汽车的位置，可以让神经网络多输出 4 个数字，标记为 $b_x$、$b_y$、$b_h$、$b_w$。将图片左上角标记为 (0, 0)，右下角标记为 (1, 1)，则有：

- 红色方框的中心点：($b_x$，$b_y$)
- 边界框的高度：$b_h$
- 边界框的宽度：$b_w$

因此，训练集不仅包含对象分类标签，还包含表示边界框的四个数字。定义目标标签 Y 如下：

$$\begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \tag{1}$$

则有：

$$P_c = 1, Y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \tag{2}$$

其中，$c_n$ 表示存在第 n 个种类的概率；如果 $P_c = 0$，表示没有检测到目标，则输出标签后面的 7 个参数都是无效的，可以忽略（用？来表示）。

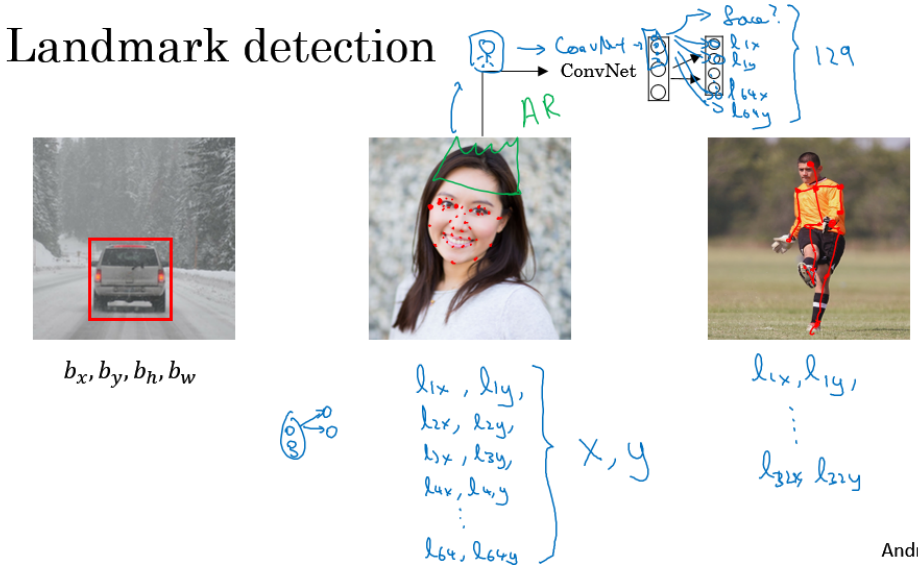
$$P_c = 0, Y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \tag{3}$$

损失函数可以表示为 $L(\hat{y}, y)$，如果使用平方误差形式，对于不同的 $P_c$ 有不同的损失函数（注意下标 i指标签的第 i 个值）：

$$\mathcal{L}(\hat{y}, y) = \begin{cases} (\hat{y_1} - y_1)^2 + (\hat{y_2} - y_2)^2 + \ldots + (\hat{y_n} - y_n)^2, & (y_1(p_c) = 1) \\ (\hat{y_1} - y_1)^2, & (y_1(p_c) = 0) \end{cases} \tag{4}$$

除了使用平方误差，也可以使用逻辑回归损失函数，类标签 $c_1, c_2, c_3$ 也可以通过 softmax 输出。相比较而言，平方误差已经能够取得比较好的效果。

**特征点检测（Landmark detection）**



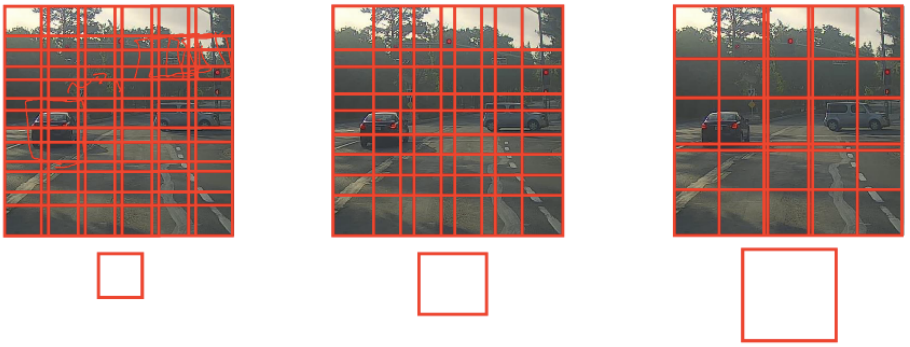

神经网络可以像标识目标的中心点位置那样，通过输出图片上的特征点，来实现对目标特征的识别。在标签中，这些特征点以多个二维坐标的形式表示。

通过检测人脸特征点可以进行情绪分类与判断，或者应用于 AR 领域等等。也可以透过检测姿态特征点来进行人体姿态检测。

## 1.2 滑窗检测

想要实现目标检测，可以采用**基于滑动窗口的目标检测（Sliding Windows Detection）**算法。该算法的步骤如下：

1. 训练集上搜集相应的各种目标图片和非目标图片，样本图片要求尺寸较小，相应目标居于图片中心位置并基本占据整张图片。
2. 使用训练集构建 CNN 模型，使得模型有较高的识别率。
3. 选择大小适宜的窗口与合适的固定步幅，对测试图片进行从左到右、从上倒下的滑动遍历。每个窗口区域使用已经训练好的 CNN 模型进行识别判断。
4. 可以选择更大的窗口，然后重复第三步的操作。

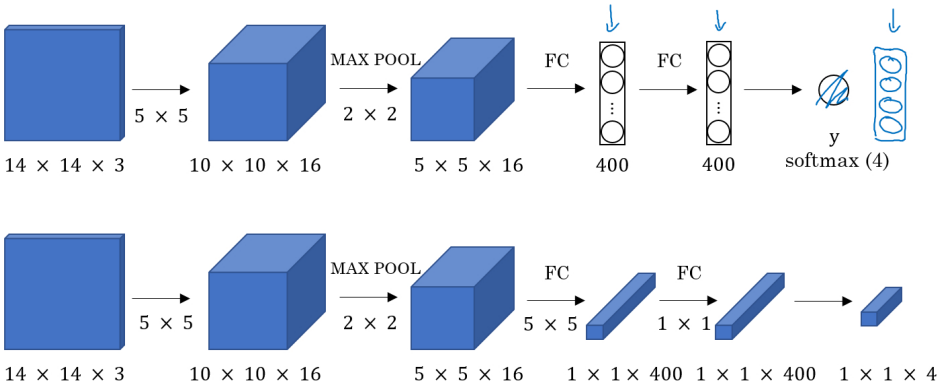


- **优点**是原理简单，且不需要人为选定目标区域；
- **缺点**是需要人为直观设定滑动窗口的大小和步幅。滑动窗口过小或过大，步幅过大均会降低目标检测的正确率。另外，每次滑动都要进行一次 CNN 网络计算，如果滑动窗口和步幅较小，计算成本往往很大。

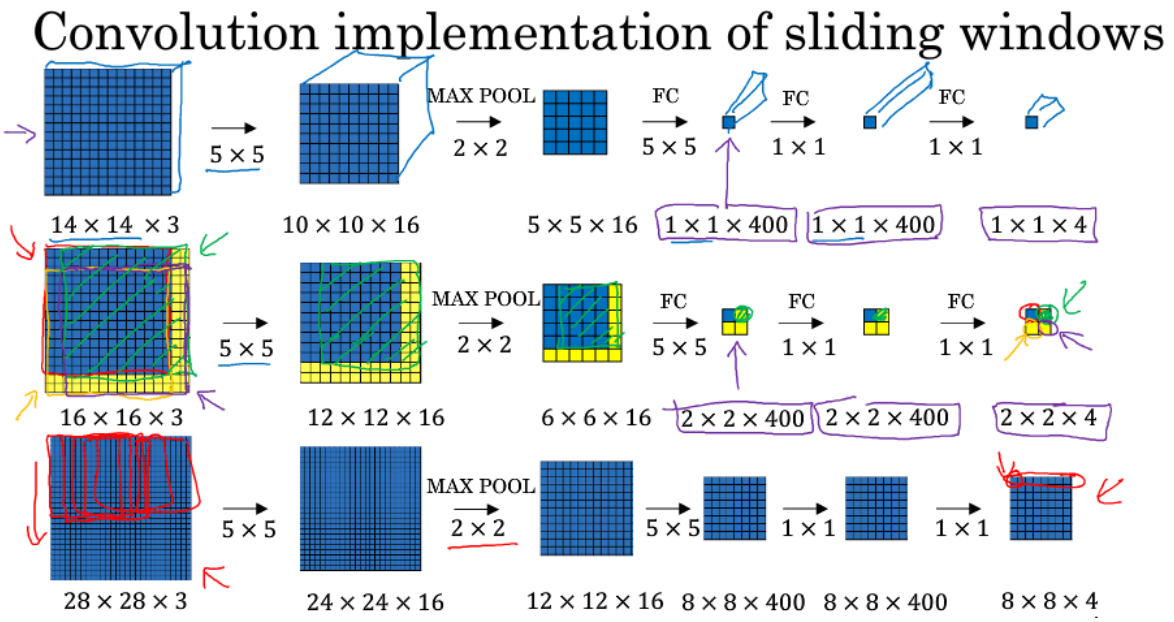
所以，滑动窗口目标检测算法虽然简单，但是性能不佳，效率较低。

**基于卷积的滑动窗口实现**

相比从较大图片多次截取，在卷积层上应用滑动窗口目标检测算法可以提高运行速度。所要做的仅是将全连接层换成卷积层，即使用与上一层尺寸一致的滤波器进行卷积运算。




上图所示为一个卷积神经网络，经过卷积、池化后，全连接过程可以看作是将池化后得到的大小为 5×5×16 的结果与 400 个大小也为 5×5×16 的卷积核分别进行卷积，输出的结果大小为 1×1×400，进一步全连接再采用 Softmax 后，最后输出的结果大小为 1×1×4。由此，全连接过程本质上还是一个卷积过程。




如图，对于 16x16x3 的图片，步长为 2，CNN 网络得到的输出层为 2x2x4。其中，2x2 表示共有 4 个窗口结果。对于更复杂的 28x28x3 的图片，得到的输出层为 8x8x4，共 64 个窗口结果。最大池化层的宽高和步长相等。

运行速度提高的原理：在滑动窗口的过程中，需要重复进行 CNN 正向计算。因此，不需要将输入图片分割成多个子集，分别执行向前传播，而是将它们作为一张图片输入给卷积网络进行一次 CNN 正向计算。这样，公共区域的计算可以共享，以降低运算成本。

这样一个方法，是Sermanet等人2014年在论文 OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks (https://arxiv.org/pdf/1312.6229.pdf) 中提出来的。

## 1.3 YOLO 算法

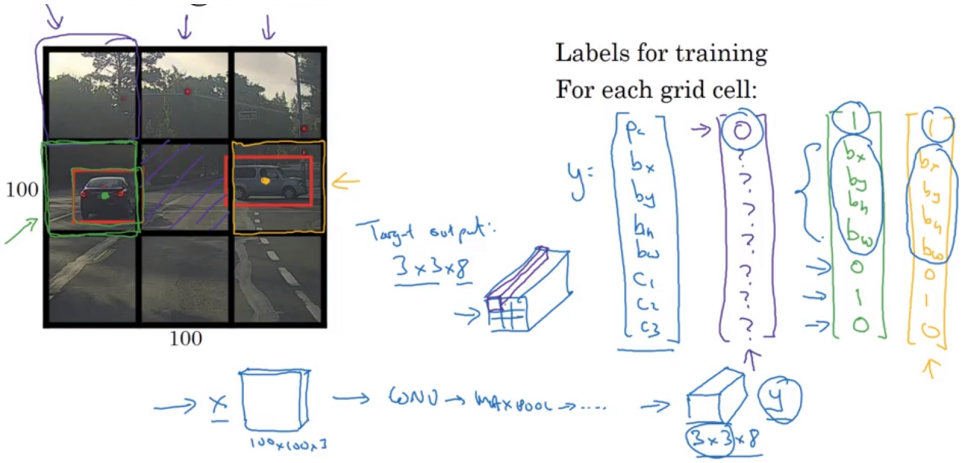
采用滑窗检测进行目标检测，难以选取到一个可以完美匹配目标位置的，大小合适的窗口。

**YOLO（You Only Look Once）**算法是Redmon等人2015年在论文[You Only Look Once: Unified, Real-Time Object Detection]中提出的另一种用于目标检测的算法。YOLO（You Only Look Once）算法可以用于得到更精确的边框。

YOLO 算法中，将输入的图像划分为 S×S 个网格（Grid Cell），对这 S×S 个网格分别指定一个标签，标签的形式如前面所述：

- $p_c$ 标识该网格中的目标存在与否。为“1”则表示存在；“0”则表示不存在，且标签中其他值都无效。
- $b_x$、$b_y$ 表示包围盒的中心坐标值，它们相对于该网格进行了归一化，也就是它们的取值范围在 0 到 1 之间；
- $b_h$、$b_w$ 表示包围盒的长度和宽度；
- $c_n$ 表示第 n 个假定类别存在的概率。

若某个目标的中心点落在某个网格，则该网格负责检测该对象。




如上面的示例中，如果将输入的图片划分为 3×3 的网格、需要检测的目标有 3 类，则每一网格部分图片的标签会是一个 8 维的列矩阵，最终输出的就是大小为 3×3×8 的结果。要得到这个结果，就要训练一个输入大小为 100×100×3，输出大小为 3×3×8 的 CNN。在实践中，可能使用更为精细的 19×19 网格，则两个目标的中点在同一个网格的概率更小。

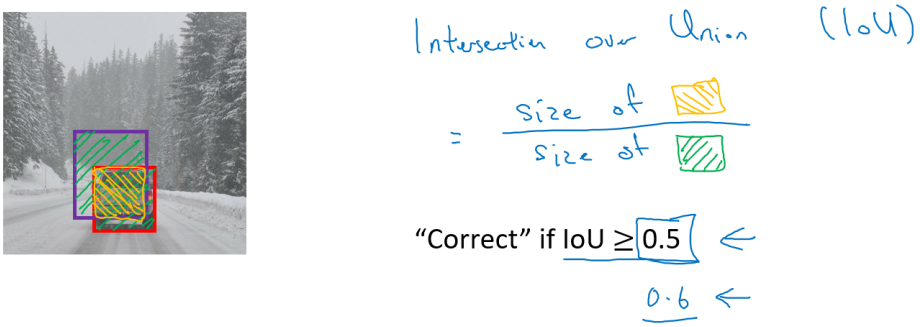
YOLO 算法的优点：

- 和图像分类和目标定位算法类似，显式输出边框坐标和大小，不会受到滑窗分类器的步长大小限制。
- 仍然只进行一次 CNN 正向计算，效率很高，甚至可以达到实时识别。

如何编码边框 $b_x$、$b_y$、$b_h$、$b_w$？YOLO 算法设 $b_x$、$b_y$、$b_h$、$b_w$ 的值是相对于网格长的比例。则 $b_x$、$b_y$ 在 0 到 1 之间，而 $b_h$、$b_w$ 可以大于 1。当然，也有其他参数化的形式，且效果可能更好。这里只是给出一个通用的表示方法。

**交并比**

预测出的目标位置的准确程度用IOU（Intersection Over Union）来衡量，它表示预测出的包围盒（Bounding Box）与实际边界（Ground Truth）的重叠度，也就是两个不同包围盒的交并比。如下图中所示，IOU就等于两个包围盒的交集面积（黄色部分）占两个包围盒的并集面积（绿色部分）的比率。一般可以约定一个阈值，以此判断预测的包围盒的准确与否。




IoU 的值在 0 ~ 1 之间，且越接近 1 表示目标的定位越准确。IoU 大于等于 0.5 时，一般可以认为预测边框是正确的，当然也可以更加严格地要求一个更高的阈值。

**非极大值抑制**

使用 YOLO 算法进行目标检测，因为是多个网格对某些目标同时进行检测，很可能会出现同一目标被多个网格检测到，并生成多个不同的包围盒的情况，这时需要通过非极大值抑制（Non-max Suppression）来筛选出其中最佳的那个。非极大值抑制（Non-max Suppression）会通过清理检测结果，找到每个目标中点所位于的网格，确保算法对每个目标只检测一次。

进行非极大值抑制的步骤如下：

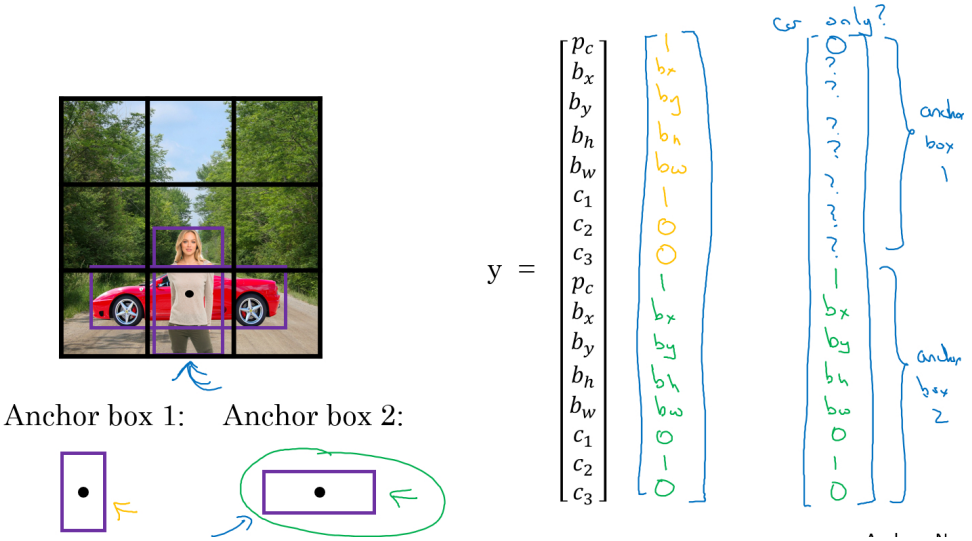
- 将包含目标中心坐标的可信度 $P_c$ 小于阈值（例如 0.6）的网格丢弃；

- 选取拥有最大 $P_c$ 的网格；
- 分别计算该网格和其他所有网格的 IoU，将 IoU 超过预设阈值的网格丢弃；
- 重复第 2~3 步，直到不存在未处理的网格。

上述步骤适用于单类别目标检测。进行多个类别目标检测时，对于每个类别，应该单独做一次非极大值抑制。

**Archor Boxes**

上述算法只适用于单目标检测，也就是每个网格只能检测一个对象。要将该算法运用在多目标检测上，需要用到 **Anchor Boxes**。在原单目标检测所用的标签中加入其他目标的标签，每个目标的标签表示形式都如上所述，一组标签即标明一个 Anchor Box，则一个网格的标签中将包含多个 Anchor Box，相当于存在多个用以标识不同目标的包围盒。




在上图示例中，我们希望同时检测人和汽车。因此，每个网格的的标签中含有两个 Anchor Box。输出的标签结果大小从 3×3×8 变为 3×3×16。若两个 $P_c$ 都大于预设阈值，则说明检测到了两个目标。

在单目标检测中，图像中的目标被分配给了包含该目标中点的那个网格；引入 Anchor Box 进行多目标检测时，图像中的目标则被分配到了包含该目标中点的那个网格以及具有最高 IoU 值的该网格的 Anchor Box。

Anchor Boxes 也有局限性，对于同一网格有三个及以上目标，或者两个目标的 Anchor Box 高度重合的情况处理不好。

Anchor Box 的形状一般通过人工选取。高级一点的方法是用 k-means 将两类对象形状聚类，选择最具代表性的 Anchor Box。

## 1.4 R-CNN

前面介绍的滑动窗口目标检测算法对一些明显没有目标的区域也进行了扫描，这降低了算法的运行效率。为了解决这个问题，R-CNN（Region CNN，带区域的 CNN）被提出。通过对输入图片运行**图像分割算法**，在不同的色块上找出**候选区域（Region Proposal）**，就只需要在这些区域上运行分类器。

R-CNN意为带区域的卷积网络，类似之前所述的滑窗检测算法，先用卷积网络训练一个能够准确识别目标的分类器，但这个算法试图选出一些区域为候选区域，只在这些区域也就是只在少数的窗口上运行分类器。候选区域的选取采用的是一种称为图像分割的算法。R-CNN 的缺点是运行速度很慢，所以有一系列后续研究工作改进。例如 Fast R-CNN（与基于卷积的滑动窗口实现相似，但得到候选区域的聚类步骤依然很慢）、Faster R-CNN（使用卷积对图片进行分割）。不过大多数时候还是比 YOLO 算法慢。

相关论文：

- R-CNN：Girshik et al., 2013. Rich feature hierarchies for accurate object detection and semantic segmentation (https://arxiv.org/pdf/1311.2524.pdf)
- Fast R-CNN：Girshik, 2015. Fast R-CNN (https://arxiv.org/pdf/1504.08083.pdf)
- Faster R-CNN：Ren et al., 2016. Faster R-CNN: Towards real-time object detection with region proposal networks (https://arxiv.org/pdf/1506.01497v3.pdf)