作者：fivetrees
https://zhuanlan.zhihu.com/p/75106112
本文已获作者授权，未经允许，不得二次转载

【前言】：2013年，R-CNN横空出世，目标检测DL世代大幕拉开。

各路豪杰快速迭代，陆续有了SPP，Fast，Faster版本，至R-FCN，速度与精度齐飞，区域推荐类网络大放异彩。

奈何，未达实时检测之基准，难获工业应用之青睐。

此时，凭速度之长，网格类检测异军突起，先有YOLO，继而SSD，更是摘实时检测之桂冠，与区域推荐类二分天下。然准确率却时遭世人诟病。

遂有JR一鼓作气，并coco，推v2，增加输出类别，成就9000。此后一年，作者隐遁江湖，逍遥twitter。偶获灵感，终推v3，横扫武林！。(很喜欢这段描述，引用自大神这里，文采斐然，膜拜大佬）

本文主要讲YOLOv1-YOLOv3的进化历程，文章如有不对的地方，还望大佬告知。

YOLOv1

https://pjreddie.com/media/files/papers/yolo.pdf

1. 基本思想

YOLOv1是典型的目标检测one stage方法，用回归的方法去做目标检测，执行速度快，达到非常高效的检测，其背后的原理和思想也非常简单。YOLOv1的基本思想是把一副图片，首先reshape成448x448大小（由于网络中使用了全连接层，所以图片的尺寸需固定大小输入到CNN中），然后将划分成SxS个单元格（原文中S=7），以每个格子所在位置和对应内容为基础，来预测：

1）**检测框，包含物体框中心相对其所在网格单元格边界的偏移**（一般是相对于单元格左上角坐标点的位置偏移，以下用x，y表示）和检测框真实宽高相对于整幅图像的比例（注意这里w，h不是实际的边界框宽和高），每个格子预测B个检测框(原文中是2），如下图；

Each bounding box consists of 5 predictions: $x, y, w, h,$ and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally

2）**每个框的Confidence**，这个confidence代表了预测框含有目标的置信度和这个预测框预测的有多准2重信息，公式和说明如下：

$$\text{confidence} = \Pr(object) \cdot \text{IOU}_{\text{pred}}^{\text{truth}}$$

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$. If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

知乎 @fivetrees

3）每个格子预测一共C个类别的概率分数，并且这个分数和物体框是不相关的，只是基于这个格子。

注意（重要细节）：

1. x，y，w，h，confidence都被限制在区间[0,1]。
2. 置信度confidence值只有2种情况，要么为0（边界框中不含目标，P(object)=0），要么为预测框与标注框的IOU，因为P(Object)只有0或1，两种可能，有目标的中心落在格子内，那么P(object)=1，否则为0，不存在（0，1）区间中的值。其他论文中置信度的定义可能跟YOLOv1有些不同，一般置信度指的是预测框中是某类别目标的概率，在[0,1]之间。
3. 每个格子预测C个类别的概率分数，而不是每个每个检测框都需要预测C个类别的概率分数。

具体实现见图：

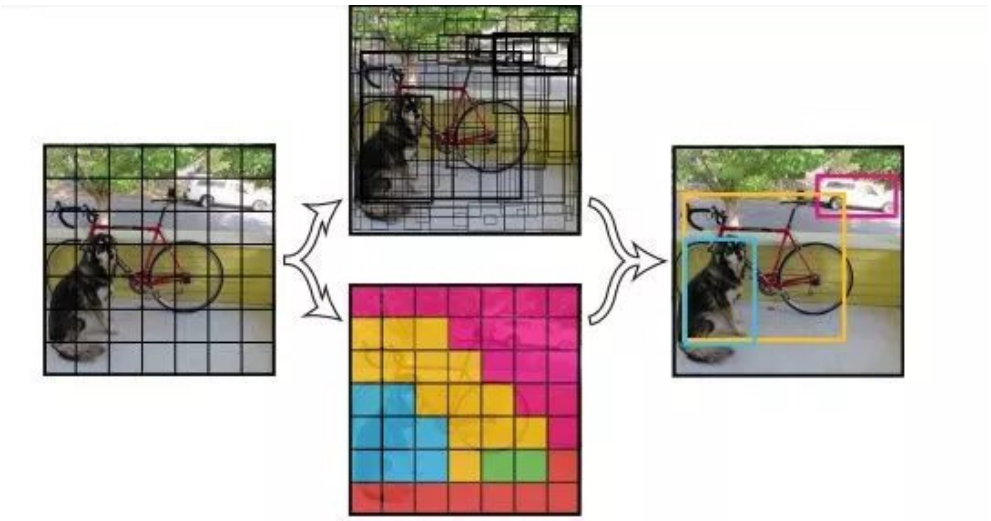


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an even grid and simultaneously predicts bounding boxes, confidence in those boxes, and class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.

知乎 @fivetrees

综上所述：每个格子需要输出的信息维度是Bx(4+1)+C=Bx5+C。在YOLO的论文中，S=7，B=2，C是PASCAL VOC的类别数20，所以最后得到的关于预测的物体信息是一个7x7x30的张量。最后从这7x7x30的张量中提取出来的预测框和类别预测的信息经过NMS，就得到了最终的物体检测结构。

2. 网络结构

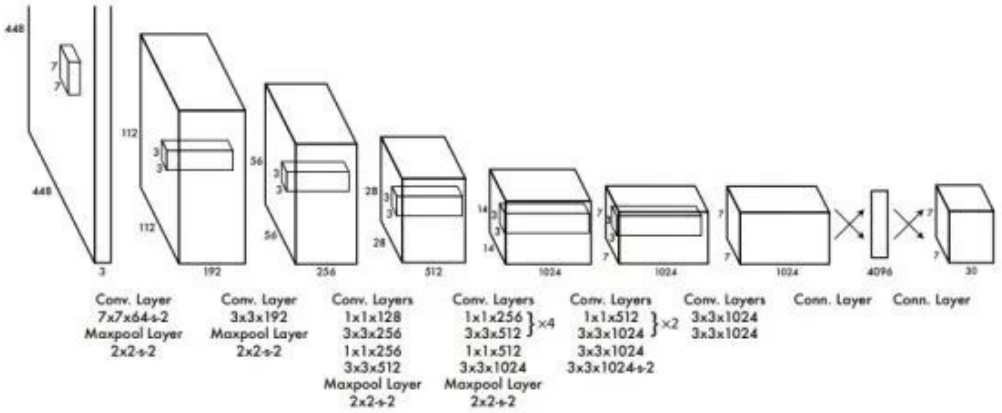
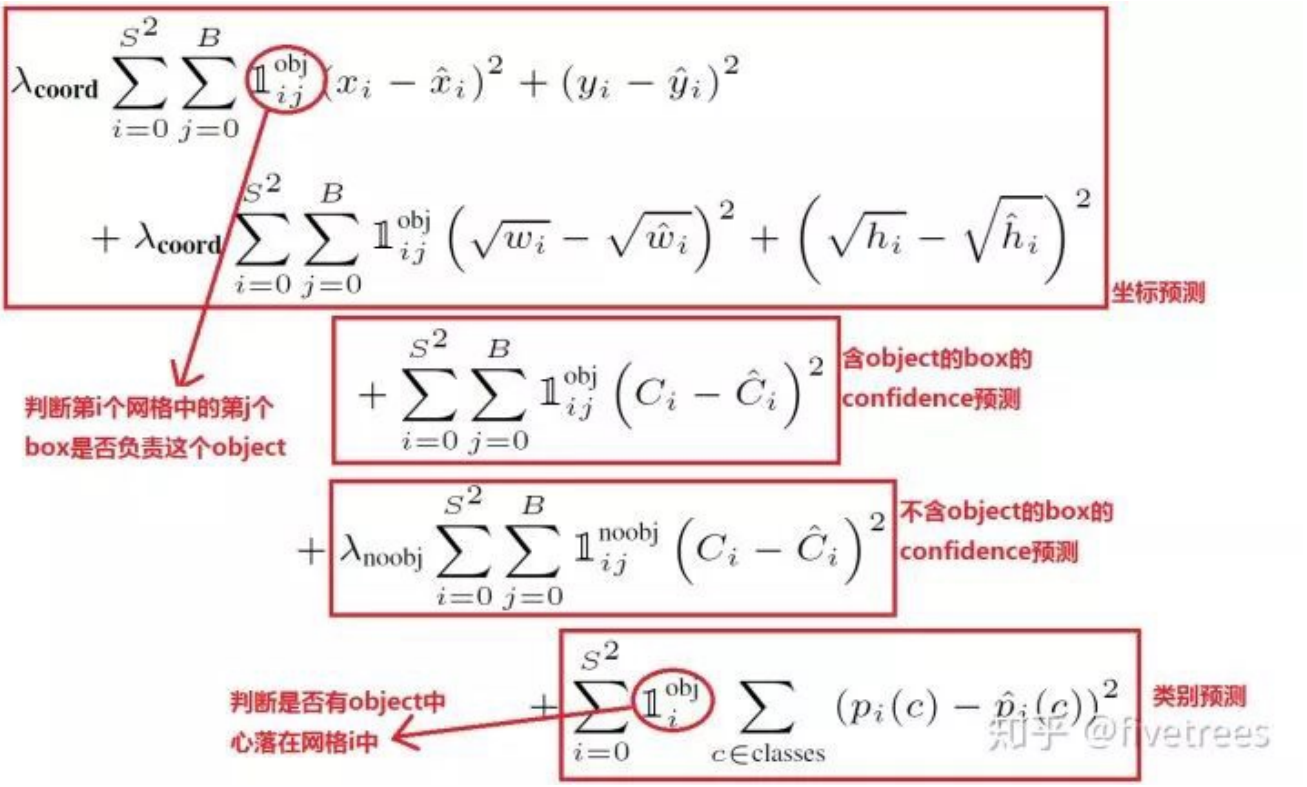


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

分析：作者的网络结构受GoogleNet启发，将GoogleNet中的Inception模块换成了1x1卷积后接3x3卷积，最终网络结构由24个卷积层和4个最大池化层和2个全连接层组成。

3. 损失函数



损失函数设计细节：

- YOLOv1对位置误差，confidence误差，分类误差均使用了均方差作为损失函数。
- 三部分误差损失（位置误差，confidence误差，分类误差），在损失函数中所占权重不一样，位置误差权重系数最大，为5。
- 由于一副图片中没有目标的网格占大多数，有目标的网格占少数，所以损失函数中对没有目标的网格中预测的bbox的confidence误差给予小的权重系数，为0.5。
- 有目标的网格中预测的bbox的confidence损失和分类损失，权重系数正常为1。
- 由于相同的位置误差对大目标和小目标的影响是不同的，相同的偏差对于小目标来说影响要比大目标大，故作者选择将预测的bbox的w,h先取其平方根，再求均方差损失。
- 一个网格预测2个bbox，在计算损失函数的时候，只取与ground truth box中IoU大的那个预测框来计算损失。

- 分类误差，只有当单元格中含有目标时才计算，没有目标的单元格的分类误差不计算在内。

4. 激活函数

- 最后一层全连接层用线性激活函数
- 其余层采用leak RELU

5. YOLOv1的缺陷

- 首先，每个单元格只预测2个bbox，然后每个单元格最后只取与gt_bbox的IOU高的那个最为最后的检测框，也只是说每个单元格最多只预测一个目标，若单个单元格有多个目标时，只能检测出其他的一个，导致小目标漏检，因此YOLOv1对小目标检测效果不好。
- 其次，虽然YOLOv1中损失函数中位置误差，对预测的w，h取平方根处理再求均方差，来缓解相同位置误差对大目标，小目标影响不同的弊端，但是作用甚微，没有根本解决问题对于小物体。小的目标的 置信度误差也会对网络优化过程造成很大的影响，从而降低了物体检测的定位准确性。
- 由于输出层为全连接层，因此在检测时，YOLO 训练模型只支持与训练图像相同的输入分辨率的图片。

YOLOv2

https://arxiv.org/pdf/1612.08242v1.pdf

1. 介绍

YOLOv2又叫YOLO9000，其能检测超过9000种类别的物体，在VOC2007数据集中在76FPS的速度下，能达到76.8%的mAP，在40FPS的速度下，能达到78.6%的mAP，很好的达到速度与精度的平衡。

2.相比于v1的改进

算法层面：

- **Anchor:引入了Faster R-CNN中使用的Anchor（猜想，Anchor是我们作目标检测的载体，最后预测的检测框位置是在Anchor的基础上进行回归的，作者认为可能比v1中的无Anchor回归效果更好，不过最近也有大量的Anchor Free文章出现，过段时间我再总结一下），注意这里作者在YOLOv2中设计的Anchor并不是像Faster R-CNN中人为事先设计的尺寸和高宽比一级个数，作者通过在所有训练图像的所有边界框上运行k-means聚类来选择锚的个数和形状(k = 5，因此它**

找到五个最常见的目标形状)。因此，YOLO的锚是特定于您正在训练(和测试)的数据集的。k-means算法找到了将所有数据点划分聚类的方法。这里的数据点是数据集中所有真实边界框的宽度和高度。但5个锚是否最佳选择?我们可以在不同数量的聚类上多次运行k-means，并计算真实标签框与它们最接近的锚框之间的平均IOU。毫无疑问，使用更多质心(k值越大)平均IOU越高，但这也意味着我们需要在每个网格单元中使用更多的检测器，并使模型运行速度变慢。对于YOLO v2，他们选择了5个锚作为召回率和模型复杂度之间的良好折衷。

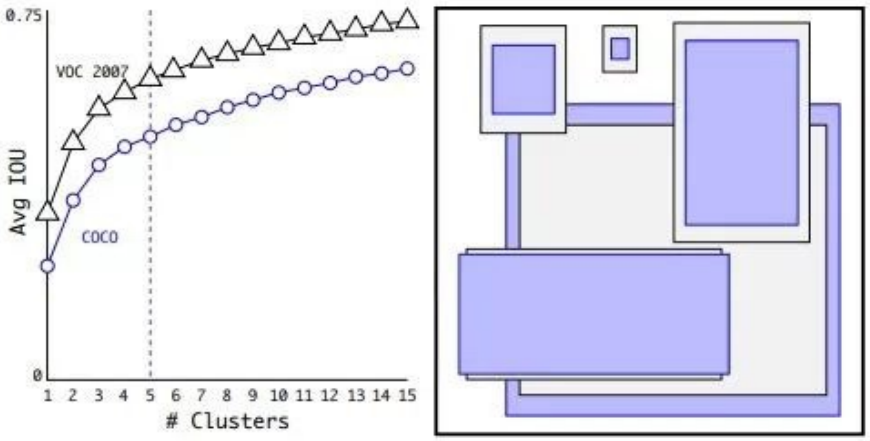
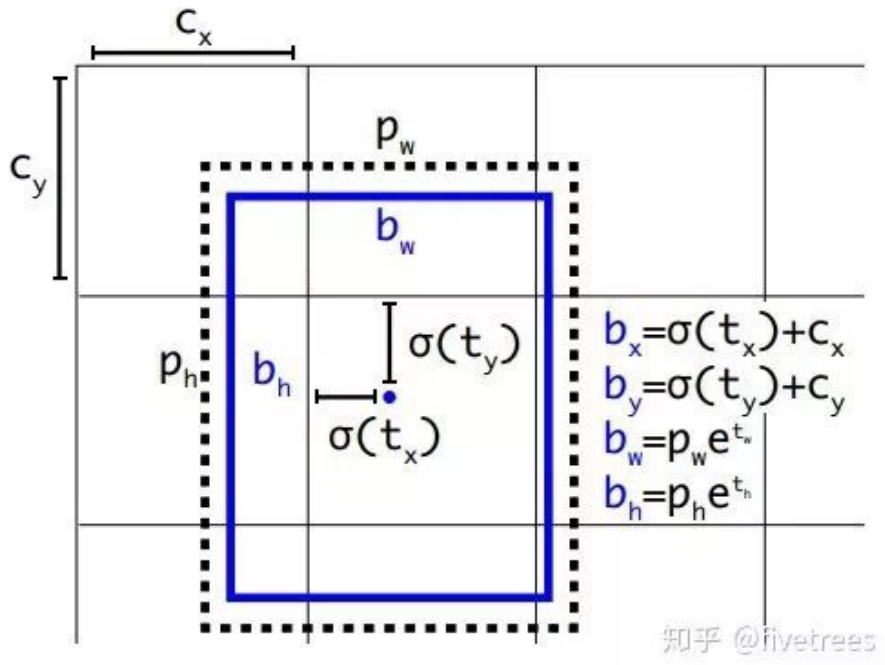


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that $k = 5$ gives a good tradeoff for recall vs. complexity of the model. 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.

知乎 @hivetree5

- 坐标预测:在这里作者虽然引入了Faster R-CNN中类似的anchor，但是作者并没有像其意义，对bbox中心坐标的预测是基于anchor坐标的偏移量得到的，而是采用了v1中预测anchor中心点相对于对于单元格左上角位置的偏移，如下图：



The network predicts 5 bounding boxes at each cell in the output feature map. The network predicts 5 coordinates for each bounding box, t_x, t_y, t_w, t_h, and t_o. If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w, p_h, then the predictions correspond to:

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

知乎 @fivetrees

- **损失函数：**

1.在计算类概率误差时，YOLOv1中仅对每个单元格计算；而YOLOv2中对每一个anchor box都会计算类概率误差。

2.YOLOv1中使用w和h的开方来缓和box的尺寸不平衡问题，而在YOLOv2中则通过赋值一个和w，h相关的权重函数达到该目的。

3.与YOLOv1不同的是修正系数的改变，YOLOv1中no_objects_loss和objects_loss分别是0.5和1，而YOLOv2中则是1和5

具体可以参考下方链接，或者直接去作者的个人网站查看源码：

https://blog.csdn.net/qq_42422981/article/details/90105149

网络层面：

- Darknet19: 与v1不同采用的是全卷积网络，取掉了v1中的全连接层，改用全局平均池化，去掉v1中最后一个池化层，增加特征的分辨率。网络共19个卷积层，5个最大池化层，具体结构见下图

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			

Table 6: Darknet-19.

知乎 @fivetrees

训练检测方面：

- **训练图像分辨率：v1在ImageNet上预训练时用的224x224尺寸的图片，正式训练时用448x448,这需要模型适应新的分辨率。YOLOv2是直接使用448x448的输入训练，随着输入分辨率的增加，模型提高了4%的mAP。**
- **使用了WordTree：通过WordTree来混合检测数据集与识别数据集之中的数据，使得这一网络结构可以实时地检测超过9000种物体分类。**
- **联合训练算法：使用这种联合训练技术同时在ImageNet和COCO数据集上进行训练。YOLO9000进一步缩小了监测数据集与识别数据集之间的代沟。联合训练算法的基本思路就是：同时在检测数据集和分类数据集上训练物体检测器（Object Detectors），用检测数据集的数据学习物体的准确位置，用分类数据集的数据来增加分类的类别量、提升健壮性。分类信息学习自ImageNet分类数据集，而物体位置检测则学习自COCO检测数据集。**
- **多尺度训练：为了提高模型的鲁棒性，在训练的时候采用了多尺度的输入进行训练，由于网络的下采样因子是32，故输入尺寸选择32的倍数288，352，...，544**

- **多尺度检测，reorg层：作者将前一层的26*26的特征图做一个reorg操作，将其变成13*13但又不破坏其大特征图的特征，然后和本层的13*13的1特征图进行concat。**

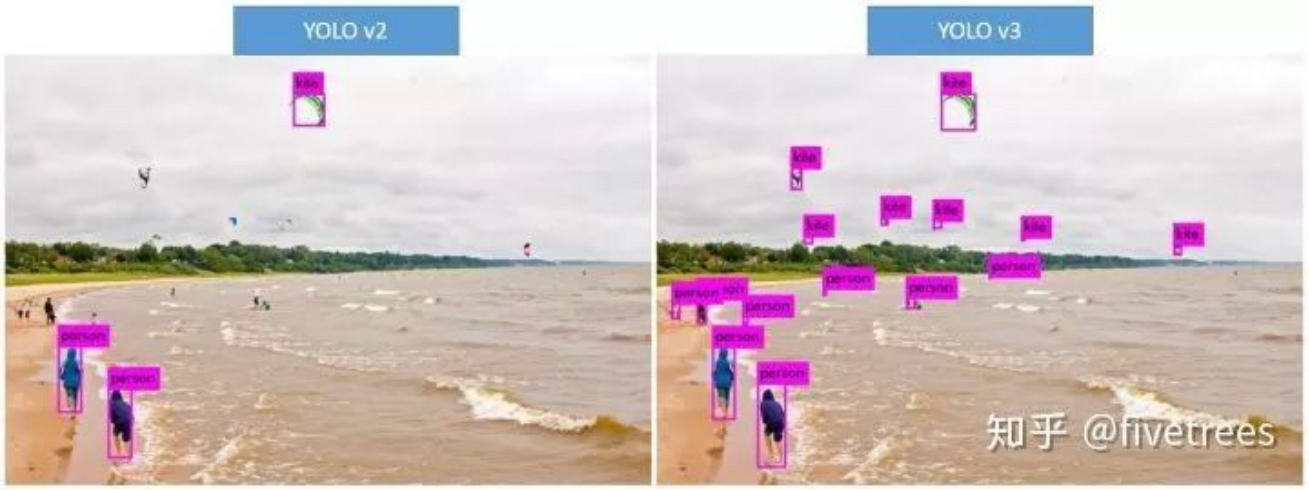
技巧方面：

- **Batch Normalization：使用Batch Normalization对网络进行优化，让网络提高了收敛性，同时还消除了对其他形式的正则化（regularization）的依赖。通过对YOLO的每一个卷积层增加Batch Normalization，最终使得mAP提高了2%，同时还使model正则化。使用Batch Normalization可以从model中去掉Dropout，而不会产生过拟合。**

YOLOv3

https://link.zhihu.com/?target=https%3A//pjreddie.com/media/files/papers/YOLOv3.pdf

先看YOLOv3的几张目标检测效果图吧，效果真实好！YOLO一直以来被诟病的对小目标检测效果不好，进步提升明显。



改进点

- **网络：Darknet53，采用简化的residual block 取代了原来 1×1 和 3×3的block; (其实就是加了一个shortcut，也是网络加深必然所要采取的手段)。这和上一点是有关系的，v2的darknet-19变成了v3的darknet-53，为啥呢？就是需要上采样啊，卷积层的数量自然就多了，另外作者还是用了一连串的3*3、1*1卷积，3*3的卷积增加channel，而1*1的卷积在于压缩3*3卷积后的特征表示。**

We use a new network for performing feature extraction. Our new network is a hybrid approach between the network used in YOLOv2, Darknet-19, and that newfangled residual network stuff. Our network uses successive 3×3 and 1×1 convolutional layers but now has some shortcut connections as well and is significantly larger. It has 53 convolutional layers so we call it.... wait for it..... Darknet-53!

知乎 @fivetrees

	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			

Table 1. **Darknet-53.**

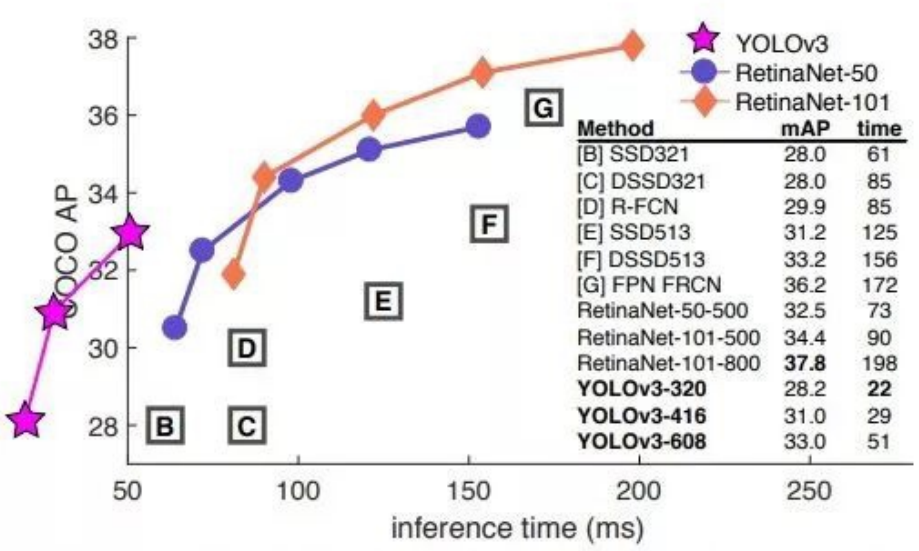
知乎 @fivetrees

- **分类损失：在YOLOv3中，每个框用多标签分类来预测边界框可能包含的类。该算法将v2中的softmax替换成了逻辑回归loss，在训练过程中使用二原交叉熵损失来进行类别预测。对于重叠的标签，多标签方法可以更好的模拟数据。**

- **跨尺度预测：YOLOv3采用多个尺度融合的方式做预测。原来YOLOv2中有一个层叫：passthrough layer，假设最后提取的特征图尺度是13*13，那么这个层的作用就是将前面一层的26*26的特征图和本层13*13的特征图进行连接，有点像ResNet。这样的操作是为了加强YOLO算法对小目标检测的精度。在YOLOv3中，作者采用了类似与FPN的上采样和融合做法（最后融合了3个尺度，其他2个尺度分别是26*26和52*52），在多给尺度的特征图上做预测，对于小目标的提升效果还是非常明显的。虽然在YOLOv3中每个网格预测3个边界框，比v2中的5个要少，但v3采用了多个尺度的特征融合，所以边界框的数量也比之前多很多。**

与其他网络对比

- mAP和相同GPU下的单张图片所需时间



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

知乎 @fivetrees

- Darknet53与其他backbone对比（256×256的图片，并进行单精度测试。运行环境为Titan X）

This new network is much more powerful than Darknet-19 but still more efficient than ResNet-101 or ResNet-152. Here are some ImageNet results:

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Table 2. **Comparison of backbones.** Accuracy, billions of operations, billion floating point operations per second, and FPS for various networks.

知乎 @fivetrees

尝试，但效果不好的工作

- **Anchor box坐标的偏移预测。作者尝试了常规的Anchor box预测方法，比如利用线性激活将坐标x、y的偏移程度预测为边界框宽度或高度的倍数。但发现这种做法降低了模型的稳定性，且效果不佳。 用线性方法预测x,y，而不是使用逻辑方法。我们尝试使用线性激活来直接预测x，y的offset，而不是逻辑激活，还降低了mAP。**

- **focal loss。我们尝试使用focal loss，但使我们的mAP降低了2%。 对于focal loss函数试图解决的问题，YOLOv3从理论上来说已经很强大了，因为它具有单独的对象预测和条件类别预测。因此，对于大多数例子来说，类别预测没有损失？或者其他的东西？我们并不完全确定。**

- **双IOU阈值和真值分配。在训练期间，Faster RCNN用了两个IOU阈值，如果预测的边框与ground truth的IoU>0.7，那它是个正样本；如果在[0.3，0.7]之间，则忽略；如果和ground truth的IoU<0.3，那它就是个负样本。作者尝试了这种思路，但效果并不好。**

总结

YOLO系列算法不断吸收目标检测同类算法的优点，如FPN，Faster-RCNN，ResNet，，将其应用与自身，不断进步，取得了较高的检测速度和检测精度，相比于其他算法更符合工业界对目标检测算法实时性的要求，简单易实现，对于嵌入式很友好，期待下一代的YOLO算法！如有错误，还望告知！

-完-