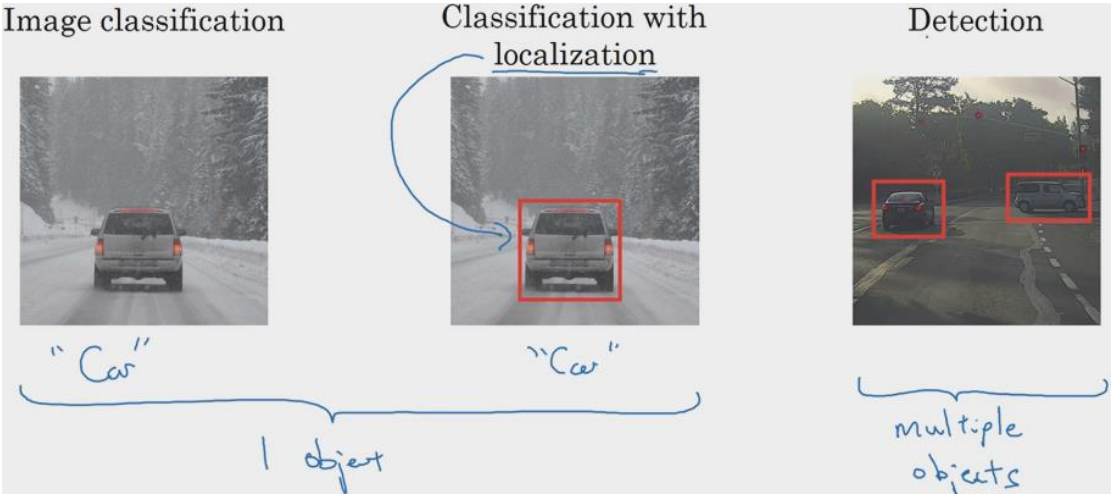
目录

Week3 目标检测 Object Detection

3.1 目标定位 Object Localization

传统的 Image classification 任务是分类输入图片中的内容（如"car"），而 Classification with localization 不仅要输出图片类别，还要用矩形框框出物体所在的位置。这两类问题中，通常仅有一个大的物体处于图片的明显位置。

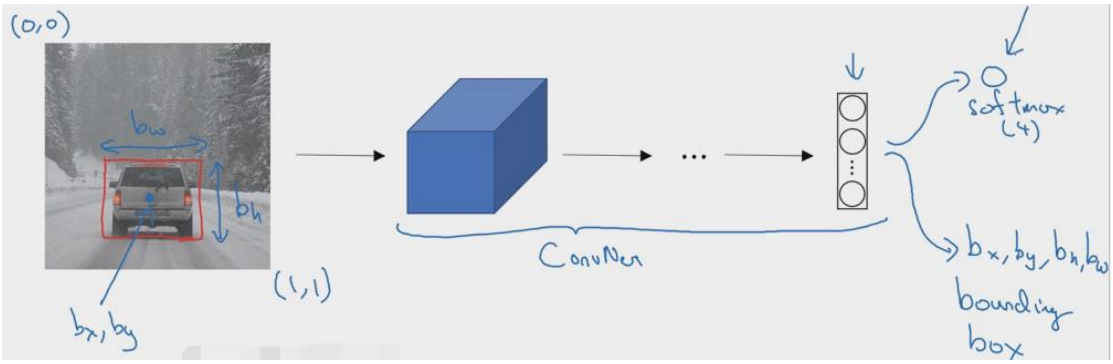
但是 Detection 问题中，图片可以包含多个对象，甚至单张图片会属于多个不同的类别，Detection 的任务不仅是分类图片，还要框出图中的各个物体。



学习 Classification 的思路可以帮助学习分类 Localization，而对象定位的思路又可以帮助学习 Detection。

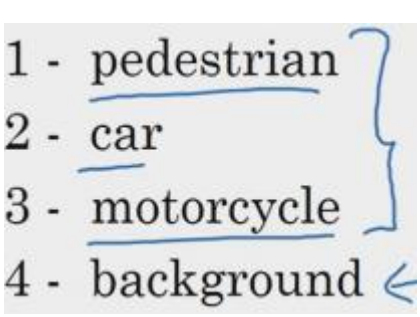
Classification with localization:

传统的图像分类问题模型最后一层是 softmax，输出 y 的各个分量对应每个物体出现的概率；而为了定位物体的位置，本模型在原有基础上，增加了四个额外的输出：bx, by, bh, bw。它们分别代表矩形框中心的横、纵坐标，矩形框的高度和宽度：



Defining the label：

假设图中最多只有一个物体，设 pc 代表图片中是否存在某个类别物体（p_c=1 代表存在，p_c=0 代表图片为 background 不含物体）。则 $y=[p_c\ b_x\ b_y\ b_h\ b_w\ c_1\ c_2\ c_3]^T$。待分类的四种物体如下：




eg.上图中 car 的图片，对应的 label 应该为$[1\ b_x\ b_y\ b_h\ b_w\ 0\ 1\ 0]^T$；而上图中雪地图片对应的 label 应该是$[0\ ?\ ?\ ?\ ?\ ?\ ?\ ?]^T$，其中'?'表示 don't care。

Defining cost function：

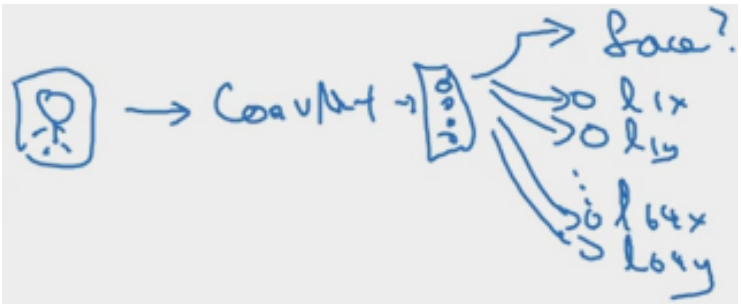
假设使用均方误差，则当 p_c=1 时，当前样本的误差为 y 所有分量的均方误差之和；当 p_c=0 即不存在物体时，误差即为 y_1 的平方：

$$\mathcal{L}(\hat{y},y)=\begin{cases}(\hat{y}_1-y_1)^2+(\hat{y}_2-y_2)^2 & \\ \quad +\cdots+(\hat{y}_8-y_8)^2 & \text{if } y_1=1\\ (\hat{y}_1-y_1)^2 & \text{if } y_1=0\end{cases}$$

3.2 特征点检测 Landmark Detection

神经网络还可以被训练用来输出图片中的特征点（landmark）的坐标。例如在人脸识别例子中，我们可以对人脸上关键的位置（如眼睛、鼻子、嘴巴）设置标记点构造训练集。因为标记点可以用来描绘人脸的特征，从而实现人脸的识别：

假设总共设置 64 个标记点，把图片作为神经网络的输入，通过一些卷积层提取特征后，最后连接 softmax 层。此时 softmax 层应有(1+64*2)个输出（y0 表示是否存在人脸）。其次，使用人们辛苦标注的训练集（图片+Landmarks）来训练这个网络：



目前一些 APP 中，自拍录视频给自己戴皇冠、加耳朵等技术，都是先基于人脸的识别，找出用户整个脸的结构，再在对应位置精确地增加特效。

People post-detection：

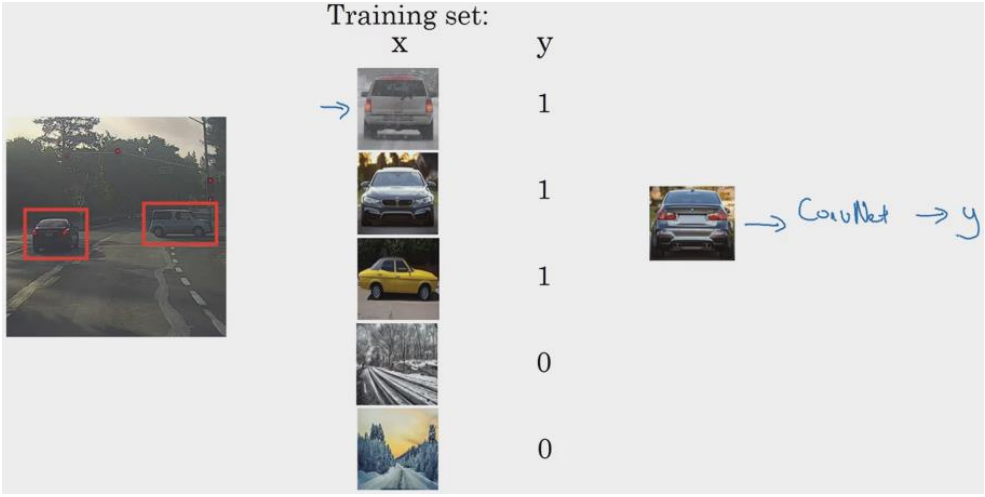
标记一些关键点（如胸口、左右肩、腰、腿等），用同样的方法，把图片和标记点作为网络的输入，并用大量数据对其训练。此后我们就可以用于判断输入图片中人物的姿态了：

3.3 目标检测 Object Detection

汽车检测举例：

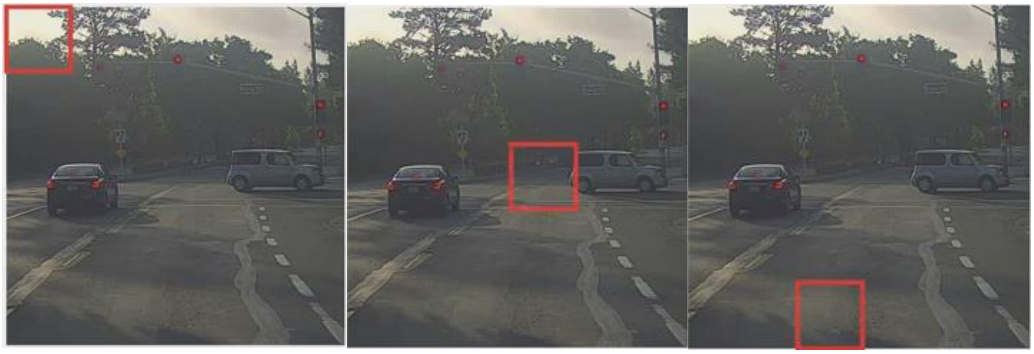
首先，训练集包含许多车的图片，这些图片都是经过较好裁剪并使得车占据图片大部分位置，它们的 label 为 1；其次还有各种非车的图片（如马路），它们的 label 为 0。

其次，把这些图片先输入给卷积网络提取特征，再预测得到 y。**训练**好这个网络之后，就可以用它来实现滑动窗口（Sliding windows）目标检测：



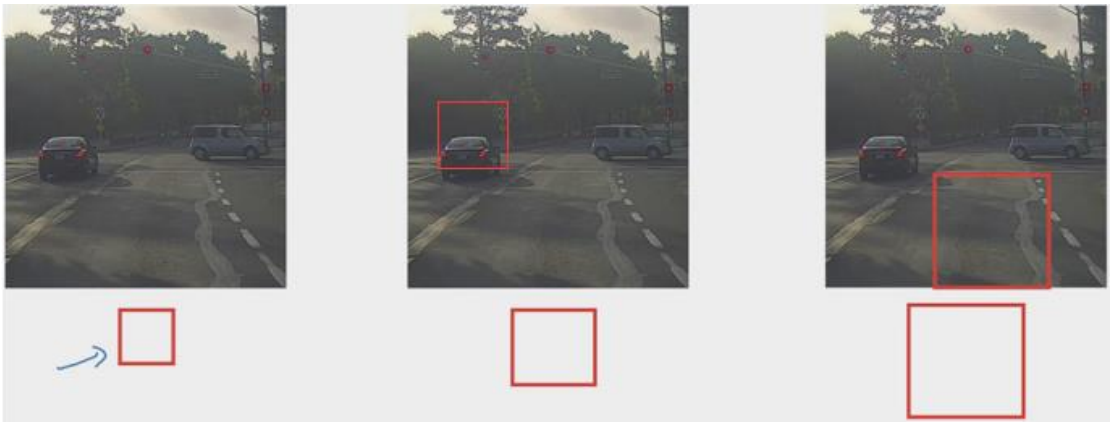
滑动窗口检测：

首先选择一个特定大小的矩形框，然后有点类似 CNN 中 kernel 的滑动，矩形框将从左到右、从上到下地在输入图片上滑动，每次步幅固定，由此遍历图像的每个区域：



每次把截取到的矩形框内的图片作为输入，传给上文训练好的网络，对每个子图片按 01 来分类。

其次，依次选择不同 size 的方框，执行以上同样的操作，遍历整个图片。过程如下：



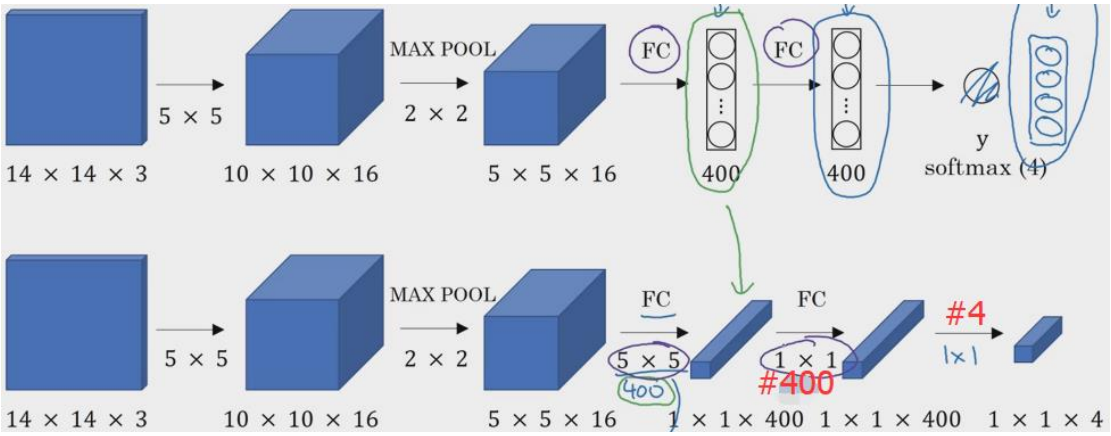
缺陷：计算量太大。因为如果步幅太小，方框不断滑动产生太多的输入；如果步幅太大，虽然计算量会减少，但是粗粒度（coarser granularity）会降低性能。

3.4 滑动窗口的卷积实现

将全连接层转换成卷积层：

原本：输入为 14*14*3 的图片，先经过 16 个 5*5kernel 的卷积，再用 2*2 的池化，得到 5*5*16 的中间结果；把这个张量展开成一个 400 维的长向量，并作为后续 **FC 层**的输入。

现在：前面的步骤不变，当得到 5*5*16 的中间结果后，使用 400 个 5*5*16 的 kernel 进行**卷积**，那么得到的输出直接就是一个 1*1*400 的长向量。从数学上来看，它其实也是类似一个 400 维的全连接层。后面连续使用两次 1×1 卷积，最后我们得到的输出为 1*1*4 的长向量，与上文等价：

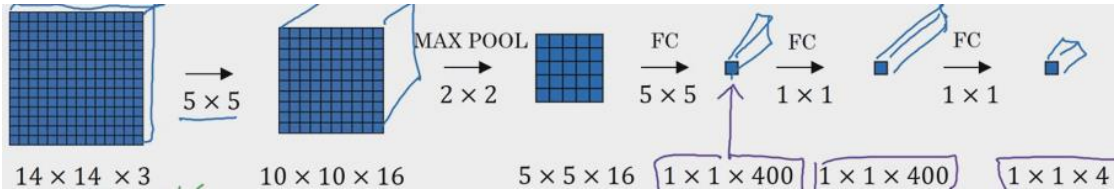


滑动窗口的卷积实现：

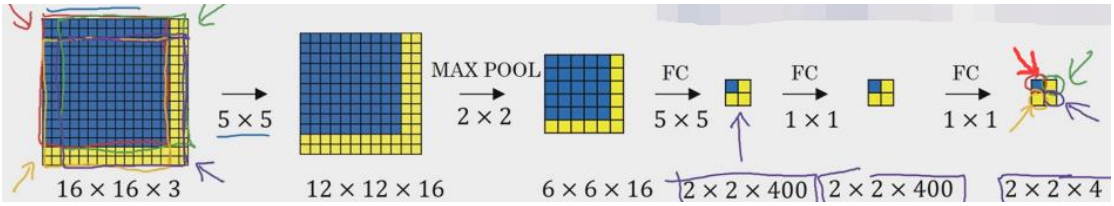
假设用于检测图片的网络输入为 14*14*3，测试的样本原图大小为 16*16*3。设步幅为 2，则窗口总共将在图片上选取 4 个矩形框（左上、右上、左下、右下）。

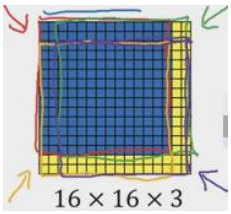
一般的做法是：依次把这 4 个局部区域作为待检测样本送给卷积网络，网络

将产生 4 个对应的输出，分别表示 4 个区域的预测结果。但是这种方法将会产生大量的冗余计算，因为中途有很多卷积的结果是重复可共享的：



于是引出新的做法[1]：直接把整个还没裁剪过的**原图**作为网络的输入，经过同样的卷积、池化步骤。再逐步通过同样的全连接层。不同的是由于输入维度不同，每次中间结果的长宽维度也不同（但通道数量是相同的）：





最终，得到的输出维度为 2*2*4，即总共 4 个输出，分别代表左上、右上、左下、右下四个矩形框的识别结果；每个输出维度为 4，因为文中假设的是四分类问题，每个维度代表相应类别的预测概率。

核心思想：不用把图片分割成许多小区域，分别传给卷积网络来预测；而是直接输入整张图片，因为在不同小区域的公共部分会有许多重复的计算。

拓展：假设原图的长度和宽度分别是 x 和 y，矩形框的长宽是 d，窗口滑动的步幅为 stride。则最后输出的维度是[(x-d)/stride, (y-d)/stride, 4]。

缺陷：边界框的位置可能不太准确

[1]参考文献：Sermanet et al.,2014,OverFeat: Integrated recognition, localization and detection using convolutional networks.

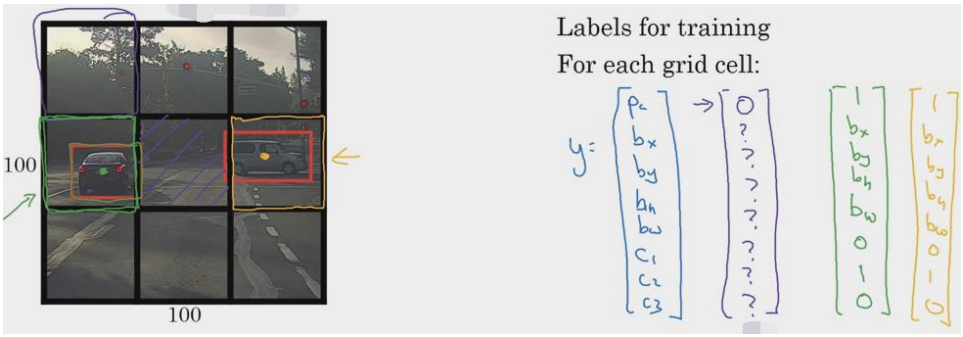
3.5 Bounding Box 预测

上文提到的算法虽然提高了效率，但是缺陷是边界框的位置不太精确。本节将通过预测边界框，来使我们得到更精准的方框。

YOLO 算法：

YOLO[1]（You Only Look Once）是 Redmon 在 2015 提出的算法。它首先将图片均匀划分成一系列网格，然后把每个网格输入 3.1 节的卷积网络（注意此处

是分割图片而非滑动），并预测该网格中是否存在待检测的物体。故下图中对应的 label 如下：



每个网格的输出是一个 8 维向量，图片总共分为 9 个网格，故整张图片对应的输出为 3*3*8。

综上，YOLO 的模型是：输入一张原图，它经过许多卷积层、池化层后，最终输出为 3*3*8（假设图片 9 等分，每个 label 维度为 8）。我们使用 BP 算法在数据集上对其训练。最后，观察输出，output[i][j]为 1 则代表第 i 行第 j 列的网格内检测到有物体，物体的具体类别参考 output[i][j][6:8]，物体的精确位置参考 output[i][j][2:5]。

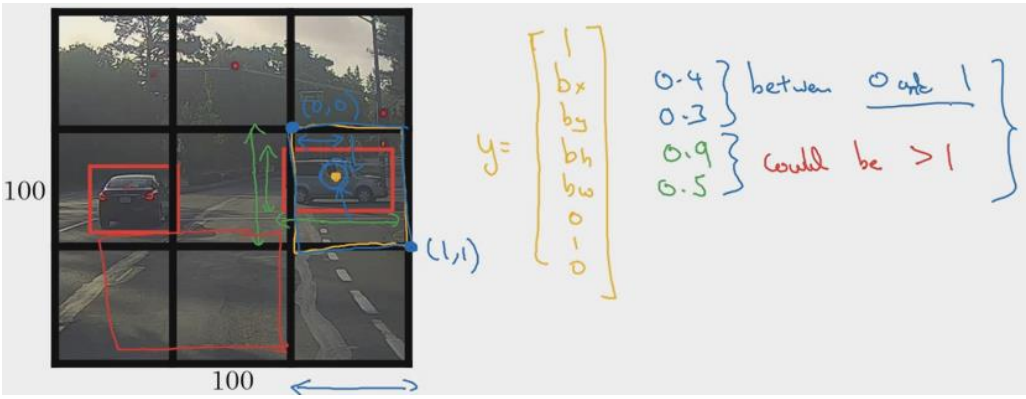
Note：

1.对于一个同时横跨多个网格的物体，我们将找出它 bounding box 的中心，然后仅把它归属于中心所属的网格。

2.本算法用到了 3.1 节的分类器，所以不仅能分类，还能输出精确的位置；其次，本算法是卷积实现（类似 3.4 节），不用将每个网格单独输入，而是一次操作同时得到所有结果，降低了计算的冗余度。而 YOLO 与 3.4 区别是并非滑动采样而是直接分割图片。

指定矩形框：

设 b_x, b_y 分别代表中心点在所属网格中的位置（值 0 到 1），b_h, b_w 代表矩形框高度和宽度与网格边长的比例（可以大于 1，因为物体大小可能超过网格大小）：

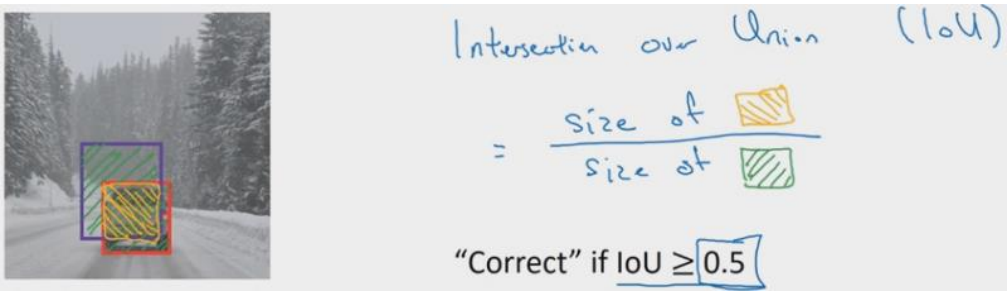


[1]Redmon et al.,2015,You Only Look Once: Unified real-time object detection.

3.6 **交并比** Intersection over union

评估物体的定位：

Intersection 函数（IoU）通过计算两个边界框交集和并集之比，来评估物体定位是否准确。当两个矩形框完全独立时，IoU=0,；当两个矩形框完美重叠时，IoU=1；若 IoU>0.5 则认为检测正确，即通常认为其阈值（threshold）为 0.5，阈值越高则要求越精准：

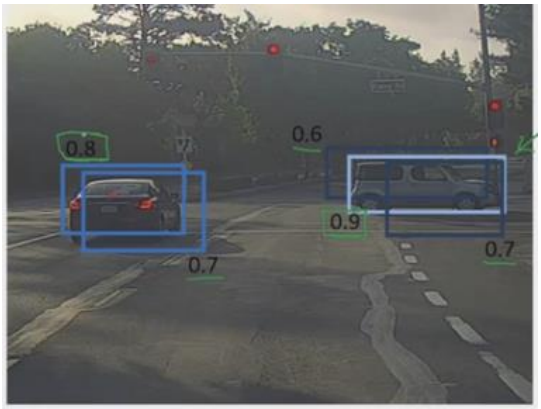


3.7 **非极大值抑制** NMS

目前学到的 Obeject Detection 算法的一个问题是：算法可能对同一个物体做了多次检测。而 NMS 算法可以确保每个对象只被检测一次，从而改善算法性能。

NMS 举例：

假设图片被分成 361 个均匀网格，传统的做法将对每个格子进行检测。那么图中车周围的格子几乎都将有 pc 为 1 的反馈，所以你可能最后会对统一物体检测很多次：



NMS 算法将清理这些检测结果，保证一辆车只会检测一次。具体做法是：

（1）检查所有 p_c 较高的矩形框，找出其中概率值最大（最可靠，如图 0.9）的一个，该矩形框理论上是目前框出物体最完美的一个；

（2）其次再寻找**该物体周围**（即 IoU>0）的框，抑止所有与当前最优框 IoU 较高（高度重叠）的框（如图 0.6，0.7）；

（3）其次，再寻找剩下框中 p_c 概率值较大的那个（或许是完美框处另一辆车的那个框，如图 0.8）；

（4）用第二步同样的方法抑止与其 IoU 较高的框（如图 0.7）；

（5）标记完所有矩形框之后，抛弃被抑制的框，仅保留被高亮的框，即得到最后的预测结果。

NMS 算法：

假设图片被分为 19*19 个矩形框，设输出维度为 5（仅识别车，类别数为 1），则预期输出为 19*19*5。则算法流程如下：

（1）先去除所有 pc<=0.6 的低质量矩形框；

（2）While(存在矩形框未被标记){

·标记其中 p_c 值最大的矩形框

·抑制所有与上步矩形框 IoU>=0.5 的剩余的矩形框 }

Tip：目前讲的 NMS 算法是假设图中只有**单个物体**。如果要进行多物体识别，最好的方法是：在输出 y 中添加额外分量，并对每个分量独立进行多次 NMS 算法。

3.8 Anchor Boxes

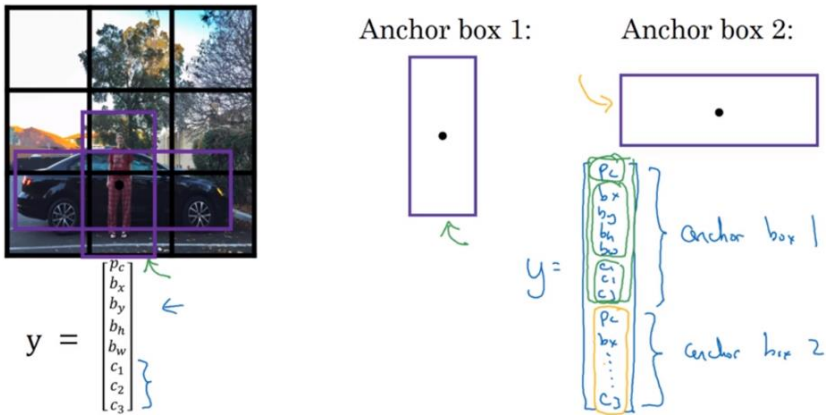
到目前为止，Objection Detection 算法仍然存在的一个问题是：每个格子只能检测出一个对象。要想同时检测出多个物体，可以使用 Anchor Boxes。

重叠的物体：

如图，车和人的中心点几乎重叠，则按原本的规则，它们将都归属与同一个网格。但传统算法每个网格最终只会输出所有待检测物体对应概率最高的那个。

Anchor box 思路：

预先定义两个（实际可更多）形状不同的 anchor box，并把预测结果和这两个 anchor box **关联**起来。由于 box1 的形状与人更吻合，而 box2 的形状与车更吻合，所以规定输出 y 的两部分分别用来预测人和车：



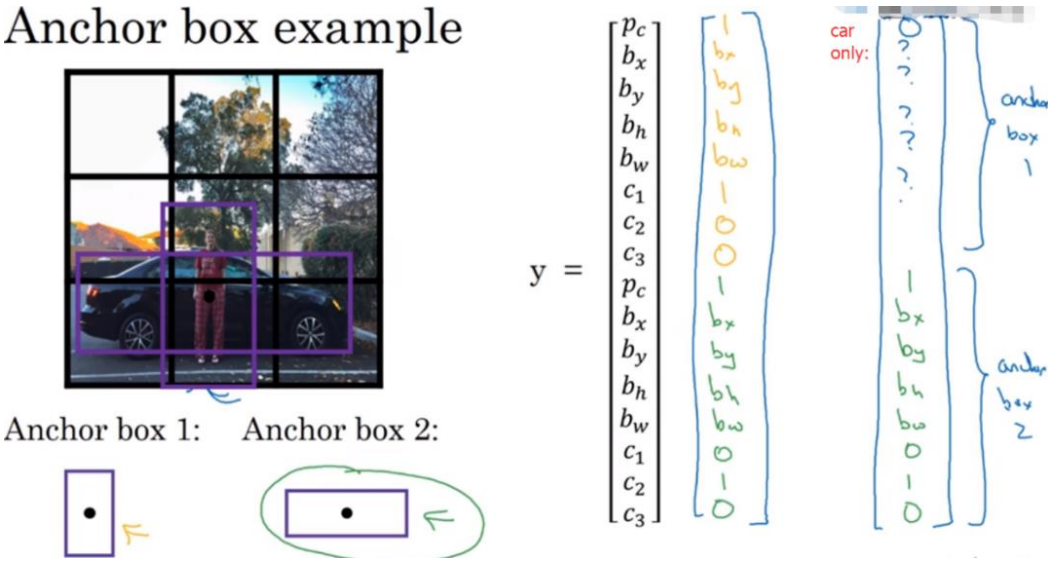
Anchor box 算法：

传统算法：图中的每个物体仅归属于包含该物体中心的那个矩形框内。

2 Anchor box：每个物体既被分配到上述同一个格子中，还被分配给与物体 IoU 最高的那个 anchor box 中（确定物体类别）。故每个物体被分配到一个(grid cell, anchor box)对中，所以现在每个输出是 16 维的。

举例：

图中第三行第二列矩形框对应的输出 y_1，而对于一个仅有车的网格，其输出可能为 y_2：



缺陷：

1.假设只有两个 anchor box，但是网格中同时有 3 类物体，虽然几率很小。

2.两个物体被分配到同一个网格中，并且二者 anchor box 形状也相同。实际上，由于划分图片比较密集，以上两种情况都很少发生。

如何选择 anchor box：

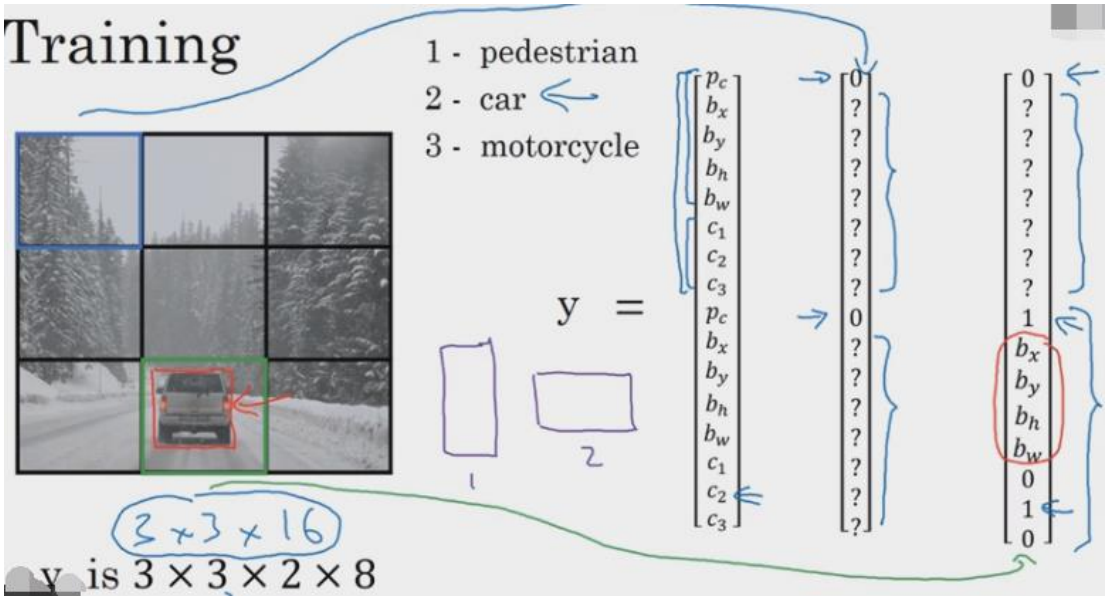
传统：人工选择。

现在：用 K-means 将两类对象类聚，并以它两来选择 box，这样得到的 box 将非常具有代表性。

3.9 YOLO 算法

训练：

设待分类的物体有 3 类（行人，汽车，摩托），anchor box 有两个，图片被分为 3*3 的网格，则整个网络的输出为 3*3*16：



对于蓝色网格 1：由于其内部不含任何物体，则其对应 label 的 p_{c1} 和 p_{c2} 都为 0，其他位全部 don't care。

对于绿色网格 8：假设训练集中对图中汽车有红色方框，假设两个 anchor box 形状分别如图紫色框，由于 box2 与红色框的 IoU 更大，故把汽车分配给训练样本 label 的下半部分。因为我们不仅要区分这里有没有物体，还要区分该物体到底是车还是人。

预测：

对于蓝色网格 1：由于 label 的 p_{c1} 和 p_{c2} 都为 0，故预测该网格中不含任何物体，不做分类。

对于绿色网格 8：由于其 p_{c2}=1，故代表存在物体，且其边框由(b_{x2}, b_{y2}, b_{h2}, b_{w2})指出，且类别由 archor box2 得知是车。

输出 NMS 结果：

（1）对每个网格，得到 2 个预测框（因为 y 是 16 维，前后 8 维分别用 box1 和 box2 的形状框出网格中的物体）；

（2）去除低概率的矩形框；

（3）对每个待识别的物体类别（p_c=i），**独立**地使用 NMS 来产生该方框集群、该类别下最后的预测结果（方框）。即每次 NMS 只操作同一类别的方框（同类物体可能有多个，因为 NMS 会保留同类但隔开的物体）。

综上，YOLO 算法能高效地同时检测出图中各类别的物体的各个对象：

3.10 Region Proposals

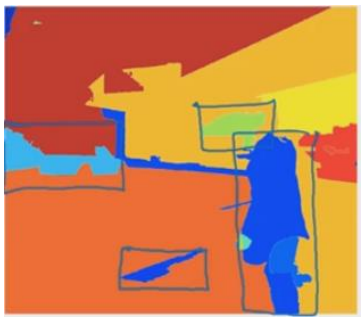
各类 Object detection 算法对比：

（1）传统滑动窗口算法：依次扫描整张图片，缺陷是效率低，进行了大量重复计算。

（2）卷积化的窗口算法：能避免重复计算，但缺点是它会在明显没有任何物体的地方浪费时间，进行计算。

（3）YOLO 算法：使用了 Anchor box，能实现每个网格同时识别多个不同类的物体；并运用 NMS 算法避免了对同一物体的多次检测。

（3）R-CNN[1]：带区域的卷积网络。RPN 网络在物体检测领域也非常有影响力，它会选出一些卷积有意义的区域来计算。而得到候选区域（Region Proposal）的方法是应用图像分割算法（Image Segmentation）。如下，它会选择这些色块对应的网格作为候选区域，并最终用分类器来预测结果。



R-CNN: Propose regions. Classify proposed regions one at a time. Output label + bounding box.

Fast R-CNN: Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions.

Faster R-CNN: Use convolutional network to propose regions.

[1]Girshik et.al, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation.