YOLO9000:
更好, 更快, 更强

Joseph Redmon*†, Ali Farhadi*†
University of Washington*, Allen Institute for AI†
http://pjreddie.com/yolo9000/

摘 要

我们介绍 YOLO9000，这是一种先进的实时物体检测系统，可以检测超过 9000 个对象类别。首先，我们提出了对 YOLO 检测方法的各种改进，既有创新的，也有先前的工作。改进模型 YOLOv2 是标准检测任务（如 PASCAL VOC 和 COCO）的最新技术。使用一种新颖的多尺度训练方法，相同的 YOLOv2 模型可以以不同的尺寸运行，在速度和准确度之间提供简单的权衡。在 67 FPS 时，YOLOv2 在 VOC 2007 上获得 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 Fast R-CNN，同时仍然运行得更快。最后，我们提出了一种联合训练对象检测和分类的方法。使用此方法，我们在 COCO 检测数据集和 ImageNet 分类数据集上同时训练 YOLO9000。我们的联合训练允许 YOLO9000 预测没有标记数据的对象类的检测。我们验证了 ImageNet 上检测任务的方法。 YOLO9000 在 ImageNet 检测验证集上获得了 19.7 mAP，尽管只有 200 个类中的 44 个具有检测数据。而在不在 COCO 的 156 个类别中，YOLO9000 获得 16.0 mAP。但是 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。它仍然可以实时运行。

1. 介绍

通用对象检测应该快速，准确并且能够识别各种各样的对象。 自从引入神经网络以来，检测框架变得越来越快速和准确。 但是，大多数检测方法仍然受限于少量的对象。

与用于分类和标记等其他任务的数据集相比，当前对象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[3],[10],[2]，而分类数据集有数百万个具有数十或数十万个类别的图像[20],[2]。

我们希望检测任务能扩展到对象分类的级别。然而，用于检测的图像标记比用于分类或标注的标记要昂贵得多（标注通常是用户免费提供的）。 因此我们不太可能

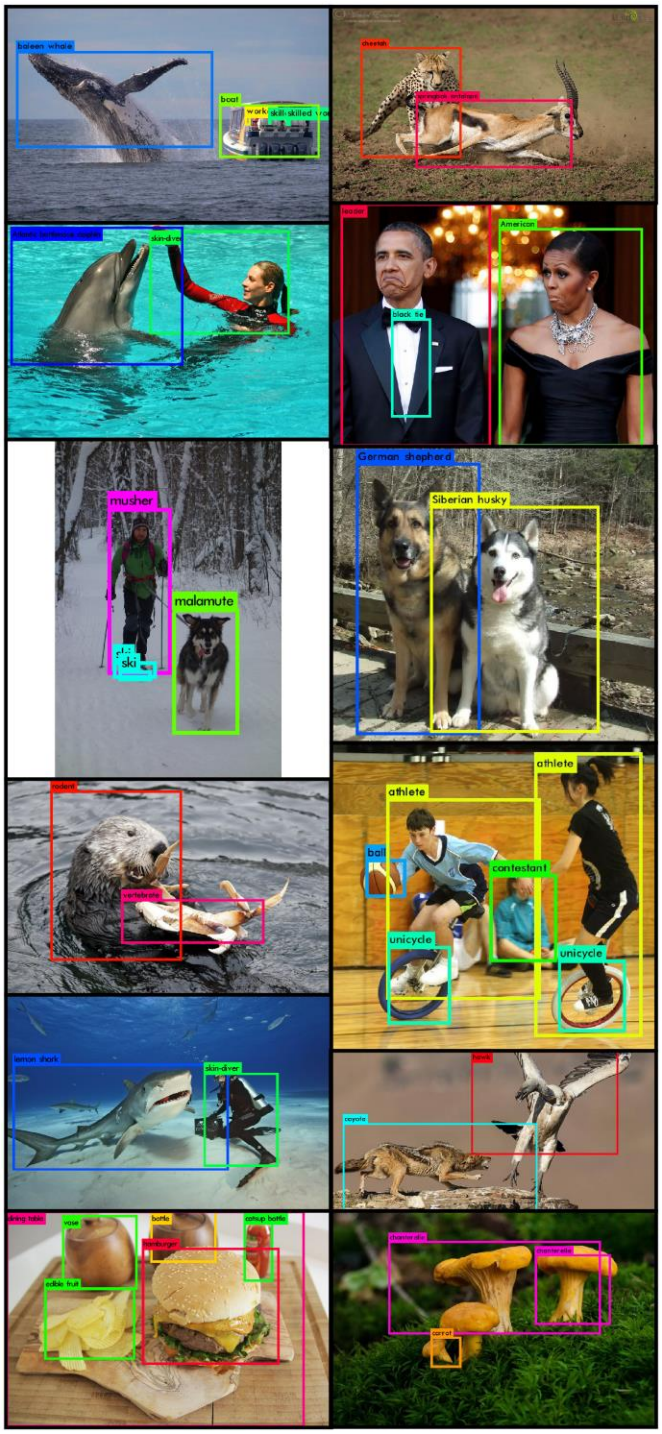

图 1：YOLO9000。 YOLO9000 可以实时检测各种对象类。

在不久的将来以与分类任务数据集相同的比例查看检测任务数据集。

我们提出了一种新方法来利用我们已有的大量分类数据，并用它来扩展当前检测系统的范围。我们的方法使用对象分类的分层视图，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物体检测器。我们的方法利用标记的检测图像来学习精确定位对象，同时使用分类图像来增加其阅读量和鲁棒性。

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物体探测器。首先，我们改进了基础 YOLO 检测系统，以生产 YOLOv2 —— 这是一种先进的实时检测器。然后我们使用我们的数据集组合方法和联合训练算法来训练来自 ImageNet 的 9000 多个类的模型以及来自 COCO 的检测数据。

我们所有的代码和预训练的模型都可以在网上获得：http://pjreddie.com/yolo9000/.

2. 更好

YOLO 相对于最先进的检测系统存在各种缺点。与 Fast R-CNN 相比，YOLO 的误差分析表明 YOLO 产生了大量的定位误差。此外，与基于区域提案的方法相比，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[6] [18] [17]。更好的性能通常取决于训练更大的网络或将多个模型集成在一起。然而，对于 YOLOv2，我们需要一种更加精确的探测器，它仍然很快。我们不是扩展我们的网络，而是简化网络，然后使图像的表达更容易学习。我们将过去工作中的各种想法与我们自己的新概念结合起来，以提高 YOLO 的性能。结果摘要见表 2。

批量归一化。批量归一化可以显著提高收敛性，同时不需要其他形式的正则化[7]。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规范模型。通过批量归一化，我们可以从模型中删除 DropOut 而不会引发过拟合。

高分辨率分类器。所有最先进的检测方法都使用 ImageNet 上预训练的分类器[16]。从 AlexNet 开始，大多数分类器都在小于 256 × 256 的输入图像上运行[8]。最初的 YOLO 在 224 × 224 训练分类器网络并将分辨率增加到 448 以进行检测。这意味着网络必须同时切换到学习对象检测和调整到新的输入分辨率。

对于 YOLOv2，我们首先在 ImageNet 的 10 个迭代（epoch）上以 448 × 448 分辨率对分类网络进行微调。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。这种高分辨率的分类网络使我们的 mAP 增加了近 4%。

带有锚框的卷积。 YOLO 直接使用卷积特征提取器顶部的完全连接层预测边界框的坐标，不同于 Faster R-CNN 直接使用手工挑选的先验预测边界框去预测坐标[15]。仅使用卷积层，Fast R-CNN 中的区域提议网络（RPN）预测锚箱的偏移和置信度。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。

我们从 YOLO 中删除完全连接层，并使用锚框来预测边界框。首先，我们消除了一个池化层，使网络卷积层的输出分辨率更高。我们还缩小网络以对 416 的输入图像进行操作而不是 448 × 448。我们这样做是因为我们在特征图中需要奇数个位置，因此只有一个中心单元。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四个位于附近的物体。 YOLO 的卷积层将图像缩小了 32 倍，因此通过使用 416 的输入图像，我们得到 13 × 13 的输出特征图。

当我们移动到锚箱时，我们还将类预测机制与空间位置分离，从而预测每个锚箱的类和对象。在 YOLO 之后，对象预测仍然预测与真实标签的 IOU，并且提议框和类预测在给定存在对象的情况下预测该类的条件概率。

使用锚箱使得我们的准确性会略有下降。 YOLO 仅预测每张图片 98 个候选框，但是使用锚框让我们的模型预测数量超过一千个。没有锚框，我们的中间模型得到 69.5 的 mAP，召回率为 81%。使用锚框我们的模型得到 69.2 的 mAP，召回率为 88%。即使 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。

维度聚类。与 YOLO 一起使用时，我们遇到了两个问题。首先是手工挑选的框尺寸。网络可以学会适当地调整框，但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良好的检测。

我们不是手动选择先验，而是在训练集边界框上运行 k-means 聚类从而自动地

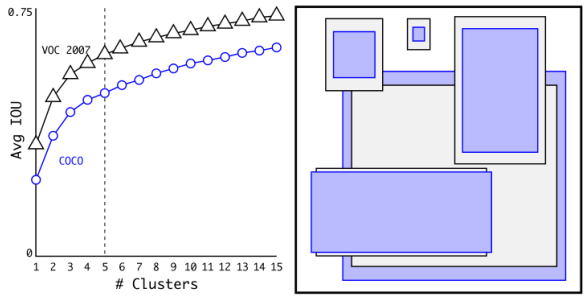

图 2：VOC 和 COCO 上的聚类框尺寸。 我们在边界框的维度上运行 k-means 聚类，以便为我们的模型获得良好的先验。 左图显示了我们获得的 k 的各种选择的平均 IOU。 我们发现 k = 5 给出了回忆与模型复杂性的良好折衷。 右图显示了 VOC 和 COCO 的相对质心。 两组原型都倾向于更薄，更高的框，而 COCO 的尺寸变化比 VOC 更大。

找到好的先验分布。 如果我们使用具有欧几里德距离的标准 k 均值，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，这与框的大小无关。 因此，对于我们的距离度量，我们使

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

我们为各种 k 值运行 k-means 并绘制具有最接近质心的平均 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。群集质心与手工挑选的锚框明显不同。我们有短而宽的框和高而瘦的框。

我们将平均 IOU 与我们的聚类策略中最接近的先验和表 1 中的手工挑选的锚框进行比较。在仅仅 5 个先验时，质心的表现类似于 9 锚框，平均 IOU 为 61.0 对比于 60.9。 如果我们使用 9 个质心，我们会看到更高的平均 IOU。 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 最接近先验框的平均 IOU。使用不同的生成方法将 VOC 2007 上的对象的平均 IOU 与其最接近、未修改的对象进行对比。 聚类比使用手工挑选的先验提供了更好的结果。

直接位置预测。 当使用带有 YOLO 的锚框时，我们遇到第二个问题：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框的（x，y）位置。 在区域提案网络中，网络预测值t_x和t_y，并且（x，y）中心坐标计算如下：

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

例如，t_x = 1 的预测会使框向右移动锚框的宽度，t_x = -1 的预测会将其向左移动相同的量。

这种公式是不受约束的，因此任何锚框都可以在图像中的任何位置结束，无论位置预测框是什么。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，而是遵循 YOLO 的方法，并预测相对于网格单元位置的位置坐标。这将基本事实限制在 0 和 1 之间。我们使用逻辑斯谛激活来约束网络预测以落在此范围内。

网络预测输出特征图中每个单元格的 5 个边界框。网络预测每个边界框的 5 个坐标，t_x，t_y，t_w，t_h和t_o。如果单元格从图像的左上角偏移（c_x, c_y）并且前面的边界框具有宽度和高度p_w，p_h，则预测对应于：

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

由于我们约束位置预测，因此参数化更容易学习，使网络更稳定。使用维度群集以及直接预测边界框中心位置可以比使用锚框的版本使得 YOLO 提高了约 5%。

细粒度特征。这个改进的 YOLO 预测了 13 × 13 特征图上的检测任务。 虽然这对于大型物体来说足够了，但它可能会受益于用于定位较小物体的更细粒度的特征。Faster R-CNN 和 SSD 都在网络中的各种特征图上运行其提议网络，以获得一系列分辨率。 我们采用一种不同的方法，只需添加一个直通层，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 中的标识映射。 这个

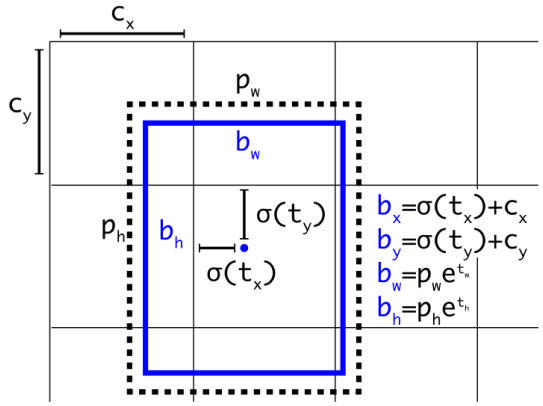

图 3：具有尺寸先验和位置预测的边界框。 我们预测框的宽度和高度作为聚类质心的偏移量。 我们使用 sigmoid 函数预测框相对于应用卷积核位置的中心坐标。

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。我们的探测器运行在此扩展特征图的顶部，以便它可以访问细粒度的功能。这使得性能提高 1%。

多尺度训练。原始 YOLO 使用输入分辨率 448 × 448。通过添加锚框，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层和池化层，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运行，因此我们将其训练到模型中。

我们不是调整输入图像大小，而是每隔几次迭代就改变一次网络。每 10 批次我们的网络随机选择一个新的图像尺寸大小。由于我们的模型缩减了 32 倍，我们从 32 的倍数中拉出来：32: {320,352,...,608}。因此，最小的选项是 320 × 320，最大的选项是 608 × 608。我们将网络调整到该维度并继续训练。

该制度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。网络以较小的尺寸运行得更快，因此 YOLOv2 可在速度和精度之间轻松权衡。

在低分辨率下，YOLOv2 作为一种简易，而又精确的探测器运行。在 288 × 288，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 仍是最先进的探测器，在 VOC 2007 上具有 78.6 mAP，同时仍然高于实时速度。有关 YOLOv2 的比较，请参见表 3

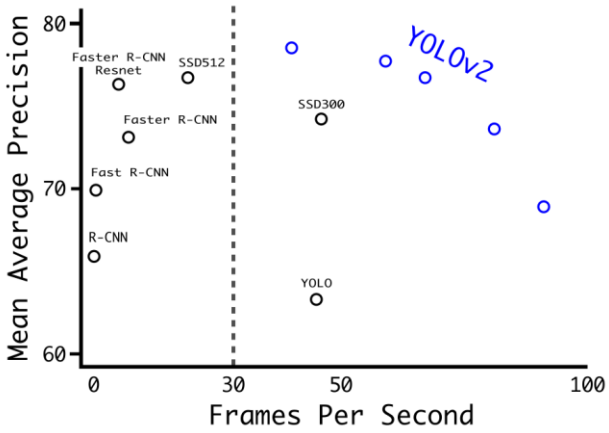

图 4：VOC 2007 的准确性和速度。

与 VOC 2007 的其他框架。见图 4。

进一步的实验。 我们训练 YOLOv2 用于检测 VOC 2012。表 4 显示了 YOLOv2 与其他最先进检测系统的性能比较。YOLOv2 在取得 73.4 mAP 下运行速度仍比竞争对手方法快。我们还在 COCO 上进行了训练，并与表 5 中的其他方法进行了比较。关于 VOC 指标（IOU = .5）YOLOv2 获得 44.0 mAP，与 SSD 和 Faster R-CNN 相当。

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 的检测框架。YOLOv2 比以前的检测方法更快，更准确。 它还可以在不同的分辨率下运行，以便在速度和准确度之间轻松权衡。 每个 YOLOv2 条目实际上是具有相同权重的相同训练模型，仅以不同的大小进行评估。 所有时间信息都在 Geforce GTX Titan X 上（原始而非 Pascal 模型）。

3. 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

	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**

表 2：从 YOLO 到 YOLOv2 的路径。 大多数列出的设计决策导致 mAP 显著增加。 两个例外是使用锚框切换到完全卷积网络和使用新网络。切换到锚框方法可以在不改变 mAP 的情况下提高召回率，同时使用新的网络减少计算 33%。

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 VOC2012 测试检测结果。 YOLOv2 与最先进的探测器（如使用 ResNet 和 SSD512 的 Faster R-CNN）相当，速度提高了 2–10 倍。

最大化性能我们设计 YOLOv2 从头就开始快速。

大多数检测框架依赖于 VGG-16 作为基本特征提取器[17]。 VGG-16 是一个功能强大，准确的分类网络，但它存在不必要的复杂。 VGG-16 的卷积层在单个图像上以 224 × 224 分辨率进行单次通过，需要 306.9 亿浮点运算。

YOLO 框架使用基于 Googlenet 架构的自定义网络[19]。该网络比 VGG-16 更快，正向通行仅使用 85.2 亿次运算。但是，它的准确性略差于 VGG-16。对于 224 × 224 的单物体，top-5 精度，YOLO 的定制训练模型获得 88.0% ImageNet，而在 VGG-16 中为 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知识之上。与 VGG 模型类似，我们大多使用 3 × 3 的卷积器，并且在进行池化步骤后将通道数加倍[17]。继 Network in Network（NIN）提出之后，我们使用全局平均池化来进行预测以及 1× 1 的卷积器来压缩 3 × 3 的卷积之间的特征表达[9]。我们使用批量归一化来稳定训练，加速收敛，并使模型正则化[7]。

我们的最终模型叫做 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 亿次操作来处理图像，但在 ImageNet 上实现了 72.9%的 top-1 精度和 91.2%的 top-5 精度。

分类训练。我们使用随机梯度下降训练网络在标准 ImageNet 1000 类分类数据集上 160 个迭代（epoch），起始学习率为 0.1，多项式速率衰减率为 4，权重衰减为 0.0005，动量为 0.9，使用 Darknet 神经网络框架[13]。在训练期间，我们使用标准数据增强技巧，包括随机裁剪，旋转和色调，饱和度和曝光变化。

如上所述，在我们对 224 × 224 的图像进行初步训练之后，我们以更大的尺寸（448）对我们的网络进行微调。对于这种微调，我们使用上述参数进行训练，但只有 10 个迭代，并且以10^{-3}的学习率开始。在这种更高的分辨率下，我们的网络可实现 76.5%的 top-1 精度和 93.3%的 top-5 精度。

检测训练。我们通过删除最后一个卷积层来修改这个网络以进行检测，并且添加三个 3 × 3 卷积层，每个卷积层有 1024 个卷积核，然后是最后的 1 × 1 卷积层，其中包含我们需要检测的输出数量。对于 VOC，我们预测 5 个框，每个框有 5 个坐标和 20 个类别，所以 125 个卷积器。我们还添加了一个直通层，从最后的 3 × 3 × 512 层到第二个至最后一个卷积层，这样我们的模型就可以使用细粒度特征。

我们训练网络 160 个迭代，起始学习率为10^{-3}，在 60 和 90 个迭代周期时除以 10。

		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 test-dev2015 的结果。 表改编自 [11]

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

我们使用 0.0005 的重量衰减和 0.9 的动量。 我们使用类似的数据增强到 YOLO 和 SSD 随机裁剪，色移等。我们对 COCO 和 VOC 使用相同的训练策略。

4. 更强

我们提出了一种联合训练分类和检测数据的机制。 我们的方法使用有标记图像进行检测，以学习特定于检测的信息，如边界框坐标和对象预测，以及如何对常见对象进行分类。它使用仅带有类标签的图像来扩展它可以检测的类别数。

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。 当它看到分类图像时，我们只反向传播模型结构的特定于分类部分的损失。

这种方法提出了一些挑战。检测数据集只有常见的对象和通用标签，如“狗”或“船”。分类数据集具有更广泛和更深的标签范围。 ImageNet 不仅仅含有“狗”的类别，它还包括“诺福克犬”，“约克夏犬”和“贝德灵顿犬”。如果我们想在两个数据集上进行训练，我们需要一种连贯的方式来合并这些标签。

大多数分类方法在所有可能的类别中使用 softmax 层来计算最终的概率分布。使用 softmax 假定类是互斥的。这给组合数据集带来了问题，例如，您不希望使用此模型组合 ImageNet 和 COCO，因为类“诺福克犬”和“狗”不是互斥的。

我们可以使用多标签模型来组合不假设互斥的数据集。这个方法忽略了我们对数据所知的所有结构，例如所有 COCO 类都是相互排斥的。

分层分类。 ImageNet 标签来自 WordNet，这是一种语言数据库，用于构建概念及其相关性[12]。在 WordNet 中，“诺福克犬”和“约克夏犬”都是“小猎犬”的上位词，它是一种“猎犬”，是一种“狗”，它是一种“犬”，等等。分类假设标签采用扁平结构，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，因为语言很复杂。例如，“狗”既是一种“犬”，也是一种“家畜”，它们都是 WordNet 中的同义词。我们不是使用完整的图形结构，而是通过从 ImageNet 中的概念构建层次结构树来简化问题。

为了构建这个树，我们检查了 ImageNet 中的视觉名词，并查看它们通过 WordNet 图形到根节点的路径，在本例中是“物理对象”。许多同义词只有一条通过图形的路径，所以首先我们将所有这些路径添加到树中。然后我们迭代地检查我们剩下的概念，并尽可能少地添加树的生长路径。因此，如果一个概念有两条通向根的路径，一条路径将三条边添加到树中，另一条路径只添加一条边，我们选择较短的路径。

最终的结果是 WordTree 的产生，一个视觉概念的层次模型。 为了使用 WordTree 进行分类，我们预测每个节点的条件概率，以获得该同义词的每个同义词的概率。 例如，在“terrier”节点，我们预测：

$$Pr(\text{Norfolk terrier}|\text{terrier})$$
$$Pr(\text{Yorkshire terrier}|\text{terrier})$$
$$Pr(\text{Bedlington terrier}|\text{terrier})$$
$$\dots$$

如果我们想要计算特定节点的绝对概率，我们只需遵循通过树到根节点的路径并乘以条件概率。 因此，如果我们想知道图片是否是诺福克犬，我们计算：

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

出于分类目的，我们假设图像包含一个对象：Pr（物理对象）= 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，我们添加了所有中间节点，将标签空间从 1000 扩展到 1369。在训练期间，我们在树上传播真实标签，这样如果一个图像标记为“诺福克犬”，它也会被标记为“狗”和“哺乳动物”。为了计算条件概率，我们的模型预测了 1369 个值的向量，我们计算了相同概念的所有系统集的 softmax，见图 5。

使用与以前相同的训练参数，我们的高分层 Darknet-19 实现了 71.9%的 top-1 精度和 90.4%的 top-5 精度。尽管增加了 369 个额外的概念，并且我们的网络预测了树形结构，但我们的精确度仅略有下降。以这种方式执行分类也具有一些益处。性能在新的或未知的对象类别上优雅地下降。例如，如果网络看到一张狗的图片但不确定它是什么类型的狗，它仍然会以高信心预测“狗”，但在下位词之间散布的信心较低。

该方法也适用于检测。现在，我们使用 YOLOv2 的对象预测器来代替 Pr（物理对象），而不是假设每个图像都有一个对象。探测器预测一个边界框

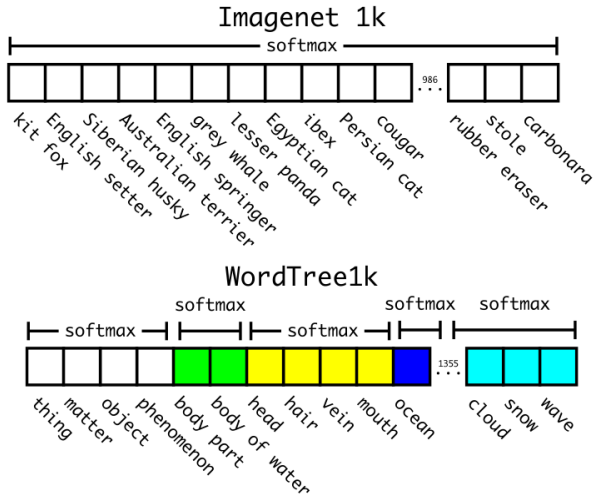

图 5：ImageNet 与 WordTree 的预测。 大多数 ImageNet 模型使用一个大的 softmax 来预测概率分布。 使用 WordTree，我们在共同下位线上执行多个 softmax 操作。

和概率树。我们遍历树，在每次分裂时获得最高置信度路径，直到达到某个阈值并且我们预测该对象类。

数据集与 WordTree 的组合。我们可以使用 WordTree 以可能的方式将多个数据集组合在一起。我们只是将数据集中的类别映射到树中的同义词集。图 6 显示了使用 WordTree 组合 ImageNet 和 COCO 标签的示例。WordNet 非常多样化，因此我们可以将此技术用于大多数数据集。

联合分类和检测。现在我们可以使用 WordTree 组合数据集，我们可以训练我们的分类和检测联合模型。我们希望训练一个超大规模的探测器，因此我们使用 COCO 检测数据集和完整的 ImageNet 版本中的前 9000 个类创建我们的组合数据集。我们还需要评估我们的方法，以便我们在 ImageNet 检测挑战中添加任何尚未包含的类。此数据集的相应 WordTree 有 9418 个类。 ImageNet 是一个更大的数据集，因此我们通过对 COCO 进行过采样来平衡数据集，好让 ImageNet 仅以 4：1 的比例放大。

使用此数据集，我们训练 YOLO9000。我们使用基本的 YOLOv2 架构，但只有 3 个先验而不是 5 个来限制输出大小。当我们的网络看到检测图像时，我们会反复传播损失。对于分类损失，我们仅反向传播等于或高于标签相应级别的损失。例如，如果标签是“狗”，我们会在树中进一步向下预测任何错误，“德国牧羊犬”与“金毛猎犬”，因为我们没有这些信息。

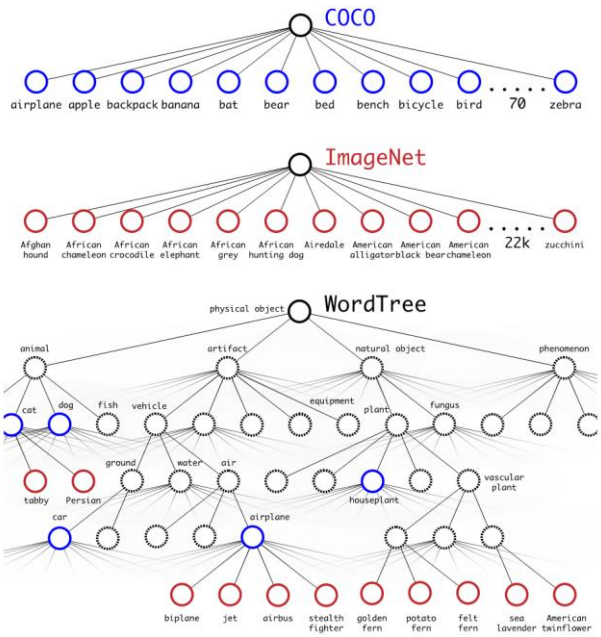

图 6：使用 WordTree 层次结构组合数据集。 使用 WordNet 概念图，我们构建了一个视觉概念的分层树。 然后，我们可以通过将数据集中的类映射到树中的同义词集来将数据集合并在一起。 这是 WordTree 的简化视图，用于说明目的。

当它看到分类图像时，我们只会反向传播分类损失。为此，我们只需找到预测该类最高概率的边界框，并计算其预测树的损失。我们还假设预测框与真实标签重叠至少 .3 IOU，我们基于此假设反向传播对象损失。

使用这种联合训练，YOLO9000 学习使用 COCO 中的检测数据查找图像中的对象，并学习使用来自 ImageNet 的数据对各种各样的对象进行分类。

我们在 ImageNet 检测任务上评估 YOLO9000。ImageNet 的检测任务与 COCO 在 44 个对象类别上共享，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 总共获得了 19.7 mAP，在不相交的 156 个对象类上有 16.0 mAP，它从未见过任何标记的检测数据。该 mAP 高于 DPM 获得的结果，但 YOLO9000 在不同的数据集上进行训练，用部分监督的方式[4]。它还可以同时检测 9000 个其他对象类别，所有这些都是实时的。

当我们分析 YOLO9000 在 ImageNet 上的表现时，我们看到它很好地学习了新的动物种类，不过它们与服装和装备等学习类别相结合。

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。具有来自 156 个弱监督类的最高和最低 AP 的类。 YOLO9000 为各种动物学习了很好的模型，但却在服装或装备等新类中挣扎。

新动物更容易学习，因为对象性预测很好地概括了 COCO 中的动物。 相反，COCO 没有针对任何服装类别的边框标签，仅限于人，因此 YOLO9000 难以模拟“太阳镜”或“泳裤”等类别。

5. 结论

我们介绍 YOLOv2 和 YOLO9000，实时检测系统。YOLOv2 是最先进的，比各种检测数据集中的其他检测系统更快。此外，它可以在各种图像尺寸下运行，以在速度和准确度之间提供平滑的权衡。

YOLO9000 是一个实时框架，通过联合优化检测和分类来检测 9000 多个对象类别。我们使用 WordTree 组合来自各种来源的数据和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对象检测之外进行推广。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段域中将是有用的。多尺度训练等训练技术可以为各种视觉任务提供优势。

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的辅助。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世界模型。

参考文献

[1] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. arXiv preprint arXiv:1512.04143, 2015. 6

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. IEEE, 2009. 1

[3] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) chal-lenge. International journal of computer vision, 88(2):303– 338, 2010. 1

[4] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/. 8

[5] R. B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015. 4, 5, 6

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learn-ing for image recognition. arXiv preprint arXiv:1512.03385, 2015. 2, 4, 5

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015. 2, 5

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012. 2

[9] M. Lin, Q. Chen, and S. Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013. 5

[10] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ra-manan, P. Dollar,´ and C. L. Zitnick. Microsoft coco: Com-mon objects in context. In European Conference on Com-puter Vision, pages 740–755. Springer, 2014. 1, 6

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed. SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015. 4, 5, 6

[12] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to wordnet: An on-line lexical database. International journal of lexicography, 3(4):235–244, 1990. 6

[13] J. Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016. 5

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640, 2015. 4, 5

[15] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: To-wards real-time object detection with region proposal net-works. arXiv preprint arXiv:1506.01497, 2015. 2, 3, 4, 5, 6

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 2015. 2

[17] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014. 2, 5

[18] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR, abs/1602.07261, 2016. 2

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014. 5

[20] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. Communications of the ACM, 59(2):64–73, 2016. 1