https://blog.csdn.net/leviopku/article/details/82588059

https://blog.csdn.net/leviopku/article/details/82588959

https://blog.csdn.net/leviopku/article/details/82660381#commentBox

YOLO v1基本思想：输入图片被划分为7x7个单元格，每个单元格独立做预测。预测框的位置、大小和物体分类都通过CNN暴力预测出来

YOLOv1 的价值，即v2/v3还保留的特性有：

1. leaky relu

相比普通ReLU，leaky并不会让负数直接为0，而是乘以一个很小的系数(恒定)，保留负数输出，但衰减负数输出；公式如下：

$$y = \begin{cases} x, x > 0 \\ 0.1x, otherwise \end{cases}$$

2. **分而治之**，用网格来划分图片区域，每块区域独立检测目标；

3. **端到端训练**。损失函数的反向传播可以贯穿整个网络，这也是one-stage检测算法的优势

YOLO v2的改进

yolo_v2的一大特点是可以”tradeoff“，翻译成中文就是”折中”。v2可以在速度和准确率上进行tradeoff，比如在67帧率下，v2在VOC2007数据集的mAP可以达到76.8；在40帧率下，mAP可以达到78.6。这样，v2就可以适应多种场景需求，在不需要快的时候，它可以把精度做很高，在不需要很准确的时候，它可以做到很快。

1. 每层卷积层后都加BN，mAP提高2%。
2. 输入可以有更高的分辨率，原来是224，现在是448，先在Imagenet上跑10个epoch，mAP提升4%
3. 尝试加入anchor机制。YOLO通过全连接层直接预测bounding box的坐标，YOLOv2去除全连接层，使用anchor框来预测bounding box。
4. 聚类anchor。在训练集的bbox上用了kmeans聚类来自动找到框，如果用标准k-means(使用欧几里得距离)，较大box会比较小box出现更多的错误。然而，我们真正想要的是能够使IOU得分更高的优选项，与box的大小没有关系。因此，对于距离判断，作者用了：

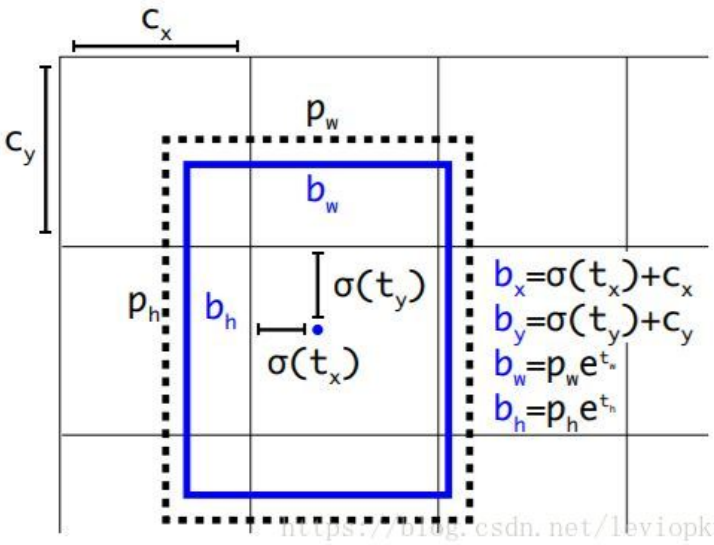
d(box, centroid) = 1 - IOU(box, centroid) 。最终k=5

5. 直接预测位置。Faster RCNN中的RPN预测box时，预测的是偏移量

$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$

例如，预测出t_x = 1意味着把框整体向右移动了一个框的距离。
这个公式没有加以限制条件，所以任何anchor box都可以偏移到图像任意的位置上。随机初始化模型会需要很长一段时间才能稳定产生可靠的offsets(偏移量)。

我们没有预测偏移量，而是遵循了YOLO的方法，直接预测对于网格单元的相对位置。



直接预测(x, y)，就像yolo_v1的做法，不过v2是预测一个相对位置，相对单元格的左上角的坐标(如上图所示)。当(x, y)被直接预测出来，那整个bounding box还差w和h需要确定。yolo_v2的做法是既有保守又有激进，x和y直接暴力预测，而w和h通过bounding box prior的调整来确定。yolo为每个bounding box预测出5个坐标(t_x, t_y, t_w, t_h, t_o)

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(\text{object}) * IOU(b, \text{object}) &= \sigma(t_o)
\end{aligned}
$$

看上面的公式也可以看出，b-box的宽和高也是同时确定出来，并不会像RPN那样通过regression来确定。p_w和p_h都是kmeans聚类之后的prior(模板框)的宽和高，yolo直接预测出偏移量t_w和t_h，相当于直接预测了bounding box的宽和高。使用聚类搭配直接位置预测法的操作，使得模型上升了5个百分点。

YOLO v3的改进：

Yolo_v3作为yolo系列目前最新的算法，对之前的算法既有保留又有改进。先分析一下yolo_v3上保留的东西：

1. “分而治之”，从yolo_v1开始，yolo算法就是通过划分单元格来做检测，只是划分的数量不一样。
2. 采用"leaky ReLU"作为激活函数。
3. 端到端进行训练。一个loss function搞定训练，只需关注输入端和输出端。
4. 从yolo_v2开始，yolo就用batch normalization作为正则化、加速收敛和避免过拟合的方法，把BN层和leaky relu层接到每一层卷积层之后。
5. 多尺度训练。在速度和准确率之间tradeoff。想速度快点，可以牺牲准确率；想准确率高点儿，可以牺牲一点速度。

改进：

1. backbone:v2是darknet-19,v3是darknet-53，Darknetconv2d_BN_Leaky，是yolo_v3的基本组件。就是卷积+BN+Leaky relu提供了替换backbone——tiny darknet，可以达到轻量高速。

Model	Top-1	Top-5	Ops	Size
AlexNet	57.0	80.3	2.27 Bn	238 MB
Darknet Reference	**61.1**	**83.0**	**0.81 Bn**	28 MB
SqueezeNet	57.5	80.3	2.17 Bn	4.8 MB
Tiny Darknet	58.7	81.7	0.98 Bn	**4.0 MB**

2.输出了3个不同尺度的feature map，predictions across scales。借鉴了FPN，用上采样的方法实现多尺度的feature map。

3.在bbox预测上。对于v3而言，在prior这里的处理有明确解释：选用的b-box priors 的k=9，对于tiny-yolo的话，k=6

每个anchor prior(名字叫anchor prior，但并不是用anchor机制)就是两个数字组成的，一个代表高度另一个代表宽度。

v3对b-box进行预测的时候，采用了logistic regression。这一波操作sao得就像RPN中的线性回归调整b-box。v3每次对b-box进行predict时，输出和v2一样都是

$(t_x, t_y, t_w, t_h, t_o)$

，然后通过公式1计算出绝对的(x, y, w, h, c)。

logistic回归用于对anchor包围的部分进行一个目标性评分(objectness score)，即这块位置是目标的可能性有多大。这一步是在predict之前进行的，可以去掉不必要anchor，可以减少计算量。

不同于faster R-CNN的是，yolo_v3只会对1个prior进行操作，也就是那个最佳prior。而logistic回归就是用来从9个anchor priors中找到objectness score(目标存在可能性得分)最高的那一个。logistic回归就是用曲线对prior相对于 objectness score映射关系的线性建模。

在评论里有同学问我关于输出的问题，看来我在这里没有说的很清楚。了解v3输出的输出是至关重要的。

第一点， 9个anchor会被三个输出张量平分的。根据大中小三种size各自取自己的anchor。

第二点，每个输出y在每个自己的网格都会输出3个预测框，这3个框是9除以3得到的，这是作者设置的，我们可以从输出张量的维度来看，13x13x255。255是怎么来的呢，3*(5+80)。80表示80个种类，5表示位置信息和置信度，3表示要输出3个prediction。在代码上来看，3*(5+80)中的3是直接由num_anchors//3得到的。

第三点，作者使用了logistic回归来对每个anchor包围的内容进行了一个目标性评分(objectness score)。根据目标性评分来选择anchor prior进行predict，而不是所有anchor prior都会有输出。

4.损失函数

在v1中使用了一种叫sum-square error的损失计算方法，就是简单的差方相加而已。

我们知道，在目标检测任务里，有几个关键信息是需要确定的:

$$(x, y), (w, h), class, confidence$$

根据关键信息的特点可以分为上述四类，损失函数应该由各自特点确定。最后加到一起就可以组成最终的loss_function

除了w，h的损失函数依然采用总方误差之外，其他部分的损失函数用的是二值交叉熵

- binary_crossentropy交叉熵损失函数，一般用于二分类：

$$loss = -\sum_{i=1}^{n} \hat{y}_i log y_i + (1-\hat{y}_i) log(1-\hat{y}_i)$$

$$\frac{\partial loss}{\partial y} = -\sum_{i=1}^{n} \frac{\hat{y}_i}{y_i} - \frac{1-\hat{y}_i}{1-y_i}$$

这个是针对概率之间的损失函数，你会发现只有y_i和$\hat{y}_i$是相等时，loss才为0，否则loss就是为一个正数。而且，概率相差越大，loss就越大。这个神奇的度量概率距离的方式称为交叉熵。