YOLOv3：增加的改进

作者：Joseph Redmon &Ali Farhadi（华盛顿大学）

原翻译 Amusi（知乎），刘祯整理

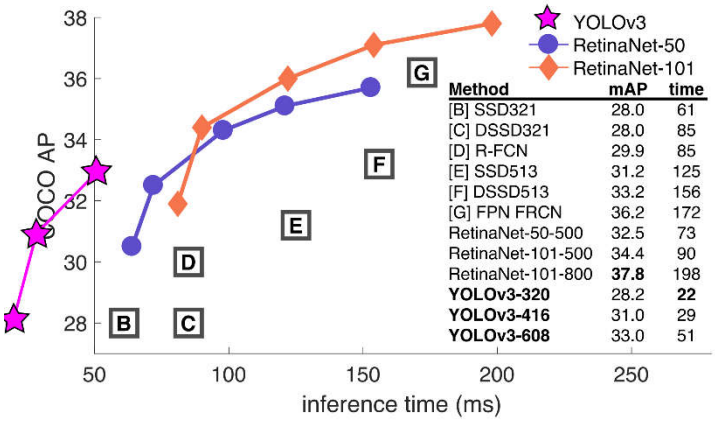
Abstract

我们给 YOLO 提供一些更新！ 我们做了一些小的设计更改以使其更好。 我们也训练了这个非常好的新网络。它比上次（YOLOv2）稍大一些，但更准确。它仍然很快，所以不用担心。在 320×320 YOLOv3 运行 22.2ms，28.2 mAP，像 SSD 一样准确，但速度快三倍。当我们看看以老的 0.5 IOU mAP 检测指标时，YOLOv3 是相当不错的。 在 Titan X 上，它在 51 ms 内实现了 57.9 的AP_{50}，与 RetinaNet 在 198 ms 内的 57.5 AP50 相当，性能相似但速度快 3.8 倍。与往常一样，所有代码均在 https://pjreddie.com/yolo/

1 Introduction

有时候，一年你主要只是在打电话，你知道吗？今年我没有做很多研究。我在 Twitter 上花了很多时间。玩了一下 GAN。去年我留下了一点点的动力[12] [1]；我设法对 YOLO 进行了一些改进。但是诚然，没有什么比这超级有趣的了，只是一小堆（bunch）改变使它变得更好。我也帮助了其他人的做一些研究。

事实上，这就是今天带给我们的。 我们有一个相机准备好的截止日期[4]，我们需要引用一些我对 YOLO 进行的随机更新，但我们没有资源。所以准备好**技术报告**！



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.

关于技术报告的好处是他们不需要介绍，你们都知道我们为什么来到这里。因此，这篇介绍性文章的结尾将为本文的其余部分提供路标（signpost）。首先我们会告诉你 YOLOv3 的详细内容。然后我们会告诉你我们是怎么做的。我们还会告诉你我们尝试过的一些没有奏效的事情。最后，我们将考虑这一切意味着什么。

2 The Deal

这里是 YOLOv3 的详细内容：我们主要从其他人那里获得好点子。我们也训练了一个比其他人更好的新分类器网络。我们将从头开始介绍整个系统，以便您能够理解这一切。

2.1 Bounding Box Prediction

在 YOLO9000 之后，我们的系统使用维度聚类（dimension clusters ）作为 anchor boxes 来预测边界框[15]。网络为每个边界框预测 4 个坐标，t_x, t_y, t_w, t_h。如果单元格从图像的左上角偏移（c_x, c_y），并且之前的边界框具有宽度和高度p_w, p_h，则预测对应于：

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

在训练期间，我们使用平方误差损失的总和。如果对于一些坐标预测的 ground truth 是 $\hat{t}_*$，我们的梯度是 ground truth（由 ground box 计算得到）减去我们的预测：$\hat{t}_* - t_*$。 通过反转上面的方程可以很容易地计算出这个 ground truth。

YOLOv3 使用逻辑回归预测每个边界框（bounding box）的对象分数。如果先前的边界框比之前的任何其他边界框重叠 ground truth 对象，则该值应该为 1。如果以前的边界框不是最好的，但是确实将 ground truth 对象重叠了一定的阈值以上，我们会忽略这个预测，按照[17]进行。我们使用阈值 0.5。与[17]不同，我们的系统只为每个 ground truth 对象分配一个边界框。如果先前的边界框未分配给 grounding box 对象，则不会对坐标或类别预测造成损失。

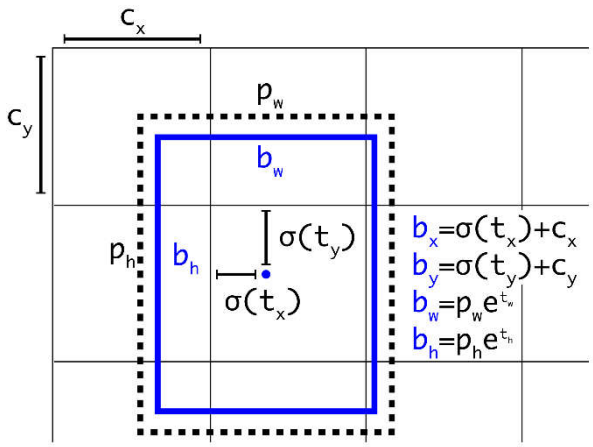


Figure 2. **Bounding boxes with dimension priors and location prediction.** We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function. This figure blatantly self-plagiarized from [15].

2.2 Class Prediction

每个框使用多标签分类来预测边界框可能包含的类。我们不使用 softmax，因为我们发现它对于高性能没有必要，相反，我们只是使用独立的逻辑分类器。 在训练过程中，我们使用二元交叉熵损失来进行类别预测。

这个公式有助于我们转向更复杂的领域，如 Open Image Dataset[7]。在这个数据集中有许多重叠的标签（如女性和人物）。使用 softmax 会强加了一个假设，即每个框中只有一个类别，但通常情况并非如此。多标签方法更好地模拟数据。

2.3 Prediction Across Scales

YOLOv3 预测 3 种不同尺度的框（boxes）。我们的系统使用类似的概念来提取这些尺度的特征，以形成金字塔网络[8]。从我们的基本特征提取器中，我们添加了几个卷积层。其中最后一个预测了 3-d 张量编码边界框，对象和类别预测。在我们的 COCO 实验[10]中，我们预测每个尺度的 3 个框，所以对于 4 个边界框偏移量，1 个目标性预测和 80 个类别预测，张量为$N \times N \times [3 * (4 + 1 + 80)]$。

接下来，我们从之前的两层中取得特征图（feature map），并将其上采样 2 倍。我们还从网络中的较早版本获取特征图，并使用 element-wise addition 将其与我们的上采样特征进行合并。这种方法使我们能够从早期特征映射中的上采样特征和更细粒度的信息中获得更有意义的语义信息。然后，我们再添加几个卷积层来处理这个组合的特征图，并最终预测出一个相似的张量，虽然现在是两倍的大小。

我们再次执行相同的设计来预测最终尺度的方框。因此，我们对第三种尺度的预测将从所有先前的计算中获益，并从早期的网络中获得细粒度的特征。

我们仍然使用 k-means 聚类来确定我们的边界框的先验。我们只是选择了 9 个聚类（clusters）和 3 个尺度（scales），然后在整个尺度上均匀分割聚类。在 COCO 数据集上，9 个聚类是：（10×13），（16×30），（33×23），（30×61），（62×45），（59×119），（116×90）；（156×198），（373×326）。

2.4 Feature Extractor

我们使用新的网络来实现特征提取。我们的新网络是用于 YOLOv2，Darknet-19 中的网络和那些新颖的残差网络的混合方法。我们的网络使用连续的 3×3 和 1×1 卷积层，但现在也有一些 shotcut 连接，该网络明显更大。它有 53 个卷积层，所以我们称之为.....稍等，

	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.**

嗯.....Darknet-53!

这个新网络比 Darknet-19 功能强大得多，而且比 ResNet-101 或 ResNet-152 更有效。以下是一些 ImageNet 结果：

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.

每个网络都使用相同的设置进行训练，并以 256×256 的单精度测试进行测试。运行时间是在 Titan X 上以 256×256 进行测量的。因此，Darknet-53 可与 state-of-the-art 的分类器相媲美，但浮点运算更少，速度更快。Darknet-53 比 ResNet-101 更好，速度更快 1：5 倍。 Darknet-53 与 ResNet-152 具有相似的性能，速度提高 2 倍。

Darknet-53 也可以实现每秒最高的测量浮点运算。这意味着网络结构可以更好地利用 GPU，从而使其评估效率更高，速度更快。这主要是因为 ResNets 的层数太多，效率不高。

2.5 Training

我们仍然训练完整的图像，没有 hard negative mining or any of that stuff 。我们使用多尺度训练，大量的 data augmentation，batch normalization，以及所有标准的东西。我们使用 Darknet 神经网络框架进行训练和测试[14]。

3 How We Do

YOLOv3 非常好！请参见表 3。就 COCO 的 mAP 指标而言，它与 SSD variants 相当，但速度提高了 3 倍。尽管如此，它仍然比像 RetinaNet 这样的其他模型落后很多。

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Table 3. I'm seriously just stealing all these tables from [9] they take soooo long to make from scratch. Ok, YOLOv3 is doing alright. Keep in mind that RetinaNet has like 3.8× longer to process an image. YOLOv3 is much better than SSD variants and comparable to state-of-the-art models on the AP_{50} metric.

然而，当我们在 IOU = 0.5（或者图表中的 AP50）看到 mAP 的“旧”检测度量时，YOLOv3 非常强大。它几乎与 RetinaNet 相当，并且远高于 SSD variants。这表明 YOLOv3 是一个非常强大的检测器，擅长为目标生成像样的框（boxes）。

但是，当我们查看“旧”检测指标在 IOU = 0.5（或图表中的AP_{50}）上的 mAP YOLOv3 非常强大。它几乎与 RetinaNet 并驾齐驱，并且远远高于它 SSD 变种。 这表明 YOLOv3 是一款性能卓越的探测器，擅长为物体制作像样的 boxes。 然而，性能随着 IOU 而显着下降阈值增加表明 YOLOv3 正在努力争取 boxes 与物体完美对齐。

在过去，YOLO 在小目标的检测上表现一直不好。然而，现在我们看到了这种趋势的逆转。 随着新的多尺度预测，我们看到 YOLOv3 具有相对较高的AP_S性能。但是，它在中等和更大尺寸的物体上的表现相对较差。需要更多的研究来达到这个目的。

当我们在AP_{50}指标上绘制精确度和速度时（见图 5），我们看到 YOLOv3 与其他检测系统相比具有显著的优势。也就是说，速度越来越快。

4 Things We Tried That Didn’t Work

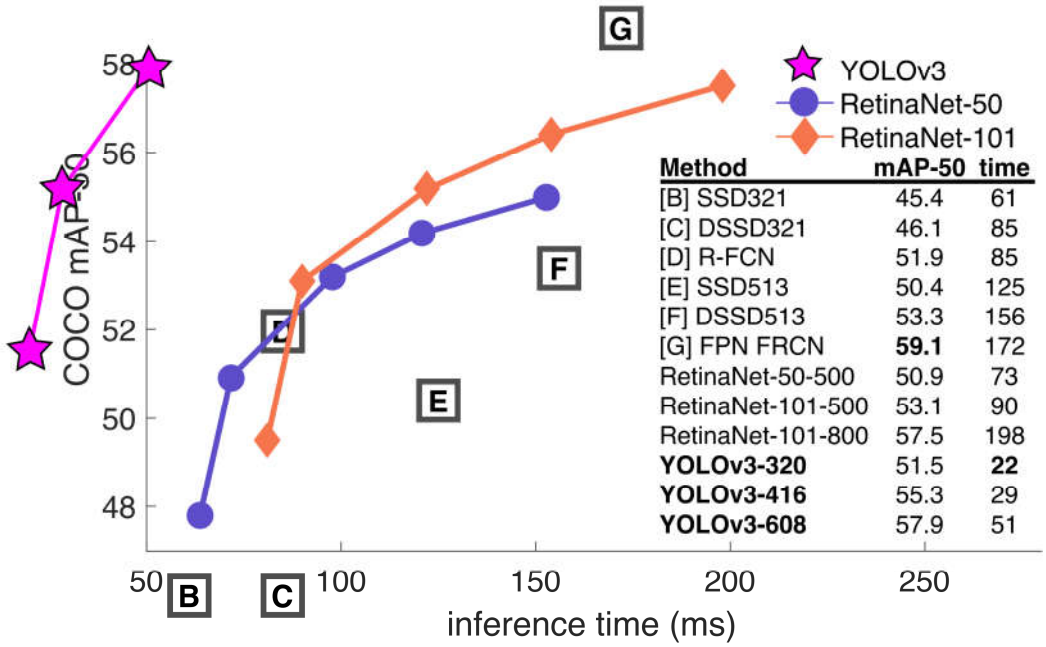
我们在研究 YOLOv3 时尝试了很多东西。很多都不起作用。这是我们可以记住的东西。

Anchor box x，y **offset predictions**。我们尝试使用正常 anchor box 预测机制，这里你使用线性激活来预测 x，y offset 作为 box 的宽度或高度的倍数。我们发现这种方法降低了模型的稳定性，并且效果不佳。

Linear x，y **predictions instead of logistic**。我们尝试使用线性激活来直接预测 x，y offeset 而不是逻辑激活。这导致 mAP 下降了几个点。

Focal loss。我们尝试使用 focal loss。它使得 mAp 降低了 2 个点。YOLOv3 对 focal loss 解决的问题可能已经很强大，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。

Dual IOU thresholds and truth assignment 。Faster R-CNN 在训练期间使用两个 IOU 阈值。如果一个预测与 ground truth 重叠达到 0.7，它就像是一个正样本，如果达到 0.3-0.7，它被忽略，如果小于 0.3，这是一个负样本的例子。我们尝试了类似的策略，但无法取得好成绩。我们非常喜欢我们目前的表述，似乎至少在局部最佳状态。有些技术可能最终会产生好的结果，也许他们只是需要一些调整来稳定训练。



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 3. Again adapted from the [9], this time displaying speed/accuracy tradeoff on the mAP at .5 IOU metric. You can tell YOLOv3 is good because it’s very high and far to the left. Can you cite your own paper? Guess who’s going to try, this guy → [16]. Oh, I forgot, we also fix a data loading bug in YOLOv2, that helped by like 2 mAP. Just sneaking this in here to not throw off layout.

5 What This All Means

YOLOv3 是一个很好的检测器。速度很快，很准确。COCO 平均 AP 介于 0.5 和 0.95 IOU 指标之间的情况并不如此。但是，对于检测度量 0.5 IOU 来说非常好。

为什么我们要改变指标？ 最初的 COCO 论文只是含有这个神秘的句子：“一旦评估服务器完成，就会添加完整的评估指标的讨论”。Russakovsky 等人报告说，人类很难区分 IOU 为 0.3 还是 0.5。“训练人们目视检查一个 IOU 值为 0.3 的边界框，并将它与 IOU 0.5 区分开来是一件非常困难的事情。”[18]如果人类很难区分这种差异，那么它有多重要？

但是也许更好的问题是：“现在我们有了这些检测器（detectors），我们要做什么？”很多做这项研究的人都在 Google 和 Facebook 上。我想至少我们知道这项技术是非常好的，绝对不会被用来收集您的个人信息，并将其出售给.....等等，您是说这就是它的用途？
那么其他大量资助视觉研究的人都是军人，他们从来没有做过任何可怕的事情，例如用新技术杀死很多人哦等等.....

我有很多希望，大多数使用计算机视觉的人都是做的快乐，研究了很多好的应用，比如计算一个国家公园内的斑马数量[13]，或者追踪它们在它们周围徘徊时的猫[19]。但是计算机视觉已经被用于可疑的应用，作为研究人员，我们有责任至少考虑我们的工作可能会造成的伤害，并考虑如何减轻它的影响。 我们非常珍惜这个世界。

最后，不要@我。（因为我终于退出了Twitter）。

参考文献

略

Rebuttal

略

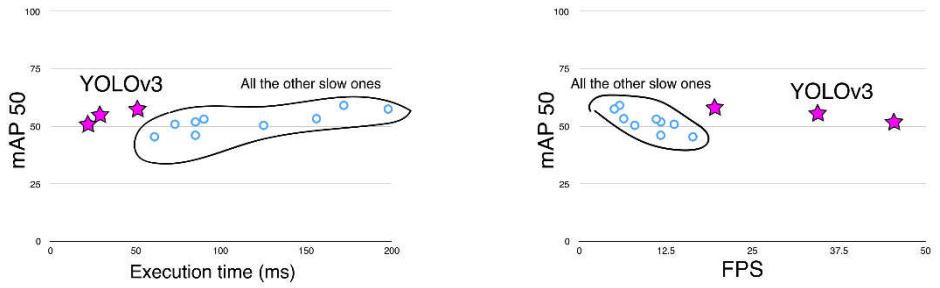


Figure 4. Zero-axis charts are probably more intellectually honest... and we can still screw with the variables to make ourselves look good!

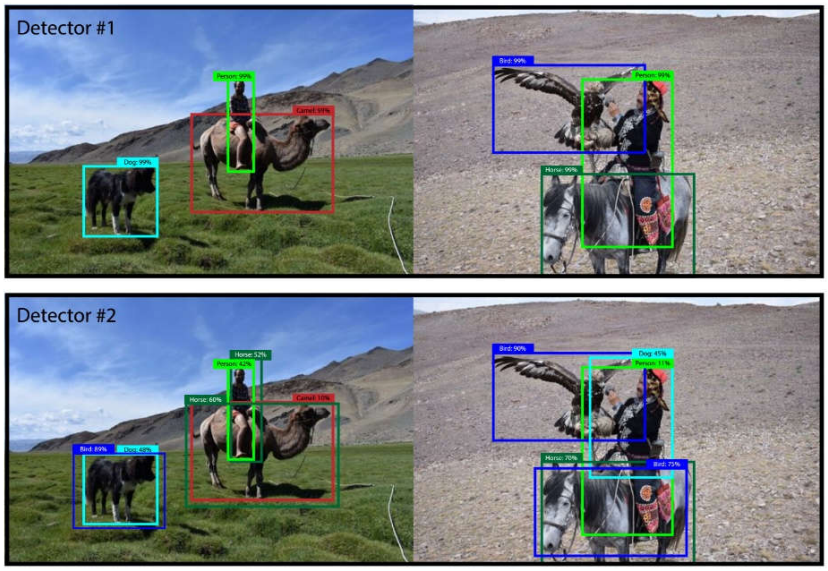


Figure 5. These two hypothetical detectors are perfect according to mAP over these two images. They are both perfect. Totally equal.