第七章　目标检测

https://www.toutiao.com/a6548986429299491331/?tt_from=weixin&utm_campaign=client_share&timestamp=1525490580&app=news_article&utm_source=weixin&iid=29839959762&utm_medium=toutiao_android&wxshare_count=1

7.1 基于候选区域的目标检测器

7.1.1 滑动窗口检测器

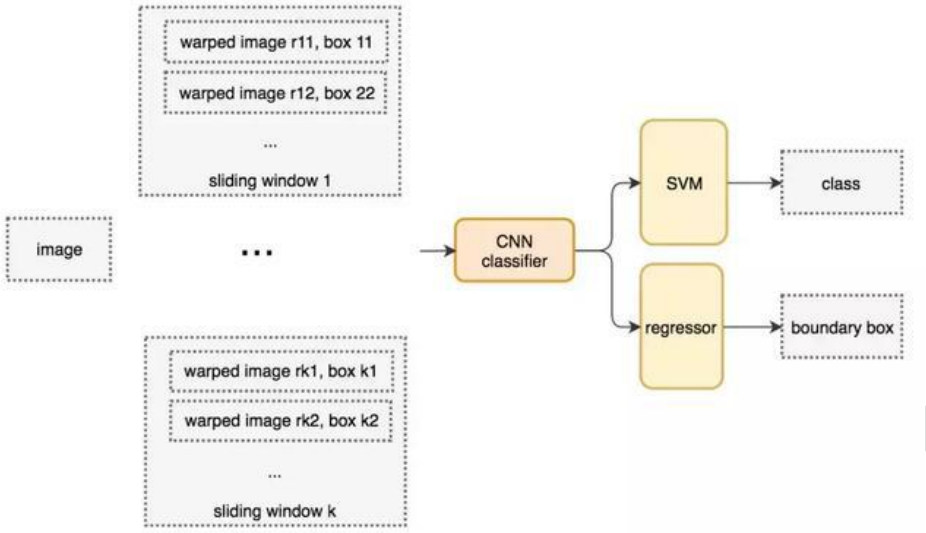
自从 AlexNet 获得 ILSVRC 2012 挑战赛冠军后，用 CNN 进行分类成为主流。一种用于目标检测的暴力方法是从左到右、从上到下滑动窗口，利用分类识别目标。为了在不同观察距离处检测不同的目标类型，我们使用不同大小和宽高比的窗口。

滑动窗口（从右到左，从上到下）

我们根据滑动窗口从图像中剪切图像块。由于很多分类器只取固定大小的图像，因此这些图像块是经过变形转换的。但是，这不影响分类准确率，因为分类器可以处理变形后的图像。

将图像变形转换成固定大小的图像

变形图像块被输入 CNN 分类器中，提取出 4096 个特征。之后，我们使用 SVM 分类器识别类别和该边界框的另一个线性回归器。



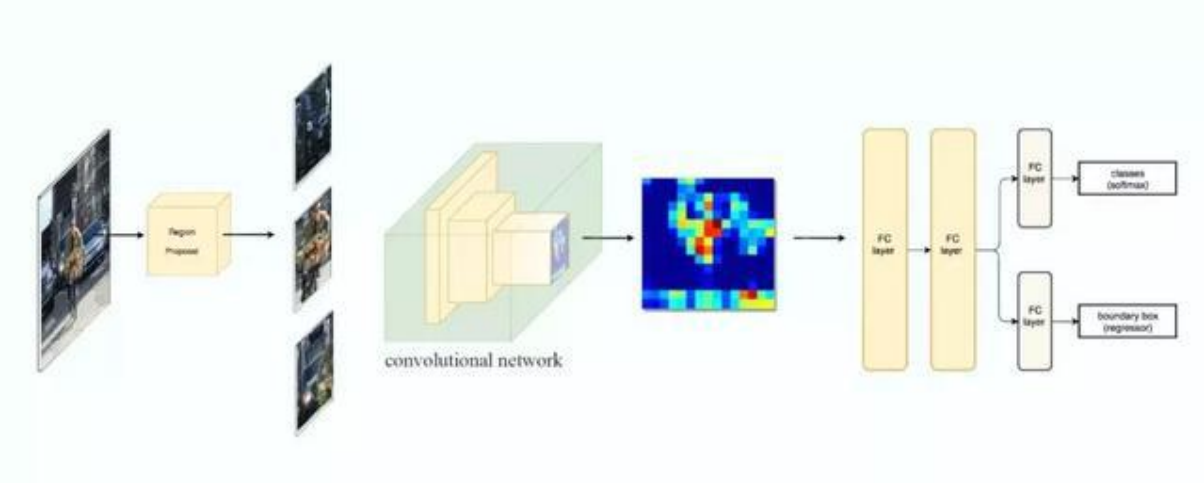
滑动窗口检测器的系统工作流程图。

7.1.2 选择性搜索

我们不使用暴力方法，而是用候选区域方法（region proposal method）创建目标检测的感兴趣区域（ROI）。在选择性搜索（selective search，SS）中，我们首先将每个像素作为一组。然后，计算每一组的纹理，并将两个最接近的组结合起来。但是为了避免单个区域吞噬其他区域，我们首先对较小的组进行分组。我们继续合并区域，直到所有区域都结合在一起。下图第一行展示了如何使区域增长，第二行中的蓝色矩形代表合并过程中所有可能的 ROI。

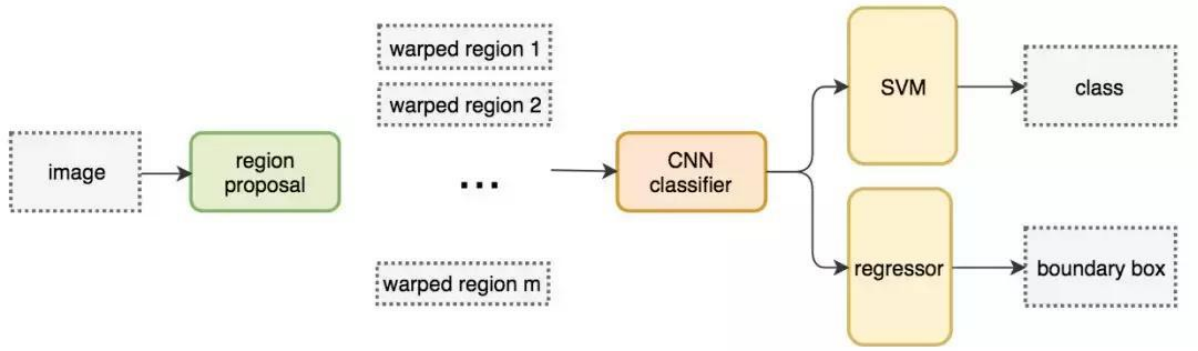
7.1.3 R-CNN

R-CNN 利用候选区域方法创建了约 2000 个 ROI。这些区域被转换为固定大小的图像，并分别馈送到卷积神经网络中。该网络架构后面会跟几个全连接层，以实现目标分类并提炼边界框。



使用候选区域、CNN、仿射层来定位目标。

以下是 R-CNN 整个系统的流程图：



通过使用更少且更高质量的 ROI，R-CNN 要比滑动窗口方法更快速、更准确。

7.1.4 边界框回归器

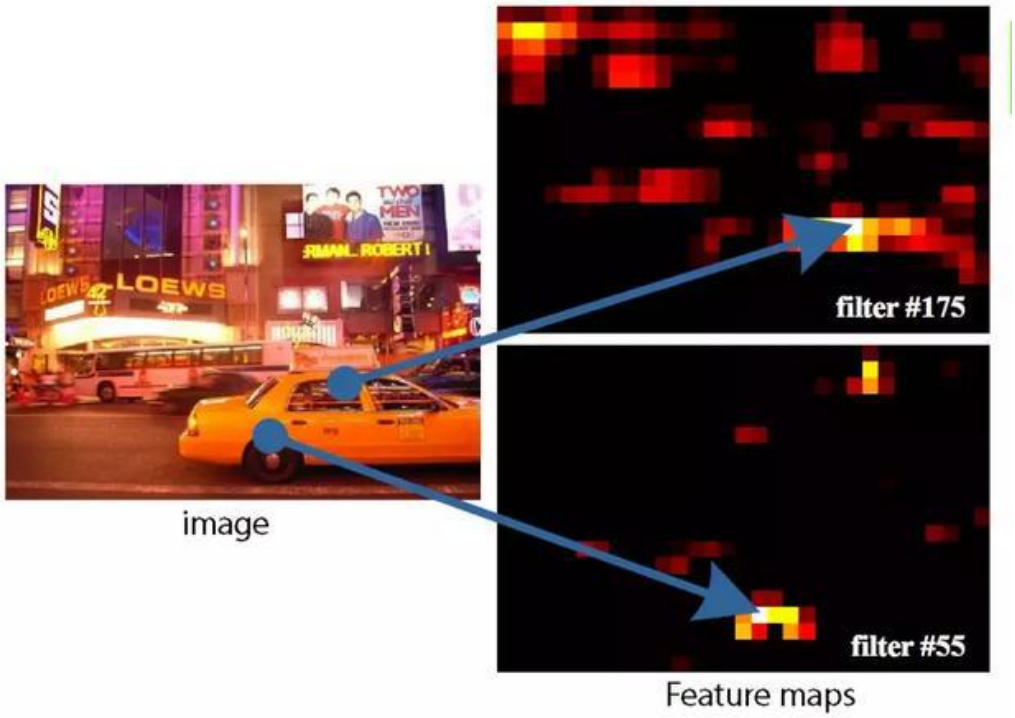
候选区域方法有非常高的计算复杂度。为了加速这个过程，我们通常会使用计算量较少的候选区域选择方法构建 ROI，并在后面使用线性回归器（使用全连接层）进一步提炼边界框。

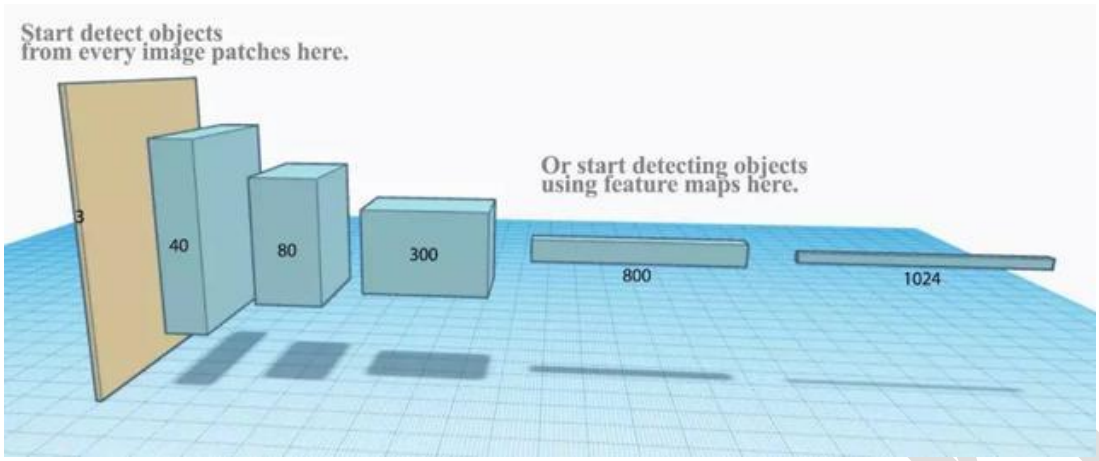
使用回归方法将蓝色的原始边界框提炼为红色的。

7.1.5 Fast R-CNN

R-CNN 需要非常多的候选区域以提升准确度，但其实有很多区域是彼此重叠的，因此 R-CNN 的训练和推断速度非常慢。如果我们有 2000 个候选区域，且每一个都需要独立地馈送到 CNN 中，那么对于不同的 ROI，我们需要重复提取 2000 次特征。

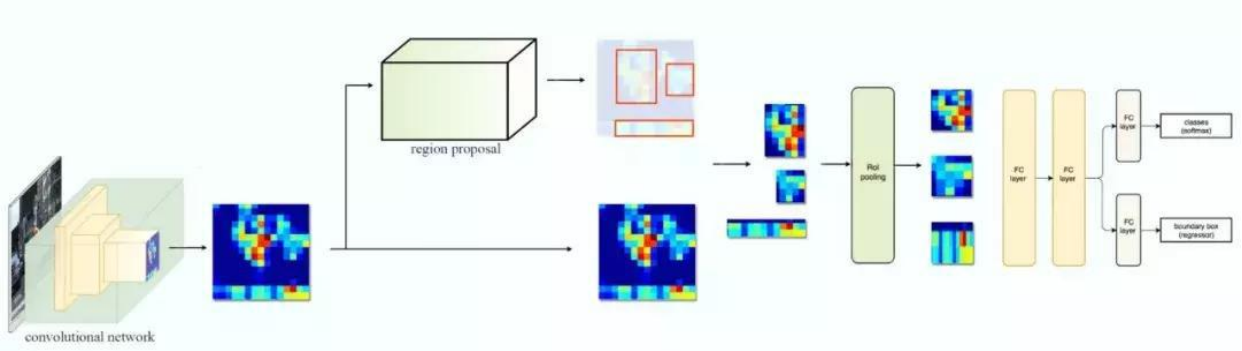
此外，CNN 中的特征图以一种密集的方式表征空间特征，那么我们能直接使用特征图代替原图来检测目标吗？



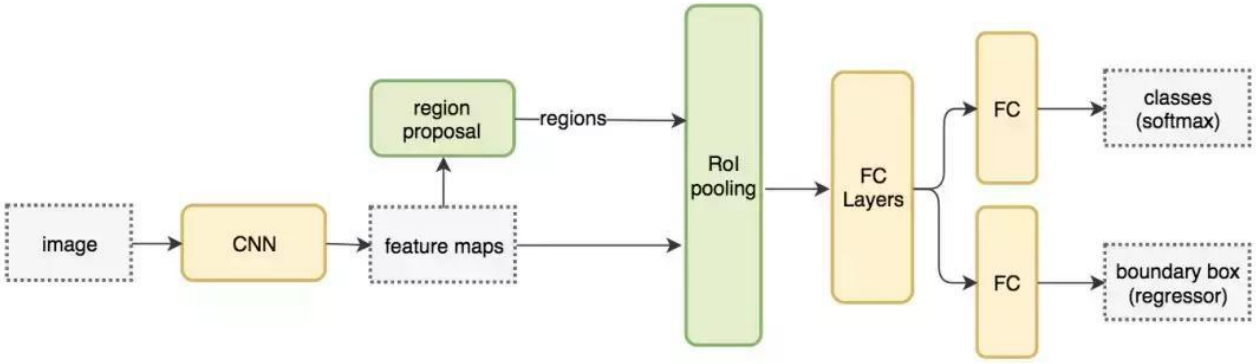


直接利用特征图计算 ROI。

Fast R-CNN 使用特征提取器（CNN）先提取整个图像的特征，而不是从头开始对每个图像块提取多次。然后，我们可以将创建候选区域的方法直接应用到提取到的特征图上。例如，Fast R-CNN 选择了 VGG16 中的卷积层 conv5 来生成 ROI，这些关注区域随后会结合对应的特征图以裁剪为特征图块，并用于目标检测任务中。我们使用 ROI 池化将特征图块转换为固定的大小，并馈送到全连接层进行分类和定位。因为 Fast-RCNN 不会重复提取特征，因此它能显著地减少处理时间。



将候选区域直接应用于特征图，并使用 RoI 池化将其转化为固定大小的特征图块。
以下是 Fast R-CNN 的流程图：



Fast R-CNN 最重要的一点就是包含特征提取器、分类器和边界框回归器在内的整个网络能通过多任务损失函数进行端到端的训练，这种多任务损失即结合了分类损失和定位损失的方法，大大提升了模型准确度。

7.1.6 ROI 池化

因为 Fast R-CNN 使用全连接层，所以我们应用 ROI 池化将不同大小的 ROI 转换为固定大小。

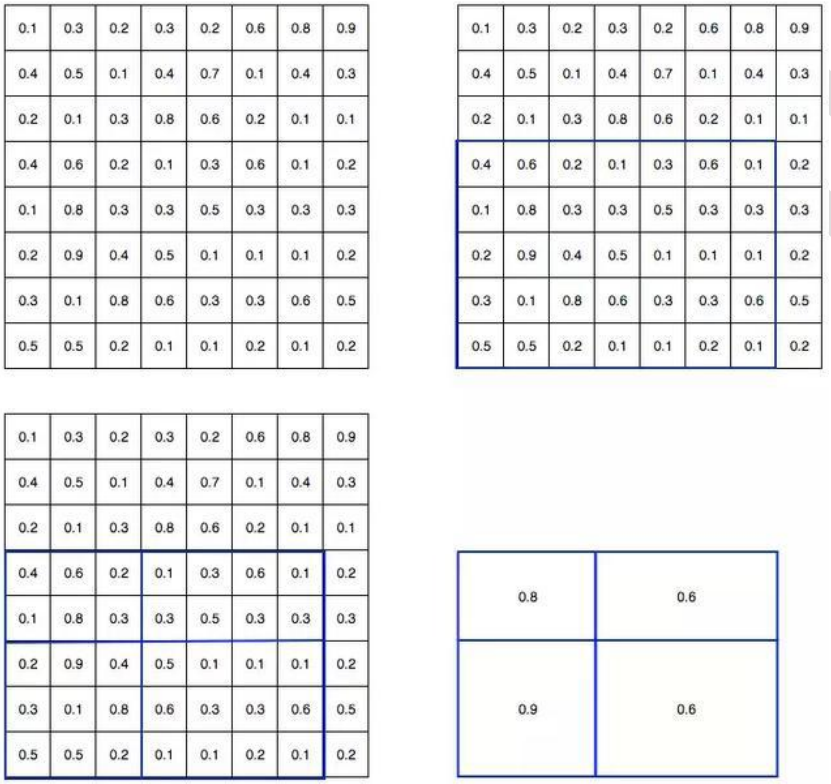
为简洁起见，我们先将 8×8 特征图转换为预定义的 2×2 大小。

下图左上角：特征图。

右上角：将 ROI（蓝色区域）与特征图重叠。

左下角：将 ROI 拆分为目标维度。例如，对于 2×2 目标，我们将 ROI 分割为 4 个大小相似或相等的部分。

右下角：找到每个部分的最大值，得到变换后的特征图。

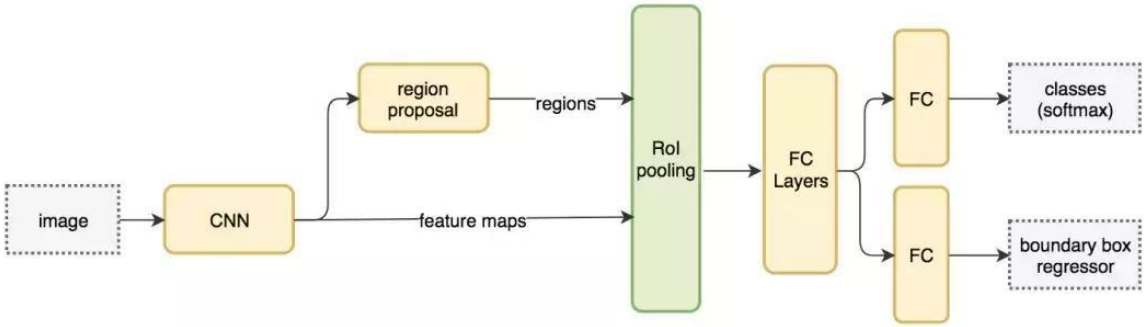


输入特征图（左上），输出特征图（右下），ROI（右上，蓝色框）。 按上述步骤得到一个 2×2 的特征图块，可以馈送至分类器和边界框回归器中。

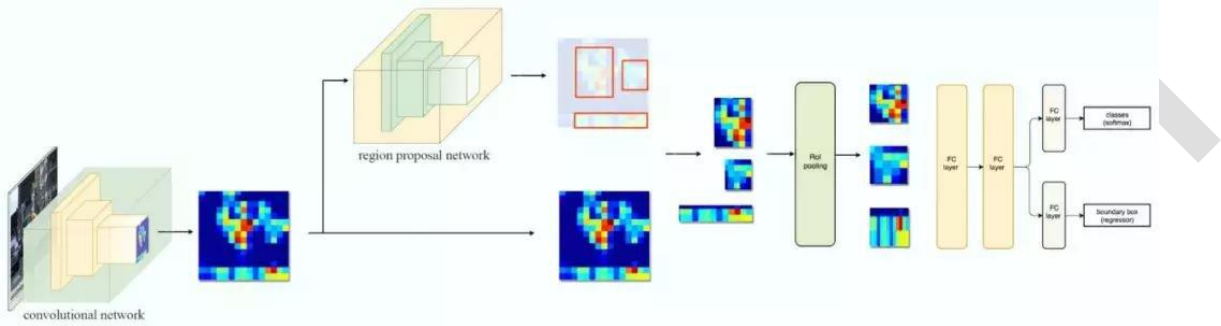
7.1.7 Faster R-CNN

Fast R-CNN 依赖于外部候选区域方法，如选择性搜索。但这些算法在 CPU 上运行且速度很慢。在测试中，Fast R-CNN 需要 2.3 秒来进行预测，其中 2 秒用于生成 2000 个 ROI。

Faster R-CNN 采用与 Fast R-CNN 相同的设计，只是它用内部深层网络代替了候选区域方法。新的候选区域网络（RPN）在生成 ROI 时效率更高，并且以每幅图像 10 毫秒的速度运行。



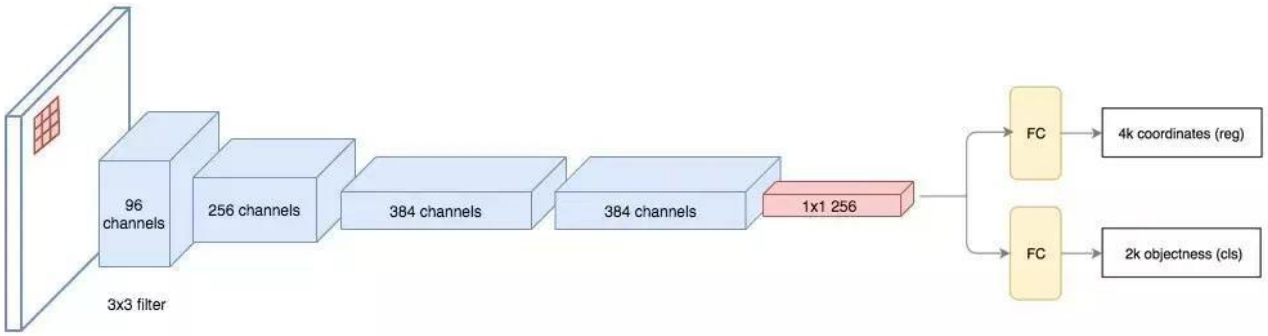
Faster R-CNN 的流程图与 Fast R-CNN 相同



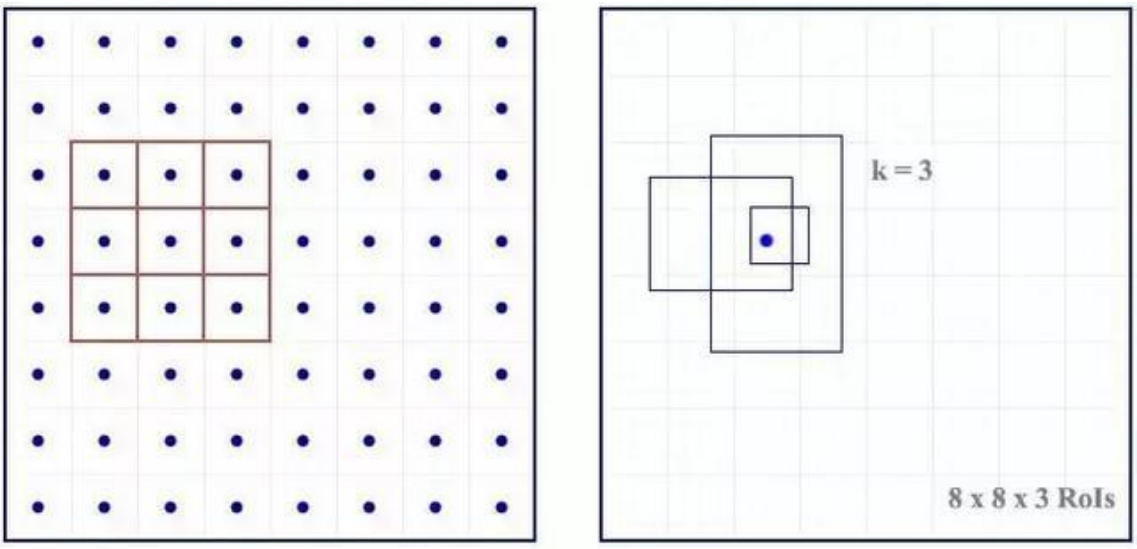
外部候选区域方法代替了内部深层网络。

7.1.8 候选区域网络

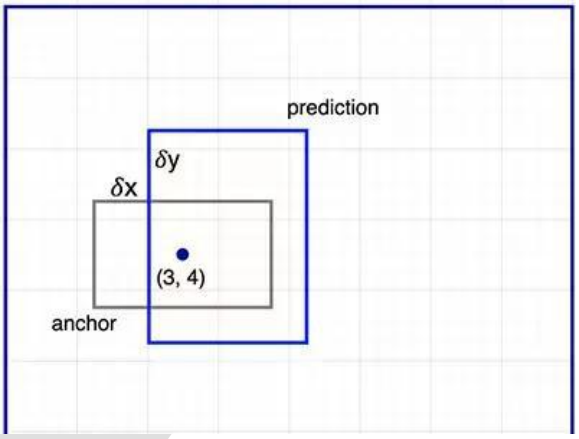
候选区域网络（RPN）将第一个卷积网络的输出特征图作为输入。它在特征图上滑动一个 3×3 的卷积核，以使用卷积网络（如下所示的 ZF 网络）构建与类别无关的候选区域。其他深度网络（如 VGG 或 ResNet）可用于更全面的特征提取，但这需要以速度为代价。ZF 网络最后会输出 256 个值，它们将馈送到两个独立的全连接层，以预测边界框和两个 objectness 分数，这两个 objectness 分数度量了边界框是否包含目标。我们其实可以使用回归器计算单个 objectness 分数，但为简洁起见，Faster R-CNN 使用只有两个类别的分类器：即带有目标的类别和不带有目标的类别。



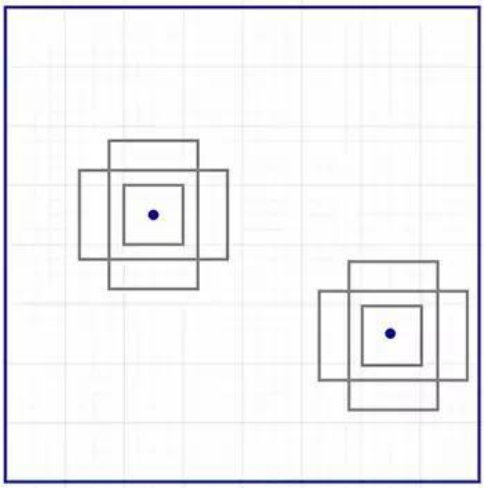
对于特征图中的每一个位置，RPN 会做 k 次预测。因此，RPN 将输出 4×k 个坐标和每个位置上 2×k 个得分。下图展示了 8×8 的特征图，且有一个 3×3 的卷积核执行运算，它最后输出 8×8×3 个 ROI（其中 k=3）。下图（右）展示了单个位置的 3 个候选区域。



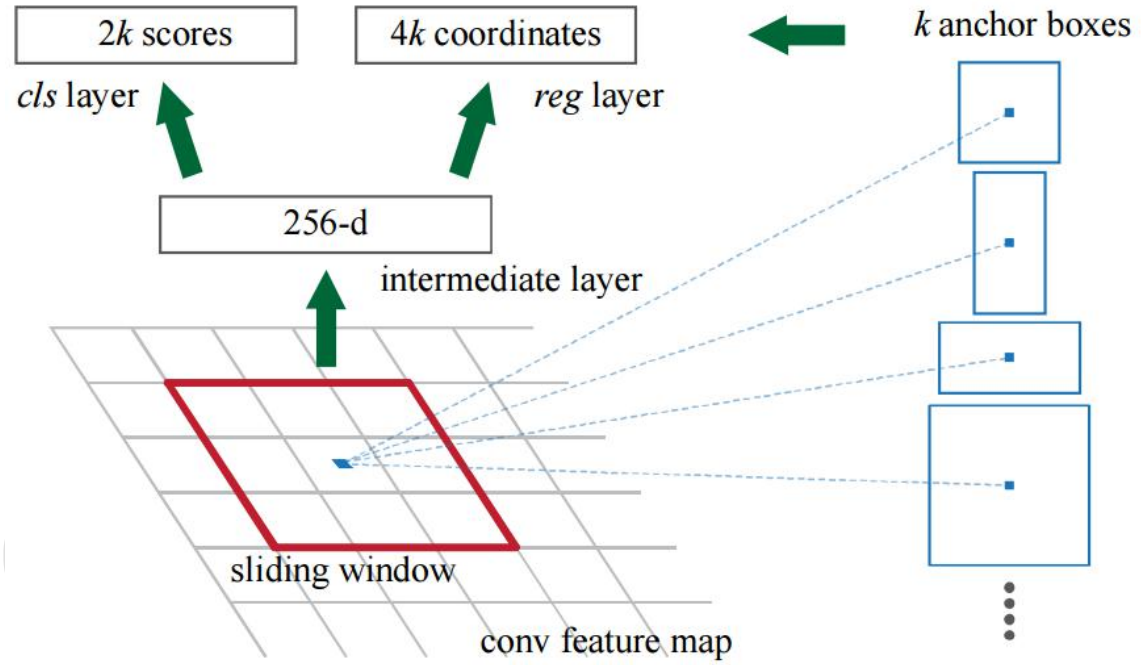
此处有 3 种猜想，稍后我们将予以完善。由于只需要一个正确猜想，因此我们最初的猜想最好涵盖不同的形状和大小。因此，Faster R-CNN 不会创建随机边界框。相反，它会预测一些与左上角名为「锚点」的参考框相关的偏移量（如 x、y）。我们限制这些偏移量的值，因此我们的猜想仍然类似于锚点。



要对每个位置进行 k 个预测，我们需要以每个位置为中心的 k 个锚点。每个预测与特定锚点相关联，但不同位置共享相同形状的锚点。



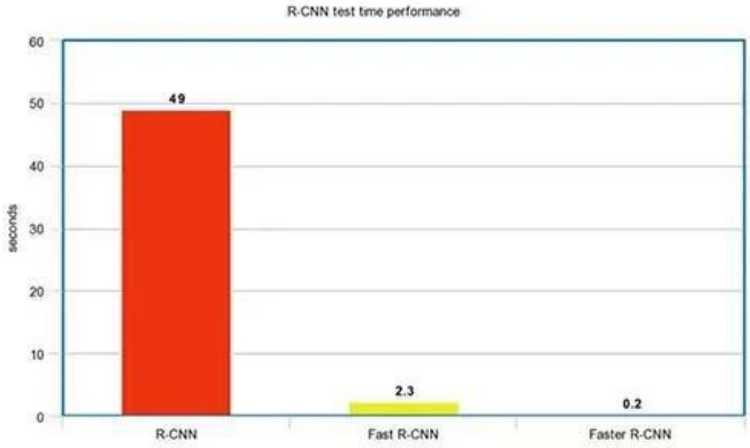
这些锚点是精心挑选的，因此它们是多样的，且覆盖具有不同比例和宽高比的现实目标。这使得我们可以以更好的猜想来指导初始训练，并允许每个预测专门用于特定的形状。该策略使早期训练更加稳定和简便。



Faster R-CNN 使用更多的锚点。它部署 9 个锚点框：3 个不同宽高比的 3 个不同大小的锚点框。每一个位置使用 9 个锚点，每个位置会生成 2×9 个 objectness 分数和 4×9 个坐标。

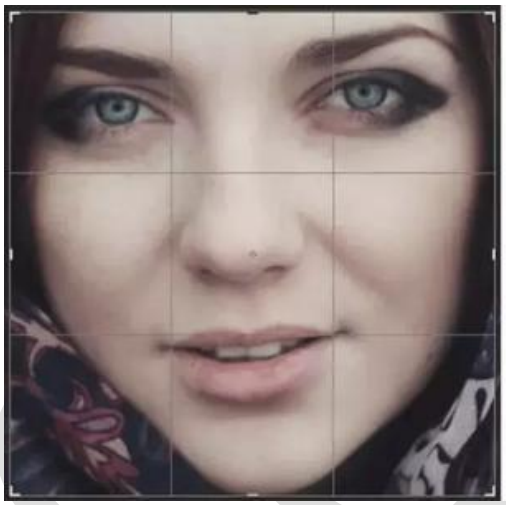
7.1.9 R-CNN 方法的性能

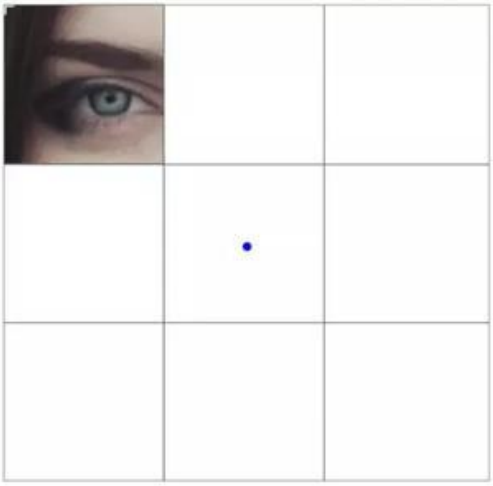
如下图所示，Faster R-CNN 的速度要快得多。



7.2 基于区域的全卷积神经网络（R-FCN）

假设我们只有一个特征图用来检测右眼。那么我们可以使用它定位人脸吗？应该可以。因为右眼应该在人脸图像的左上角，所以我们可以利用这一点定位整个人脸。



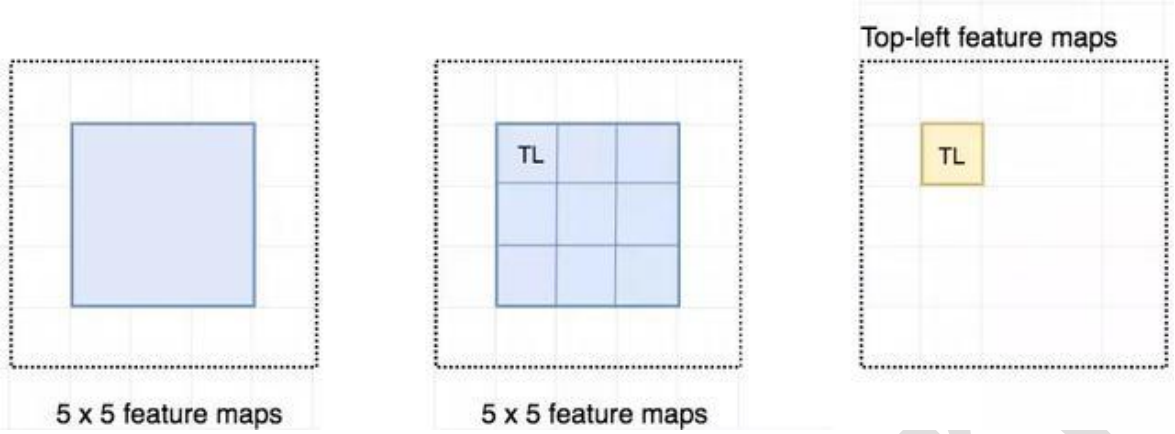


如果我们还有其他用来检测左眼、鼻子或嘴巴的特征图，那么我们可以将检测结果结合起来，更好地定位人脸。

现在我们回顾一下所有问题。在 Faster R-CNN 中，检测器使用了多个全连接层进行预测。如果有 2000 个 ROI，那么成本非常高。

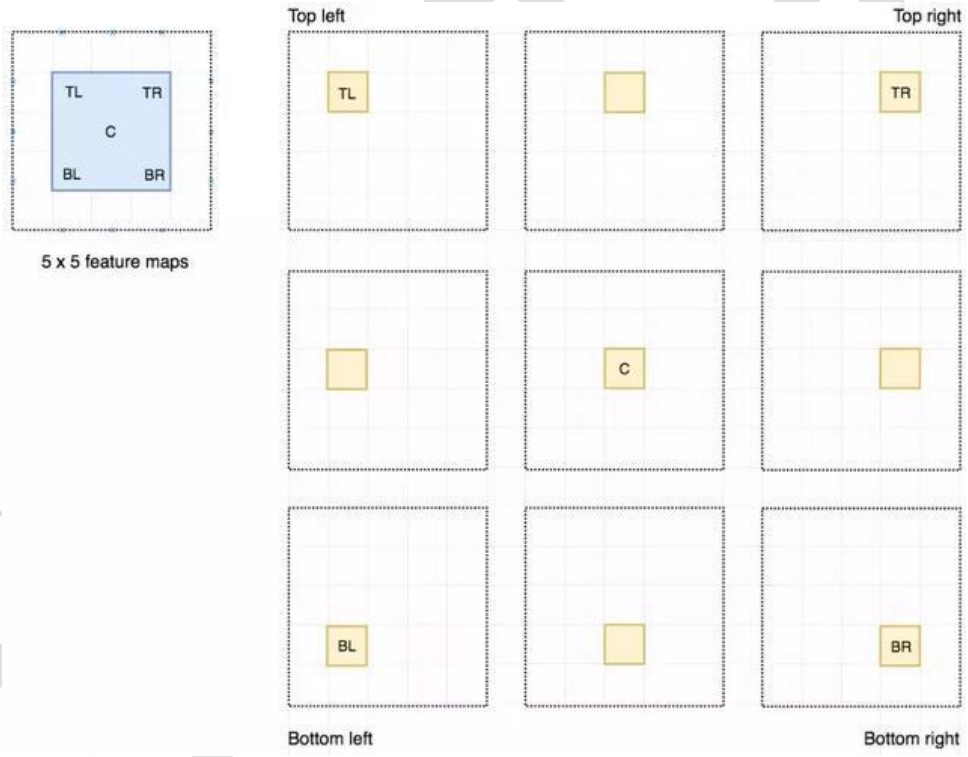
R-FCN 通过减少每个 ROI 所需的工作量实现加速。上面基于区域的特征图与 ROI 是独立的，可以在每个 ROI 之外单独计算。剩下的工作就比较简单了，因此 R-FCN 的速度比 Faster R-CNN 快。

现在我们来看一下 5 × 5 的特征图 M，内部包含一个蓝色方块。我们将方块平均分成 3 × 3 个区域。现在，我们在 M 中创建了一个新的特征图，来检测方块的左上角（TL）。这个新的特征图如下图（右）所示。只有黄色的网格单元 [2, 2] 处于激活状态。



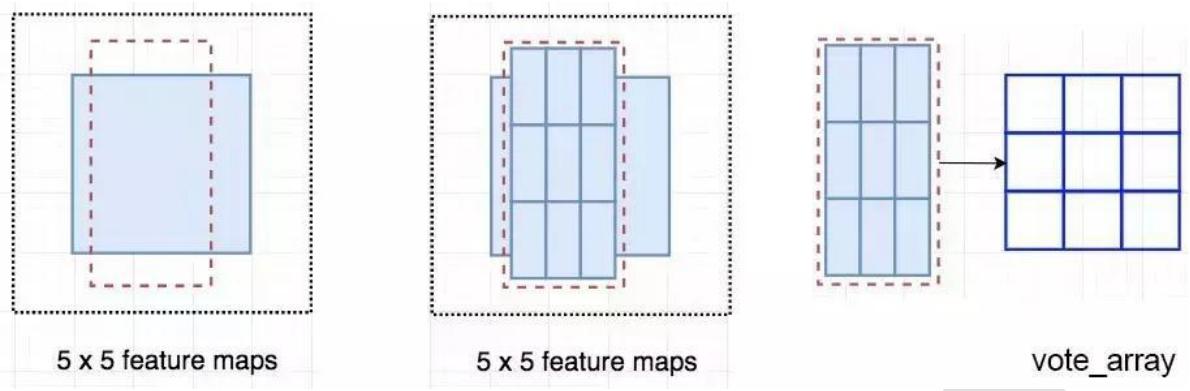
在左侧创建一个新的特征图，用于检测目标的左上角。

我们将方块分成 9 个部分，由此创建了 9 个特征图，每个用来检测对应的目标区域。这些特征图叫作位置敏感得分图（position-sensitive score map），因为每个图检测目标的子区域（计算其得分）。



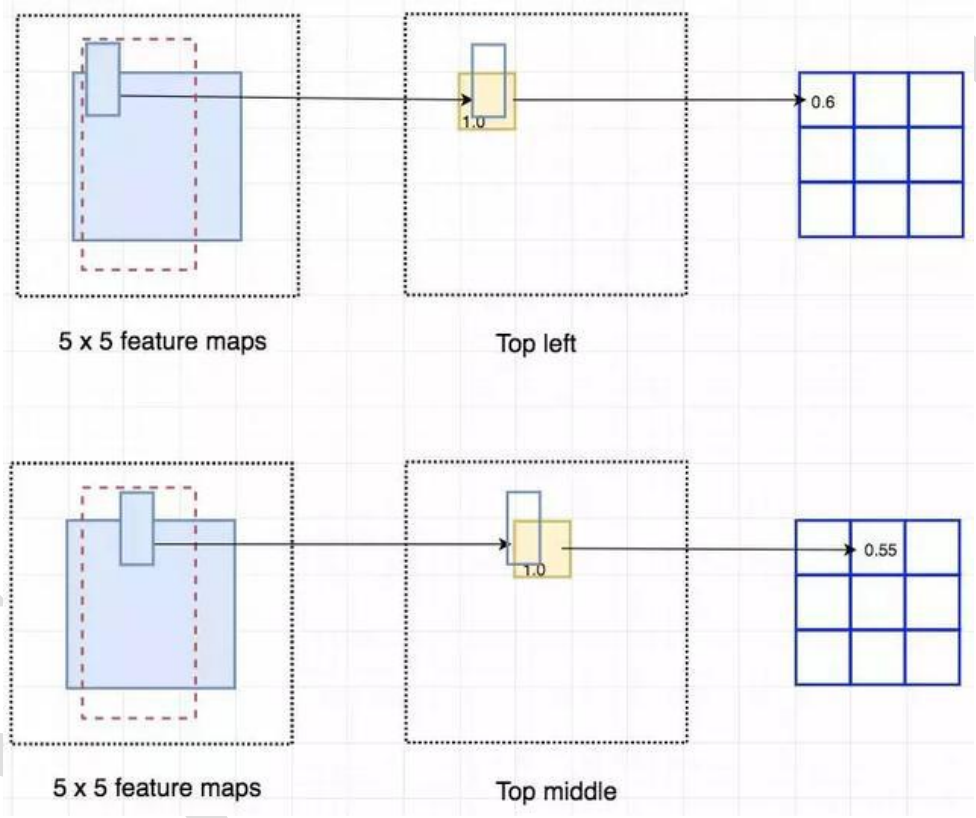
生成 9 个得分图

下图中红色虚线矩形是建议的 ROI。我们将其分割成 3 × 3 个区域，并询问每个区域包含目标对应部分的概率是多少。例如，左上角 ROI 区域包含左眼的概率。我们将结果存储成 3 × 3 vote 数组，如下图（右）所示。例如，vote_array[0][0] 包含左上角区域是否包含目标对应部分的得分。



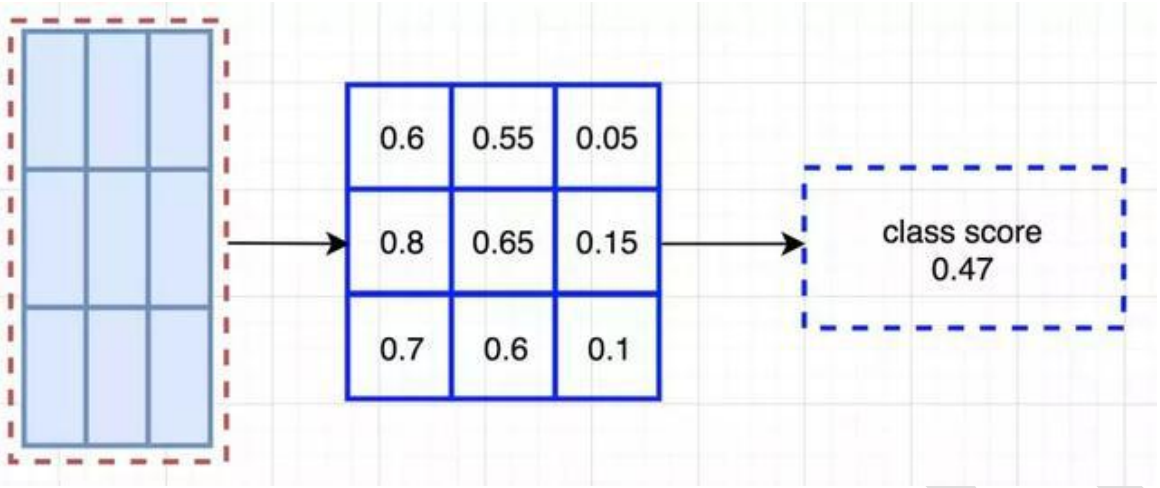
将 ROI 应用到特征图上，输出一个 3 x 3 数组。

将得分图和 ROI 映射到 vote 数组的过程叫作位置敏感 ROI 池化（position-sensitive ROI-pool）。该过程与前面讨论过的 ROI 池化非常接近。



将 ROI 的一部分叠加到对应的得分图上，计算 V[i][j]。

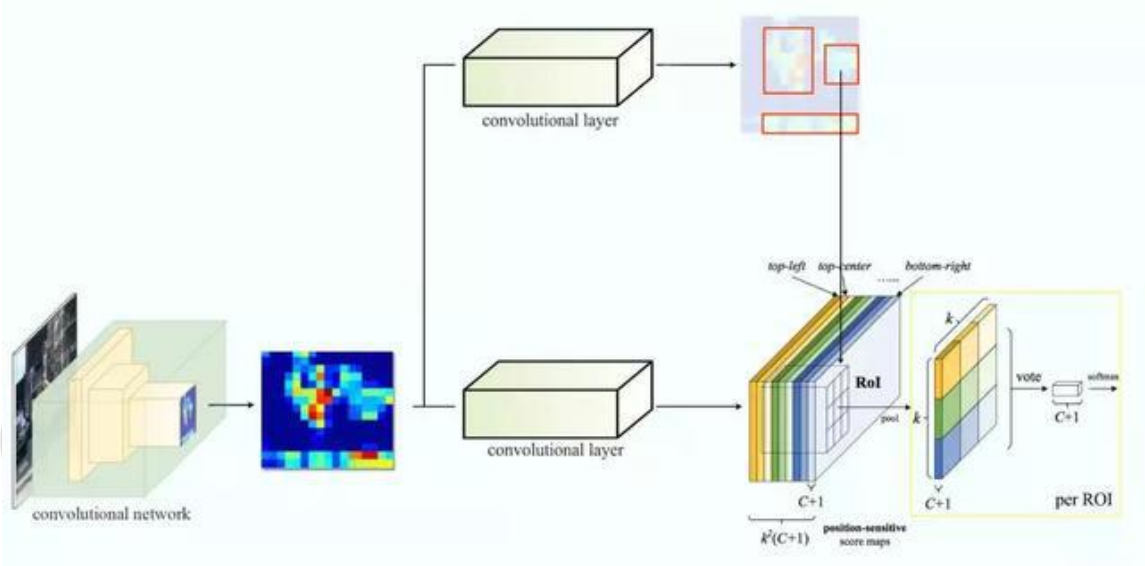
在计算出位置敏感 ROI 池化的所有值后，类别得分是其所有元素得分的平均值。



ROI 池化

假如我们有 C 个类别要检测。我们将其扩展为 C + 1 个类别，这样就为背景（非目标）增加了一个新的类别。每个类别有 3 × 3 个得分图，因此一共有 (C+1) × 3 × 3 个得分图。使用每个类别的得分图可以预测出该类别的类别得分。然后我们对这些得分应用 softmax 函数，计算出每个类别的概率。

以下是数据流图，在我们的案例中，k=3。

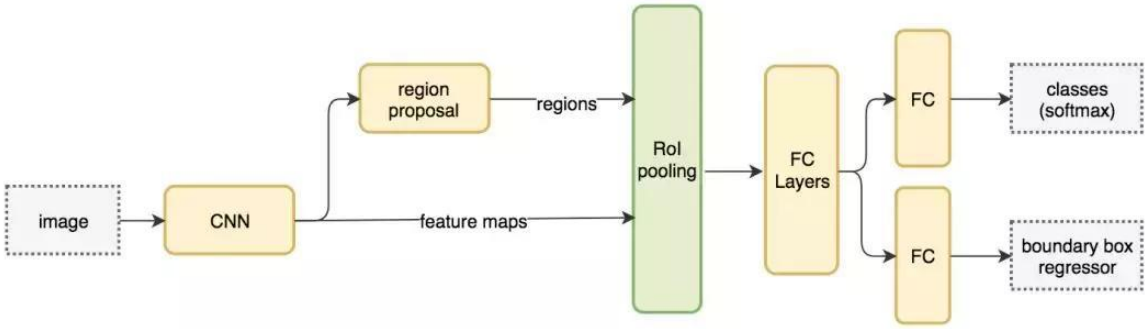


7.3 单次目标检测器

我们将对单次目标检测器（包括 SSD、YOLO、YOLOv2、YOLOv3）进行综述。我们将分析 FPN 以理解多尺度特征图如何提高准确率，特别是小目标的检测，其在单次检测器中的检测效果通常很差。然后我们将分析 Focal loss 和 RetinaNet，看看它们是如何解决训练过程中的类别不平衡问题的。

7.3.1 单次检测器

Faster R-CNN 中，在分类器之后有一个专用的候选区域网络。



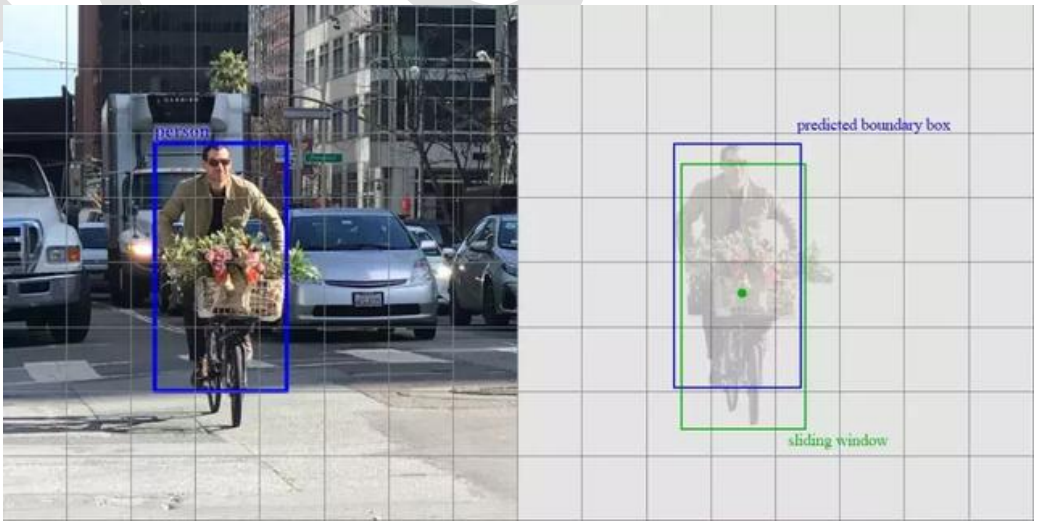
Faster R-CNN 工作流

基于区域的检测器是很准确的，但需要付出代价。Faster R-CNN 在 PASCAL VOC 2007 测试集上每秒处理 7 帧的图像（7 FPS）。和 R-FCN 类似，研究者通过减少每个 ROI 的工作量来精简流程。

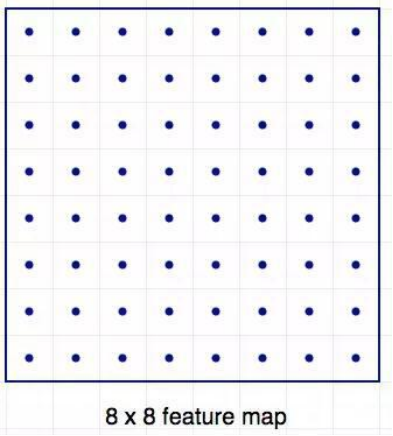
作为替代，我们是否需要一个分离的候选区域步骤？我们可以直接在一个步骤内得到边界框和类别吗？

7.3.2 滑动窗口进行预测

让我们再看一下滑动窗口检测器。我们可以通过在特征图上滑动窗口来检测目标。对于不同的目标类型，我们使用不同的窗口类型。以前的滑动窗口方法的致命错误在于使用窗口作为最终的边界框，这就需要非常多的形状来覆盖大部分目标。更有效的方法是将窗口当做初始猜想，这样我们就得到了从当前滑动窗口同时预测类别和边界框的检测器。

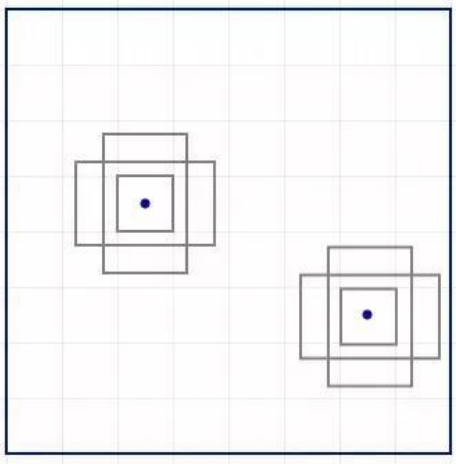


基于滑动窗口进行预测　这个概念和 Faster R-CNN 中的锚点很相似。然而，单次检测器会同时预测边界框和类别。例如，我们有一个 8 × 8 特征图，并在每个位置做出 k 个预测，即总共有 8 × 8 × k 个预测结果。



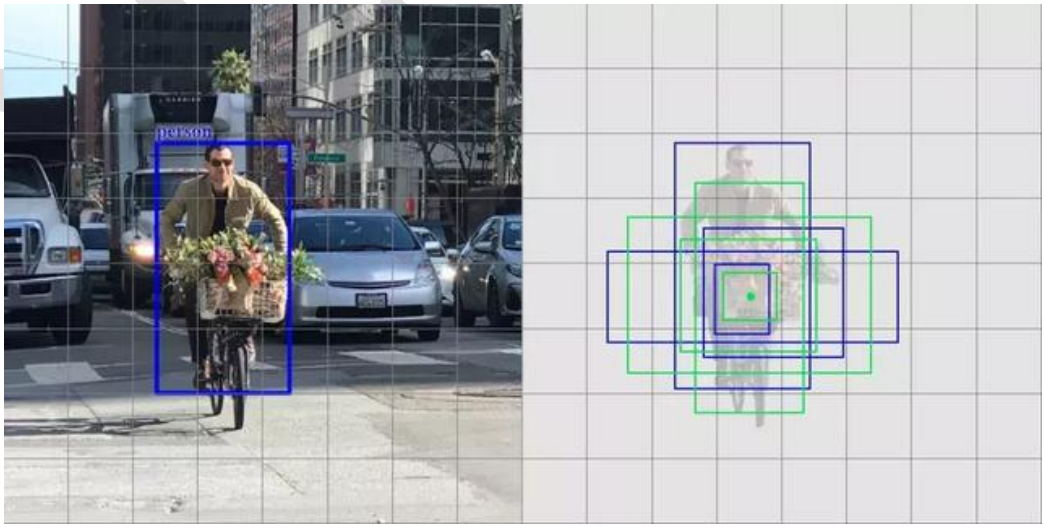
64 个位置

在每个位置，我们有 k 个锚点（锚点是固定的初始边界框猜想），一个锚点对应一个特定位置。我们使用相同的 锚点形状仔细地选择锚点和每个位置。



使用 4 个锚点在每个位置做出 4 个预测。

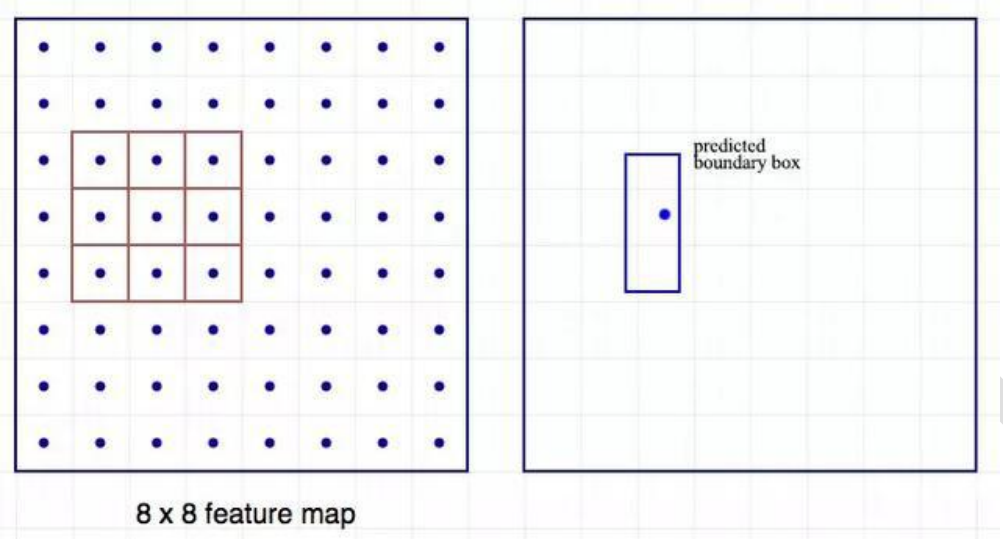
以下是 4 个锚点（绿色）和 4 个对应预测（蓝色），每个预测对应一个特定锚点。



4 个预测，每个预测对应一个锚点。

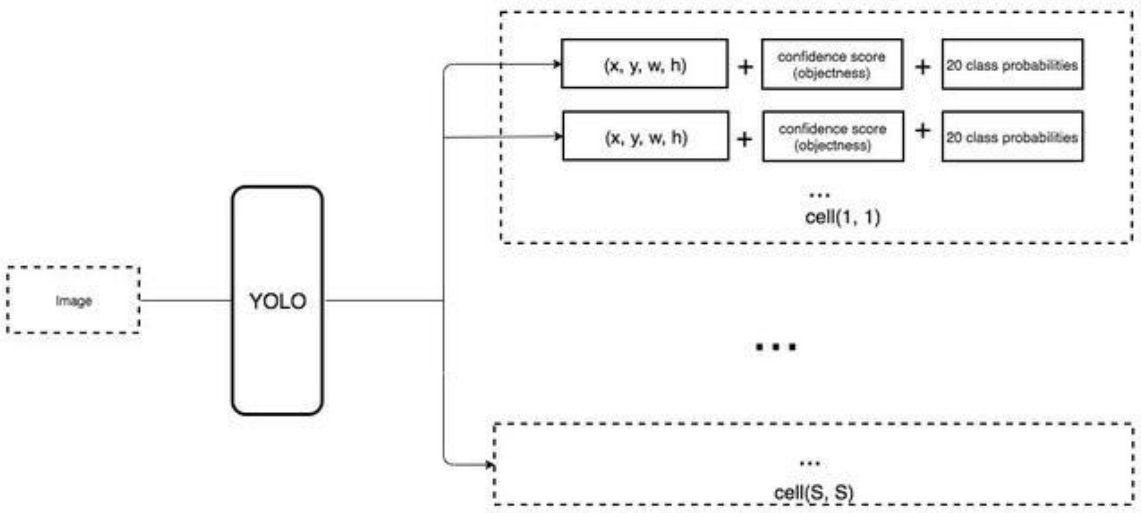
在 Faster R-CNN 中，我们使用卷积核来做 5 个参数的预测：4 个参数对应某个锚点的

预测边框，1 个参数对应 objectness 置信度得分。因此 3× 3× D × 5 卷积核将特征图从 8 × 8 × D 转换为 8 × 8 × 5。



使用 3x3 卷积核计算预测。

在单次检测器中，卷积核还预测 C 个类别概率以执行分类（每个概率对应一个类别）。因此我们应用一个 3× 3× D × 25 卷积核将特征图从 8 × 8 × D 转换为 8 × 8 × 25（C=20）。

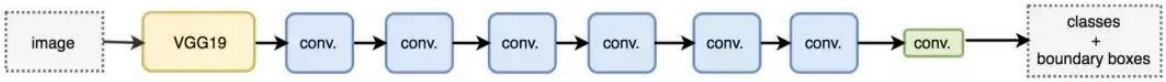


每个位置做出 k 个预测，每个预测有 25 个参数。

单次检测器通常需要在准确率和实时处理速度之间进行权衡。它们在检测太近距离或太小的目标时容易出现问题。在下图中，左下角有 9 个圣诞老人，但某个单次检测器只检测出了 5 个。

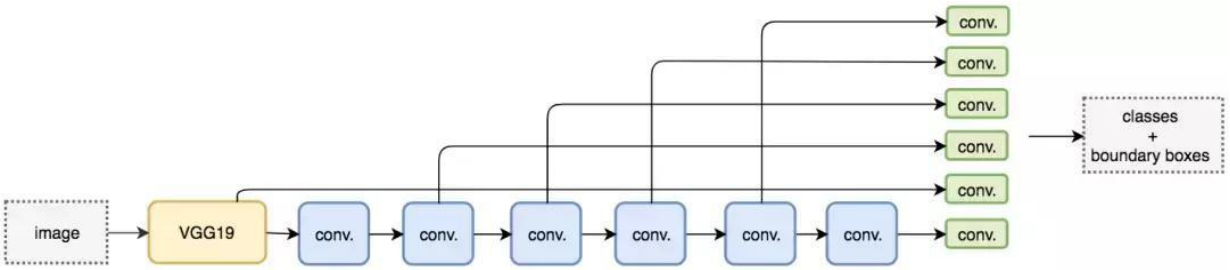
7.3.3 SSD

SSD 是使用 VGG19 网络作为特征提取器（和 Faster R-CNN 中使用的 CNN 一样）的单次检测器。我们在该网络之后添加自定义卷积层（蓝色），并使用卷积核（绿色）执行预测。



同时对类别和位置执行单次预测。

然而，卷积层降低了空间维度和分辨率。因此上述模型仅可以检测较大的目标。为了解决该问题，我们从多个特征图上执行独立的目标检测。



使用多尺度特征图用于检测。

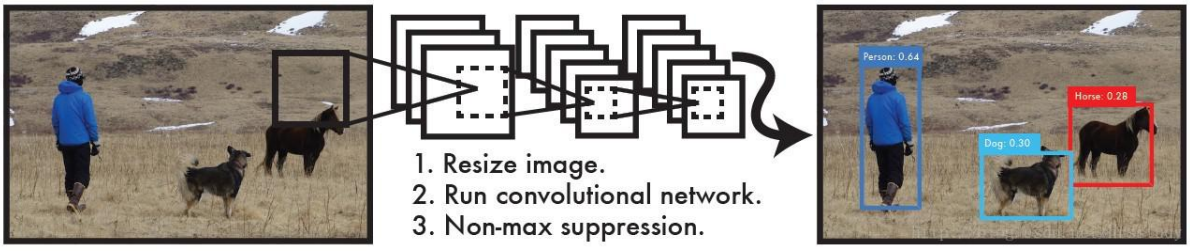
7.4 YOLO 系列

7.4.1 YOLOv1 介绍

YOLO 之前的物体检测系统使用分类器来完成物体检测任务。为了检测一个物体，这些物体检测系统要在一张测试图的不同位置和不同尺寸的 bounding box 上使用该物体的分类器去评估是否有该物体。如 DPM 系统，要使用一个滑窗（sliding window）在整张图像上均匀滑动，用分类器评估是否有物体。

在 DPM 之后提出的其他方法，如 R-CNN 方法使用 region proposal 来生成整张图像中可能包含待检测物体的 potential bounding boxes，然后用分类器来评估这些 boxes，接着通过 post-processing 来改善 bounding boxes，消除重复的检测目标，并基于整个场景中的其他物体重新对 boxes 进行打分。整个流程执行下来很慢，而且因为这些环节都是分开训练的，检测性能很难进行优化。

作者设计了 YOLO（You Only Look Once: Unified, Real-Time Object Detection），将物体检测任务当做回归问题（regression problem）来处理，直接通过整张图片的所有像素得到 bounding box 的坐标、box 中包含物体的置信度和 class probabilities。通过 YOLO，每张图像只需要看一眼就能得出图像中都有哪些物体和这些物体的位置。

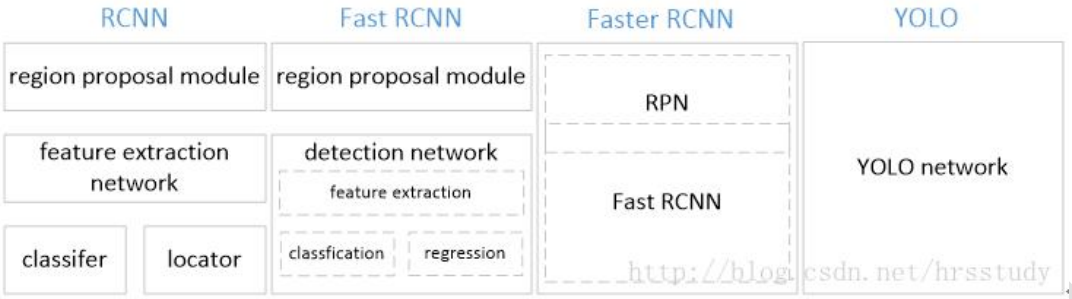


如图所示，使用 YOLO 来检测物体，其流程是非常简单明了的：

1、将图像 resize 到 448 * 448 作为神经网络的输入

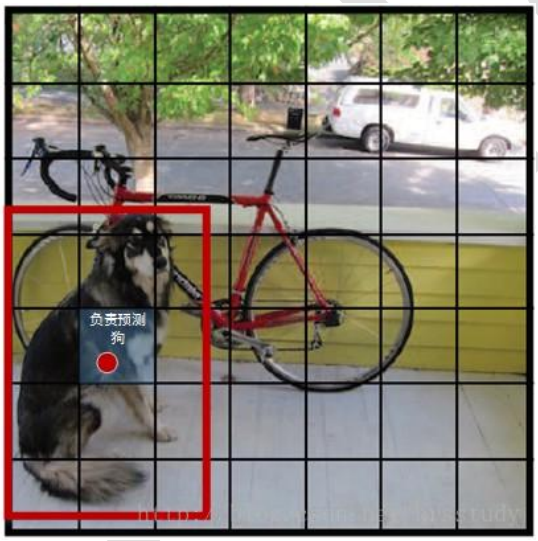
2、运行神经网络，得到一些 bounding box 坐标、box 中包含物体的置信度和 class probabilities

3、进行非极大值抑制，筛选 Boxes 下图是各物体检测系统的检测流程对比：



Unified Detection

YOLO 将输入图像划分为 S*S 的栅格，每个栅格负责检测中心落在该栅格中的物体,如下图所示：



每一个栅格预测 B 个 bounding boxes，以及这些 bounding boxes 的 confidence scores。这个 confidence scores 反映了模型对于这个栅格的预测：该栅格是否含有物体，以及这个 box 的坐标预测的有多准。

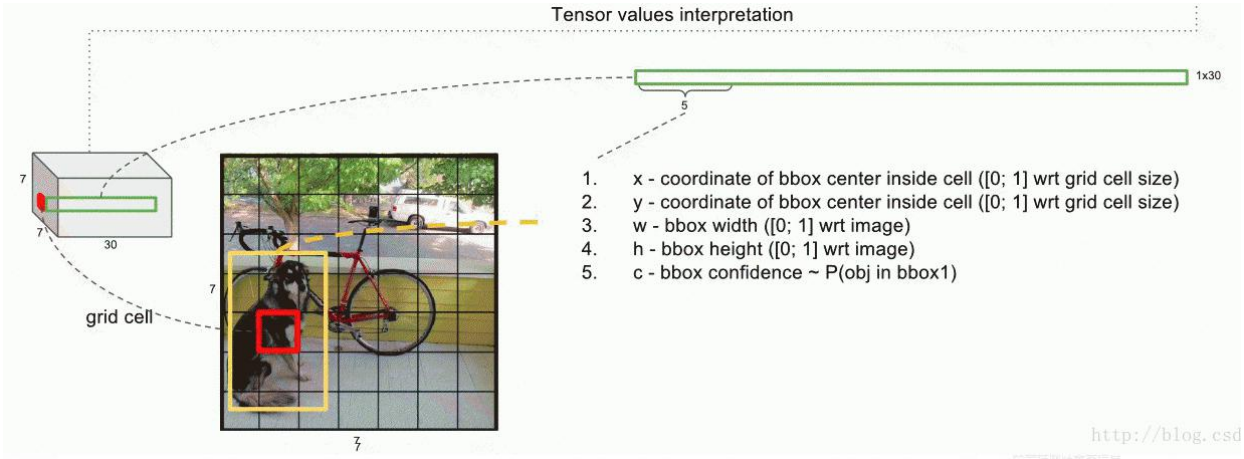
如果这个栅格中不存在一个 object，则 confidence score 应该为 0；否则的话，confidence score 则为 predicted bounding box 与 ground truth box 之间的 IOU（intersection over union)。

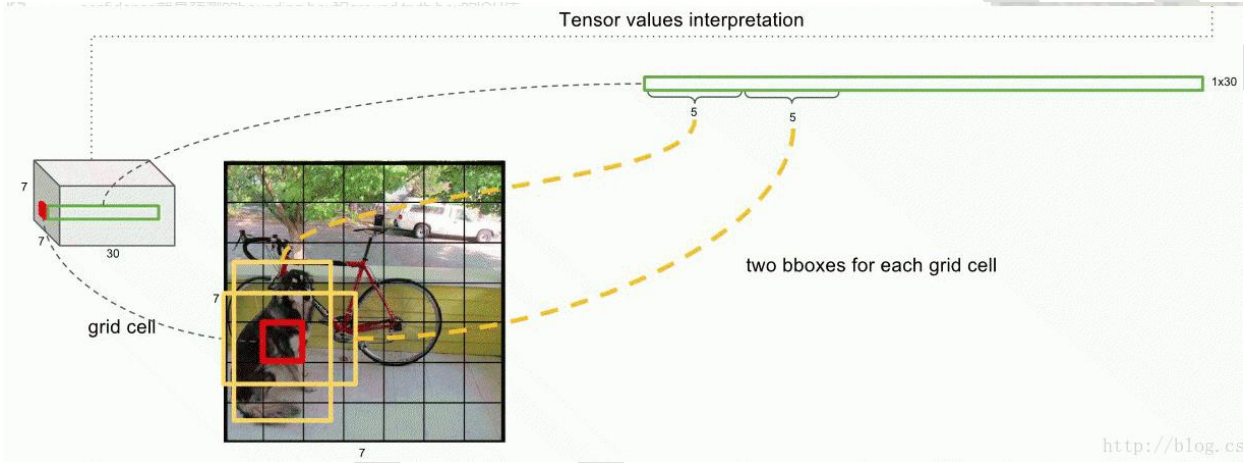
YOLO 对每个 bounding box 有 5 个 predictions：x, y, w, h, and confidence。

坐标 x,y 代表了预测的 bounding box 的中心与栅格边界的相对值。

坐标 w,h 代表了预测的 bounding box 的 width、height 相对于整幅图像 width,height 的比例。

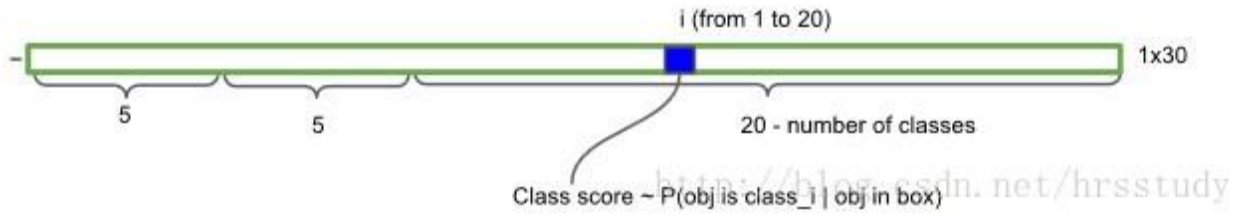
confidence 就是预测的 bounding box 和 ground truth box 的 IOU 值。





每一个栅格还要预测 C 个 conditional class probability（条件类别概率）：Pr(Classi|Object)。即在一个栅格包含一个 Object 的前提下，它属于某个类的概率。

我们只为每个栅格预测一组（C 个）类概率，而不考虑框 B 的数量。

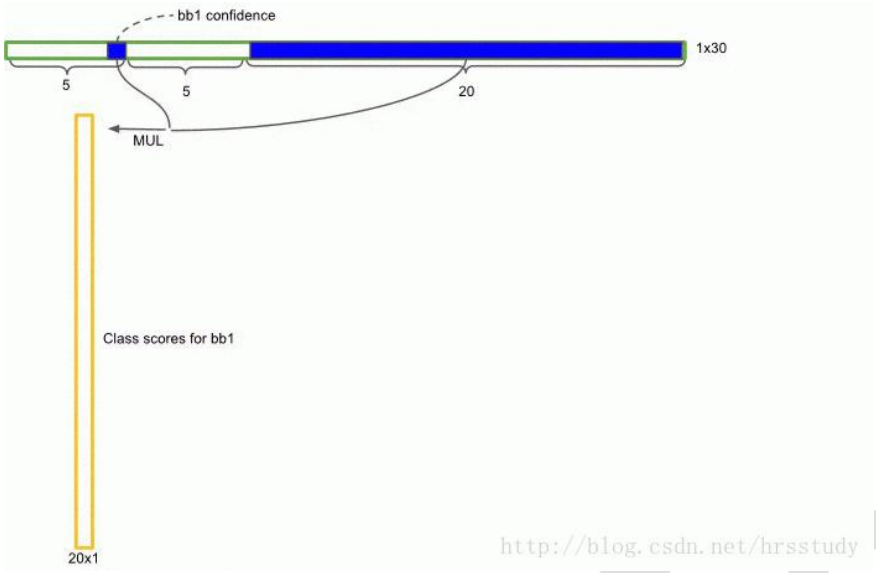


注意： conditional class probability 信息是针对每个网格的。

confidence 信息是针对每个 bounding box 的。

在测试阶段，将每个栅格的 conditional class probabilities 与每个 bounding box 的 confidence 相乘：

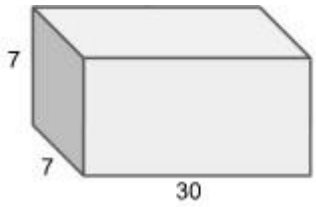
这样既可得到每个 bounding box 的具体类别的 confidence score。 这乘积既包含了 bounding box 中预测的 class 的 probability 信息，也反映了 bounding box 是否含有 Object 和 bounding box 坐标的准确度。

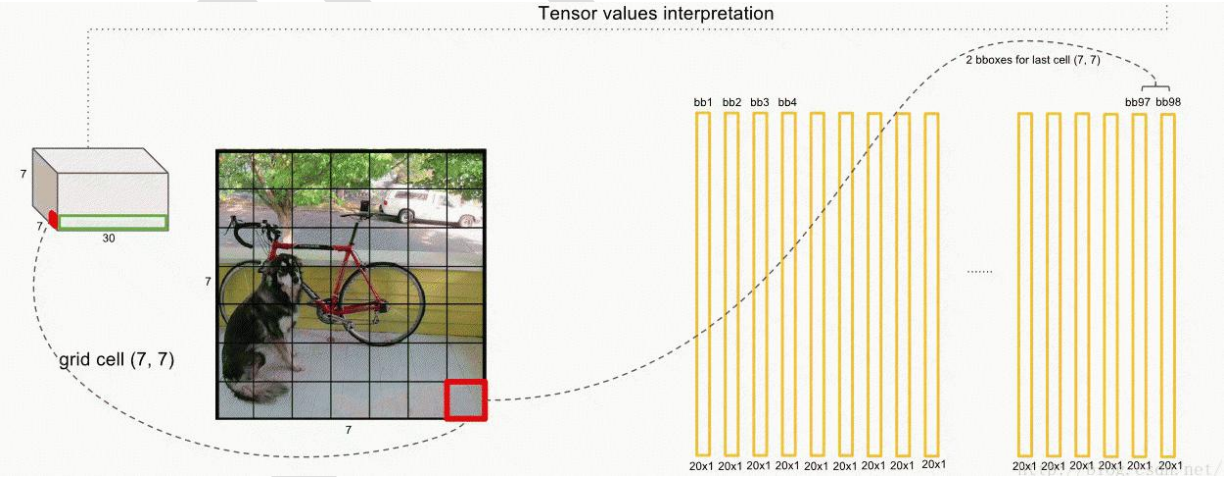


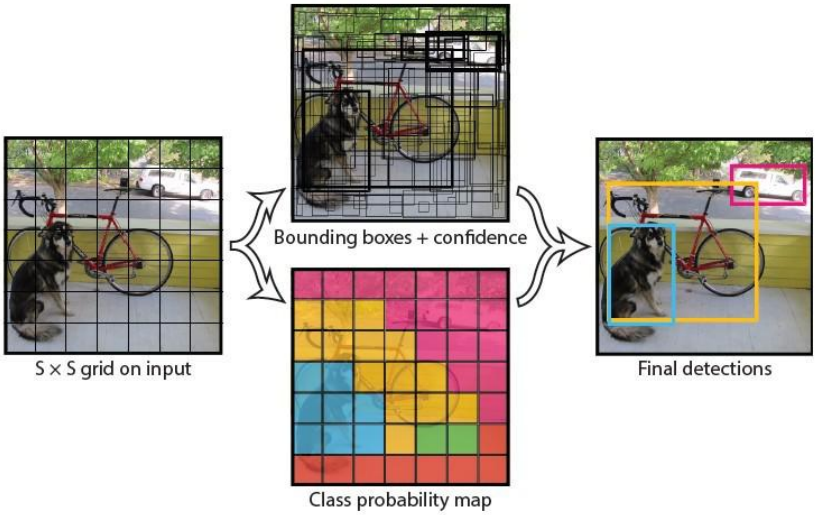
将 YOLO 用于 PASCAL VOC 数据集时：

论文使用的 S=7，即将一张图像分为 7×7=49 个栅格每一个栅格预测 B=2 个 boxes（每个 box 有 x,y,w,h,confidence，5 个预测值），同时 C=20（PASCAL 数据集中有 20 个类别）。

因此，最后的 prediction 是 7×7×30 ｛ 即 S * S *（ B * 5 + C）｝的 Tensor。

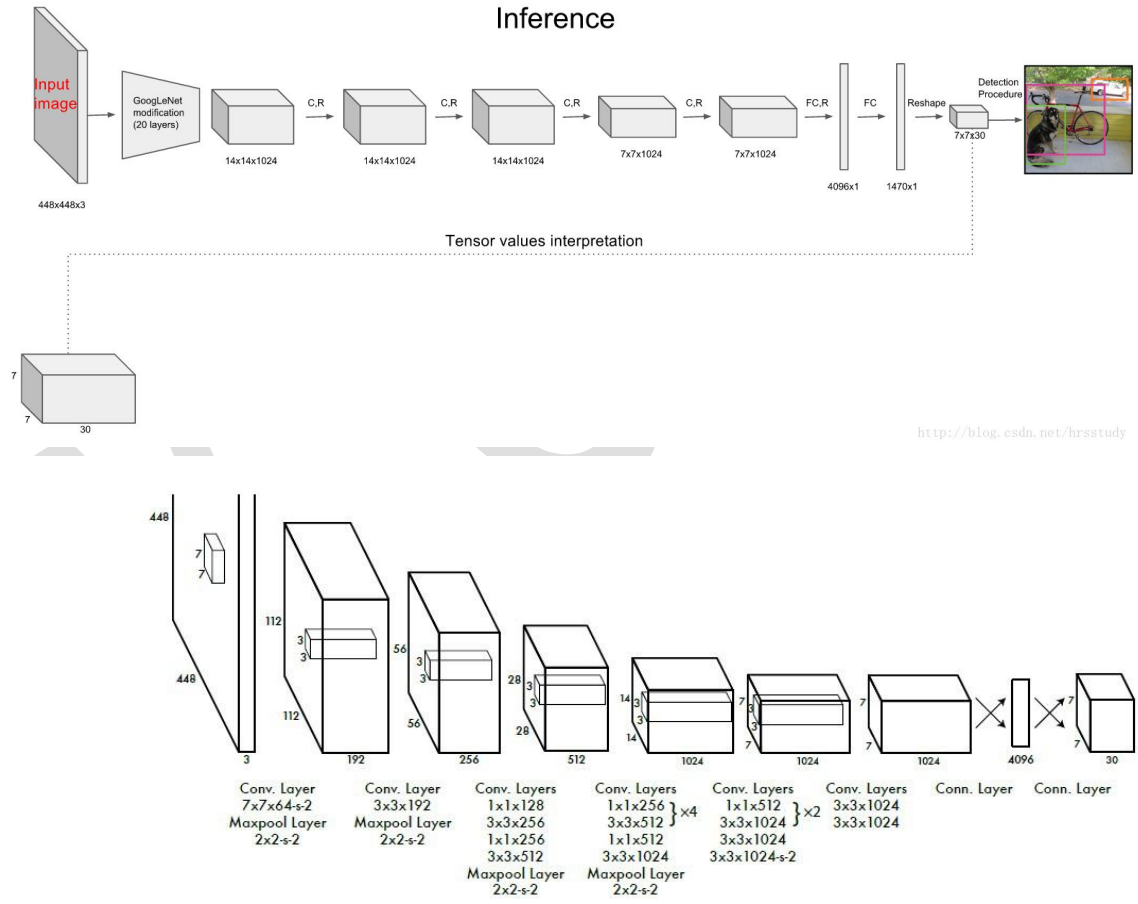






Network Design

YOLO 检测网络包括 24 个卷积层和 2 个全连接层，如图所示：



其中，卷积层用来提取图像特征，全连接层用来预测图像位置和类别概率值。

YOLO 网络借鉴了 GoogLeNet 分类网络结构。不同的是，YOLO 未使用 inception module，而是使用 1x1 卷积层（此处 1x1 卷积层的存在是为了跨通道信息整合）+3x3 卷积层简单替代。

Training

首先利用 ImageNet 1000-class 的分类任务数据集 Pretrain 卷积层。使用上述网络中的前 20 个卷积层，加上一个 average-pooling layer，最后加一个全连接层，作为 Pretrain 的网络。训练大约一周的时间，使得在 ImageNet 2012 的验证数据集 Top-5 的精度达到 88%，这个结果跟 GoogleNet 的效果相当。

将 Pretrain 的结果的前 20 层卷积层应用到 Detection 中，并加入剩下的 4 个卷积层及 2 个全连接。

同时为了获取更精细化的结果，将输入图像的分辨率由 224* 224 提升到 448* 448。

将所有的预测结果都归一化到 0~1，使用 Leaky RELU 作为激活函数。

为了防止过拟合，在第一个全连接层后面接了一个 ratio=0.5 的 Dropout 层。

为了提高精度，对原始图像做数据提升。

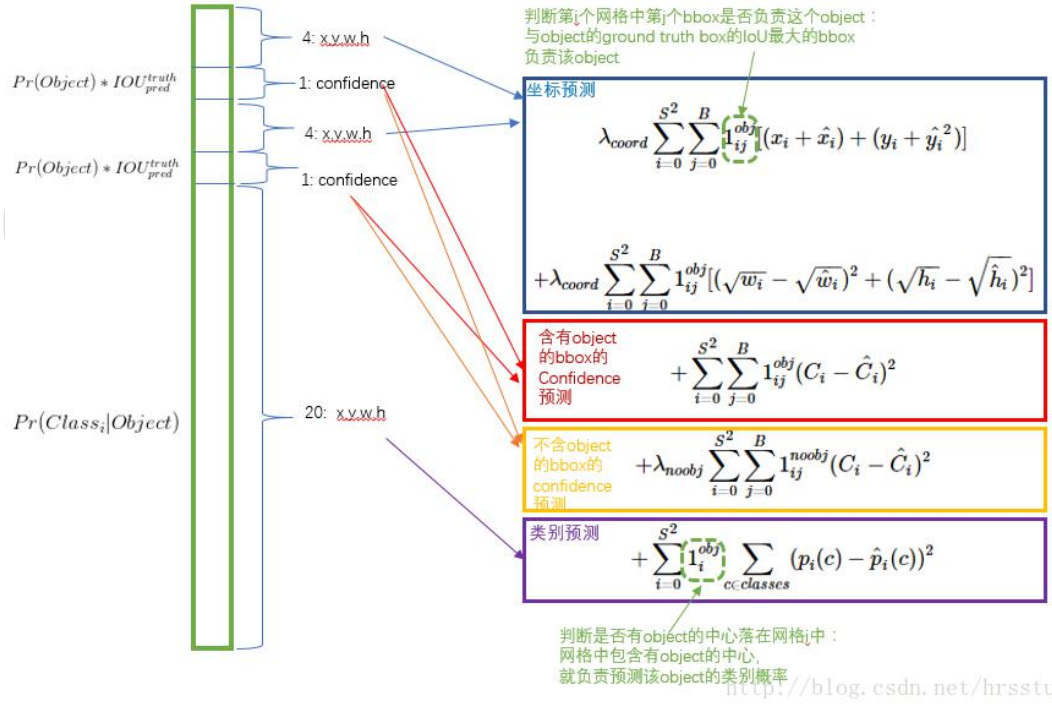
损失函数

损失函数的设计目标就是让坐标（x,y,w,h），confidence，classification 这个三个方面达到很好的平衡。

简单的全部采用了 sum-squared error loss 来做这件事会有以下不足：

a) 8 维的 localization error 和 20 维的 classification error 同等重要显然是不合理的。

b) 如果一些栅格中没有 object（一幅图中这种栅格很多），那么就会将这些栅格中的 bounding box 的 confidence 置为 0，相比于较少的有 object 的栅格，这些不包含物体的栅格对梯度更新的贡献会远大于包含物体的栅格对梯度更新的贡献，这会导致网络不稳定甚至发散。



解决方案如下：

更重视 8 维的坐标预测，给这些损失前面赋予更大的 loss weight，记为 λcoord，在 pascal VOC 训练中取 5。（上图蓝色框）

对没有 object 的 bbox 的 confidence loss，赋予小的 loss weight，记为 λ_{noobj}，在 pascal VOC 训练中取 0.5。（上图橙色框）

有 object 的 bbox 的 confidence loss（上图红色框）和类别的 loss （上图紫色框）的 loss weight 正常取 1。

对不同大小的 bbox 预测中，相比于大 bbox 预测偏一点，小 box 预测偏相同的尺寸对 IOU 的影响更大。而 sum-square error loss 中对同样的偏移 loss 是一样。

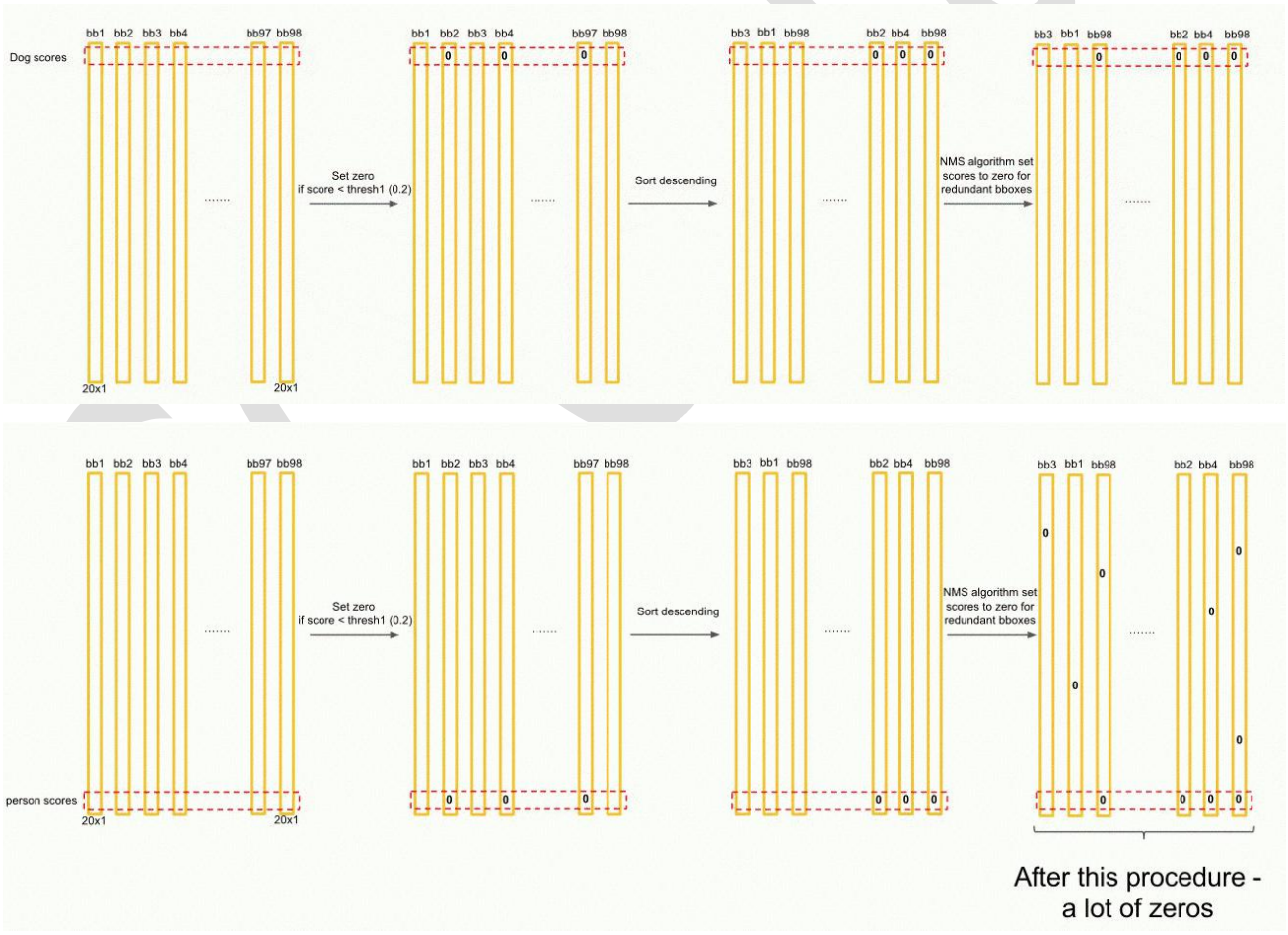
为了缓和这个问题，作者用了一个巧妙的办法，就是将 box 的 width 和 height 取平方根代替原本的 height 和 width。

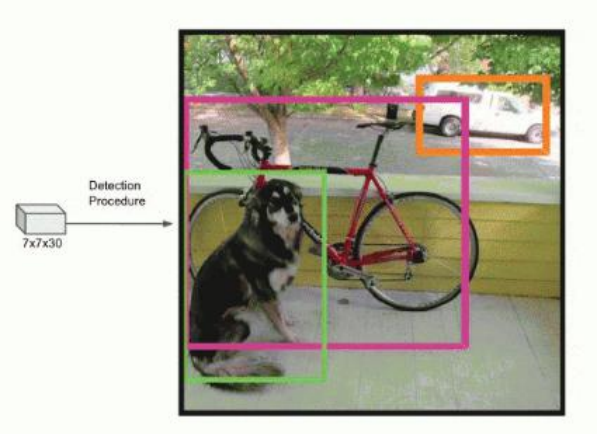
在 YOLO 中，每个栅格预测多个 bounding box，但在网络模型的训练中，希望每一个物体最后由一个 bounding box predictor 来负责预测。

因此，当前哪一个 predictor 预测的 bounding box 与 ground truth box 的 IOU 最大，这个 predictor 就负责 predict object。

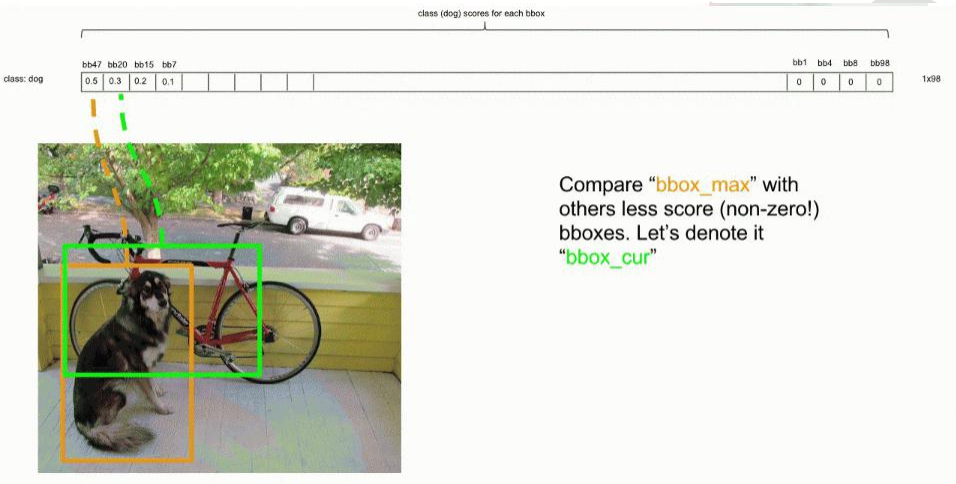
这会使得每个 predictor 可以专门的负责特定的物体检测。随着训练的进行，每一个 predictor 对特定的物体尺寸、长宽比的物体的类别的预测会越来越好。

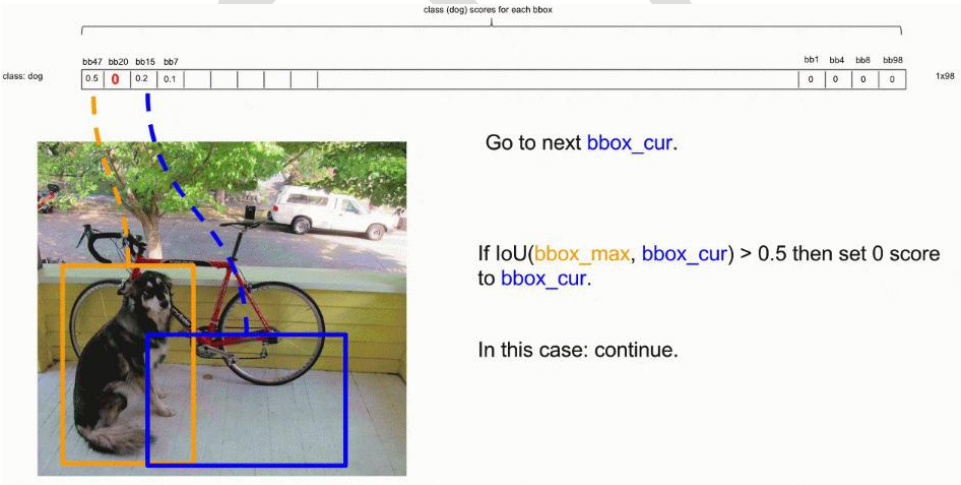
神经网络输出后的检测流程

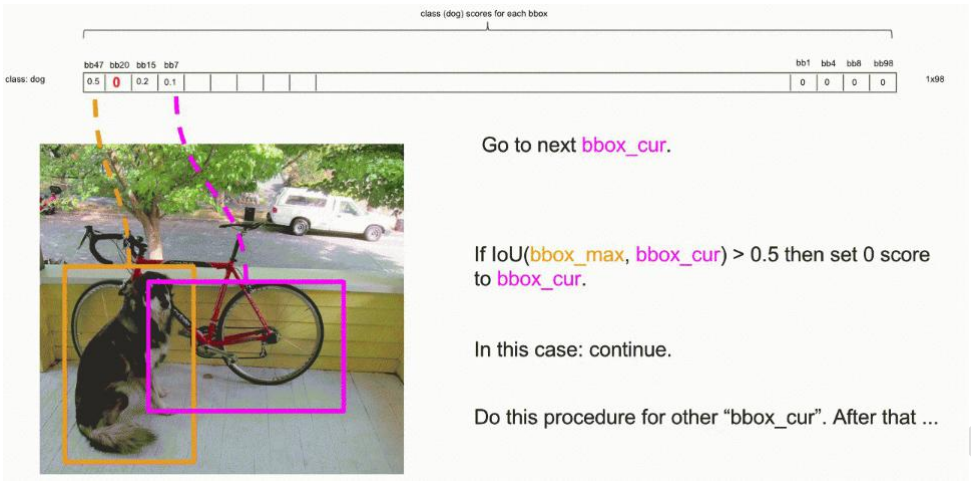



Detection
Procedure
7x7x30

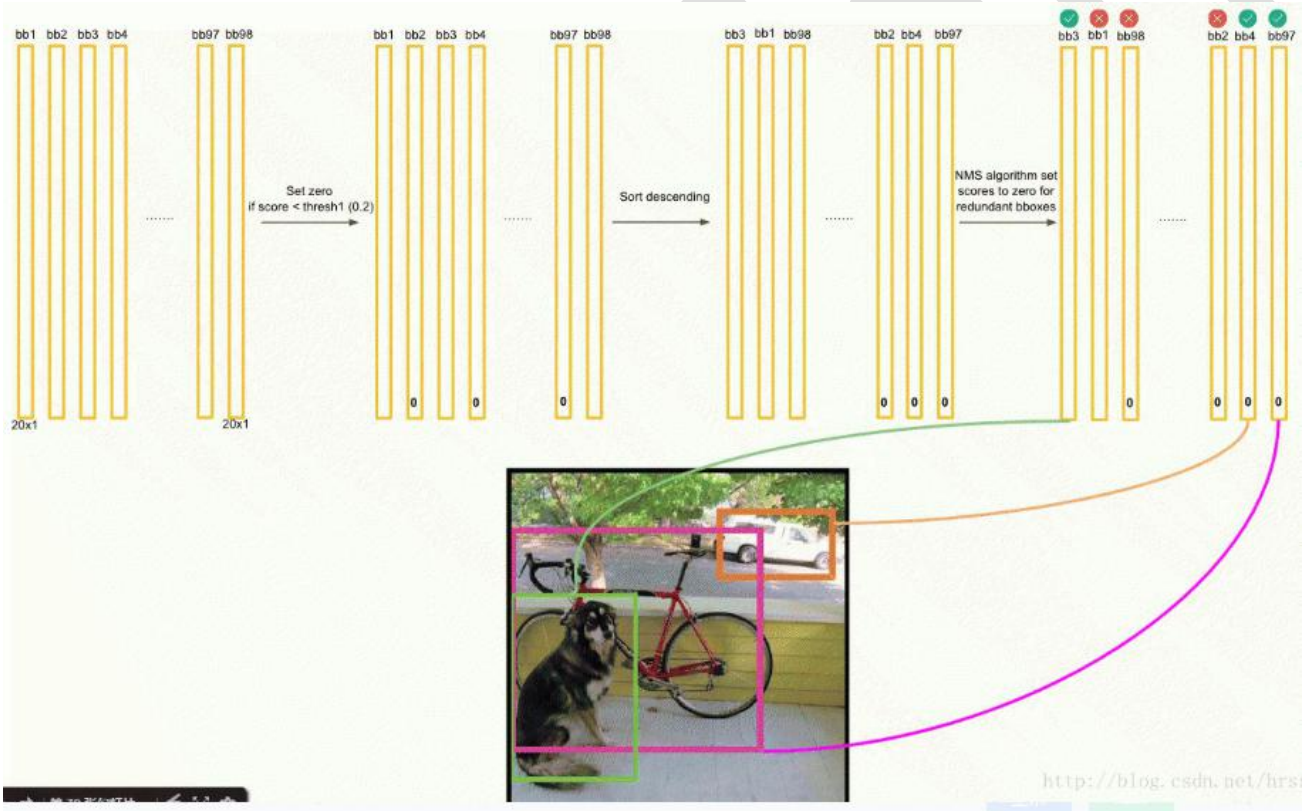
非极大值抑制


class (dog) scores for each bbox
bb47 bb20 bb15 bb7
bb1 bb4 bb8 bb98
class: dog
0.5 0.3 0.2 0.1
0 0 0 0
1x98
Compare “bbox_max” with others less score (non-zero!) bboxes. Let’s denote it “bbox_cur”


class (dog) scores for each bbox
bb47 bb20 bb15 bb7
bb1 bb4 bb8 bb98
class: dog
0.5 0 0.2 0.1
0 0 0 0
1x98
Go to next bbox_cur.
If IoU(bbox_max, bbox_cur) > 0.5 then set 0 score to bbox_cur.
In this case: continue.



获取 Object Detect 结果



7.4.2 YOLOv1 模型优缺点？

1、YOLO 检测物体非常快。

因为没有复杂的检测流程，只需要将图像输入到神经网络就可以得到检测结果，YOLO 可以非常快的完成物体检测任务。标准版本的 YOLO 在 Titan X 的 GPU 上能达到 45 FPS。更快的 Fast YOLO 检测速度可以达到 155 FPS。而且，YOLO 的 mAP 是之前其他实时物体检测系统的两倍以上。

2、YOLO 可以很好的避免背景错误，产生 false positives。

不像其他物体检测系统使用了滑窗或 region proposal，分类器只能得到图像的局部信息。YOLO 在训练和测试时都能够看到一整张图像的信息，因此 YOLO 在检测物体时能很好的利用上下文信息，从而不容易在背景上预测出错误的物体信息。和 Fast-R-CNN 相比，YOLO 的背景错误不到 Fast-R-CNN 的一半。

3、YOLO 可以学到物体的泛化特征。

当 YOLO 在自然图像上做训练，在艺术作品上做测试时，YOLO 表现的性能比 DPM、R-CNN 等之前的物体检测系统要好很多。因为 YOLO 可以学习到高度泛化的特征，从而迁移到其他领域。

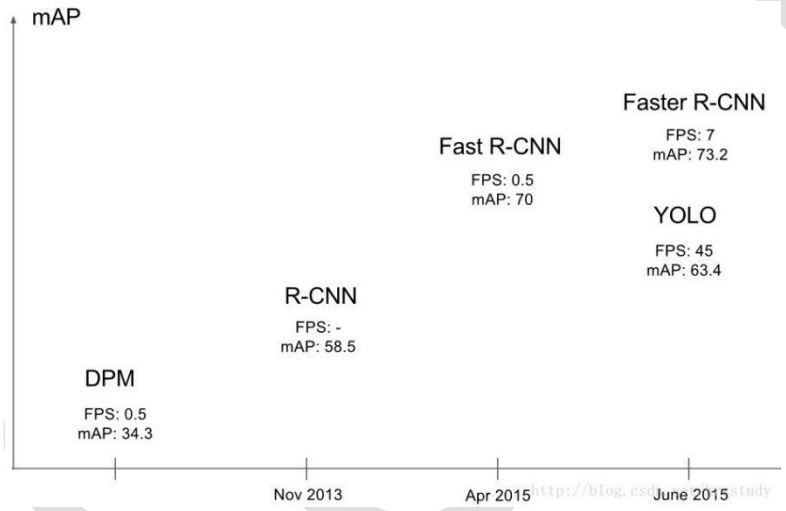
尽管 YOLO 有这些优点，它也有一些缺点：

1、YOLO 的物体检测精度低于其他 state-of-the-art 的物体检测系统。

2、YOLO 容易产生物体的定位错误。

3、YOLO 对小物体的检测效果不好（尤其是密集的小物体，因为一个栅格只能预测 2 个物体）。

下图是各物体检测系统的检测性能对比：



7.4.3 YOLOv2

YOLOv2 相比 YOLOv1 做了很多方面的改进，这也使得 YOLOv2 的 mAP 有显著的提升，并且 YOLOv2 的速度依然很快，保持着自己作为 one-stage 方法的优势，YOLOv2 和 Faster R-CNN, SSD 等模型的对比如图 1 所示。

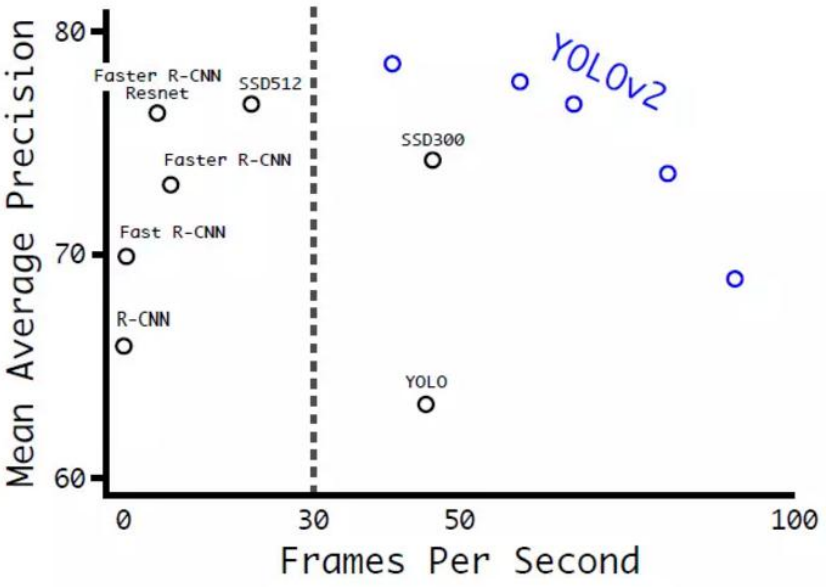


图 1：YOLOv2 与其它模型在 VOC 2007 数据集上的效果对比

YOLOv1 虽然检测速度很快，但是在检测精度上却不如 R-CNN 系检测方法，YOLOv1 在物体定位方面（localization）不够准确，并且召回率（recall）较低。YOLOv2 共提出了几种改进策略来提升 YOLO 模型的定位准确度和召回率，从而提高 mAP，YOLOv2 在改进中遵循一个原则：保持检测速度，这也是 YOLO 模型的一大优势。YOLOv2 的改进策略如图 2 所示，可以看出，大部分的改进方法都可以比较显著提升模型的 mAP。

7.4.4 YOLOv2 改进策略

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

图 2：YOLOv2 相比 YOLOv1 的改进策略

（1）Batch Normalization

Batch Normalization 可以提升模型收敛速度，而且可以起到一定正则化效果，降低模型的过拟合。在 YOLOv2 中，每个卷积层后面都添加了 Batch Normalization 层，并且不再使用 droput。使用 Batch Normalization 后，YOLOv2 的 mAP 提升了 2.4%。

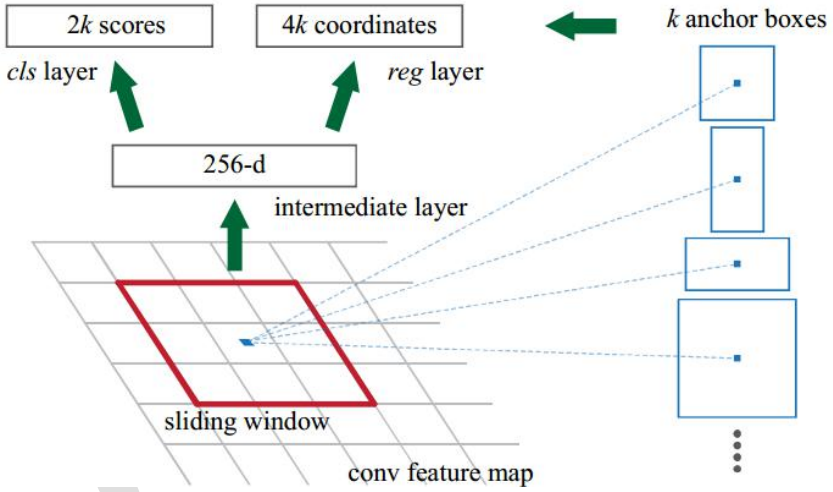
（2）High Resolution Classifier

目前的目标检测方法中，基本上都会使用 ImageNet 预训练过的模型（classifier）来提取特征，如果用的是 AlexNet 网络，那么输入图片会被 resize 到不足 256 * 256，导致分辨率不够高，给检测带来困难。为此，新的 YOLO 网络把分辨率直接提升到了 448 * 448，这也意味之原有的网络模型必须进行某种调整以适应新的分辨率输入。

对于 YOLOv2，作者首先对分类网络（自定义的 darknet）进行了 fine tune，分辨率改成 448 * 448，在 ImageNet 数据集上训练 10 轮（10 epochs），训练后的网络就可以适应高分辨率的输入了。然后，作者对检测网络部分（也就是后半部分）也进行 fine tune。这样通过提升输入的分辨率，mAP 获得了 4%的提升。

（3）Convolutional With Anchor Boxes

之前的 YOLO 利用全连接层的数据完成边框的预测，导致丢失较多的空间信息，定位不准。作者在这一版本中借鉴了 Faster R-CNN 中的 anchor 思想，回顾一下，anchor 是 RNP 网络中的一个关键步骤，说的是在卷积特征图上进行滑窗操作，每一个中心可以预测 9 种不同大小的建议框。



为了引入 anchor boxes 来预测 bounding boxes，作者在网络中果断去掉了全连接层。剩下的具体怎么操作呢？

首先，作者去掉了后面的一个池化层以确保输出的卷积特征图有更高的分辨率。然后，通过缩减网络，让图片输入分辨率为 416 * 416，这一步的目的是为了让后面产生的卷积特征图宽高都为奇数，这样就可以产生一个 center cell。

作者观察到，大物体通常占据了图像的中间位置，就可以只用中心的一个 cell 来预测这些物体的位置，否则就要用中间的 4 个 cell 来进行预测，这个技巧可稍稍提升效率。

最后，YOLOv2 使用了卷积层降采样（factor 为 32），使得输入卷积网络的 416 * 416 图片

最终得到 13 * 13 的卷积特征图（416/32=13）。

加入了 anchor boxes 后，可以预料到的结果是召回率上升，准确率下降。我们来计算一下，假设每个 cell 预测 9 个建议框，那么总共会预测 13 * 13 * 9 = 1521 个 boxes，而之前的网络仅仅预测 7 * 7 * 2 = 98 个 boxes。具体数据为：没有 anchor boxes，模型 recall 为 81%，mAP 为 69.5%；加入 anchor boxes，模型 recall 为 88%，mAP 为 69.2%。这样看来，准确率只有小幅度的下降，而召回率则提升了 7%，说明可以通过进一步的工作来加强准确率，的确有改进空间。

（4）Dimension Clusters（维度聚类）

作者在使用 anchor 的时候遇到了两个问题，第一个是 anchor boxes 的宽高维度往往是精选的先验框（hand-picked priors），虽说在训练过程中网络也会学习调整 boxes 的宽高维度，最终得到准确的 bounding boxes。但是，如果一开始就选择了更好的、更有代表性的先验 boxes 维度，那么网络就更容易学到准确的预测位置。

和以前的精选 boxes 维度不同，作者使用了 K-means 聚类方法类训练 bounding boxes，可以自动找到更好的 boxes 宽高维度。传统的 K-means 聚类方法使用的是欧氏距离函数，也就意味着较大的 boxes 会比较小的 boxes 产生更多的 error，聚类结果可能会偏离。为此，作者采用的评判标准是 IOU 得分（也就是 boxes 之间的交集除以并集），这样的话，error 就和 box 的尺度无关了，最终的距离函数为：

$$d(box, centroid) = 1 - TOU(box, centroid)$$

作者通过改进的 K-means 对训练集中的 boxes 进行了聚类，判别标准是平均 IOU 得分，聚类结果如下图：

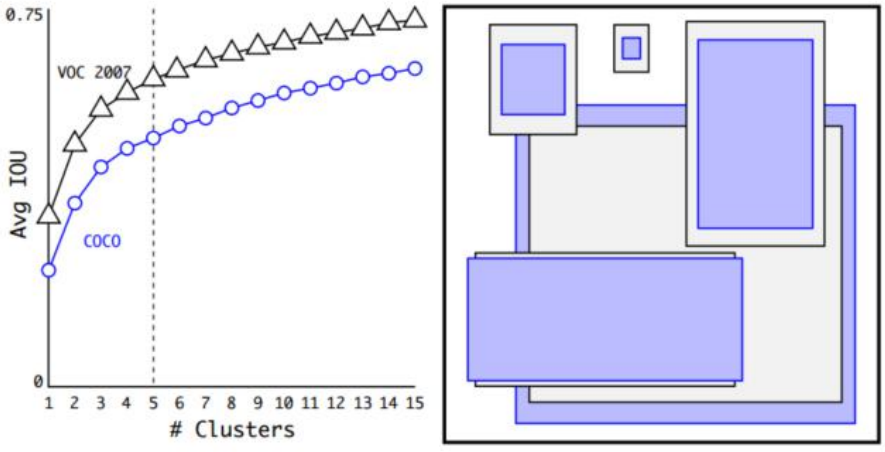


图 3：数据集 VOC 和 COCO 上的边界框聚类分析结果

可以看到，平衡复杂度和 IOU 之后，最终得到 k 值为 5，意味着作者选择了 5 种大小的 box 维度来进行定位预测，这与手动精选的 box 维度不同。结果中扁长的框较少，而瘦高的框更多（这符合行人的特征），这种结论如不通过聚类实验恐怕是发现不了的。

当然，作者也做了实验来对比两种策略的优劣，如下图，使用聚类方法，仅仅 5 种 boxes 的召回率就和 Faster R-CNN 的 9 种相当。说明 K-means 方法的引入使得生成的 boxes 更具有代表性，为后面的检测任务提供了便利。

Box Generation	#	Avg IOU
Cluster SSE	5	58.7
Cluster IOU	5	61.0
Anchor Boxes [15]	9	60.9
Cluster IOU	9	67.2

（5）New Network：Darknet-19

YOLOv2 采用了一个新的基础模型（特征提取器），称为 Darknet-19，包括 19 个卷积层和 5 个 maxpooling 层，如图 4 所示。Darknet-19 与 VGG16 模型设计原则是一致的，主要采用 3*3 卷积，采用 2*2 的 maxpooling 层之后，特征图维度降低 2 倍，而同时将特征图的 channles 增加两倍。与 NIN(Network in Network)类似，Darknet-19 最终采用 global avgpooling 做预测，并且在 3*3 卷积之间使用 1*1 卷积来压缩特征图 channles 以降低模型计算量和参数。Darknet-19 每个卷积层后面同样使用了 batch norm 层以加快收敛速度，降低模型过拟合。在 ImageNet 分类数据集上，Darknet-19 的 top-1 准确度为 72.9%，top-5 准确度为 91.2%，但是模型参数相对小一些。使用 Darknet-19 之后，YOLOv2 的 mAP 值没有显著提升，但是计算量却可以减少约 33%。

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

（6） Direct location prediction

前面讲到，YOLOv2 借鉴 RPN 网络使用 anchor boxes 来预测边界框相对先验框的 offsets。边界框的实际中心位置(x, y)，需要根据预测的坐标偏移值(t_x, t_y)，先验框的尺度(w_a, h_a)以及中心坐标(x_a, y_a)（特征图每个位置的中心点）来计算：

$$x = (t_x \times w_a) - x_a = (t_y \times h_a) - y_a$$

但是上面的公式是无约束的，预测的边界框很容易向任何方向偏移，如当$t_x = 1$时边界框将向右偏移先验框的一个宽度大小，而当$t_x = -1$时边界框将向左偏移先验框的一个宽度大小，因此每个位置预测的边界框可以落在图片任何位置，这导致模型的不稳定性，在训练时需要很长时间来预测出正确的 offsets。所以，YOLOv2 弃用了这种预测方式，而是沿用 YOLOv1 的方法，就是预测边界框中心点相对于对应 cell 左上角位置的相对偏移值，为了将边界框中心点约束在当前 cell 中，使用 sigmoid 函数处理偏移值，这样预测的偏移值在(0,1)范围内（每个 cell 的尺度看做 1）。总结来看，根据边界框预测的 4 个 offsets：t_x, t_y, t_w, t_h，可以按如下公式计算出边界框实际位置和大小：

$$b_x = \sigma t_x + c_x, b_y = \sigma t_y + c_y,$$
$$b_w = p_w e^{t_w}, b_h = p_h e^{t_h},$$

其中为(c_x, c_y)为 cell 的左上角坐标，如图 5 所示，在计算时每个 cell 的尺度为 1，所以当前 cell 的左上角坐标为(1,1)。由于 sigmoid 函数的处理，边界框的中心位置会约束在当前 cell 内部，防止偏移过多。而p_w和p_h是先验框的宽度与长度，前面说过它们的值也是相对于特征图大小的，在特征图中每个 cell 的长和宽均为 1。这里记特征图的大小为(W, H)（在文中是(13,13)），这样我们可以将边界框相对于整张图片的位置和大小计算出来（4 个值均在 0 和 1 之间）：

$$b_x = (\sigma t_x + c_x)/W, b_y = (\sigma t_y + c_y)/H,$$
$$b_w = (p_w e^{t_w})/W, b_h = (p_h e^{t_h})/H,$$

如果再将上面的 4 个值分别乘以图片的宽度和长度（像素点值）就可以得到边界框的最终位置和大小了。这就是 YOLOv2 边界框的整个解码过程。约束了边界框的位置预测值使得模型更容易稳定训练，结合聚类分析得到先验框与这种预测方法，YOLOv2 的 mAP 值提升了约 5%。

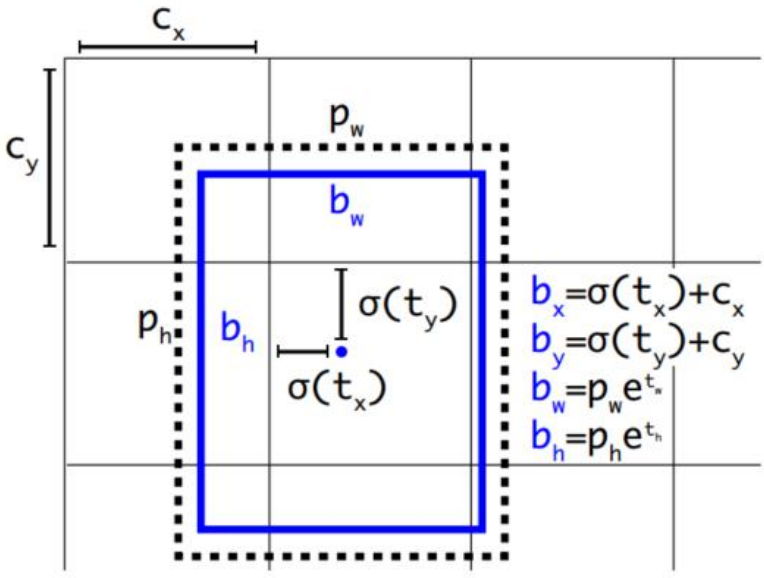


图 5：边界框位置与大小的计算示例图

（7）Fine-Grained Features

YOLOv2 的输入图片大小为 416*416，经过 5 次 maxpooling 之后得到 13*13 大小的特征图，并以此特征图采用卷积做预测。13*13 大小的特征图对检测大物体是足够了，但是对于小物体还需要更精细的特征图（Fine-Grained Features）。因此 SSD 使用了多尺度的特征图来分别检测不同大小的物体，前面更精细的特征图可以用来预测小物体。YOLOv2 提出了一种 passthrough 层来利用更精细的特征图。YOLOv2 所利用的 Fine-Grained Features 是 26*26 大小的特征图（最后一个 maxpooling 层的输入），对于 Darknet-19 模型来说就是大小为 26*26*512 的特征图。passthrough 层与 ResNet 网络的 shortcut 类似，以前面更高分辨率的特征图为输入，然后将其连接到后面的低分辨率特征图上。前面的特征图维度是后面的特征图的 2 倍，passthrough 层抽取前面层的每个 2*2 的局部区域，然后将其转化为 channel 维度，对于 26*26*512 的特征图，经 passthrough 层处理之后就变成了 13*13*2048 的新特征图（特征图大小降低 4 倍，而 channles 增加 4 倍，图 6 为一个实例），这样就可以与后面的 13*13*1024 特征图连接在一起形成 13*13*3072 的特征图，然后在此特征图基础上卷积做预测。在 YOLO 的 C 源码中，passthrough 层称为 reorg layer。

另外，作者在后期的实现中借鉴了 ResNet 网络，不是直接对高分辨特征图处理，而是增加了一个中间卷积层，先采用 64 个 1*1 卷积核进行卷积，然后再进行 passthrough 处理，这样 26*26*512 的特征图得到 13*13*256 的特征图。这算是实现上的一个小细节。使用 Fine-Grained Features 之后 YOLOv2 的性能有 1%的提升。

（8）Multi-Scale Training

由于 YOLOv2 模型中只有卷积层和池化层，所以 YOLOv2 的输入可以不限于 416*416 大小的图片。为了增强模型的鲁棒性，YOLOv2 采用了多尺度输入训练策略，具体来说就是在训

练过程中每间隔一定的 iterations 之后改变模型的输入图片大小。由于 YOLOv2 的下采样总步长为 32，输入图片大小选择一系列为 32 倍数的值：输入图片最小为 320*320，此时对应的特征图大小为 10*10（不是奇数了，确实有点尴尬），而输入图片最大为 608*608,对应的特征图大小为 19*19,在训练过程，每隔 10 个 iterations 随机选择一种输入图片大小，然后只需要修改对最后检测层的处理就可以重新训练。

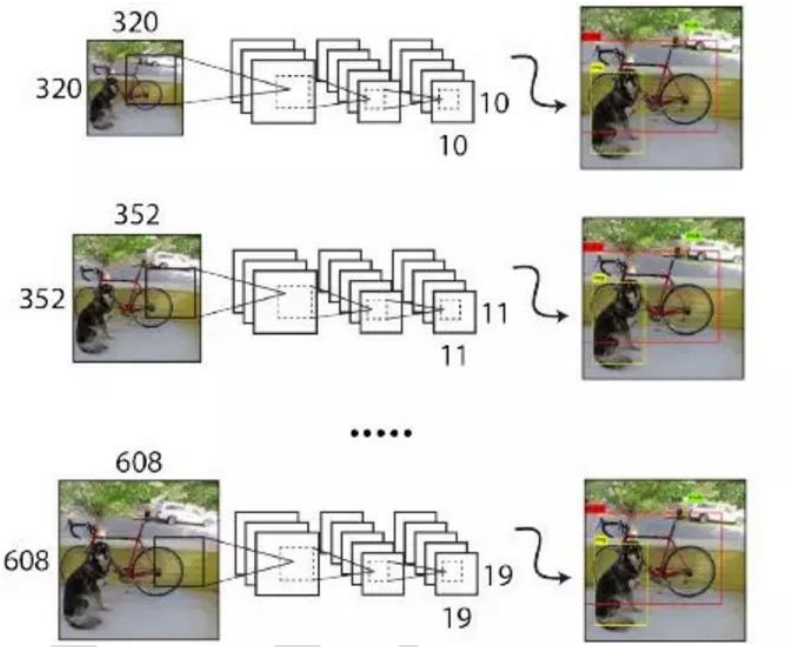


图 7：Multi-Scale Training

采用 Multi-Scale Training 策略，YOLOv2 可以适应不同大小的图片，并且预测出很好的结果。在测试时，YOLOv2 可以采用不同大小的图片作为输入，在 VOC 2007 数据集上的效果如下图所示。可以看到采用较小分辨率时，YOLOv2 的 mAP 值略低，但是速度更快，而采用高分辨输入时，mAP 值更高，但是速度略有下降，对于 544*544,mAP 高达 78.6%。注意，这只是测试时输入图片大小不同，而实际上用的是同一个模型（采用 Multi-Scale Training 训练）。

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

图 8：YOLOv2 在 VOC 2007 数据集上的性能对比

总结来看，虽然 YOLOv2 做了很多改进，但是大部分都是借鉴其它论文的一些技巧，如 Faster R-CNN 的 anchor boxes，YOLOv2 采用 anchor boxes 和卷积做预测，这基本上与 SSD 模型（单尺度特征图的 SSD）非常类似了，而且 SSD 也是借鉴了 Faster R-CNN 的 RPN 网络。从某种意义上来说，YOLOv2 和 SSD 这两个 one-stage 模型与 RPN 网络本质上无异，只不过 RPN 不做类别的预测，只是简单地区分物体与背景。在 two-stage 方法中，RPN 起到的作用是给出 region proposals，其实就是作出粗糙的检测，所以另外增加了一个 stage，即采用 R-CNN 网络来进一步提升检测的准确度（包括给出类别预测）。而对于 one-stage 方法，它们想要一步到位，直接采用“RPN”网络作出精确的预测，要因此要在网络设计上做很多的 tricks。YOLOv2 的一大创新是采用 Multi-Scale Training 策略，这样同一个模型其实就可以适应多种大小的图片了。

7.4.5 YOLOv2 的训练

YOLOv2 的训练主要包括三个阶段。

第一阶段：先在 ImageNet 分类数据集上预训练 Darknet-19，此时模型输入为 224*224,共训练 160 个 epochs。

第二阶段：将网络的输入调整为 448*448,继续在 ImageNet 数据集上 finetune 分类模型，训练 10 个 epochs，此时分类模型的 top-1 准确度为 76.5%，而 top-5 准确度为 93.3%。

第三个阶段：修改 Darknet-19 分类模型为检测模型，并在检测数据集上继续 finetune 网络。

网络修改包括（网路结构可视化）：移除最后一个卷积层、global avgpooling 层以及 softmax 层，并且新增了三个 3*3*2014 卷积层，同时增加了一个 passthrough 层，最后使用 1*1 卷积层输出预测结果。

7.4.6 YOLO9000

YOLO9000 是在 YOLOv2 的基础上提出的一种可以检测超过 9000 个类别的模型，其主要贡献点在于提出了一种分类和检测的联合训练策略。众多周知，检测数据集的标注要比分类数据集打标签繁琐的多，所以 ImageNet 分类数据集比 VOC 等检测数据集高出几个数量级。在 YOLO 中，边界框的预测其实并不依赖于物体的标签，所以 YOLO 可以实现在分类和检测数据集上的联合训练。对于检测数据集，可以用来学习预测物体的边界框、置信度以及为物体分类，而对于分类数据集可以仅用来学习分类，但是其可以大大扩充模型所能检测的物体种类。

作者选择在 COCO 和 ImageNet 数据集上进行联合训练，但是遇到的第一问题是两者的类别并不是完全互斥的，比如"Norfolk terrier"明显属于"dog"，所以作者提出了一种层级分类方法（Hierarchical classification），主要思路是根据各个类别之间的从属关系（根据 WordNet）建立

一种树结构 WordTree，结合 COCO 和 ImageNet 建立的 WordTree 如下图所示：

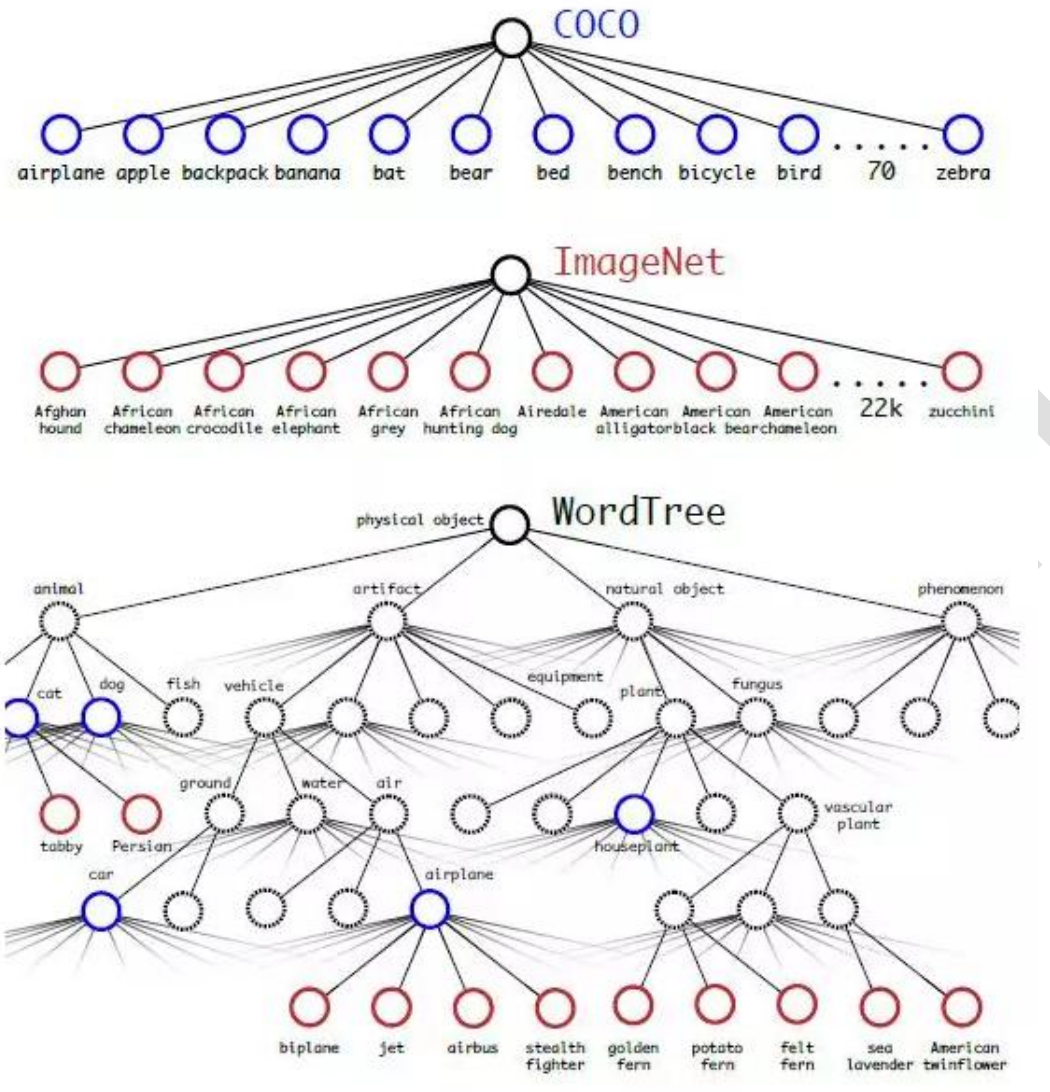


图 12：基于 COCO 和 ImageNet 数据集建立的 WordTree

WordTree 中的根节点为"physical object"，每个节点的子节点都属于同一子类，可以对它们进行 softmax 处理。在给出某个类别的预测概率时，需要找到其所在的位置，遍历这个 path，然后计算 path 上各个节点的概率之积。

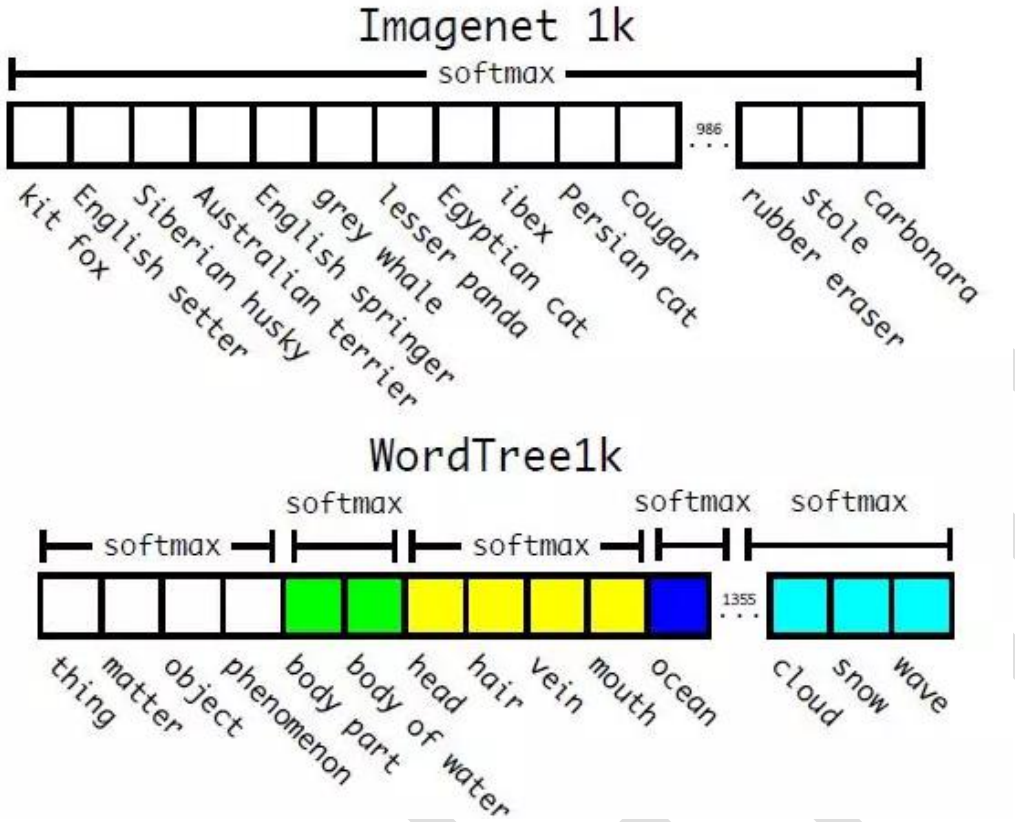
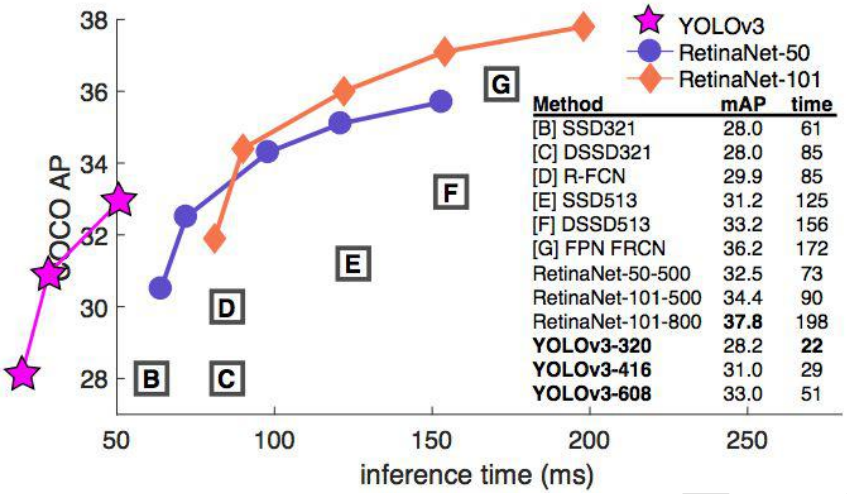


图 13：ImageNet 与 WordTree 预测的对比

在训练时，如果是检测样本，按照 YOLOv2 的 loss 计算误差，而对于分类样本，只计算分类误差。在预测时，YOLOv2 给出的置信度就是 Pr(physicalobject)，同时会给出边界框位置以及一个树状概率图。在这个概率图中找到概率最高的路径，当达到某一个阈值时停止，就用当前节点表示预测的类别。 通过联合训练策略，YOLO9000 可以快速检测出超过 9000 个类别的物体，总体 mAP 值为 19,7%。我觉得这是作者在这篇论文作出的最大的贡献，因为 YOLOv2 的改进策略亮点并不是很突出，但是 YOLO9000 算是开创之举。

7.4.7 YOLOv3

YOLOv3 在 Pascal Titan X 上处理 608x608 图像速度达到 20FPS，在 COCO test-dev 上 mAP@0.5 达到 57.9%，与 RetinaNet（FocalLoss 论文所提出的单阶段网络）的结果相近，并且速度快 4 倍. YOLO v3 的模型比之前的模型复杂了不少，可以通过改变模型结构的大小来权衡速度与精度。 速度对比如下：



YOLOv3 在实现相同准确度下要显著地比其它检测方法快。时间都是在采用 M40 或 Titan X 等相同 GPU 下测量的。

简而言之，YOLOv3 的先验检测（Prior detection）系统将分类器或定位器重新用于执行检测任务。他们将模型应用于图像的多个位置和尺度。而那些评分较高的区域就可以视为检测结果。此外，相对于其它目标检测方法，我们使用了完全不同的方法。我们将一个单神经网络应用于整张图像，该网络将图像划分为不同的区域，因而预测每一块区域的边界框和概率，这些边界框会通过预测的概率加权。我们的模型相比于基于分类器的系统有一些优势。它在测试时会查看整个图像，所以它的预测利用了图像中的全局信息。与需要数千张单一目标图像的 R-CNN 不同，它通过单一网络评估进行预测。这令 YOLOv3 非常快，一般它比 R-CNN 快 1000 倍、比 Fast R-CNN 快 100 倍。

7.4.8 YOLOv3 改进

1、多尺度预测 （类 FPN）

每种尺度预测 3 个 box, anchor 的设计方式仍然使用聚类,得到 9 个聚类中心,将其按照大小均分给 3 中尺度.

尺度 1: 在基础网络之后添加一些卷积层再输出 box 信息.

尺度 2: 从尺度 1 中的倒数第二层的卷积层上采样(x2)再与最后一个 16x16 大小的特征图相加,再次通过多个卷积后输出 box 信息.相比尺度 1 变大两倍.

尺度 3: 与尺度 2 类似,使用了 32x32 大小的特征图.

2、更好的基础分类网络（类 ResNet）和分类器 darknet-53,见下图。

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

基础网络 Darknet-53

3.分类器-类别预测：

YOLOv3 不使用 Softmax 对每个框进行分类，主要考虑因素有两个：

Softmax 使得每个框分配一个类别（score 最大的一个），而对于 Open Images 这种数据集，目标可能有重叠的类别标签，因此 Softmax 不适用于多标签分类。

Softmax 可被独立的多个 logistic 分类器替代，且准确率不会下降。

分类损失采用 binary cross-entropy loss.

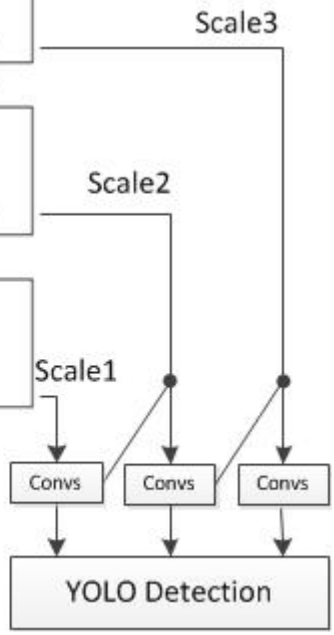
仿 ResNet, 与 ResNet-101 或 ResNet-152 准确率接近,但速度更快.对比如下:

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [13]	74.1	91.8	7.29	1246	**171**
ResNet-101[3]	77.1	93.7	19.7	1039	53
ResNet-152 [3]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

主干架构的性能对比：准确率（top-1 误差、top-5 误差）、运算次数（/十亿）、每秒浮点数运算次数（/十亿），以及 FPS 值。

检测结构如下：

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			



	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

从中看出，YOLOv3 表现得不错。RetinaNet 需要大约 3.8 倍的时间来处理一张图像，YOLOv3 相比 SSD 变体要好得多，并在 AP_50 指标上和当前最佳模型有得一拼

YOLOv3 在 mAP@0.5 及小目标 APs 上具有不错的结果,但随着 IOU 的增大,性能下降,说明 YOLOv3 不能很好地与 ground truth 切合.

边框预测

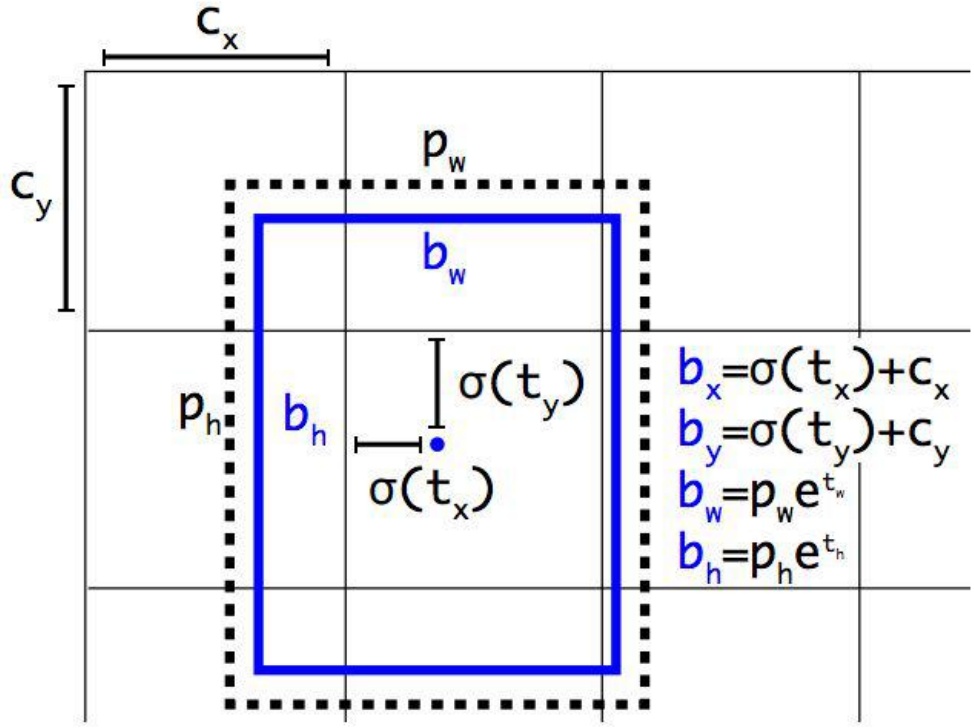


图 2：带有维度先验和定位预测的边界框。我们边界框的宽和高以作为离聚类中心的位移，并使用 Sigmoid 函数预测边界框相对于滤波器应用位置的中心坐标。

仍采用之前的 logis 其中 cx,cy 是网格的坐标偏移量,pw,ph 是预设的 anchor box 的边长.最终得到的边框坐标值是 b*,而网络学习目标是 t*，用 sigmod 函数、指数转换。

优缺点

优点

（1）快速,pipline 简单.

（2）背景误检率低。

（3）通用性强。

YOLO 对于艺术类作品中的物体检测同样适用。它对非自然图像物体的检测率远远高于 DPM 和 RCNN 系列检测方法。

但相比 RCNN 系列物体检测方法，YOLO 具有以下缺点：

（1）识别物体位置精准性差。

（2）召回率低。在每个网格中预测两个 box 这种约束方式减少了对同一目标的多次检测(R-CNN 使用的 region proposal 方式重叠较多),相比 R-CNN 使用 Selective Search 产生 2000 个 proposal（RCNN 测试时每张超过 40 秒）,yolo 仅使用 7x7x2 个.