YOLOv3: An Incremental Improvement

YOLOv3：增量式的改进

Joseph Redmon Ali Farhadi
University of Washington

Abstract

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320×320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57:9 AP50 in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

摘要

我们对 YOLO 进行了一系列更新！它包含一堆小设计，可以使系统的性能得到更新。我们也训练了一个新的、比较大的神经网络。虽然比上一版更大一些，但是精度也提高了。不用担心，它的速度依然很快。YOLOv3 在 320×320 输入图像上运行时只需 22ms，并能达到 28.2mAP，其精度和 SSD 相当，但速度要快上 3 倍。使用之前 0.5 IOU mAP 的检测指标，YOLOv3 的效果是相当不错。YOLOv3 使用 Titan X GPU，其耗时 51ms 检测精度达到 57.9 AP_{50}，与 RetinaNet 相比，其精度只有 57.5 AP_{50}，但却耗时 198ms，相同性能的条件下

YOLOv3 速度比 RetinaNet 快 3.8 倍。与之前一样，所有代码在网址：https://pjreddie.com/yolo/。

1. Introduction

Sometimes you just kinda phone it in for a year, you know? I didn't do a whole lot of research this year. Spent a lot of time on Twitter. Played around with GANs a little. I had a little momentum left over from last year [12] [1]; I managed to make some improvements to YOLO. But, honestly, nothing like super interesting, just a bunch of small changes that make it better. I also helped out with other people's research a little.

1. 引言

有时候，一年内你主要都在玩手机，你知道吗？今年我没有做很多研究。我在 Twitter 上花了很多时间。研究了一下 GAN。去年我留下了一点点的精力[12] [1]；我设法对 YOLO 进行了一些改进。但是，实话实说，除了仅仅一些小的改变使得它变得更好之外，没有什么超级有趣的事情。我也稍微帮助了其他人的一些研究。

Actually, that's what brings us here today. We have a camera-ready deadline [4] and we need to cite some of the random updates I made to YOLO but we don't have a source. So get ready for a TECH REPORT!

其实，这就是今天我要讲的内容。我们有一篇论文快截稿了，并且我们还缺一篇关于 YOLO 更新内容的文章作为引用，但是我们没有引用来源。因此准备写一篇技术报告！

The great thing about tech reports is that they don't need intros, y'all know why we're here. So the end of this introduction will signpost for the rest of the paper. First we'll tell you what the deal is with YOLOv3. Then we'll tell you how we do. We'll also tell you about some things we tried that didn't work. Finally we'll contemplate what this all means.

技术报告的好处是他们不需要引言，你们都知道我为什么写这个。所以引言的结尾可以作为阅读本文剩余内容的一个指引。首先我们会告诉你 YOLOv3 的方案。其次我们会告诉你我们是如何实现的。我们也会告诉你我们尝试过但并不奏效的一些事情。最后我们将探讨这些的意义。

2. The Deal

So here's the deal with YOLOv3: We mostly took good ideas from other people. We also trained a new classifier network that's better than the other ones. We'll just take you through the whole system from scratch so you can understand it all.

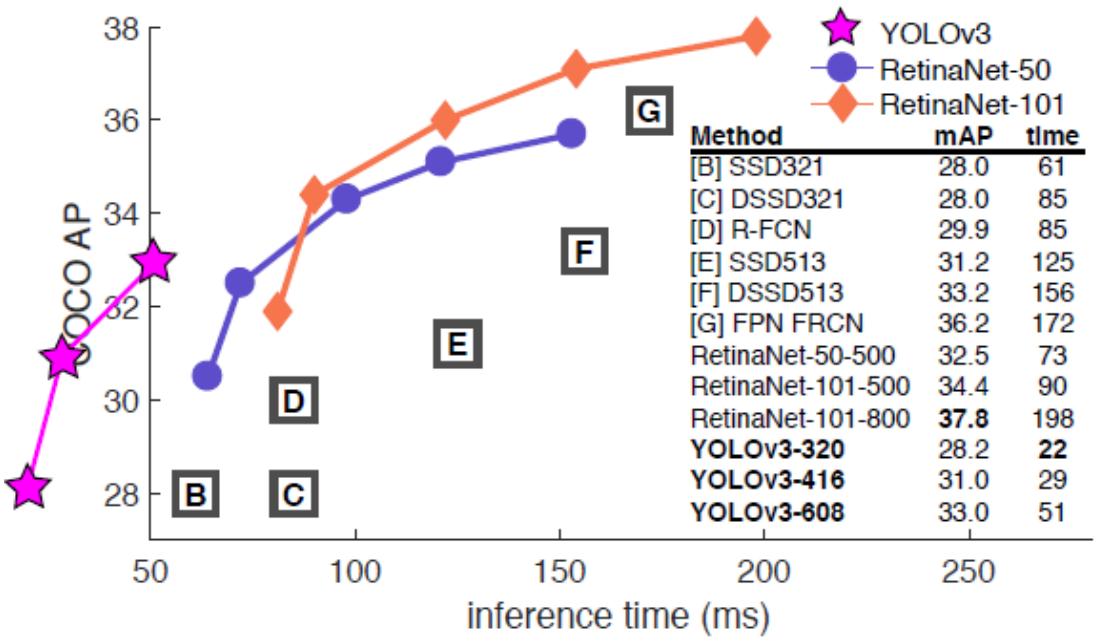


Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

2. 方案

这节主要介绍 YOLOv3 的方案：我们主要从其他人的研究工作里获得了一些好思路、好想法。我们还训练了一个新的、比其他网络更好的分类网络。为了方便您理解，我们将带您从头到尾贯穿整个模型系统。

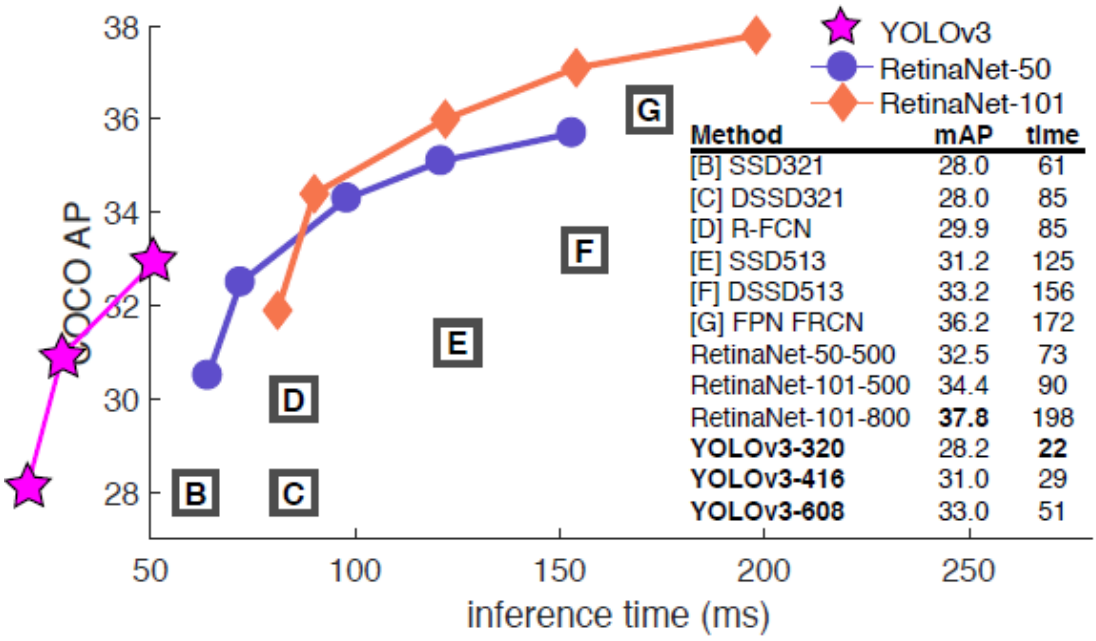


图 1.这个图来自 Focal Loss 论文[9]。YOLOv3 的运行速度明显快于其他具有可比性能的检测方法。检测时间基于 M40 或 Titan X（这两个基本上是相同的 GPU）。

2.1. Bounding Box Prediction

Following YOLO9000 our system predicts bounding boxes using dimension clusters as anchor boxes [15]. The network predicts 4 coordinates for each bounding box, tx, ty, tw, th. If the cell is offset from the top left corner of the image by (cx; cy) and the bounding box prior has width and height pw, ph, then the predictions correspond to:

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

2.1 边界框预测

按照 YOLO9000，我们的系统也使用维度聚类得到的 anchor 框来预测边界框[15]。网络为每个边界框预测的 4 个坐标：tx、ty、tw、th。假设格子距离图像的左上角偏移量为（cx，cy），先验边界框宽度和高度分别为：pw、ph，则预测结果对应为：

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

During training we use sum of squared error loss. If the ground truth for some coordinate prediction is ^t* our gradient is the ground truth value (computed from the ground truth box) minus our prediction: ^t* - t*. This ground truth value can be easily computed by inverting the equations above.

训练时我们使用误差平方和损失。如果某个预测坐标的真值是t^*，那么梯度就是真值（从真值框计算而得）和预测值之差：t^*-t*。真实值可以很容易地通过变换上述公式得到。

YOLOv3 predicts an objectness score for each bounding box using logistic regression. This should be 1 if the bounding box prior overlaps a ground truth object by more than any other bounding box prior. If the bounding box prior is not the best but does overlap a ground truth object by more than some threshold we ignore the prediction, following [17]. We use the threshold of .5. Unlike [17] our system only assigns one bounding box prior for each ground truth object. If a bounding box prior is not assigned to a ground truth object it incurs no loss for coordinate or class predictions, only objectness.

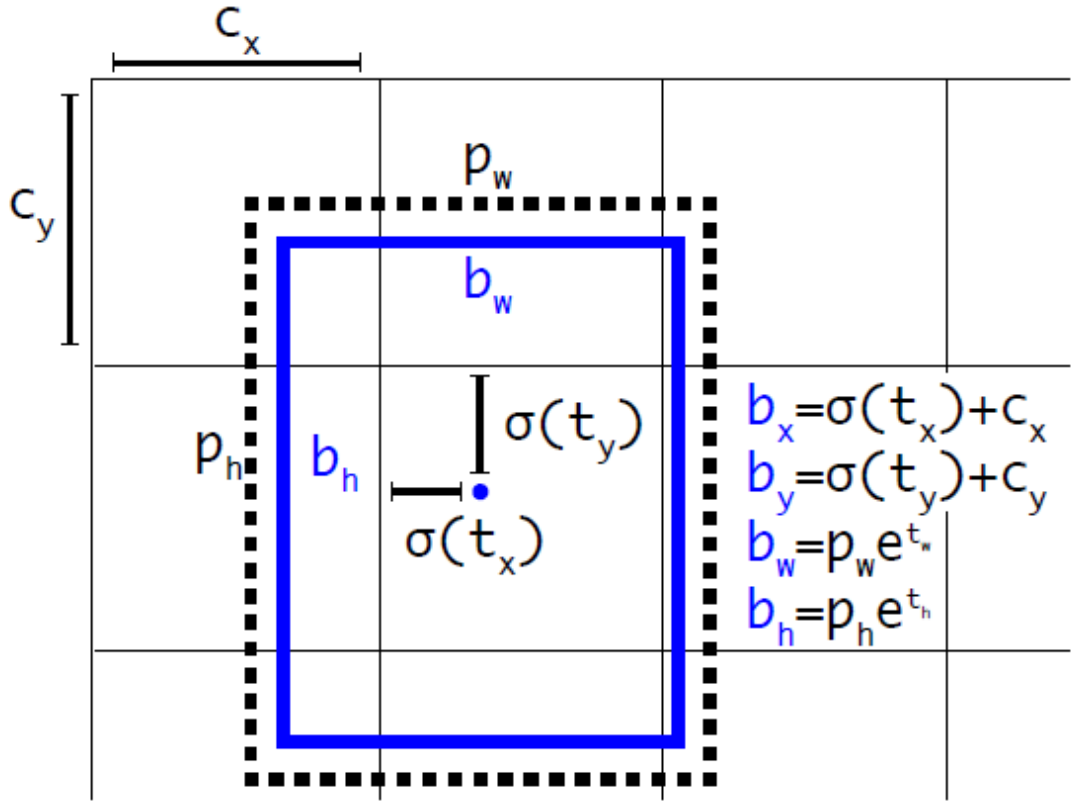


Figure 2. Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function. This figure blatantly self-plagiarized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

YOLOv3 使用逻辑回归预测每个边界框是目标的分数。如果真实标签框与某个边界框重叠的面积比与其他任何边界框都大，那么这个先验边界框得分为 1。按照[17]的做法，如果先验边界框不是最好的，但是确实与目标的真实标签框重叠的面积大于阈值，我们就会忽略这个预测。我们使用阈值为 0.5。与[17]不同，我们的系统只为每个真实目标分配一个边界框。如果先验边界框未分配到真实目标，则不会产生坐标或类别预测的损失，只会产生是否是目标的损失。

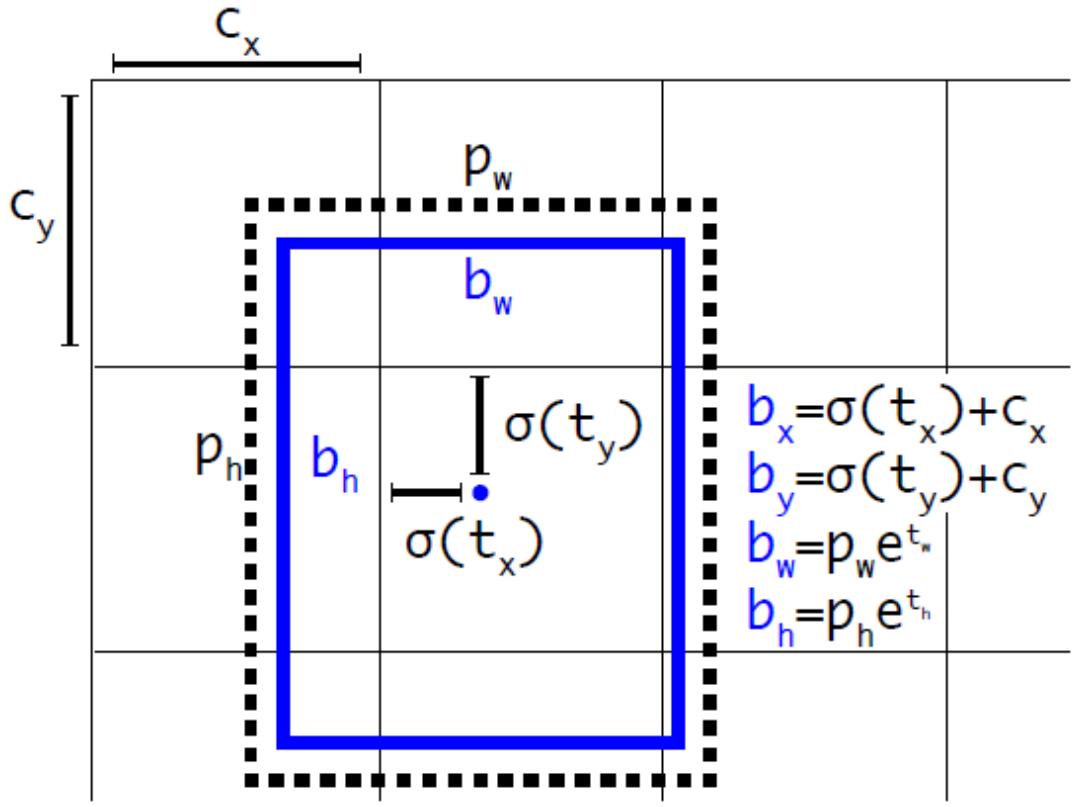


图 2.维度先验和位置预测的边界框。我们使用聚类质心的偏移量预测框的宽度和高度。我们使用 sigmoid 函数预测相对于滤波器应用位置的框的中心坐标。这个图公然引用于自己的论文[15]。

2.2. Class Prediction

Each box predicts the classes the bounding box may contain using multilabel classification. We do not use a softmax as we have found it is unnecessary for good performance, instead we simply use independent logistic classifiers. During training we use binary cross-entropy loss for the class predictions.

2.2 分类预测

每个边界框都会使用多标签分类来预测框中可能包含的类。我们不用 softmax，而是用单独的逻辑分类器，因为我们发现前者对于提升网络性能没什么作用。在训练过程中，我们用 binary cross-entropy（二元交叉熵）损失来预测类别。

This formulation helps when we move to more complex domains like the Open Images Dataset [7]. In this dataset there are many overlapping labels (i.e. Woman and Person). Using a softmax imposes the assumption that each box has exactly one class which is often not the case. A multilabel approach better models the data.

当我们转向更复杂的领域，例如 Open Images Dataset [7]，上面的这种改变将变得很有用。这个数据集中有许多重叠的标签（例如女性和人）。使用 softmax 会强加这样一个假设——即每个框恰好只有一个类别，但通常情况并非如此。多标签的方式可以更好地模拟数据。

2.3. Predictions Across Scales

YOLOv3 predicts boxes at 3 different scales. Our system extracts features from those scales using a similar concept to feature pyramid

networks [8]. From our base feature extractor we add several convolutional layers. The last of these predicts a 3-d tensor encoding bounding box, objectness, and class predictions. In our experiments with COCO [10] we predict 3 boxes at each scale so the tensor is N×N×[3×(4+1+80)] for the 4 bounding box offsets, 1 objectness prediction, and 80 class predictions.

2.3 跨尺度预测

YOLOv3 预测 3 种不同尺度的框。我们的系统使用类似特征金字塔网络的相似概念，并从这些尺度中提取特征[8]。在我们的基础特征提取器上添加了几个卷积层。其中最后一个卷积层预测了一个编码边界框、是否是目标和类别预测结果的三维张量。在我们的 COCO 实验[8]中，我们为每个尺度预测 3 个框，所以对于每个边界框的 4 个偏移量、1 个目标预测和 80 个类别预测，最终的张量大小为 N×N×[3×(4+1+80)]。

Next we take the feature map from 2 layers previous and upsample it by 2×. We also take a feature map from earlier in the network and merge it with our upsampled features using concatenation. This method allows us to get more meaningful semantic information from the upsampled features and finer-grained information from the earlier feature map. We then add a few more convolutional layers to process this combined feature map, and eventually predict a similar tensor, although now twice the size.

接下来，我们从前面的 2 个层中取得特征图，并将其上采样 2 倍。我们还从网络中的较前的层中获取特征图，并将其与我们的上采样特征图进行拼接。这种方法使我们能够从上采样的特征图中获得更有意义的语义信息，同时可以从更前的层中获取更细粒度的信息。然后，我们添加几个卷积层来处理这个特征映射组合，并最终预测出一个相似的、大小是原先两倍的张量。

We perform the same design one more time to predict boxes for the final scale. Thus our predictions for the 3rd scale benefit from all the prior computation as well as fine-grained features from early on in the network.

我们再次使用相同的设计来预测最终尺寸的边界框。因此，第三个尺寸的预测将既能从所有先前的计算，又能从网络前面的层中的细粒度的特征中获益。

We still use k-means clustering to determine our bounding box priors. We just sort of chose 9 clusters and 3 scales arbitrarily and then divide up the clusters evenly across scales. On the COCO dataset the 9 clusters were: (10×13)、(16×30)、(33×23)、(30×61)、(62×45)、(59×119)、(116 × 90)、(156 × 198)、(373 × 326).

我们仍然使用 k-means 聚类来确定我们的先验边界框。我们只是选择了 9 个类和 3 个尺度，然后在所有尺度上将聚类均匀地分开。在 COCO 数据集上，9 个聚类分别为(10×13)、(16×30)、(33×23)、(30×61)、(62×45)、(59×119)、(116 × 90)、(156 × 198)、(373 × 326)。

2.4. Feature Extractor

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!

	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.

2.4 特征提取器

我们使用一个新的网络来进行特征提取。我们的新网络融合了 YOLOv2、Darknet-19 和新发明的残差网络的思想。我们的网络使用

连续的 3×3 和 1×1 卷积层，而且现在多了一些快捷连接（shortcut connetction），而且规模更大。它有 53 个卷积层，所以我们称之为... Darknet-53！

	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			

表 1. Darknet-53.

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.

这个新网络比 Darknet-19 功能强大很多，并且仍然比 ResNet-101 或 ResNet-152 更高效。以下是一些 ImageNet 上的结果：

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

表 2.不同 backbones 的各种网络在准确度、Bn Ops（十亿操作数）、BFLOP/s（每秒十亿浮点操作）和 FPS 上的比较。

Each network is trained with identical settings and tested at 256×256, single crop accuracy. Run times are measured on a Titan X at 256×256. Thus Darknet-53 performs on par with state-of-the-art classifiers but with fewer floating point operations and more speed. Darknet-53 is better than ResNet-101 and 1:5faster. Darknet-53 has similar performance to ResNet-152 and is 2× faster.

每个网络都使用相同的设置进行训练，并在 256×256 的图像上进行单精度测试。运行时间是在 Titan X 上用 256×256 图像进行测量的。因此，Darknet-53 可与最先进的分类器相媲美，但浮点运算更少，速度更快。Darknet-53 比 ResNet-101 更好，且速度快 1.5 倍。Darknet-53 与 ResNet-152 相比性能差不多，但速度快比其 2 倍。

Darknet-53 also achieves the highest measured floating point operations per second. This means the network structure better utilizes the GPU, making it more efficient to evaluate and thus faster. That's mostly because ResNets have just way too many layers and aren't very efficient.

Darknet-53 也实现了最高的每秒浮点运算测量。这意味着网络结构可以更好地利用 GPU，使它的评测更加高效、更快。这主要是因为 ResNets 的层数太多，效率不高。

2.5. Training

We still train on full images with no hard negative mining or any of that stuff. We use multi-scale training, lots of data augmentation, batch normalization, all the standard stuff. We use the Darknet neural network framework for training and testing [14].

2.5 训练

我们仍然在完整的图像上进行训练，没有使用难负样本挖掘（hard negative mining）或其他类似的方法。我们使用多尺度训练，使用大量的数据增强、批量标准化等标准的操作。我们使用 Darknet 神经网络框架进行训练和测试[12]。

3. How We Do

YOLOv3 is pretty good! See table 3. In terms of COCOs weird average mean AP metric it is on par with the SSD variants but is 3× faster. It is still quite a bit behind other models like RetinaNet in this metric though.

	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.

3 我们是如何做的

YOLOv3 表现非常好!请看表 3。就 COCO 的平均 AP 指标而言，它与 SSD 类的模型相当，但速度提高了 3 倍。尽管如此，它仍然在这个指标上比像 RetinaNet 这样的其他模型差些。

	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

表 3.我很认真地从[9]中“窃取”了所有这些表格，他们花了很长时间才从头开始制作。好的，YOLOv3 没问题。请记住，RetinaNet 处理图像的时间要长 3.8 倍。YOLOv3 比 SSD 变体要好得多，可与 AP50 指标上的最新模型相媲美。

However, when we look at the “old” detection metric of mAP at IOU= :5 (or AP_{50} in the chart) YOLOv3 is very strong. It is almost on par with RetinaNet and far above the SSD variants. This indicates that YOLOv3 is a very strong detector that excels at producing decent boxes for objects. However, performance drops significantly as the IOU

threshold increases indicating YOLOv3 struggles to get the boxes perfectly aligned with the object.

然而，当我们使用“旧的”检测指标——在 IOU=0.5 的 mAP（或图表中的 AP_{50}）时，YOLOv3 非常强大。其性能几乎与 RetinaNet 相当，并且远强于 SSD。这表明 YOLOv3 是一个非常强大的检测器，擅长为目标生成恰当的框。然而，随着 IOU 阈值增加，性能显著下降，这表明 YOLOv3 预测的边界框与目标不能完美对齐。

In the past YOLO struggled with small objects. However, now we see a reversal in that trend. With the new multi-scale predictions we see YOLOv3 has relatively high APS performance. However, it has comparatively worse performance on medium and larger size objects. More investigation is needed to get to the bottom of this.

之前的 YOLO 不擅长检测小物体。但是，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 APS 性能。但是，它在中等和更大尺寸的物体上的表现相对较差。需要更多的研究来深入了解这一点。

When we plot accuracy vs speed on the AP_{50} metric (see figure 5) we see YOLOv3 has significant benefits over other detection systems. Namely, it’s faster and better.

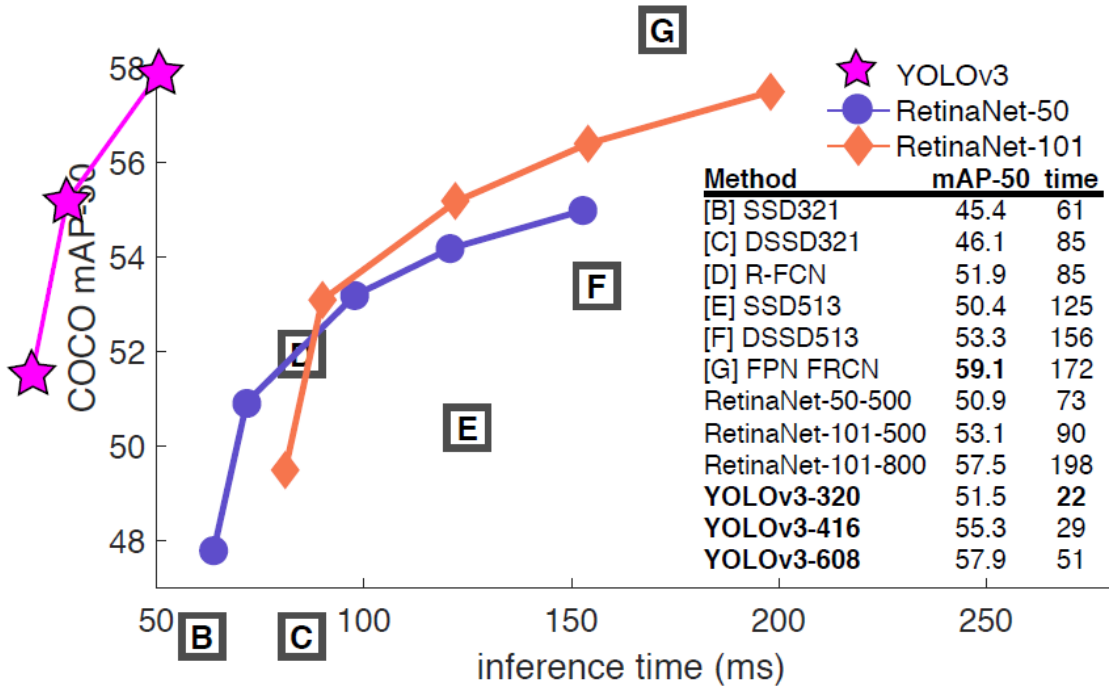
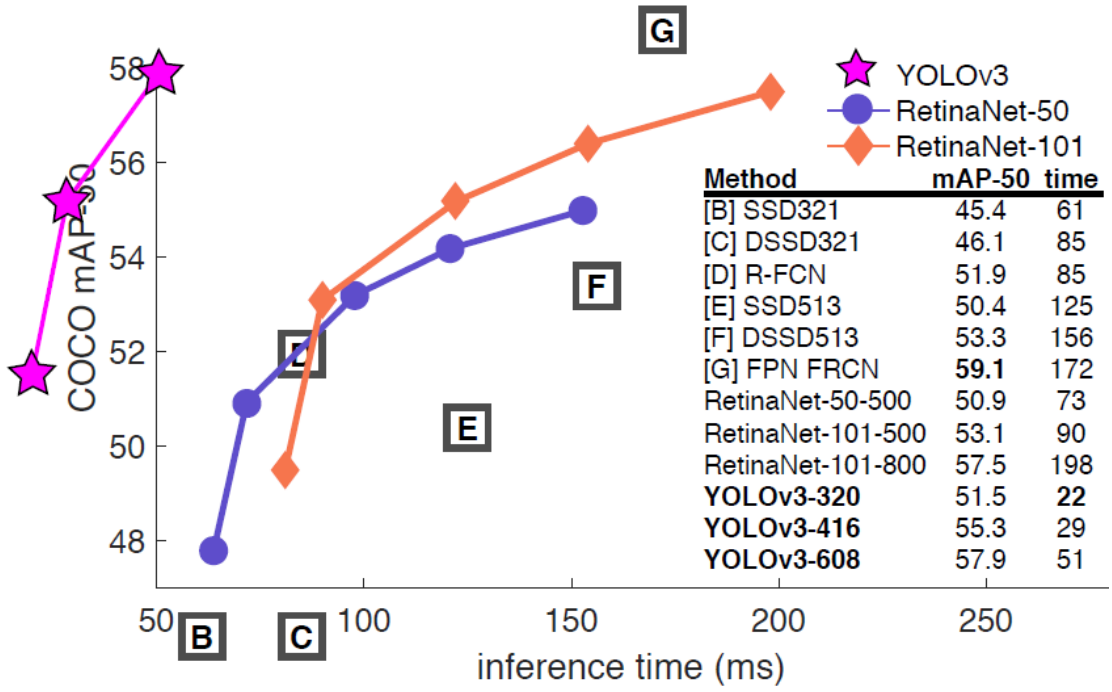


Figure 3. Again adapted from the [9], this time displaying speed/accuracy tradeoff on the mAP at .5 IOU metric. You can tell YOLOv3 is good because it's very high and far to the left. Can you cite your own paper? Guess who's going to try, this guy ! [16]. Oh, I forgot, we also fix a data loading bug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

当我们在 AP_{50} 指标上绘制准确度和速度关系图时（见图 3），我们看到 YOLOv3 与其他检测系统相比具有显着的优势。也就是说，速度更快、性能更好。



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

图 3. 再次改编自[9]，这次显示的是在 0.5 IOU 指标上速度/准确度的折衷。你可以说 YOLOv3 是好的，因为它非常高并且在左边很远。你能引用你自己的论文吗？猜猜谁会去尝试，这个人→[16]。哦，我忘了，我们还修复了 YOLOv2 中的数据加载 bug，该 bug 的修复提升了 2 mAP。将 YOLOv3 结果潜入这幅图中而没有改变原始布局。

4. ThingsWe Tried That Didn't Work

We tried lots of stuff while we were working on YOLOv3. A lot of it didn't work. Here's the stuff we can remember.

4 失败的尝试

我们在研究 YOLOv3 时尝试了很多东西，但很多都不起作用。下面是我们要记住的血的教训。

Anchor box x; y offset predictions. We tried using the normal anchor box prediction mechanism where you predict the x; y offset as a

multiple of the box width or height using a linear activation. We found this formulation decreased model stability and didn't work very well.

Anchor 框的 x、y 偏移预测。我们尝试使用常规的 Anchor 框预测机制，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. We tried using a linear activation to directly predict the x, y offset instead of the logistic activation. This led to a couple point drop in mAP.

用线性激活代替逻辑激活函数进行 x、y 预测。我们尝试使用线性激活代替逻辑激活来直接预测 x、y 偏移。这个改变导致 MAP 下降了几个点。

Focal loss. We tried using focal loss. It dropped our mAP about 2 points. YOLOv3 may already be robust to the problem focal loss is trying to solve because it has separate objectness predictions and conditional class predictions. Thus for most examples there is no loss from the class predictions? Or something? We aren't totally sure.

focal loss。我们尝试使用 focal loss。它使得 mAP 下降 2 个点。YOLOv3 可能已经对 focal loss 试图解决的问题具有鲁棒性，因为它具有单独的目标预测和条件类别预测。因此，对于大多数样本来说，类别预测没有损失？或者有一些？我们并不完全确定。

Dual IOU thresholds and truth assignment. Faster RCNN uses two IOU thresholds during training. If a prediction overlaps the ground truth by .7 it is as a positive example, by [:3□:7] it is ignored, less than .3 for all ground truth objects it is a negative example. We tried a similar strategy but couldn't get good results.

双 IOU 阈值和真值分配。Faster R-CNN 在训练期间使用两个 IOU 阈值。如果一个预测与真实标签框重叠超过 0.7，它就是一个正样本，若重叠为[0.3，0.7]之间，那么它会被忽略，若它与所有的真实标签框的 IOU 小于 0.3，那么一个负样本。我们尝试了类似的策略，但无法取得好的结果。

We quite like our current formulation, it seems to be at a local optima at least. It is possible that some of these techniques could eventually produce good results, perhaps they just need some tuning to stabilize the training.

我们非常喜欢目前的更新，它似乎至少在局部达到了最佳。有些方法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。

5. What This All Means

YOLOv3 is a good detector. It's fast, it's accurate. It's not as great on the COCO average AP between .5 and .95 IOU metric. But it's very good on the old detection metric of .5 IOU.

5 这一切意味着什么

YOLOv3 是一个很好的检测器。速度很快、很准确。它在 COCO 平均 AP 介于 0.5 和 0.95 IOU 之间的指标的上并不理想。但是，对于旧的 0.5 IOU 检测指标上效果非常好。

Why did we switch metrics anyway? The original COCO paper just has this cryptic sentence: “A full discussion of evaluation metrics will be added once the evaluation server is complete”. Russakovsky et al report that that humans have a hard time distinguishing an IOU of .3 from .5! “Training humans to visually inspect a bounding box with IOU of 0.3 and distinguish it from one with IOU 0.5 is surprisingly difficult.” [18] If humans have a hard time telling the difference, how much does it matter?

为什么我们要改变指标？COCO 的原论文只是有这样一句含糊不清的句子：“一旦评估服务器完成，就会生成全面评测指标”。Russakovsky 等人的报告说，人们很难区分 0.3 和 0.5 的 IOU。“训练人类用视觉检查 0.3 IOU 的边界框，并且与 0.5 IOU 的框区别开来是非常困难的。“[16]如果人类很难说出差异，那么它也没有多重要吧？

But maybe a better question is: “What are we going to do with these detectors now that we have them?” A lot of the people doing this research are at Google and Facebook. I guess at least we know the technology is in good hands and definitely won’t be used to harvest your personal information and sell it to.... wait, you’re saying that’s exactly what it will be used for?? Oh.

但是也许更好的问题是：“现在我们有了这些检测器，我们要做什么？”很多做关于这方面的研究的人都受聘于 Google 和 Facebook。我想至少我们知道这项技术在好人的手中，绝对不会被用来收集您的个人信息并将其出售给.....等等，您是说这正是它的用途？oh。

Well the other people heavily funding vision research are the military and they’ve never done anything horrible like killing lots of people with new technology oh wait.....(Footnote: The author is funded by the Office of Naval Research and Google.)

其他花大钱资助视觉研究的人还有军方，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，等等.....（脚注：作者由 the Office of Naval Research and Google 资助支持。）

I have a lot of hope that most of the people using computer vision are just doing happy, good stuff with it, like counting the number of zebras in a national park [13], or tracking their cat as it wanders around their house [19]. But computer vision is already being put to questionable use and as researchers we have a responsibility to at least consider the harm our work might be doing and think of ways to mitigate it. We owe the world that much.

我强烈地希望，大多数使用计算机视觉的人都用它来做一些快乐且有益的事情，比如计算一个国家公园里斑马的数量[11]，或者追踪在附近徘徊的猫[17]。但是计算机视觉已经有很多可疑的用途，作为

研究人员，我们有责任考虑我们的工作可能造成的损害，并思考如何减轻它的影响。我们欠这个世界太多。

In closing, do not@me. (Because I finally quit Twitter).

最后，不要再@我了。（因为哥已经退出 Twitter 这个是非之地了）。

References

参考文献

[1] Analogy. Wikipedia, Mar 2018. 1

[2] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2):303–338, 2010. 6

[3] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659, 2017. 3

[4] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. Iqa: Visual question answering in interactive environments. arXiv preprint arXiv:1712.03316, 2017. 1

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. 3

[6] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z.Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. 3

[7] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. Dataset available from https://github.com/openimages, 2017. 2

[8] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2117–2125, 2017. 2, 3

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll ´ar. Focal loss for dense object detection. arXiv preprint arXiv:1708.02002, 2017. 1, 3, 4

[10] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll ´ar, and C. L. Zitnick. Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer, 2014. 2

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.- Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pages

21–37. Springer, 2016. 3
[12] I. Newton. Philosophiae naturalis principia mathematica. William Dawson & Sons Ltd., London, 1687. 1
[13] J. Parham, J. Crall, C. Stewart, T. Berger-Wolf, and D. Rubenstein. Animal population censusing at scale with citizen science and photographic identification. 2017. 4
[14] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016. 3
[15] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 6517–6525. IEEE, 2017. 1, 2, 3
[16] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv, 2018. 4
[17] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497, 2015. 2
[18] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2121–2131, 2015. 4
[19] M. Scott. Smart camera gimbal bot scanlime:027, Dec 2017. 4
[20] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond skip connections: Top-down modulation for object detection. arXiv preprint arXiv:1612.06851, 2016. 3
[21] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. 2017. 3