YOLOv2原文翻译 YOLO9000: Better, Faster, Stronger

Abstract（摘要）

我们将介绍一个先进的，实时目标检测的网络YOLO9000，它可以检测超过9000个类别的物体。首先，我们针对YOLO检测网络提出了许多从以前的工作中得出的，新颖的改进。改进后的网络称为YOLOv2，在标准的检测任务中，比如PASCAL VOC和COCO，它也是一个先进的目标检测网络。使用新颖的多尺度训练方法，相同的YOLOv2模型可以在不同的图像大小下运行，并在速度和准确度之间提供简单的权衡。在VOC 2007数据集上，67FPS时，YOLOv2实现了76.8 mAP。在40 FPS时，YOLOv2获得78.6 mAP，优于最先进的方法，如使用ResNet和SSD的Faster R-CNN，同时运行速度明显更快。最后，我们提出了一个**联合训练目标检测和分类任务的方法**。使用这个方法，我们在COCO检测数据集和ImageNet分类数据集上同时训练YOLO9000网络。我们的联合训练使得**YOLO9000可以预测没有标签的检测数据的对象类别**。我们在ImageNet的检测任务中验证了网络的有效性。YOLO9000在ImageNet检测验证集上获得了19.7 mAP，尽管在200个类中只有44个类具有检测数据。在COCO不包含的156个类别中，YOLO9000获得了16.0 mAP。 但YOLO可以检测到超过200个类别; 它预测了超过9000种不同对象类别的检测。并且它仍然可以实时运行。

1. Introduction（介绍）

通用目标检测应该是快速的、准确的并且能够识别各种各样的物体。由于神经网络的发展，检测网络已经变得越来越快速和准确。然而，大多数检测网络仍然受限于很少的物体范围内。

与用于分类和标记等其他任务的数据集相比，当前目标检测数据集是有限的。最常见的检测数据集包含数千到数十万个具有数十到数百个标签的图像。而分类数据集具有数百万个具有数十或数十万个类别的图像。

我们希望检测能扩展到物体分类的级别。然而，用于检测的标签图像比用于分类或标记的标签要昂贵得多（标签通常是用户免费提供的）。因此，我们不太可能在不久的将来看到与分类数据集相同规模的检测数据集。

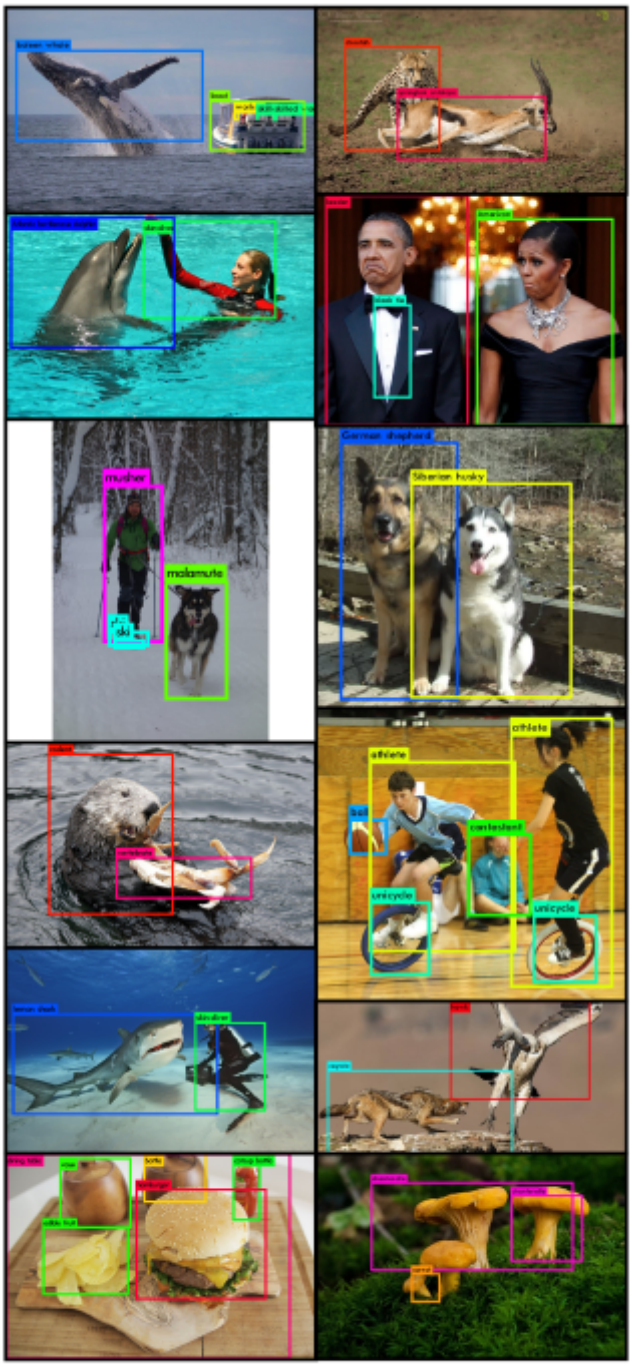


Figure 1: YOLO9000. YOLO9000 can detect a wide variety of object classes in real-time.

> 图1:**YOLO9000**。YOLO9000可以实时检测多种对象类

我们提出了一种新方法来利用我们已有的大量的分类数据，并用它来扩展当前检测系统的范围。我们的方法使用物体分类的分层视图，允许我们将不同的数据集组合在一起。

我们还提出了一种联合训练算法，该算法允许我们在检测和分类数据上训练目标检测网络。我们的方法利用标记的检测图像来学习精确定位物体，同时使用分类图像来增加其词汇量和鲁棒性。

使用这种方法，我们训练YOLO9000，一个可以检测超过9000种不同物体类别的实时目标检测网络。首先，我们改进了基础YOLO检测网络，以得到YOLOv2，这是一种先进的实时检测网络。然后我们使用数据集组合方法和联合训练算法来训练来自ImageNet以及COCO检测数据的超过9000个类的模型。

我们所有的代码和预训练模型可以在网上找到：http://pjreddie.com/yolo9000/

2. Better（更好）

与先进的检测系统相比，YOLO存在各种缺点。与Fast R-CNN相比，YOLO的错误分析表明YOLO产生了大量的定位误差。 此外，与基于region proposal的方法相比，YOLO具有相对较低的召回率。 因此，我们主要关注改善召回率和定位，同时保持分类准确性。

计算机视觉通常趋向于更大，更深的网络。更好的性能通常取决于训练更大的网络或集合多个模型。但是，对于YOLOv2，我们需要更准确的检测，但速度仍然很快。我们没有扩展网络，而是**简化网络**，然后使特征表示更容易学习。我们将过去工作中的各种想法与我们自己的新思想结合起来，以提高YOLO的性能。结果摘要见表2。

	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**

序号	描述		YOLO							YOLOv2
1	batch norm代替dropout防overfit	batch norm?		✓	✓	✓	✓	✓	✓	✓
2	448x448 finetune ImageNet	hi-res classifier?			✓	✓	✓	✓	✓	✓
3	grid: 13 x 13代替7 x7，416 / 32 = 13为奇数	convolutional?				✓	✓	✓	✓	✓
4	除去FC，加入anchors，提高recall	anchor boxes?				✓	✓			
5	设计新分类网络darknet-19作为基础网络	new network?					✓	✓	✓	✓
6	Kmeans离线选取anchor个数k=5	dimension priors?						✓	✓	✓
7	新的位置预测方法	location prediction?						✓	✓	✓
8	增加跳级路特征 => 结合高低分辨率	passthrough?							✓	✓
9	FCN能适用多尺度(32倍数)，尺度=> 分格数	multi-scale?								✓
10	-	hi-res detector?								✓
		VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

yolo v2

表2：**从YOLO到YOLOv2的改变**。列出的大多数设计决策都会导致mAP的显著增加。有两个例外是切换到带有锚定框的全卷积网络和使用新网络。在使用新的网络切割计算时，切换到锚定框方法在不改变mAP情况下提高了召回率33%

Batch Normalization. **批量归一化**导致收敛的显着改善，同时消除了对其他形式的正则化的需要。通过在YOLO中的所有卷积层上添加批量归一化，我们可以使mAP提高2%以上。批量归一化也有助于模型正则化。通过批量归一化，我们可以从模型中删除dropout而不会过度拟合。

High Resolution Classifier.（高分辨率分类器）所有最先进的检测方法都使用在ImageNet上预先训练的分类器。从AlexNet开始，大多数分类器在小于256 * 256的输入图像上运行。原始的YOLO训练分类器网络为224 * 224并增加分辨率为448 * 448进行检测。这意味着网络必须同时切换到学习目标检测并调整到新的输入分辨率。

对于YOLOv2，我们首先在ImageNet上以完整的448 * 448分辨率微调分类网络10个epoch（迭代周期）。这使网络有时间调整其卷积核，以便在更高分辨率的输入上更好地工作。然后我们在检测时对生成的网络进行微调。高分辨率分类网络使我们的mAP增加了近4%。

Convolutional With Anchor Boxes.（锚框卷积） YOLO直接使用卷积特征提取器顶部的全连接层预测边界框的坐标。而不是直接预测坐标的Faster R-CNN使用手工挑选的先验预测边界框。仅使用卷积层，Faster R-CNN中的区域提议网络（RPN）预测anchor boxes的偏移和置信度。由于预测层是卷积的，因此RPN在特征图中的每个位置预测这些偏移。预测偏移而不是坐标简化了问题，使网络更容易学习。

我们移除了YOLO网络中的全连接层并使用anchor boxes来预测边界框。首先，我们移除一个pooling层，使网络卷积层的输出分辨率更高。我们还缩小网络用来在416 * 416输入图像而不是448 * 448上操作。我们这样做是因为我们想要在我们的特征图中有奇数个位置，所以只有一个中心单元格。物体，特别是大物体，往往占据图像的中心，因此最好在中心有一个位置来预测这些物体而不是附近的四个位置。YOLO的卷积层将图像缩小了32倍，因此通过使用416 * 416的输入图像，我们得到13 * 13的输出特征图。

当我们使用anchor boxes时，我们还将类预测机制与空间位置预测分离，而不是预测每个anchor box的类和objectness（指anchor boxes中是否有物体）。同YOLO网络一样，objectness预测仍然预测真实框与预测框之间的IOU（Pr(Object) * IOU），并且类别预测在给定存在对象的情况下预测该类的条件概率（Pr(Class[i] | Object)）。

使用anchor boxes我们的准确性会略有下降。 YOLO仅预测每张图片98个boxes，但是使用anchor boxes我们的模型预测超过1000个boxes。没有anchor boxes，我们的中间模型获得69.5 mAP，召回率为81%。使用anchor boxes，我们的模型获得69.2 mAP，召回率为88%。虽然mAP减少，召回的增加也意味着我们的模型有更大的改进空间。

Dimension Clusters.（维度群集） 在YOLO网络中使用anchor boxes时，我们遇到了两个问题。首先是box维度是手工挑选的。网络可以学会适当地调整框，如果我们从网络中选择更好的先验，我们可以让网络更容易学习预测好的检测。

我们不是手动选择先验，而是在训练集边界框上运行k-means聚类，以自动找到好的先验。 如果我们使用具有欧几里德距离的标准k-means，那么较大的框会产生比较小的框更多的误差。然而，我们真正想要的是能够获得良好IOU分数的先验，这与框boxes的大小无关。 因此，对于我们的距离度量，我们使用：

$$d(\text{box}, \text{centroid}) = 1 - \text{IOU}(\text{box}, \text{centroid})$$

我们为各种k值运行k-means并绘制具有最接近聚类中心的平均IOU，参见图2。我们选择k = 5作为模型复杂度和高召回率之间的良好权衡。聚类中心与手工挑选的anchor boxes明显不同。宽，短的boxes更少，相反高，瘦的boxes更多。

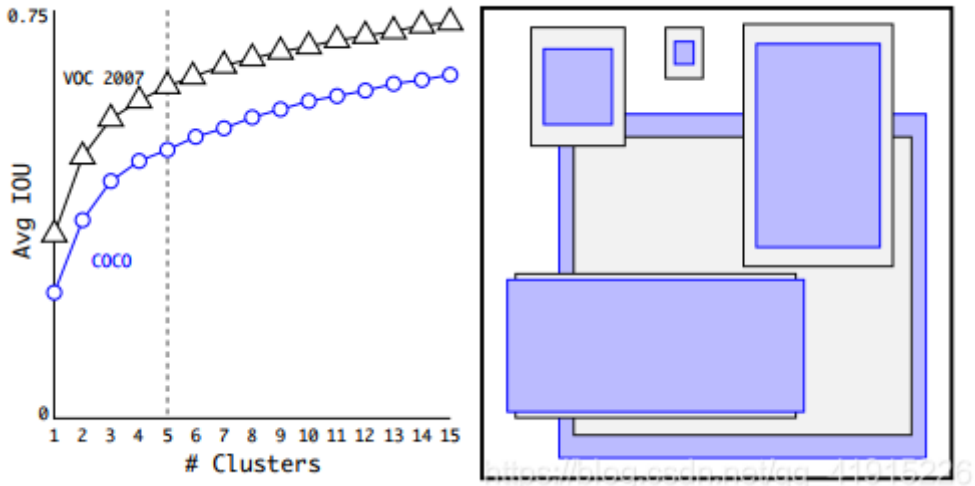


图2:**VOC和COCO上的聚类框维度。**我们对边界框的维数进行k-means聚类，得到模型的良好先验。左边的图片显示了我们对k的各种选择得到的平均IOU。我们发现k=5给出了一个很好的取舍，即回忆和模型的复杂性。右图显示了VOC和COCO的相对质心。这两组优先级都倾向于更薄、更高的框，而COCO在尺寸上的差异大于VOC。

我们比较聚类策略和手工挑选anchor boxes的平均IOU，如表1所示。只有5个聚类中心时的性能与9个anchor boxes性能类似，即平均IOU为61.0 VS. 60.9。如果我们使用9个聚类中心，我们会看到更高的平均IOU (67.2)。这表明使用k-means生成我们的边界框会以更好的表示方式启动模型，使任务更容易学习。

Box Generation	#	Avg IOU
Cluster SSE	5	58.7
Cluster IOU	5	61.0
Anchor Boxes [15]	9	60.9
Cluster IOU	9	67.2

表1:**VOC 2007上最接近优先级的框的平均IOU。**使用不同生成方法之前，VOC 2007上最接近、未修改的对象的平均IOU。聚类结果比使用人工挑选的先验值要好得多。

Direct Location Prediction.（直接定位预测） 在YOLO网络中使用anchor boxes时，我们遇到了第二个问题：模型不稳定，特别是在早期迭代期间。大多数不稳定性来自于预测box的（x, y）位置。在区域提议网络RPN中，网络预测值 tx 和 ty，并且（x, y）中心坐

标计算为：

$$x = (t_x * w_a) - x_a$$
$$y = (t_y * h_a) - y_a$$

例如，tx = 1的预测会将框box向右移动anchor box的宽度，tx = -1的预测会将其向左移动相同的量。

这个公式是不受约束的，因此任何anchor box都可以在图像中的任何位置结束，无论预测box的位置如何。随机初始化，模型需要很长时间才能稳定以预测合理的偏移。

我们不是预测偏移offset，而是遵循YOLO的方法，预测相对于网格单元的位置坐标。 这将真实框限制在0和1之间。我们使用逻辑激活logistics activation来约束网络预测落在此范围内。

网络在输出的feature map中预测每个单元格的5个边界框。网络预测每个边界框的 5 个坐标，tx，ty，tw，th和to。如果单元格从图像的左上角偏移（cx, cy）并且前面的边界框具有宽度和高度pw，ph，则预测对应于：

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

由于我们约束位置进行预测，因此参数化更容易学习，使网络更稳定。使用维度聚类以及直接预测边界框中心位置可使YOLO比使用anchor boxes的版本提高近5%。

Fine-Grained Features. （细粒度特征）改进后的YOLO在13 *13特征图上预测检测。虽然这对于大型物体来说已足够，但它可能会受益于用于定位较小对象的更细粒度的特征。Faster R-CNN和SSD都在网络中的各种特征图上运行其提议网络RPN，以获得一系列分辨率。我们采用不同的方法，只需添加一个直通层(passthrough layer)，以26 * 26的分辨率从较早的层中获取特征。

直通层通过将相邻特征堆叠到不同的通道而不是空间位置，将较高分辨率的特征与低分辨率特征连接起来，类似于ResNet中的identity mappings。这将 26 * 26 * 512 特征图转换为 13 * 13 * 2048 特征图，可以与原始特征连接。我们的检测器运行在这个扩展的特征图之上，因此它可以访问细粒度的特征。这使得性能提高1%。

Multi-Scale Training.(多尺度训练) 最初的YOLO使用输入分辨率为448 * 448。通过添加锚定框，我们将分辨率更改为416 * 416。但是，由于我们的模型仅使用卷积和池化层，

因此可以动态调整大小。我们希望YOLOv2能够在不同尺寸的图像上鲁棒运行，因此我们将其训练到模型中。

我们并没有固定输入图像的尺寸，而是每隔几次迭代就改变一次网络。每10批（batch）我们的网络随机选择一个新的图像尺寸大小。由于我们的模型下采样率为32，我们从32的倍数中提取{320；352；…；608}。因此最小的选项是320×320，最大的是608×608。我们将网络调整到这个维度并继续训练。

这种方式迫使网络学习如何在各种输入维度上做好预测。这意味着同一网络可以预测不同分辨率的检测。网络在较小的图像尺寸运行得更快，因此YOLOv2可在速度和精度之间轻松权衡。

在低分辨率下，*YOLOv2*作为相当精确的检测器运行。在288 * 288分辨率时，它的运行速度超过90 FPS，mAP几乎与Fast R-CNN一样好。这使其成为较小GPU，高帧率视频或多视频流的理想选择。

在高分辨率下，*YOLOv2*是最先进的检测器，在VOC 2007上具有78.6 mAP，同时仍然高于实时速度。有关YOLOv2与VOC 2007其他框架的比较，请参阅表3，图4.

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

表3:**PASCAL VOC 2007检测框架。**
YOLOv2比以前的检测方法更快更准确。它还可以以不同的分辨率运行，以便在速度和精度之间进行简单的权衡。每个YOLOv2条目实际上是具有相同权重的相同训练模型，只是在不同的大小下求值。所有的时间信息都在Geforce GTX Titan X上（原始的，不是Pascal模型）。

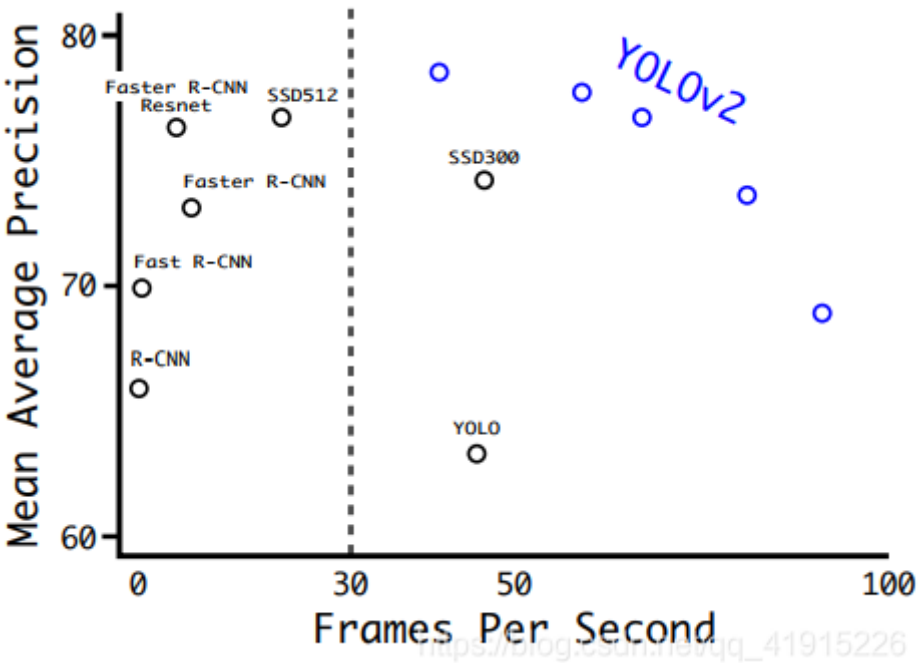


图4:**VOC 2007的准确性和速度。**

进一步实验. 我们训练YOLOv2用于检测VOC 2012数据集。表4 展示了YOLOv2与其他最先进的检测系统的性能比较。YOLOv2达到73.4 mAP，同时检测速度比其它方法快得多。我们还在COCO数据集上进行训练并在表5 中的其他方法进行比较。在VOC指标（IOU = 0.5）上，YOLOv2达到44.0 mAP，与SSD和更快的R-CNN相当。

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
Faster R-CNN [15]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [14]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

表4:**PASCAL VOC12012测试检测结果。**YOLOv2的性能与最先进的探测器相当，如带有ResNet和SSD512的Faster R-CNN，速度快2-10倍。

		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast R-CNN [5]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast R-CNN[1]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster R-CNN[15]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [1]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster R-CNN[10]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300 [11]	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512 [11]	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**
YOLOv2 [11]	trainval35k	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4

表5:**COCO测试结果-dev2015。**

3. Faster（更快）

我们希望检测准确，但我们也希望检测速度快。大多数检测应用程序（如机器人或自动驾驶汽车）都依赖于低延迟预测。为了最大限度地提高性能，我们将YOLOv2设计为从头开始加速。

大多数检测框架依赖于VGG-16作为基本特征提取器。VGG-16是一个功能强大，准确的分类网络，但它非常复杂。VGG-16的卷积层需要306.9亿浮点运算，在224 * 224分辨率的单个图像上进行单次检测。

YOLO框架使用基于GoogLenet 架构的自定义网络。该网络比VGG-16更快，在前向计算时仅使用85.2亿次浮点运算。 然而，它的准确性略差于VGG-16 对于224 * 224 single-crop的top-5 accuracy，YOLO定制型在ImageNet上为88.0%，而VGG-16为90.0%。

Darknet-19. 我们提出了一种新的分类模型作为YOLOv2的基础。我们的模型建立在网络设计的先前工作以及该领域的常识之上。与VGG型号类似，我们大多使用 3 * 3 卷积核，并且在进行池化步骤后将通道数量增加一倍。效仿Network in Network (NIN)，我们使用全局平均池化来进行预测以及1*1卷积核来压缩3 * 3卷积之间的特征表示。我们使用批量归一化 (BN) 来稳定训练，加速收敛，并使模型正则化。

我们的最终模型名为Darknet-19，有19个卷积层和5个最大池化层。有关完整说明，请参阅表6。Darknet-19仅需要55.8亿次浮点运算来处理图像，但在ImageNet上实现了

72.9% 的top-1精度和 91.2% top-5精度。

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			

表6: **Darknet-19**

分类培训。我们使用Darknet神经网络框架，在标准ImageNet 1000类分类数据集上训练网络，使用随机梯度下降（起始学习率为0:1）、多项式速率衰减（幂为4）、权重衰减（权重衰减为0.0005）和动量（动量为0.9）对160个时代进行分类。在训练过程中，我们使用标准的数据增强技巧，包括随机作物、旋转、色调、饱和度和曝光偏移。

如上所述，在224×224的初始图像训练之后，我们将网络微调为更大的448。对于这种微调，我们使用上述参数进行训练，但仅针对10个阶段，并以10^{-3}的学习速率开始。在这种高分辨率下，我们的网络达到了76.5%的前1精度和93.3%的前5精度。

检测培训。我们通过去掉最后一个卷积层，并在三个3×3卷积层上加上1024个滤波器，每个滤波器后加上最后一个1×1卷积层，再加上需要检测的输出数，来修改该网络进行检测。对于VOC，我们预测5个框，每个框有5个坐标，每个框有20个类，因此125个过滤器。我们还添加了一个从最后的3×3×512层到第二到最后的卷积层的穿过层，这样我们的模型就可以使用细粒度特征。

我们对网络进行160个阶段的训练，起始学习率为10^{-3}，在60和90个迭代周期将其除以10。我们使用0.0005的重量衰减和0.9的动量。我们使用类似于YOLO和SSD的随机作物、颜色变化等数据增强。我们对COCO和VOC使用相同的训练策略。

4. Stronger（更强）

我们提出了一个联合训练分类和检测数据的机制。我们的方法使用标记为检测的图像来学习边界框坐标预测和目标之类的特定检测信息以及如何对常见目标进行分类。它使用仅具有类别标签的图像来扩展可检测类别的数量。

在训练期间，我们混合来自检测和分类数据集的图像。当我们的网络看到标记为检测的图像时，我们可以基于完整的YOLOv2损失函数进行反向传播。当它看到一个分类图像时，我们只能从该架构的分类特定部分反向传播损失。

这种方法提出了一些新的挑战。检测数据集只有通用目标和通用标签，如“狗”或“船”。分类数据集具有更广更深的标签范围。ImageNet有超过一百种品种的狗，包括Norfolk terrier，Yorkshire terrier和Bedlington terrier。如果我们想在两个数据集上训练，我们需要一个连贯的方式来合并这些标签。

大多数分类方法使用跨所有可能类别的softmax层来计算最终的概率分布。使用softmax假定这些类是相互排斥的。这给数据集的组合带来了问题，例如你不想用这个模型来组合ImageNet和COCO，因为类Norfolk terrier和dog不是相互排斥的。

我们可以改为使用多标签模型来组合不假定互斥的数据集。这种方法忽略了我们已知的关于数据的所有结构，例如，所有的COCO类是互斥的。

Hierarchical classification（分层分类。）ImageNet标签是从WordNet中提取的，这是一个构建概念及其相互关系的语言数据库。在WordNet中，Norfolk terrier和Yorkshire terrier都是terrier的下义词，terrier是一种hunting dog，hunting dog是dog，dog是canine等。分类的大多数方法为标签假设一个扁平结构，但是对于组合数据集，结构正是我们所需要的。

WordNet的结构是有向图，而不是树，因为语言是复杂的。例如，dog既是一种canine，也是一种domestic animal，它们都是WordNet中的同义词。我们不是使用完整的图结构，而是通过从ImageNet的概念中构建分层树来简化问题。

为了构建这棵树，我们检查了ImageNet中的视觉名词，并查看它们通过WordNet图到根节点的路径，在这种情况下是“物理对象”。许多同义词通过图只有一条路径，所以首先我们将所有这些路径添加到我们的树中。然后我们反复检查我们留下的概念，并尽可能少地添加生长树的路径。所以如果一个概念有两条路径到一个根，一条路径会给我们的树增加三条边，另一条只增加一条边，我们选择更短的路径。

最终的结果是WordTree，一个视觉概念的分层模型。为了使用WordTree进行分类，我们预测每个节点的条件概率，以得到同义词集合中每个同义词下义词的概率。例如，在terrier节点我们预测：

$$
\begin{aligned}
&Pr(\text{Norfolk terrier}|\text{terrier}) \\
&Pr(\text{Yorkshire terrier}|\text{terrier}) \\
&Pr(\text{Bedlington terrier}|\text{terrier}) \\
&\qquad\qquad \ldots
\end{aligned}
$$

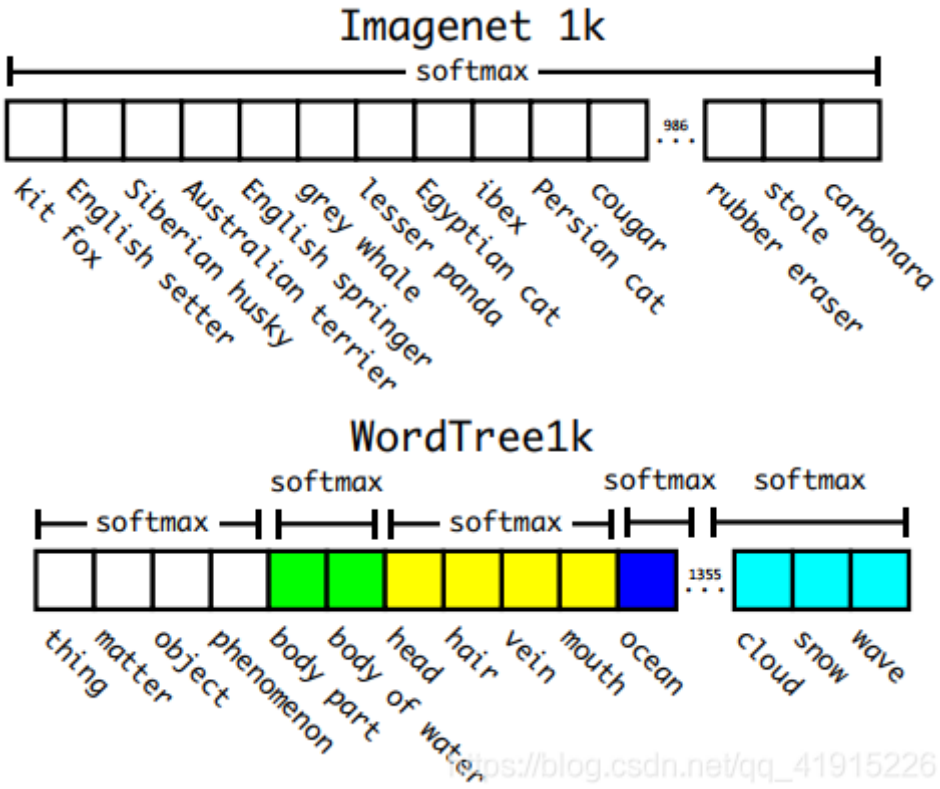
如果我们想要计算一个特定节点的绝对概率，我们只需沿着通过树到达根节点的路径，再乘以条件概率。所以如果我们想知道一张图片是否是Norfolk terrier，我们计算：

$$
\begin{aligned}
Pr(\text{Norfolk terrier}) = {} & Pr(\text{Norfolk terrier}|\text{terrier}) \\
& * Pr(\text{terrier}|\text{hunting dog}) \\
& \quad * \ldots * \\
& * Pr(\text{mammal}|Pr(\text{animal}) \\
& * Pr(\text{animal}|\text{physical object})
\end{aligned}
$$

为了分类目的，我们假定图像包含一个目标：$\Pr(\text{physical object}) = 1$。

为了验证这种方法，我们在使用1000类ImageNet构建的WordTree上训练Darknet-19模型。为了构建WordTree1k，我们添加了所有将标签空间从1000扩展到1369的中间节点。在训练过程中，我们将真实标签向树上面传播，以便如果图像被标记为Norfolk terrier，则它也被标记为dog和mammal等。为了计算条件概率，我们的模型预测了具有1369个值

的向量，并且我们计算了相同概念的下义词在所有同义词集上的softmax，见图5。



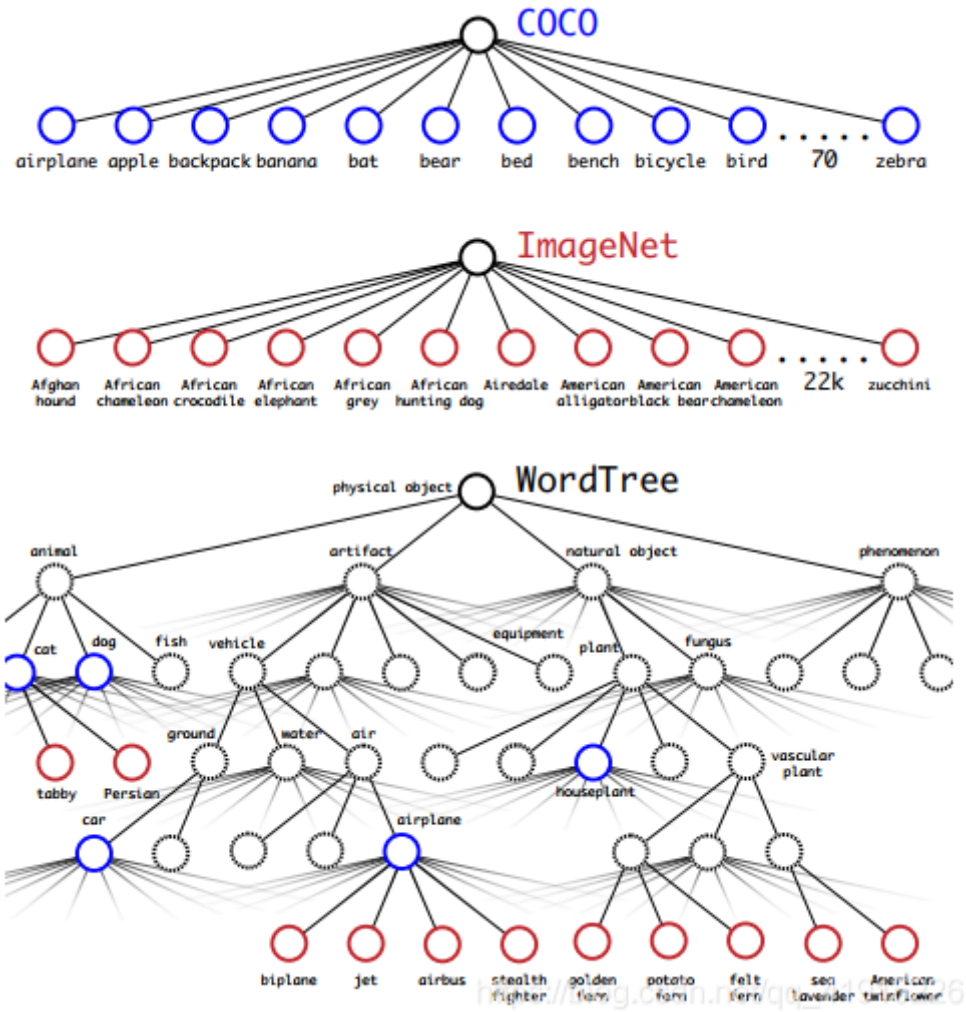
> 图5：**在ImageNet与WordTree上的预测。**大多数ImageNet模型使用一个较大的softmax来预测概率分布。使用WordTree，我们可以在共同的下义词上执行多次softmax操作。

使用与以前相同的训练参数，我们的分级Darknet-19达到71.9%的top-1准确率和90.4%的top-5准确率。尽管增加了369个额外的概念，而且我们的网络预测了一个树状结构，但我们的准确率仅下降了一点点。以这种方式进行分类也有一些好处。在新的或未知的目标类别上性能会优雅地降低。例如，如果网络看到一只狗的照片，但不确定它是什么类型的狗，它仍然会高度自信地预测“狗”，但是在下义位扩展之间有更低的置信度。

这个构想也适用于检测。现在，我们不是假定每张图像都有一个目标，而是使用YOLOv2的目标预测器给我们$Pr(\text{physical object})$的值。检测器预测边界框和概率树。我们遍历树，在每个分割中采用最高的置信度路径，直到达到某个阈值，然后我们预测目标类。

数据集与WordTree的组合。我们可以使用WordTree以合理的方式将多个数据集组合在一起。我们只需将数据集中的类别映射到树中的语法集。图6显示了一个使用WordTree合并ImageNet和COCO中的标签的示例。WordNet是非常多样化的，所以我们可以在大多数

数据集中使用这种技术。



> 图6：**使用WordTree层次结构组合数据集。**使用WordNet概念图，我们构建了一个可视化概念的层次树。然后，我们可以通过将数据集中的类映射到树中的语法集来将数据集合并在一起。这是一个简化的WordTree视图，用于演示。

联合分类和检测。现在我们可以使用WordTree组合数据集，我们可以在分类和检测上训练联合模型。我们想要训练一个非常大规模的检测器，所以我们使用COCO检测数据集和完整的ImageNet版本中的前9000个类来创建我们的组合数据集。我们还需要评估我们的方法，以便从ImageNet检测挑战中添加任何尚未包含的类。该数据集的相应WordTree有9418个类别。ImageNet是一个更大的数据集，所以我们通过对COCO进行过采样来平衡数据集，使得ImageNet仅仅大于4:1的比例。

我们用这个数据集训练YOLO9000。使用基本的YOLOv2架构，但只使用3个prior而不是5来限制输出大小。当我们的网络看到检测图像时，我们会像正常情况一样反向传播损耗。对于分类损失，我们只在标签的相应级别或更高级别上反向传播损失。例如，如果标签是“dog”，我们将错误分配给树下更远的预测，“German Shepherd”与“Golden Retriever”，因为我们没有这些信息。

当它看到分类图像时，我们只反向传播分类损失。我们只需找到预测该类的最高概率的边界框，然后计算其预测树上的损失。我们还假设预测框与ground truth（地面真值）标签重叠至少0.3IOU，并基于此假设反向传播目标损失。

使用这种联合训练，YOLO9000学习使用COCO中的检测数据来查找图像中的目标，并学习使用来自ImageNet的数据对各种目标进行分类。

我们在ImageNet检测任务上评估YOLO9000。ImageNet的检测任务与COCO共享44个目标类别，这意味着YOLO9000只能看到大多数测试图像的分类数据，而不是检测数据。YOLO9000在从未见过任何标记的检测数据的情况下，整体上获得了19.7 mAP，在不相交的156个目标类别中获得了16.0 mAP。这个mAP高于DPM的结果，但是YOLO9000在不同的数据集上训练，只有部分监督。它也同时检测9000个其他目标类别，所有的都是实时的。

当我们分析YOLO9000在ImageNet上的表现时，我们发现它很好地学习了新的动物种类，但是却在像服装和设备这样的学习类别中表现差劲。新动物更容易学习，因为目标预测可以从COCO中的动物泛化的很好。相反，COCO没有任何类型的衣服的边界框标签，只针对人，因此YOLO9000正在努力建模“墨镜”或“泳裤”等类别。

diaper	0.0
horizontal bar	0.0
rubber eraser	0.0
sunglasses	0.0
swimming trunks	0.0
...	
red panda	50.7
fox	52.1
koala bear	54.3
tiger	61.0
armadillo	61.7

> 表7:**ImageNet上的YOLO9000最佳和最差类。**
> 156个弱监督类中AP最高和最低的类。YOLO9000学习各种动物的好模型，但却在与服装或设备等新课程作斗争

5. Conclusion（结论）

我们介绍实时检测系统YOLOv2和YOLO9000。**YOLOv2是先进的检测网络**，在各种检测数据集中比其他检测系统更快。此外，它可以在各种图像尺寸下运行，以在速度和精度之间提供平滑的折衷。

YOLO9000是一个实时框架，通过联合优化检测和分类来检测9000多个物体类别。我们使用WordTree组合来自各种来源的数据，以及我们的联合优化技术以同时在ImageNet和COCO上进行训练。YOLO9000是缩小检测和分类之间数据集大小差距的重要一步。

我们的许多技术都在目标检测之外进行推广。 ImageNet的WordTree表示为图像分类提供了更丰富，更详细的输出空间。使用分层分类的数据集组合在分类和分割中将是有用的。多尺度训练等训练技术可以为各种视觉任务带来好处。

对于未来的工作，我们希望使用类似的技术进行弱监督图像分割。我们还计划使用更强大的匹配策略来改进我们的检测结果，以便在训练期间为分类数据分配弱标签。计算机视觉受到大量标记数据带来的好处。我们将继续寻找将不同来源和结构数据结合在一起的方法，以制作更强大的视觉模型。