YOLO算法详解+完整代码详解

@浪里小白龙 2019-06-06 13:49:53

YOLOv2

特点：darknet网路；使用先验框；多尺度训练

针对YOLOv1的缺点进行改进，下图是YOLOv2做的一些改进，大部分的改进都能提升mAP

	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**

Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

图片来自：https://blog.csdn.net/wfei101/article/details/79398563

Batch normalization（每个卷积之后进行批归一化）

批归一化可以提升模型的收敛速度，而且起到正则化的效果，降低模型的的过拟合，YOLOv2中，每个卷积层后面都添加了batch normalization，不再使用dropout，mAP提高了2.4%

High-resolution classifier（使用448*448大小的图片训练darknet-19）

YOLOv1中GoogleNet训练的图片大小是224*224，YOLOv2在ImageNet数据上使用448*448来finetune分类网络这一中间过程（10epochs）,使得模型在监测数据集之前已经适用高分别率输入

Convolutional with anchor boxes（采用先验框）

YOLOv1最后直接使用全连接层对边界框进行预测，其中边界框的高度是相对整张照片大小的，而由于各个图片中存在不同尺寸和长宽比的物体，YOLOv1在训练过程中学习适应不同物体的形状是比较困难的，这也导致YOLOv1在精确定位方面的表现较差。

YOLOv2借鉴Faster-CNN的RPN网络的先验框，PRN对CNN特征提取器得到的特征图进行卷积来预测每个位置的边界框以及置信度（是否有目标），并且各个位置设置不同尺寸和比例的先验框，所有RPN预测的是边界框相对于先验框的偏移值，使用先验框使得更容易学习

为了使检测框的分辨率更高，移除其中的一个pool层，在检测模型中输入是**416*416**大小的，YOLOv2模型下采样总步长是32，因此得到的特征图大小为13*13，只有一个中心位置。对于一些大的物体，他们的中心点落在图片中心位置，此时使用特征图的一个中心点去预测物体的边界框相对容易一些。**所以YOLOv2中包保证最终的特征图有奇数个位置**

YOLOv1中每个gird cell值预测一套分类概率值（其实是置信度下的条件概率值）供两个boxes分享。

YOLOv2使用了anchor boxes之后，每个位置的各个anchor box都单独预测一套分类概率值

使用了anchor box之后mAP稍微下降（原因是什么？）但是召回率有81%提升到88%。

Dimension clusters（使用聚类选择先验框的个数）

在Faster-RCNN和SSD中，先验框的长和宽是主观手动设置的，设置的先验框比较合适则更容易学习，反之。因此YOLOv2对训练集的边界框做K-MEANS聚类分析。先验框的主要目的是为了使得预测框和ground truth的IOU更好，所以聚类分析时选用box与聚类中心box之间的IOU值作为距离标准：

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

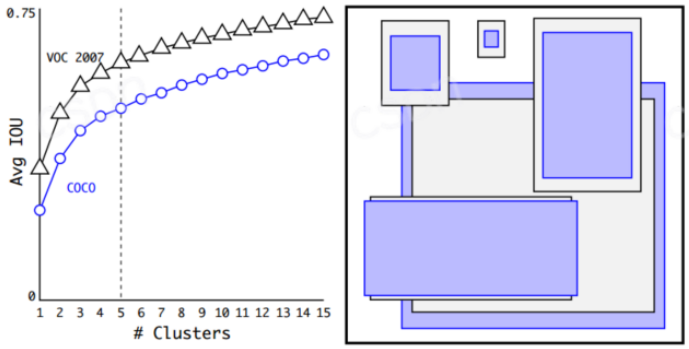


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that $k = 5$ gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. Both sets of priors favor thinner, taller boxes while COCO has greater variation in size than VOC.

图片来自：https://blog.csdn.net/wfei101/article/details/79398563

上图中随着聚类中心的增加，平均IOU也在增加，但是综合模型复杂度和召回率，作者选了**5个聚类中心**作为先验框如右边的图像所示：**所以就是设置了5个先验框**

COCO: (0.57273, 0.677385), (1.87446, 2.06253), (3.33843, 5.47434), (7.88282, 3.52778), (9.77052, 9.16828)

VOC: (1.3221, 1.73145), (3.19275, 4.00944), (5.05587, 8.09892), (9.47112, 4.84053), (11.2364, 10.0071)

相对于预测的特征图（13*13）

New network：Darknet-19

使用新设计的基础模型（特征提取器），称作darknet-19，包括19个卷积层5个maxpooling层，darknet的设计与VGG16的设计原理一致，主要采用3*3卷积，采用2*2max pooling层之后，特征图维度降低2倍，而同时特征图的channels增加2倍，最后采用global avgpooling做预测。并在3*3卷积之间添加1*1卷积压缩通道。

darknet每个卷积层之后使用了batch normalization。

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 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 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 × 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			

Table 6: Darknet-19.

图片来自：https://blog.csdn.net/mieleizhi0522/article/details/79887066

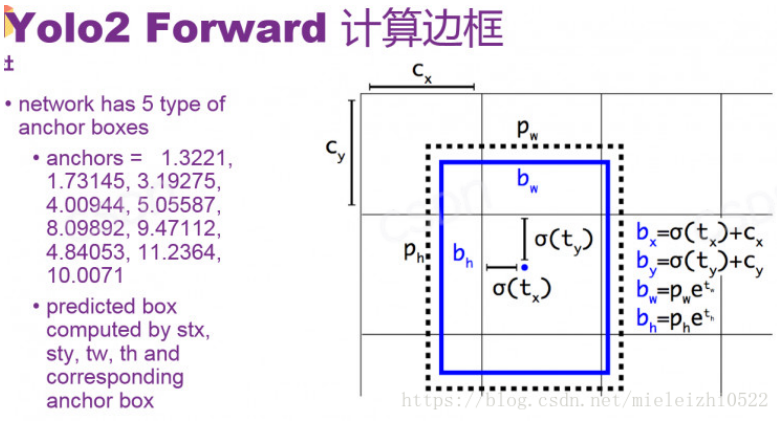
Direct location prediction（采用左上方为参考点，计算以左上方为参考点的偏移值）

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(tx,ty)(tx,ty)，先验框的尺度(wa,ha)(wa,ha)以及中心坐标(xa,ya)(xa,ya)（特征图每个位置的中心点）来计算：

x=(tx*wa)−xa,y=(ty*ha)−ya

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，这导致模型的不稳定性，需要很长时间预测出正确的offsets。所以YOLOv2弃用这种方法，而是用YOLOv1的方法，就是：**预测边框中心点相对于对应网格左上角的相对偏移值**，使用sigmoid函数处理偏移值，根据边界框预测tx,ty,tw,th4个offsets值。



图片来源：https://blog.csdn.net/mieleizhi0522/article/details/79887066

$$b_x = \sigma(t_x) + c_x, b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{tw}, b_h = p_h e^{th}$$

边框实际位置的大小为：

其中(cx,xy)为cell的左上角坐标，pw和ph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）：

$$b_x = (\sigma(t_x) + c_x)/W, b_y = (\sigma(t_y) + c_y)/H$$

$$b_w = p_w e^{tw}/W, b_h = p_h e^{th}/H$$

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

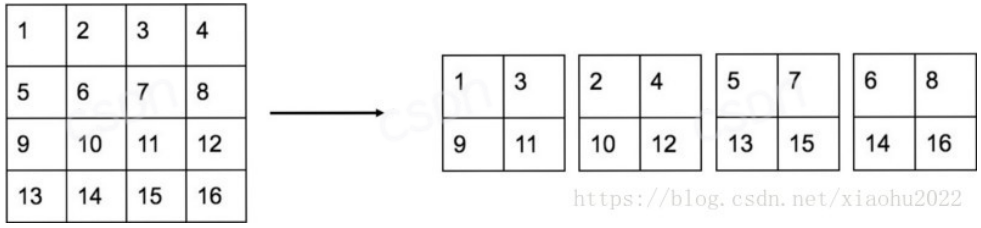
这就是YOLOv2边界框的整个解码过程。约束了边界框的位置预测值使得模型更容易稳定训练，结合聚类分析得到先验框与这种预测方法，YOLOv2的mAP值提升了约5%。

Fine-gained features（passthrough层：将26*26大小的特征图和13*13大小的特征图连接起来）

YOLOv2输入图片大小是416*416,5次max pooling之后得到13*13大小的特征图，对于大物体13*13足够了，但是对于小物体还是不够精细。

SSD采用了多尺度的特征图分别检测不同大小的物体，YOLOv2使用passthrough层来利用更精细的特征图。使用的是最后一个max pooling的输入， 对于darknet-19就是26*26*512大小（将其特征图降低4倍，channel增加4倍，变为13*13*2048），然后连接13*13*1024大小的feature map，有点儿像ResNet.

passthrough层示意图：



图片来源为：https://blog.csdn.net/xiaohu2022

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1×11×1卷积核进行卷积，然后再进行passthrough处理，这样26×26×51226×26×512的特征图得到13×13×25613×13×256的特征图。使用Fine-Grained Features之后YOLOv2的性能有1%的提升。

Multi-scale training（多输入尺度训练：一个重要的创新点）

YOLOv2模型只有卷积核池化层，所以输入大小没有限制，为了增强鲁棒性，YOLOv2采用了多尺度训练策略，就是在训练过程中每经过一定的itreations之后改变模型输入图片大小，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。

YOLOv2的训练

1. 在ImageNet训练Draknet-19，模型输入为**224*224**，共160个epochs
2. 调整网络输入为448*448，继续在ImageNet上finetune分类模型，训练10epoches
3. 修改Darknet-16分类模型为检测模型，并在监测数据集上继续finetune模型

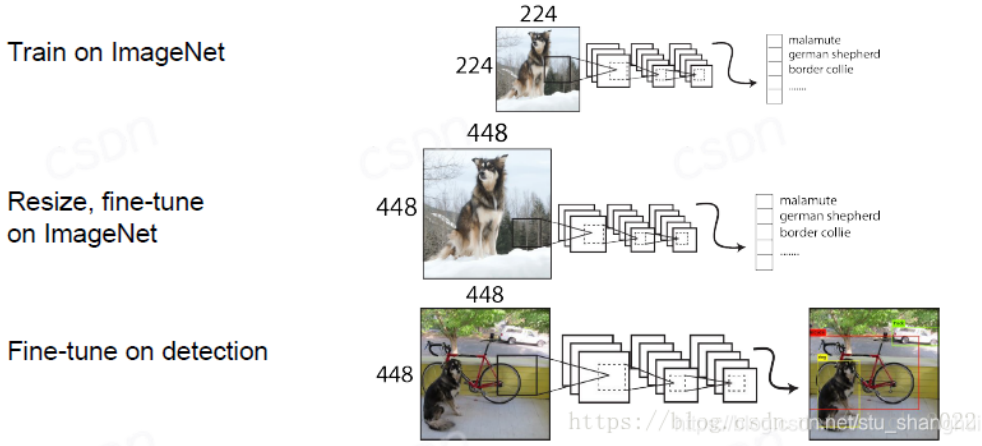
网络修改包括：

1. 移除最后一个卷积层、global avgpooling层和softmax层
2. 增加3个3*3*2014卷积层
3. 增加passthrough层
4. 最后使用1*1卷积层输出预测结果，输出的channel数为num_anchors*(5+num_calsses)

Num_anchors在文中为5，num_classes=20是类别个数，5是坐标值和置信度

VOC数据集中输出channel是125， COCO数据集为425，以VOC数据集为例，最终的预测矩阵为T（shape为（batch_size, 13, 13, 125）,将其reshape为（batch_size,13,13,5,25）,其中T[:, :, :, :, 0:4]为边界框的位置和大小(tx,ty,tw,th)， T[:,:,:,:, 4]是边界框的置信度，T[:,:,:,:,5:]为类别预测值）

下图是YOLOv2训练的三个阶段：



图片来源：https://blog.csdn.net/stu_shan022

YOLOv2的先验框匹配

对于训练图片中的ground truth，如果其中心店落在某个cell内，那么该cell内的**5个先验框**所对应的边界框负责预测它，具体是哪个预测它，需要在训练中确定，即由那个与ground truth的IOU最大的边界框预测它，而剩余的边界框不与该ground truth匹配。YOLOv2中同样需要假定每个cell中至多含有一个ground truth，与ground truth匹配的先验框计算坐标误差、置信度误差（此时表示target=1）以及分类误差，而其他的边界框只计算置信度误差（这个表示target=0）

损失函数

$$
\begin{aligned}
loss_t = \sum_{i=0}^{W}\sum_{j=0}^{H}\sum_{k=0}^{A} \quad & 1_{Max\,IOU < Thresh}\,\lambda_{noobj} * \left(-b_{ijk}^{o}\right)^2 \\
& + 1_{\,t<12800}\,\lambda_{prior} * \sum_{r\,\epsilon\,(x,y,w,h)} \left(prior_k^r - b_{ijk}^r\right)^2 \\
& + 1_k^{truth}\Big(\,\lambda_{coord} * \sum_{r\,\epsilon\,(x,y,w,h)} \left(truth^r - b_{ijk}^r\right)^2 \\
& \quad + \lambda_{obj} * \left(IOU_{truth}^k - b_{ijk}^o\right)^2 \\
& \quad + \lambda_{class} * \Big(\sum_{c=1}^{C} \left(truth^c - b_{ijk}^c\right)^2\Big)\Big)
\end{aligned}
$$

图片来源：https://blog.csdn.net/xiaohu2022

说明：W,H表示特征图（13*13）的宽和高，A指先验框的数目，λ 值是各个loss部分的权重系数

1. 第一行loss是计算background的置信值；先计算各个预测框和所有ground truth的IOU值，并且取最大值ma_IOU,如果该值小于阈值（YOLOV2中使用的是0.6），就标记为background，同时计算noobj的执行度
2. 第二行计算先验框与预测框的坐标误差，只在前12800个iterations间计算
3. 第三行计算ground truth框和预测框的坐标误差，

1. 首先确定其中心点落在哪个cell上
2. 然后计算这个cell的5个先验框与ground truth的IOU值（不考虑坐标，只考虑形状，所以会将先验框和ground truth的中心点都偏移到同一位置，然后计算IOU，IOU最大的那个先验框与ground truth匹配，对应的先验框用来预测ground truth）

4. 第四行计算目标置信度，在YOLOv1中target=1，而YOLOv2增加了一个控制参数rescore，当其为1时，target取预测框与ground truth的真实IOU值。对于那些没有与ground truth匹配的先验框（与预测框对应），除去那些Max_IOU低于阈值的，其它的就全部忽略，不计算任何误差
5. 第五行计算分类误差

需要注意的是：在计算boxes的W,H误差时，YOLOv1使用了平方根降低boxes的大小对误差的影响，而YOLOv2直接计算，但是根据ground truth的大小对权重系数进行了修正

我在代码中加入了详细的注释，权重文件也压缩在文件中，直接可以使用。

代码下载地址：https://pan.baidu.com/s/1GvIyLp1j_R5Uc1RHaSWKtw

密码：ur1d