本教程分为5个部分：

第1部分（本文）：理解 YOLO 的原理

第2部分：创建网络结构

第3部分：实现网络的前向传递

第4部分：目标分阈值和非极大值抑制

第5部分：网络的输入和输出

文章目录 [隐藏]

背景知识

理解卷积神经网络的原理，包括 Residual Blocks, skip connections, 和 Upsampling。

目标检测的定义、边界框回归、IoU 和非极大值抑制。

PyTorch 的基本用法。

什么是 YOLO?

YOLO 的全称是 You Only Look Once。它是一种基于深度卷积神经网络的目标检测器。我们先了解 YOLO 的工作原理。

全卷积神经网络 FCN

YOLO 仅仅使用卷积层，这种仅适用卷基层的网络我们称之为全卷积神经网络（Fully Convolutional Network）。YOLO 拥有 75 个卷积层，还有 skip connections 和 上采样 Upsampling 层。它使用步幅为 2 的卷积层对特征图进行下采样，而不是使用池化层，这有助于防止通常由池化导致的低级特征丢失。

作为 FCN，YOLO 对于输入图像的大小并没有要求。然而，在实践中，我们可能想要固定输入的大小，以防止后续一些问题的出现。这其中的一个重要原因是：如果我们希望按 batch 处理图像（batch 由 GPU 并行处理，这样可以提升速度），我们就需要固定所有图像的高度和宽度。这就需要将多个图像整合进一个大的 batch（将将许多 PyTorch Tensors 合并成一个）。

YOLO 通过 **stride（步幅）**对图像进行上采样。例如，如果网络的步幅是 32，则大小为 416×416 的输入图像将产生 13×13 的输出。

输出

一般来讲，卷积层所学习的特征会被传递到分类器/回归器进行预测（边界框的坐标、类别标签等）。

在 YOLO 中，预测是通过 1 x 1 的卷积层完成的。

现在要注意的是，我们的输出都是特征图（feature map），因为使用 1 x 1 的卷基层，所以每次输出的特征图都和之前的特征图是一样大小的。在 YOLO v3上，预测图就是每个可以预测固定数量边界框的单元格。

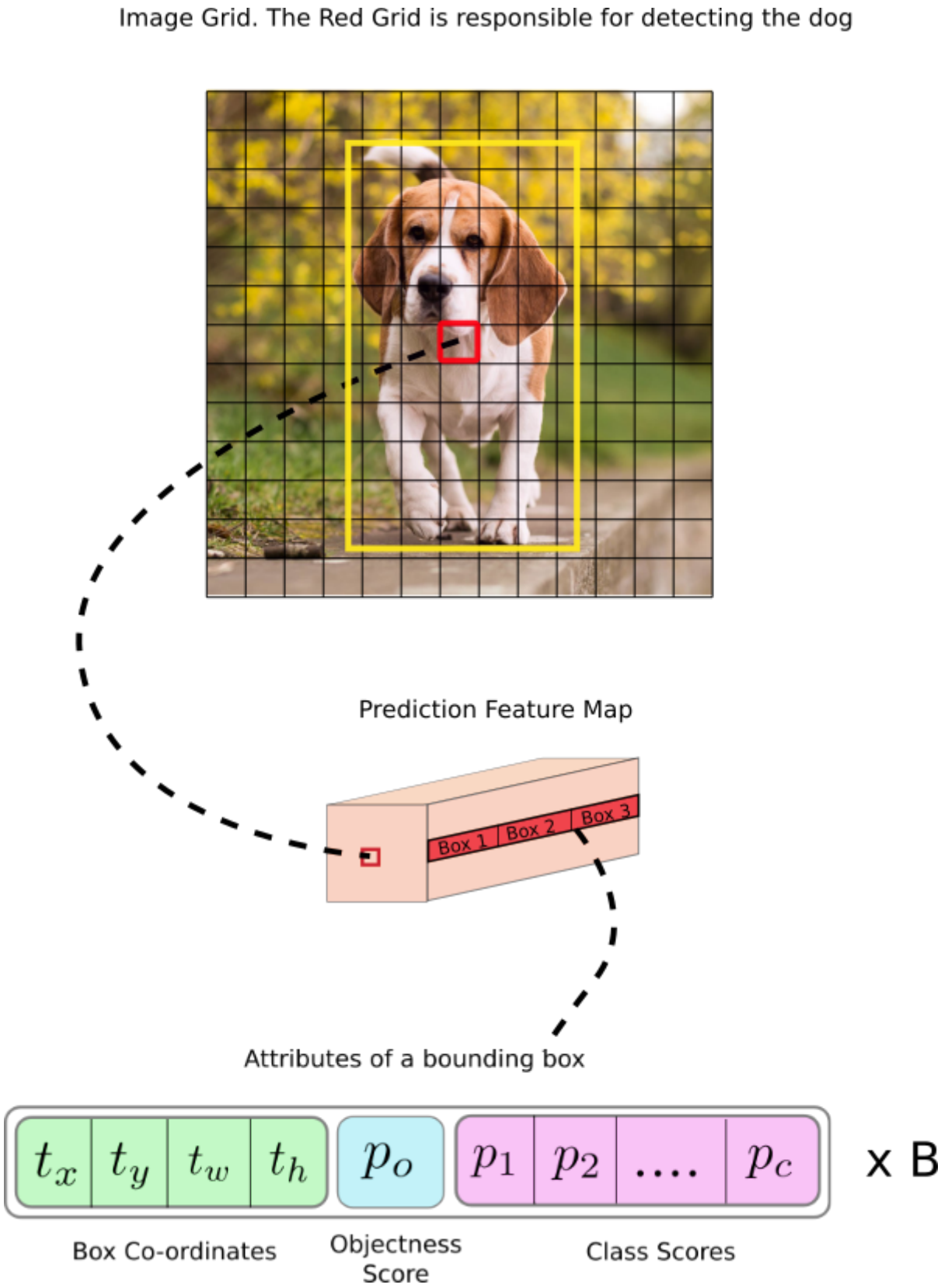
对于网络的深度，我们的特征图包含 `(B x (5 C))` 个条目， `B` 代表每个单元可以预测的边界框数量。根据 YOLO 的论文，这些 `B` 边界框中的每一个都可能专门用于检测某种对象。每个边界框都有 `5 C` 个属性，分别描述每个边界框的中心坐标、维度、objectness 分数和 C 类置信度。YOLO v3 在每个单元中预测 3 个边界框。

如果对象的中心位于单元格的感受野内，你会希望特征图的每个单元格都可以通过其中一个边界框预测对象。（感受野是输入图像对于单元格可见的区域。）

这与 YOLO 是如何训练的有关，只有一个边界框负责检测任意给定对象。首先，我们必须确定这个边界框属于哪个单元格。

因此，我们需要切分输入图像，把它拆成维度等于最终特征图的网格。

让我们思考下面一个例子，其中输入图像大小是 416×416，网络的步幅是 32。如之前所述，特征图的维度会是 13×13。随后，我们将输入图像分为 13×13 个网格。



输入图像中包含了真值对象框中心的网格会作为负责预测对象的单元格。在图像中，它是被标记为红色的单元格，其中包含了真值框的中心（被标记为黄色）。

现在，红色单元格是网格中第七行的第七个。我们现在使特征图中第七行第七个单元格（特征图中的对应单元格）作为检测狗的单元。

现在，这个单元格可以预测三个边界框。哪个将会分配给狗的真值标签？为了理解这一点，我们必须理解锚点的概念。

请注意，我们在这里讨论的单元格是预测特征图上的单元格，我们将输入图像分隔成网格，以确定预测特征图的哪个单元格负责预测对象。

锚点框（Anchor Box）

预测边界框的宽度和高度看起来非常合理，但在实践中，训练会带来不稳定的梯度。所以，现在大部分目标检测器都是预测对数空间（log-space）变换，或者预测与预训练默认边界框（即锚点）之间的偏移。

然后，这些变换被应用到锚点框来获得预测。YOLO v3 有三个锚点，所以每个单元格会预测 3 个边界框。

回到前面的问题，负责检测狗的边界框的锚点有最高的 IoU，且有真值框。

预测

下面的公式描述了网络输出是如何转换，以获得边界框预测结果的。

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

中心坐标

注意：我们使用 sigmoid 函数进行中心坐标预测。这使得输出值在 0 和 1 之间。原因如下：

正常情况下，YOLO 不会预测边界框中心的确切坐标。它预测：

与预测目标的网格单元左上角相关的偏移；

使用特征图单元的维度（1）进行归一化的偏移。

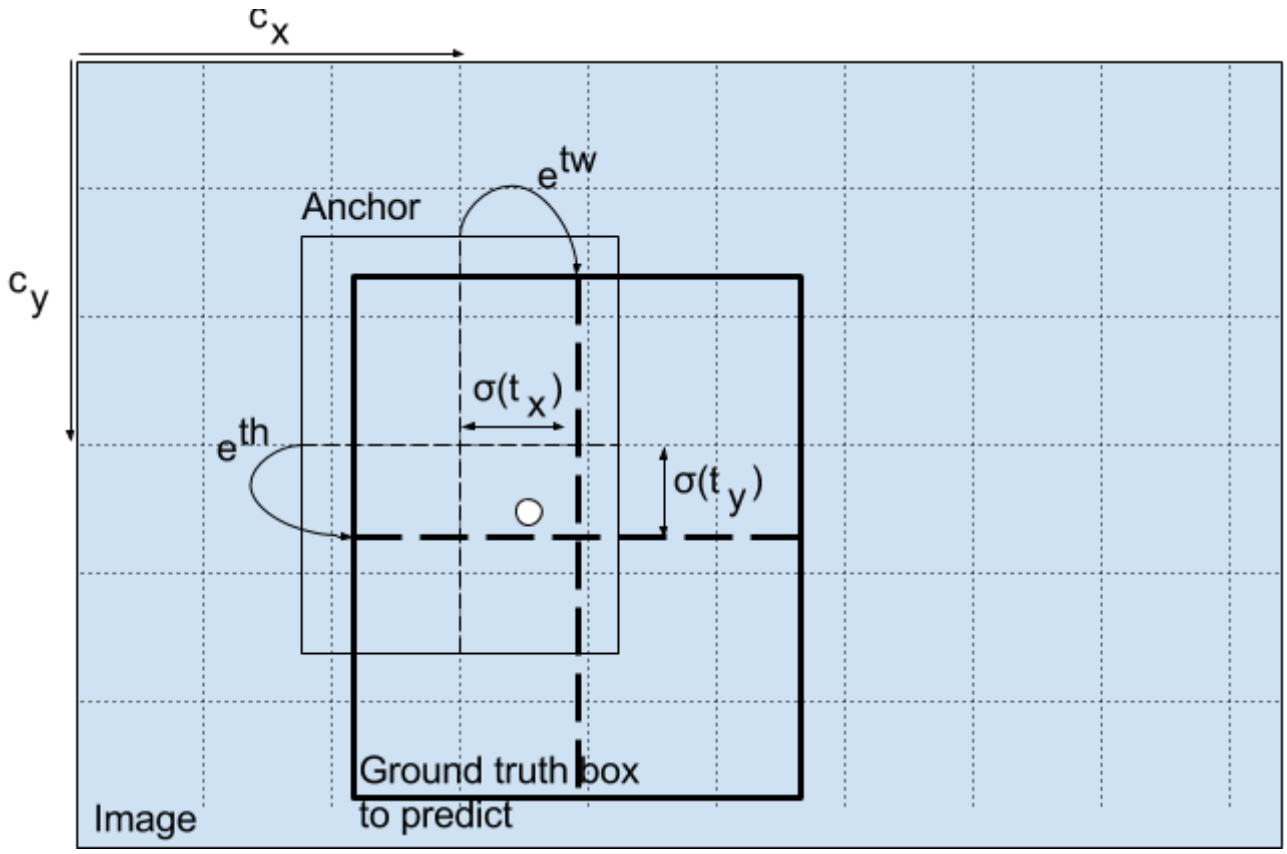
以我们的图像为例。如果中心的预测是 (0.4, 0.7)，则中心在 13 x 13 特征图上的坐标是 (6.4, 6.7)（红色单元的左上角坐标是 (6,6)）。

但是，如果预测到的 x,y 坐标大于 1，比如 (1.2, 0.7)。那么中心坐标是 (7.2, 6.7)。注意该中心在红色单元右侧的单元中，或第 7 行的第 8 个单元。这打破了 YOLO 背后的理论，因为如果我们假设红色框负责预测目标狗，那么狗的中心必须在红色单元中，不应该在它旁边的网格单元中。

因此，为了解决这个问题，我们对输出执行 sigmoid 函数，将输出压缩到区间 0 到 1 之间，有效确保中心处于执行预测的网格单元中。

边界框的维度

我们对输出执行对数空间变换，然后乘锚点，来预测边界框的维度。



检测器输出在最终预测之前的变换过程，图源：http://christopher5106.github.io/

得出的预测 bw 和 bh 使用图像的高和宽进行归一化。即，如果包含目标（狗）的框的预测 bx 和 by 是 (0.3, 0.8)，那么 13 x 13 特征图的实际宽和高是 (13 x 0.3, 13 x 0.8)。

Objectness 分数

Object 分数表示目标在边界框内的概率。红色网格和相邻网格的 Object 分数应该接近 1，而角落处的网格的 Object 分数可能接近 0。

objectness 分数的计算也使用 sigmoid 函数，因此它可以被理解为概率。

类别置信度

类别置信度表示检测到的对象属于某个类别的概率（如狗、猫、香蕉、汽车等）。在 v3 之前，YOLO 需要对类别分数执行 softmax 函数操作。

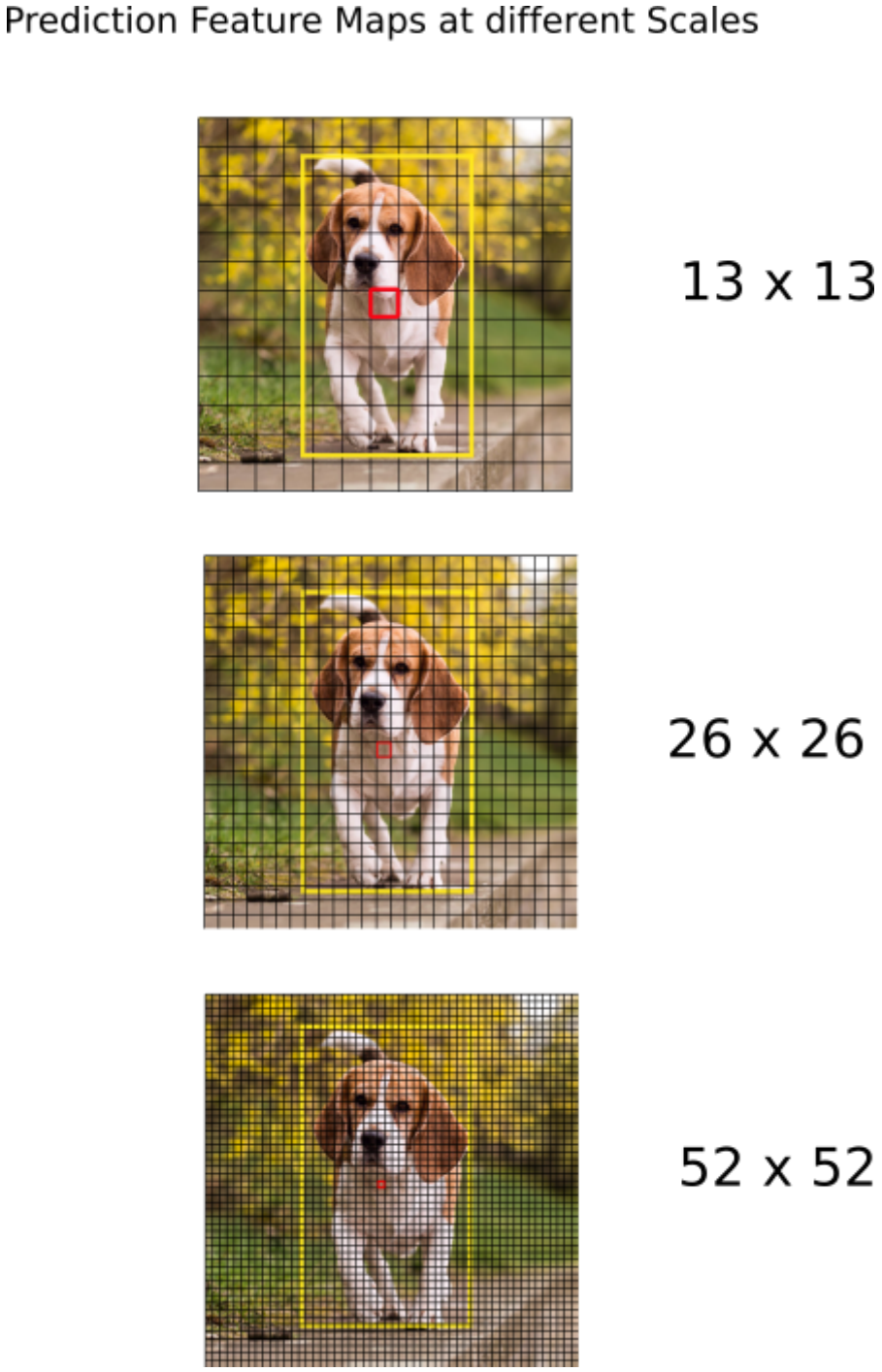
但是，YOLO v3 舍弃了这种设计，作者选择使用 sigmoid 函数。因为对类别分数执行 softmax 操作的前提是类别是互斥的。简言之，如果对象属于一个类别，那么必须确保其不属于另一个类别。这在我们设置检测器的 COCO 数据集上是正确的。但是，当出现类别「女性」（Women）和「人」（Person）时，该假设不可行。这就是作者选择不使用 Softmax 激活函数的原因。

在不同尺度上的预测

YOLO v3 在 3 个不同尺度上进行预测。检测层用于在三个不同大小的特征图上执行预测，特征图步幅分别是 32、16、8。这意味着，当输入图像大小是 416 x 416 时，我们在尺度 13 x 13、26 x 26 和 52 x 52 上执行检测。

该网络在第一个检测层之前对输入图像执行下采样，检测层使用步幅为 32 的层的特征图执行检测。随后在执行因子为 2 的上采样后，并与前一个层的特征图（特征图大小相同）拼接。另一个检测在步幅为 16 的层中执行。重复同样的上采样步骤，最后一个检测在步幅为 8 的层中执行。

在每个尺度上，每个单元使用 3 个锚点预测 3 个边界框，锚点的总数为 9（不同尺度的锚点不同）。



作者称这帮助 YOLO v3 在检测较小目标时取得更好的性能，而这正是 YOLO 之前版本经常被抱怨的地方。上采样可以帮助该网络学习细粒度特征，帮助检测较小目标。

输出处理

对于大小为 416 x 416 的图像，YOLO 预测 ((52 x 52) (26 x 26) 13 x 13)) x 3 = 10647 个边界框。但是，我们的示例中只有一个对象——一只狗。那么我们怎么才能将检测次数从 10647 减少到 1 呢?

目标置信度阈值：首先，我们根据它们的 objectness 分数过滤边界框。通常，分数低于阈值的边界框会被忽略。

非极大值抑制：非极大值抑制（NMS）可解决对同一个图像的多次检测的问题。例如，红色网格单元的 3 个边界框可以检测一个框，或者临近网格可检测相同对象。

Multiple Grids may detect the same object
NMS is used to remove multiple detections

实现

YOLO 只能检测出属于训练所用数据集中类别的对象。我们的检测器将使用官方权重文件，这些权重通过在 COCO 数据集上训练网络而获得，因此我们可以检测 80 个对象类别。

该教程的第一部分到此结束。这部分详细讲解了 YOLO 算法。如果你想深度了解 YOLO 的工作原理、训练过程和与其他检测器的性能规避，可阅读原始论文：

1. YOLO V1: You Only Look Once: Unified, Real-Time Object Detection
2. YOLO V2: YOLO9000: Better, Faster, Stronger
3. YOLO V3: An Incremental Improvement