YOLOv3原文翻译 YOLOv3：An Incremental Improvement

Abstract（摘要）

我们为YOLO提供了一系列更新！它包含一堆小设计，可以使系统的性能得到更新；也包含一个新训练的、非常棒的神经网络，虽然比上一版更大一些，但精度也提高了。不用担心，虽然体量大了点，它的速度还是有保障的。在输入320×320的图片后，YOLOv3能在22毫秒内完成处理，并取得28.2mAP的成绩。它的精度和SSD相当，但速度要快上3倍。和旧版数据相比，v3版进步明显。在Titan X环境下，YOLOv3的检测精度为57.9AP5057.9AP50，用时51ms；而RetinaNet的精度只有57.5AP5057.5AP50，但却需要198ms，相当于YOLOv3的3.8倍。

1. Introduction（介绍）

你知道有时候你会花一年的时间只是打打电话吗？今年我没有做很多研究。我在Twitter上花了很多时间。也玩了一下GAN。去年我留下了一点点的动力；我设法对YOLO进行了一些改进。但是诚然，没有什么能像这个超级有趣了，只是一小堆（bunch）改变使它变得更好。我也帮助其他人做了一些研究。

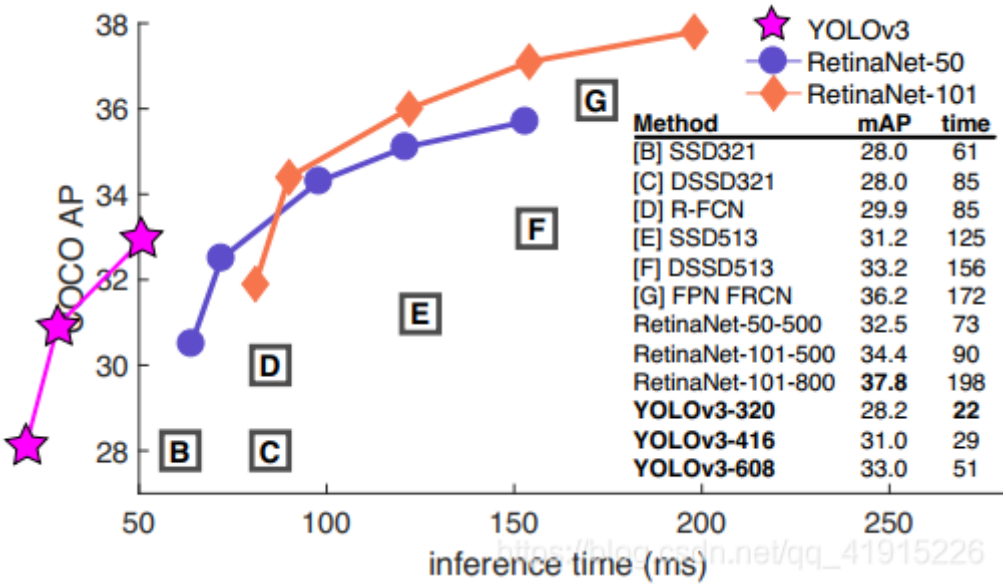
其实，这就是今天带给我们的。我们有一个最终稿截止日期（camera-ready deadline），我们需要引用一些我对YOLO做的随机更新，但我们没有来源。所以开始为技术报告做准备！

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，我们来思考这一切意味着什么。

2. The Deal

谈到YOLOv3的更新情况，其实大多数时候我们就是直接把别人的好点子拿来用了。我们还训练了一个全新的、比其他网络更好的分类网络。为了方便你理解，让我们从头开始慢

慢介绍。



> 图1.我们采用了焦距损失论文的这一数字。YOLOv3的运行速度明显快于其他性能相当的检测方法。无论是M40还是Titan X，它们基本上是同一个GPU。

2.1 Bounding Box Prediction（边界框预测）

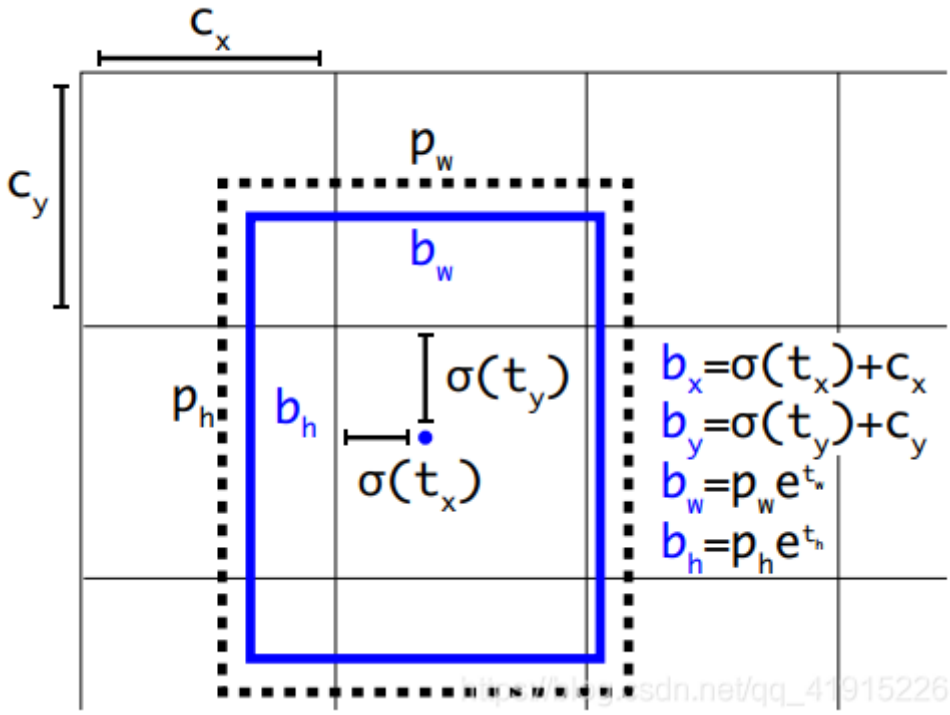
在YOLO9000后，我们的系统开始用维度集群（dimension clusters）固定锚定框（anchor box）来选定边界框。神经网络会为每个边界框预测4个坐标：tx、ty、tw、th。如果目标cell距离图像左上角的边距是（cx, cy），且它对应边界框的宽和高为pw、ph，那么网络的预测值会是：

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

在训练期间，我们会计算平方误差损失。如果预测坐标的ground truth是t^*，那相应的梯度就是ground truth值和预测值的差：t^*-t*。通过对上面的公式变形，可以很容易地计算这个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。



> 图2.**带有维度先验和位置预测的边界框**。我们预测框的宽度和高度是由集群重心的偏移量决定的。我们使用sigmoid函数来预测框的中心坐标相对于滤波器的位置。这一数字公然自我抄袭自[15]。

2.2 Class Prediction（类预测）

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是非必要的，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。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（跨尺度预测）

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[6]。从我们的基本特征提取器中，我们添加了几个卷积层。最后一种方法预测了一个三维张量编码的边界框、对象性和类预测。在我们的COCO实验中，我们在每个尺度上预测了3个框，因此张量是N×N×[3*(4 + 1 + 80)]，用于4个边界盒偏移量、1个客观预测和80个类预测。

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，并使用连接将其与我们的上采样特征合并。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，尽管现在它的大小是原来两倍。

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）。

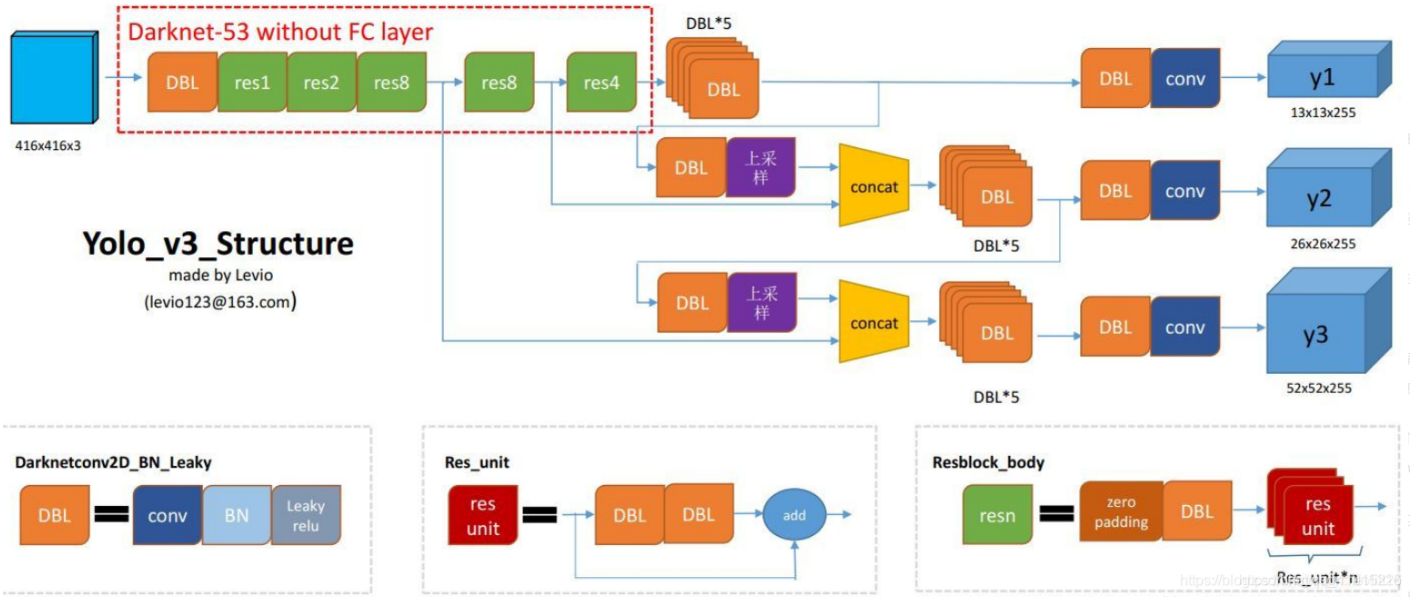
2.4 Feature Extractor（特征提取器）

我们使用新的网络来实现特征提取。我们的新网络是基于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… ***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网络结构



> YOLOV3网络结构。
> DBL:代码中的Darknetconv2d_BN_Leaky，是yolo_v3的基本组件。就是卷积+BN+Leaky relu。
> resn：n代表数字，有res1，res2, … ,res8等等，表示这个res_block里含有多少个res_unit。
> concat：张量拼接。将darknet中间层和后面的某一层的上采样进行拼接。拼接的操作和残差层add的操作是不一样的，拼接会扩充张量的维度，而add只是直接相加不会导致张量维度的改变。

这个新型网络在性能上远超Darknet-19，在效率上同样优于ResNet-101和ResNet-152。下表是在ImageNet上的实验结果：

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.**主干网比较**。准确度，数十亿次运算，每秒数十亿次浮点运算，以及各种网络的FPS

每个网络都使用相同的设置进行训练，输入256×256的图片，并进行单精度测试。运行环境为Titan X。我们得出的结论是Darknet-53在精度上可以与最先进的分类器相媲美，同时它的浮点运算更少，速度也更快。和ResNet-101相比，Darknet-53的速度是前者的1.5倍；而ResNet-152和它性能相似，但用时却是它的2倍以上。

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，使其预测效率更高，速度更快。这主要是因为ResNets的层数太多，效率不高。

2.5 Training（训练）

我们只是输入完整的图像，并没有做其他处理。实验过程中涉及的多尺寸训练、大量数据增强和批量归一化batch normalization等操作均符合标准。模型训练和测试的框架是Darknet神经网络。

3. How We Do（我们怎么做）

YOLOv3的表现非常好！请参见表3，就COCO数据集的平均mAP成绩而言，它与SSD变体相当，但速度提高了3倍。尽管如此，它仍然比像RetinaNet这样的模型要差一点。

	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.我真的只是从[9]偷了这些表，他们花了太长时间从头开始做。YOLOv3做得很好。 记住，RetinaNet处理图像的时间大约是3.8倍。YOLOv3比SSD变种要好得多，在AP50指标上可与最先进的型号相媲美。

如果仔细看这个表，我们可以发现在IOU=.5（即表中的AP50）时，YOLOv3非常强大。它几乎与RetinaNet相当，并且远高于SSD变体。这就证明了它其实是一款非常灵活的检测器，擅长为检测对象生成合适的边界框。然而，随着IOU阈值增加，YOLOv3的性能开始同步下降，这时它预测的边界框就不能做到完美对齐了。

在过去，YOLO一直致力于对小型对象进行检测。但现在我们可以预见其中的演变趋势，随着新的多尺寸预测功能上线，YOLOv3将具备更高的APS性能。但是它目前在中等尺寸或大尺寸物体上的表现还相对较差，仍需进一步的完善。

当我们基于AP50指标绘制精度和速度时（参照图5），我们发现YOLOv3与其他检测系统相比具有显著优势。也就是说，它的速度正在越来越快。

4. Things We Tried That Didn’t Work（我们尝试过却没有成功的工作）

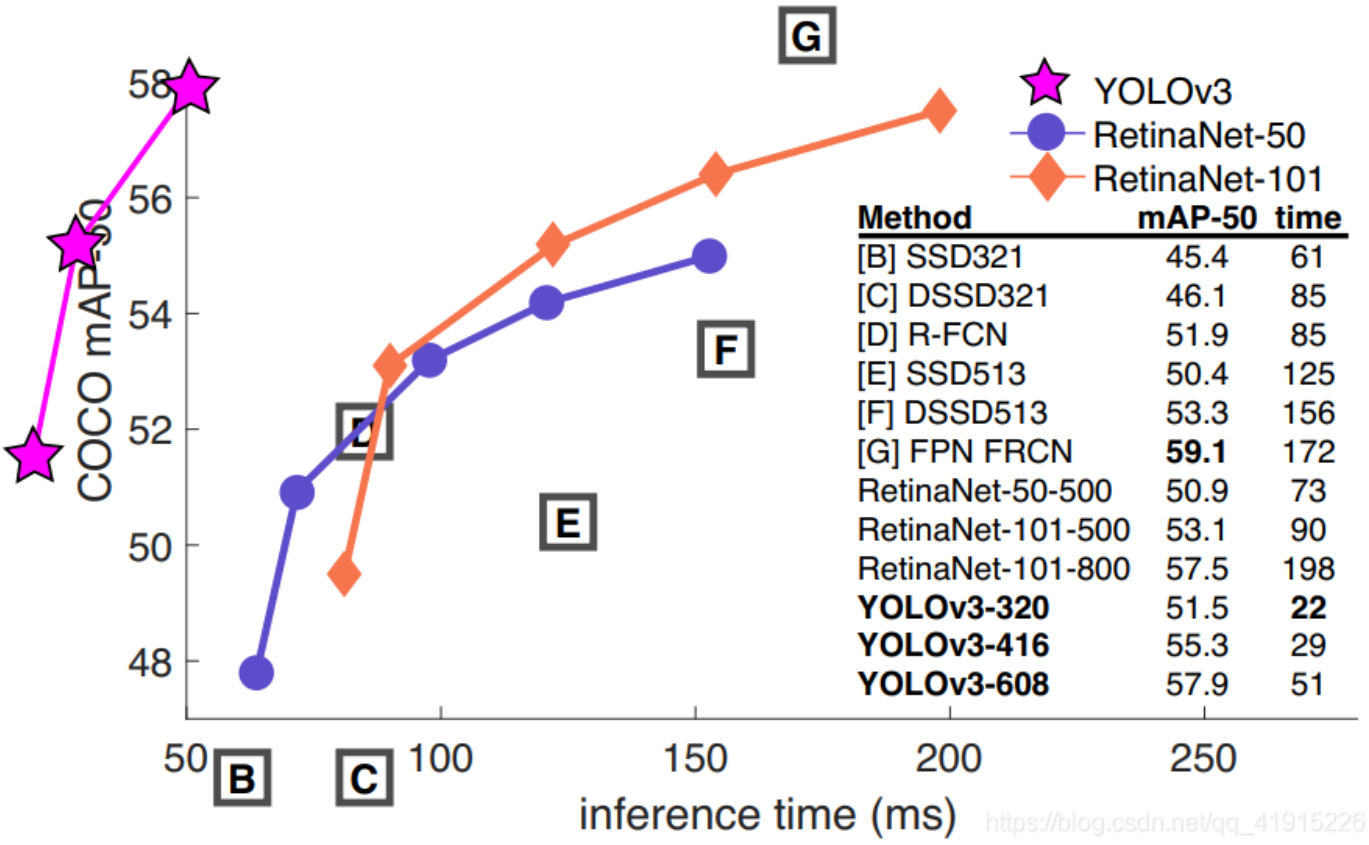
我们在研究YOLOv3时尝试了很多东西，以下是我们还记得的一些失败案例。

Anchor box（锚定框）坐标的偏移预测。我们尝试了常规的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。但我们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。这降低了mAP成绩。

focal loss（焦点损失）。我们尝试使用focal loss，但它使我们的mAP降低了2点。 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0.7的ground truth重合，那它是个正面的结果；如果在[0.3—0.7]之间，则忽略；如果和0.3的ground truth重合，那它就是个负面的结果。我们尝试了这种思路，但效果并不好。我们对现在的更新状况很满意，它看起来已经是最佳状态。有些技术可能会产生更好的结果，但我们还需要对它们做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

> 图3.再次改编自[9]，这一次显示速度/精度权衡在地图上0.5 IOU的度量。你可以说YOLOv3很好，因为它很高而且在左边很远。哦，我忘了，我们还修复了YOLOv2中的一个数据加载错误，它是通过类似于2 mAP的方式来帮助修复的。偷偷把这个弄进来，免得破坏布局。

5. What This All Means（这一切意味着什么）

YOLOv3是一个很好的检测器，它速度快，精度又高。虽然基于0.3和0.95的新指标，它在COCO上的成绩不如人意，但对于旧的检测指标0.5 IOU，它还是非常不错的。

为什么我们要改变指标呢？最初的COCO论文里只有这样一句含糊其词的话：一旦评估完成，就会生成评估指标结果。Russakovsky等人曾经有一份报告，说人类很难区分0.3与0.5的IOU：“训练人们用肉眼区别IOU值为0.3的边界框和0.5的边界框是一件非常困难的事”。如果人类都难以区分这种差异，那这个指标有多重要？

但也许更好的一个问题是：现在我们有了这些检测器，我们能用它们来干嘛？很多从事这方面研究的人都受雇于Google和Facebook，我想至少我们知道如果这项技术发展得完善，那他们绝对不会把它用来收集你的个人信息然后卖给……等等，你把事实说出来了？噢！

那么其他巨资资助计算机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，比如用新技术杀死很多人……呸呸呸

我有很多希望！我希望大多数人会把计算机视觉技术用于快乐的、幸福的事情上，比如计算国家公园里斑马的数量，或者追踪小区附近到底有多少猫。但是计算机视觉技术的应用已经步入歧途了，作为研究人员，我们有责任思考自己的工作可能带给社会的危害，并考虑怎么减轻这种危害。我们非常珍惜这个世界。

最后，不要在Twitter上@我。（我已经不玩Twitter了）

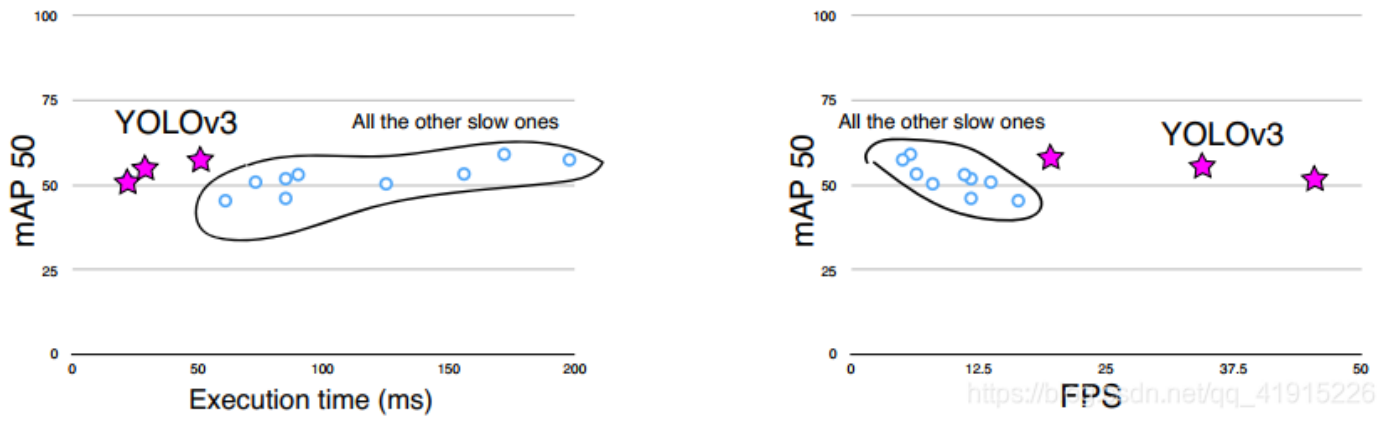


图4.零轴图表可能在智力上更诚实……我们仍然可以通过改变变量来解决问题

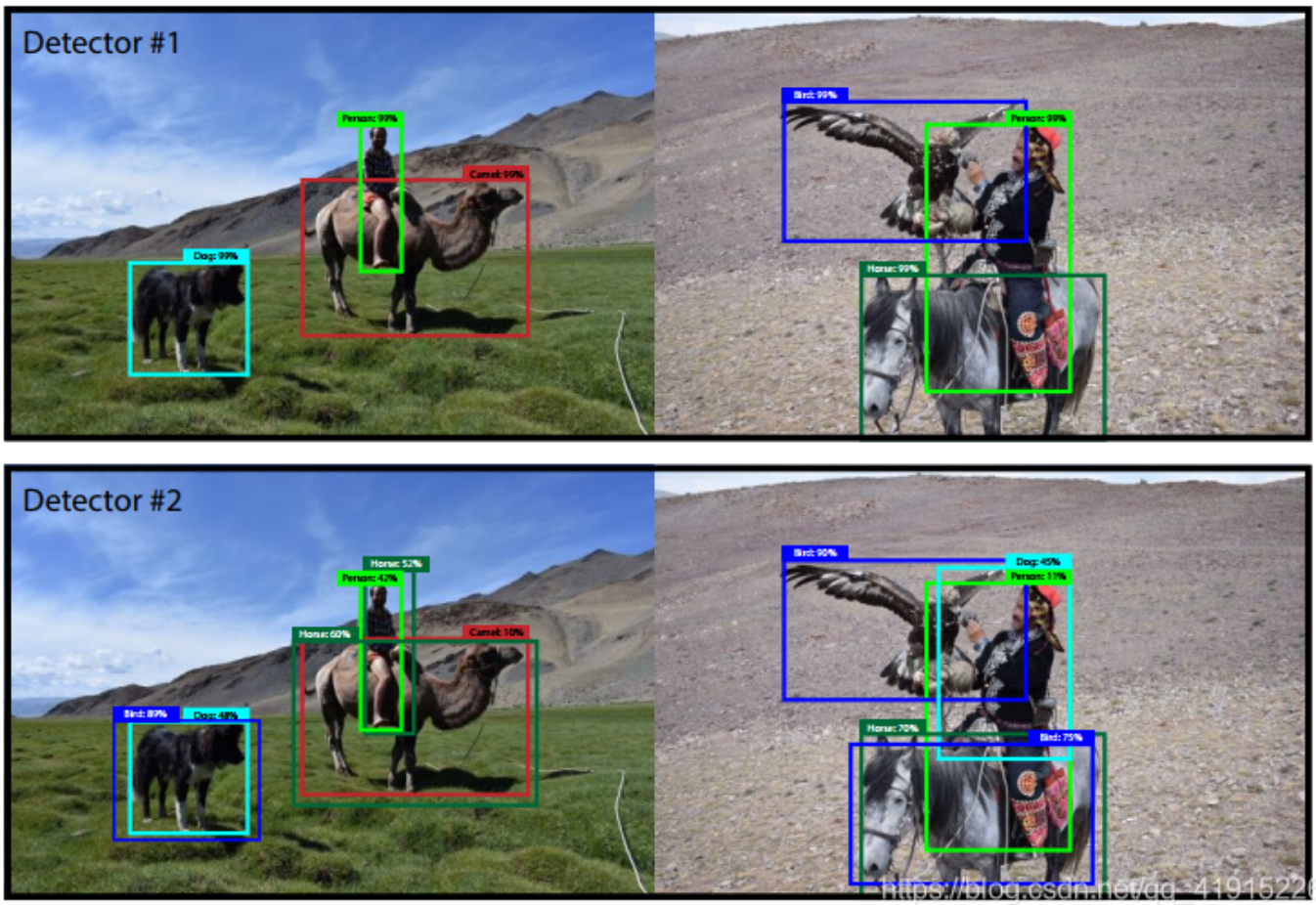


图5.根据这两幅图像的映射，这两个假设的探测器是完美的。他们都是完美的。完全平等。