系列：PYTORCH 中的 YOLO 对象探测器

如何在 PyTorch 中从头开始实现 YOLO （v3） 对象探测器：第 1 部分

有关从头开始构建 YOLO v3 探测器的教程，详细说明了如何从配置文件创建网络体系结构、加载权重和设计输入/输出管道。

对象检测是一个从深度学习的最新发展中获益匪浅的域。近年来，人们开发许多对象检测算法，其中一些包括YLO、SSD、掩码RCNN和视网膜网。

在过去的几个月里，我一直在研究实验室改进物体检测。从这次体验中，最大的收获之一是认识到学习对象检测的最佳方法就是从头开始自己实现算法。这正是我们将在本教程中完成的内容。

我们将使用PyTorch实现基于YLO v3的对象探测器，这是更快的对象检测算法之一。

本教程的代码设计为在 Python 3.5 和 PyTorch **0.4** 上运行。它可以在这个Github存储库上找到它的全部。

本教程分为 5 个部分：

1. 第 1 部分（本部分）： 了解 YOLO 的工作原理
2. 第 2 部分：创建网络体系结构的层

3. 第3部分：实现网络的转发传递
4. 第 4 部分：对象度分数阈值和非最大抑制
5. 第5部分：设计输入和输出管道

先决条件

- 您应该了解卷积神经网络是如何工作的。这还包括剩余块、跳过连接和上采样的知识。
- 什么是对象检测、边界框回归、IoU 和非最大抑制。
- 基本 PyTorch 用法。您应该能够轻松创建简单的神经网络。

我已经在帖子的末尾提供了链接， 以防你在任何方面都达不到。

什么是约洛?

约洛代表你只看一次。它是一种对象检测器，它使用深度卷积神经网络所学的功能来检测物体。在用代码摆脱脏东西之前，我们必须了解 YOLO 的工作原理。

完全卷积神经网络

YOLO 只使用卷积层，使它成为一个完全卷积网络 （FCN）。它有75个卷积层，跳过连接和上采样层。未使用池形式，并且使用带步幅 2 的卷积图层来对要素贴图进行下采样。这有助于防止通常归因于池的低级功能丢失。

作为 FCN，YOLO 与输入图像的大小是不变的。然而，在实践中，我们可能希望坚持一个恒定的输入大小，由于各种问题，只显示他们的头，当我们实现算法。

其中一个大问题是，如果我们想要分批处理图像（图像可以由GPU并行处理，从而加快速度），我们需要拥有固定高度和宽度的所有图像。这是需要将多个图像串联成一个大

批（将许多 PyTorch tensor 串联到一个）

网络通过称为网络步幅的系数**对图像进行**向下采样。例如，如果网络的步幅为 32，则大小为 416 x 416 的输入图像将产生大小为 13 x 13 的输出。通常，**网络中任何图层的步幅等于图层输出小于网络输入图像的系数。**

解释输出

通常，（如所有对象探测器的情况），卷积层所学的要素将传递到分类器/回归器上，该器进行检测预测（边界框的坐标、类标签等）。

在 YOLO 中，使用使用 1 x 1 卷积的卷积层完成预测。

现在，首先要注意的是，我们的**输出是一个要素图**。由于我们使用了 1 x 1 卷积，因此预测地图的大小正是之前要素地图的大小。在 YOLO v3（及其后代）中，解释此预测映射的方式是每个单元格都可以预测固定数量的边界框。

> *虽然描述要素图中一个单元的技术正确术语是神经元，但称它为细胞会使它在我们的上下文中更加直观。*

从深度上讲，要素图中包含（B x（5 + C））条目。B 表示每个单元格可以预测的边界框数。根据本文，这些B边界框可能专门检测某种物体。每个边界框都有*5 ° C*属性，用于描述每个边界框的中心坐标、尺寸、*对象度分数*和 C 类置信度。YOLO v3 预测每个单元格有 3 个边界框。

如果对象的中心位于该单元格的接受字段中，则要素贴图的每个单元格都通过其中一个边界框预测对象。（接收字段是单元格可见的输入图像区域。有关进一步澄清，请参阅

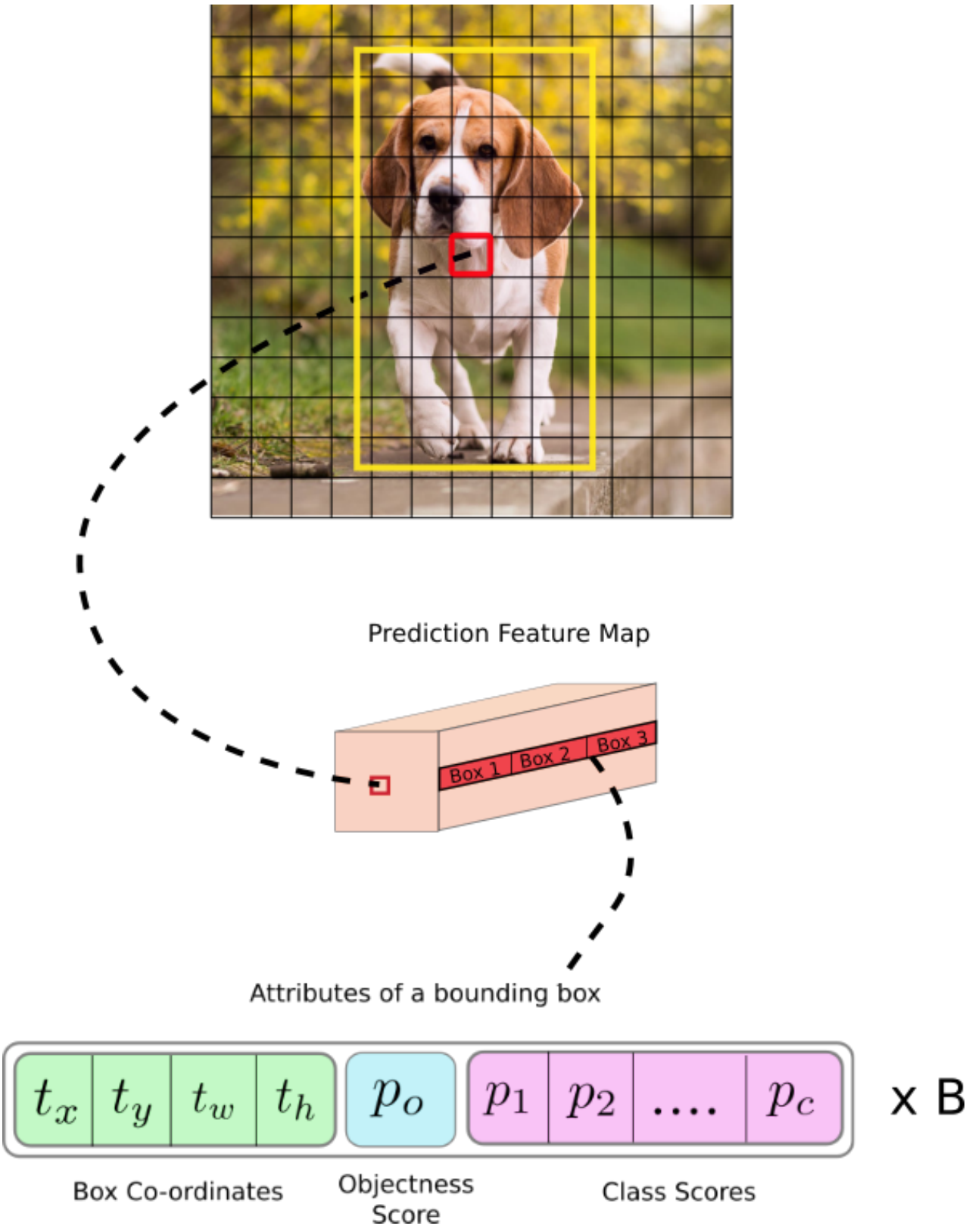
卷积神经网络上的链接）。

这与 YOLO 的训练方法有关，其中只有一个边界框负责检测任何给定的对象。首先，我们必须确定这个边界框属于哪个单元格。

为此，我们将输入**图像划分**为与最终要素图相等的尺寸网格。

让我们考虑下面的示例，其中输入图像为 416 x 416，网络步幅为 32。如前面指出，要素地图的尺寸将为 13 x 13。然后，我们将输入图像分成 13 x 13 个单元格。

Image Grid. The Red Grid is responsible for detecting the dog



然后，选择*包含对象地面*真理框中心的单元格（在输入图像上）作为负责预测对象的单元格。在图像中，它是标记为红色的单元格，其中包含地面真理框的中心（标记为黄色）。

现在，红细胞是网格上第 7 行的第 7 个单元格。现在，我们将要素图第 7 行中的第7 个单

元格（要素图上的相应单元格）分配给负责检测狗的单元格。

现在，这个单元格可以预测三个边界框。哪一个将被分配到狗的地面真相标签上？为了理解这一点，我们必须绕一个锚的概念。

> *请注意，我们在这里谈论的单元格是预测要素图上的单元格。我们将输入**图像划分**为网格，只是为了确定预测要素**图的哪个单元格**负责预测*

锚箱

预测边界框的宽度和高度可能有意义，但在实践中，这会导致训练期间的梯度不稳定。相反，大多数现代对象探测器预测日志空间转换，或简单地偏移到预定义的默认边界框称为**锚点**。

然后，这些变换应用于锚点框以获取预测。YOLO v3 有三个锚点，这将导致预测每个单元格三个边界框。

回到我们之前的问题， 负责检测狗的界框将是一个锚有最高的 IoU 与地面真相框。

进行预测

以下公式描述了如何转换网络输出以获取边界框预测。

$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

bx， *by*， *bw*， *bh*是我们预测的 x， y 中心坐标、 宽度和高度。*tx*， *ty*， *tw*，这是网络输出的。*cx 和 cy*是网格的左上角坐标。*pw*和*ph*是框的锚点尺寸。

中心坐标

请注意，我们正在通过 sigmoid 函数运行中心坐标预测。这会强制输出的值介于 0 和 1 之间。为什么会这样？跟我来

通常，YOLO 不会预测边界框中心的绝对坐标。它预测偏移量，这些偏移量是：

- 相对于预测对象的网格单元格的左上角。
- 由要素贴图中的单元格尺寸规范化，即 1。

例如，考虑我们的狗形象的情况。如果中心的预测是 （0.4， 0.7），则这意味着中心位于 13 x 13 要素图上的 （6.4， 6.7）。（因为红细胞的左上角坐标为 （6，6）。

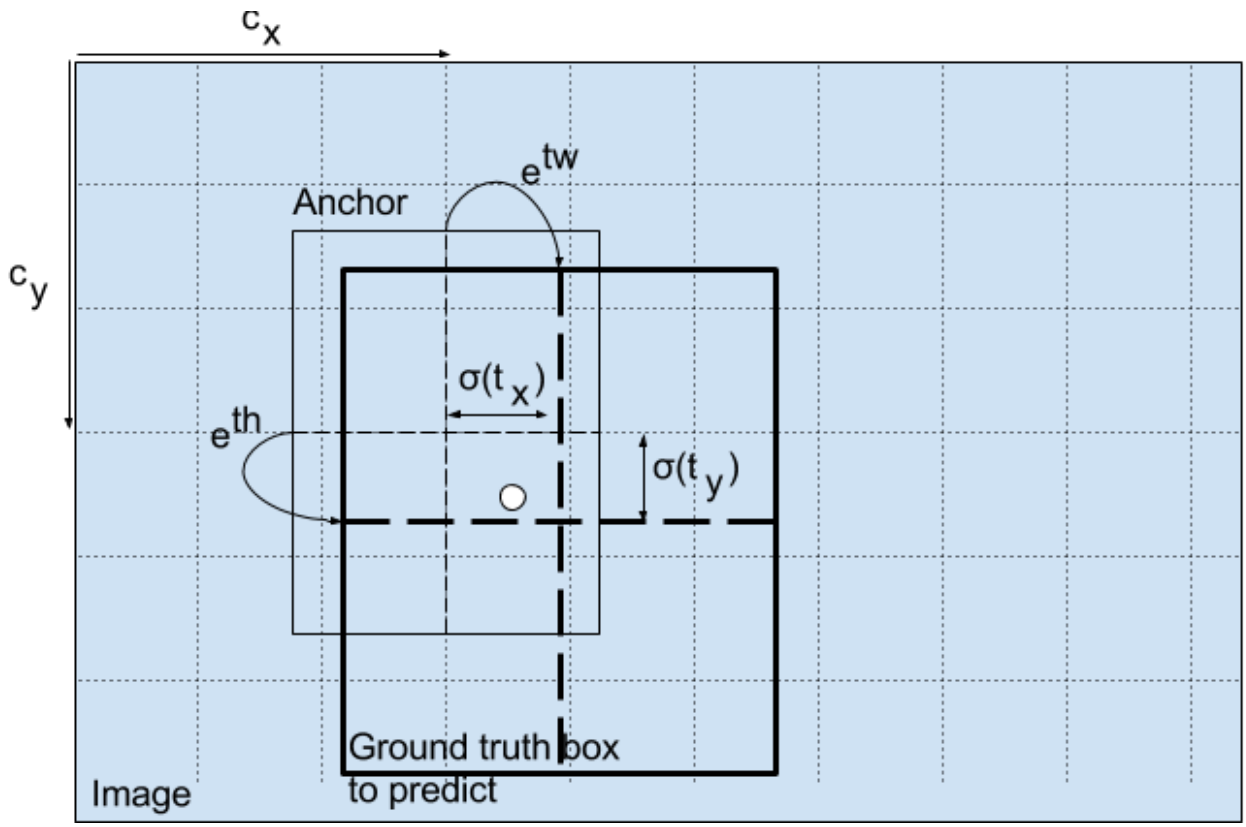
但是，等待，如果预测的 x，y 坐标大于 1，例如 （1.2， 0.7），会发生什么。这意味着中心位于 （7.2， 6.7）。请注意，中心现在位于单元格中，就在单元格的右边，即第 7

行的第 8 个单元格。**这打破了YLO**背后的理论，因为如果我们假设红盒负责预测狗，狗的中心必须位于红细胞，而不是它旁边的一个。

因此，为了解决此问题，输出通过 sigmoid 函数传递，该函数将输出压缩到 0 到 1 的范围内，从而有效地将中心留在正在预测的网格中。

边界框的尺寸

通过将日志空间变换应用于输出，然后与锚点相乘来预测边界框的尺寸。



如何转换探测器输出以给出最终预测。图像积分。http://christopher5106.github.io/

由此产生的预测，*bw*和*bh*，由图像的高度和宽度进行规范化。（培训标签选择方式）。因此，如果预测*bx 和包含*狗的框的预测是 （0.3， 0.8），则 13 x 13 要素图上的实际宽度和高度为 （13 x 0.3， 13 x 0.8）。

对象分数

对象分数表示对象包含在边界框中的概率。对于红色网格和相邻网格，它应该是近 1，而对于角的网格，它应该是近 0。

对象度分数也通过 sigmoid 传递，因为它要被解释为概率。

班级信心

类置信度表示属于特定类（狗、猫、香蕉、汽车等）的检测到的对象的概率。在 v3 之前，YOLO 习惯于将班级分数软到最大。

但是，该设计选择在 v3 中已被丢弃，作者已选择使用 sigmoid 代替。原因是 Softmaxing 类分数假定类是互斥的。简单地说，如果一个对象属于一个类，那么它保证它不能属于另一个类。这对于COCO数据库来说是正确的，我们将基于该数据库。

然而，当我们有像女人和人这样的课程时，*这种假设*可能*站不住脚*。这就是作者避开使用 Softmax 激活的原因。

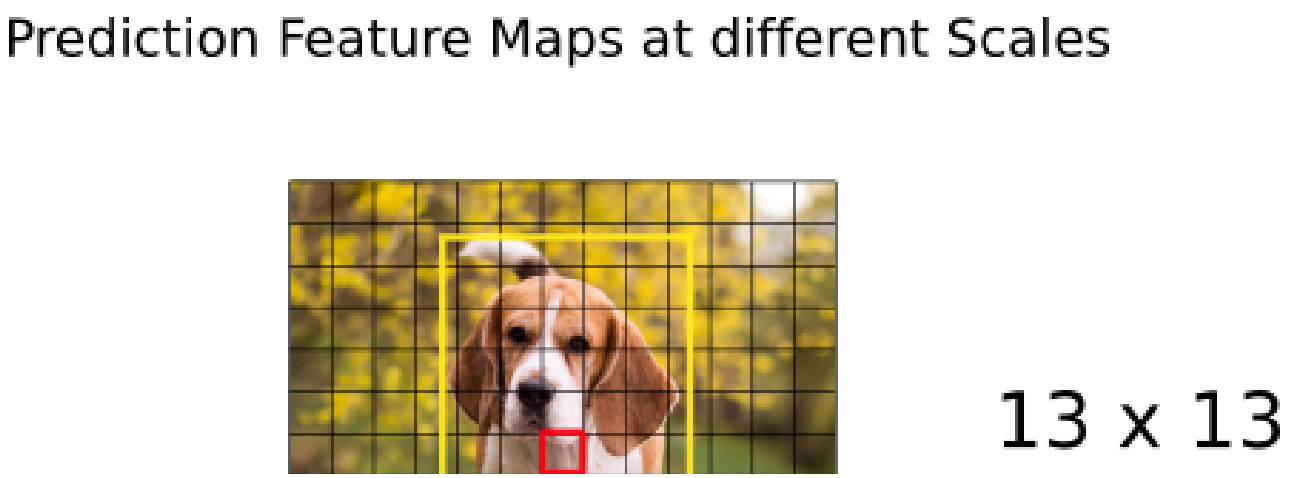
不同比例的预测。

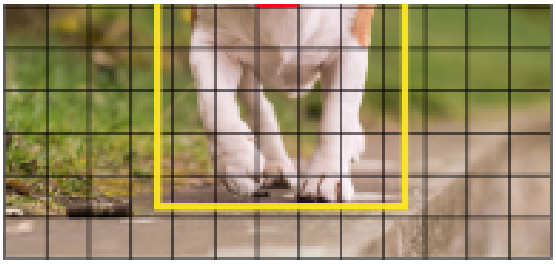
YOLO v3 对 3 个不同比例进行预测。检测层用于在三种不同尺寸的特征图上进行检测，分别有**32**、**16**、**8**步幅。这意味着，输入为 416 x 416 时，我们在刻度 13 x 13、26 x 26 和 52 x 52 上进行检测。

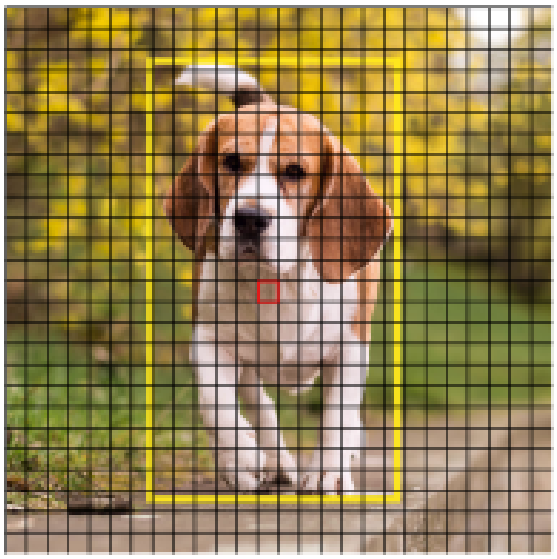
网络将输入图像向下采样，直到第一个检测层，其中使用步幅为 32 的图层的特征映射进行检测。此外，图层的上采样系数为 2，并与其他具有相同要素贴幅大小的图层的要素

贴图串联。另一个检测现在在步幅 16 的层上进行。重复相同的上采样过程，并在步幅 8 层进行最终检测。

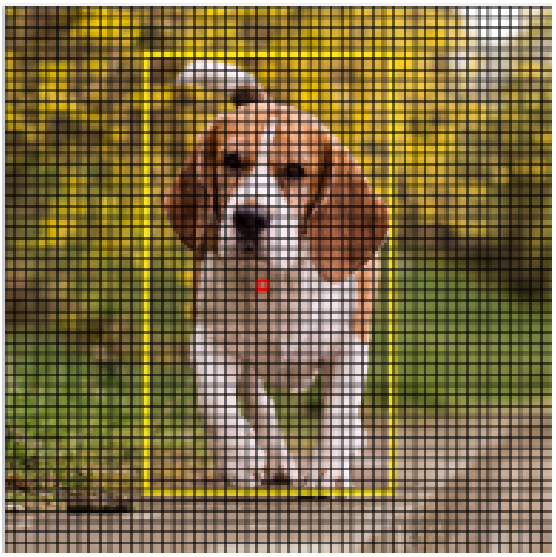
在每个比例下，每个单元格使用 3 个锚点预测 3 个边界框，使使用的锚点总数为 9。（不同比例的锚点不同）







26 x 26



52 x 52

作者报告说，这有助于YLO v3更好地检测小物体，这是YOLO早期版本的经常投诉。

Upsampling 可以帮助网络学习细粒度功能，这些功能有助于检测小对象。

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

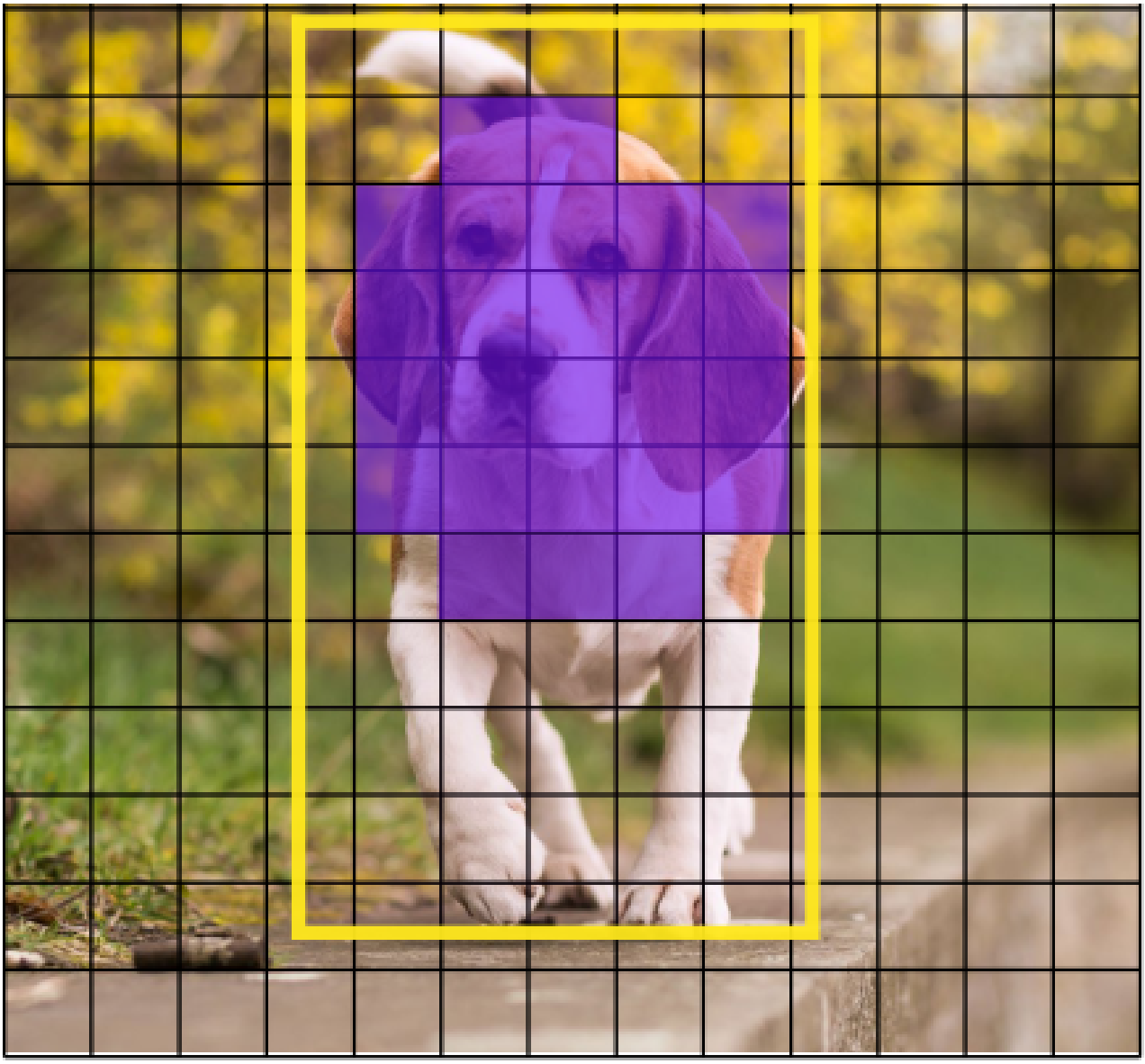
3 = **10647 边界框**。然而，在我们的图像的情况下，只有一个对象，一只狗。我们如何将检测从 10647 减至 1？

按对象置信度阈值

首先，我们根据框的客观性分数进行筛选。通常，忽略分数低于阈值的框。

非最大抑制

NMS 打算解决同一图像的多次检测问题。例如，红色网格单元格的所有 3 个边界框可能检测到一个框，或者相邻的单元格可能检测到同一个对象。



如果你不知道 Nms， 我提供了一个网站的链接， 解释同样的。

我们的实施

YOLO 只能检测属于用于训练网络的数据集中存在的类的对象。我们将使用我们的探测器的官方重量文件。这些权重是通过在COCO数据集上训练网络获得的，因此我们可以检测80个对象类别。

第一部分就是这样。这篇文章对 YOLO 算法的介绍足够，使您能够实现探测器。但是，

如果您想要深入了解 YOLO 的工作原理、训练方式以及与其他探测器相比的性能，您可以阅读原始文件，下面我提供了这些文件的链接。

这就是这部分。下一部分，我们实现各种层需要放在一起的探测器。

进一步阅读

1. YOLO V1：您只看一次：统一、实时的对象检测
2. Yolo V2： YOLO9000： 更好，更快，更强
3. YOLO V3：增量改进
4. 卷积神经网络
5. 边界框回归（附录 C）
6. Iou
7. 非最大超
8. 皮托奇官方教程

Ayoosh Kathuria 目前是印度国防研究与发展组织的实习生，他正致力于改进颗粒状视频中的物体检测。当他不工作时， 他要么睡觉， 要么在吉他上弹粉红色的弗洛伊德。你可以和他联系LinkedIn，或者看看他在Github 做什么