YOLO9000: Better, Faster, Stronger

YOLO9000：更好、更快、更强

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

Abstract

We introduce YOLO9000, a state-of-the-art, real-time object detection system that can detect over 9000 object categories. First we propose various improvements to the YOLO detection method, both novel and drawn from prior work. The improved model, YOLOv2, is state-of-the-art on standard detection tasks like PASCAL VOC and COCO. Using a novel, multi-scale training method the same YOLOv2 model can run at varying sizes, offering an easy tradeoff between speed and accuracy. At 67 FPS, YOLOv2 gets 76.8 mAP on VOC 2007. At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster R-CNN with ResNet and SSD while still running significantly faster. Finally we propose a method to jointly train on object detection and classification. Using this method we train YOLO9000 simultaneously on the COCO detection dataset and the ImageNet classification dataset. Our joint training allows YOLO9000 to predict detections for object classes that don’t have labelled detection data. We validate our approach on the ImageNet detection task. YOLO9000 gets 19.7 mAP on the ImageNet detection validation set despite only having detection data for 44 of the 200 classes. On the 156 classes not in COCO, YOLO9000 gets 16.0 mAP. But YOLO can detect more than just 200 classes; it predicts detections for more than 9000 different object categories. And it still runs in real-time.

摘要

我们引入了一个先进的实时目标检测系统 YOLO9000，可以检测超过 9000 个目标类别。首先，我们提出了对 YOLO 检测方法的各种改进，既有新发明的一些东西，也参考了前人的工作。改进后的模型 YOLOv2 在 PASCAL VOC 和 COCO 等标准检测任务上性能是最好的。使用一种新颖的、多尺度训练方法，同样的 YOLOv2 模型可以以不同的尺度运行，从而在速度和准确性之间获得了良好的权衡。以 67FPS 的检测速度，YOLOv2 在 VOC 2007 上获得了 76.8 mAP。而检测速度 40FPS 时，YOLOv2 获得了 78.6 mAP，比使用 ResNet 的 Faster R-CNN 和 SSD 等先进方法表现更出色，同时仍然运行速度显著更快。最后我们提出了一种联合训练目标检测与分类的方法。使用这种方法，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 预测未标注的检测数据目标类别的检测结果。我们在 ImageNet 检测任务上验证了我们的方法。YOLO9000 在 ImageNet 检测验证集上获得 19.7 mAP，尽管 200 个类别中只有 44 个具有检测数据。不在 COCO 中的 156 个类别上，YOLO9000 获得 16.0 mAP。但 YOLO 可以检测到 200 多个类别；它预测超过 9000 个不同目标类别的检测结果。并且它仍然能实时运行。

1. Introduction

General purpose object detection should be fast, accurate, and able to recognize a wide variety of objects. Since the introduction of neural networks, detection frameworks have become increasingly fast and accurate. However, most detection methods are still constrained to a small set of objects.

1. 引言

通用目的的目标检测系统应该是快速的、准确的，并且能够识别各种各样的目标。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一小部分目标。

Current object detection datasets are limited compared to datasets for other tasks like classification and tagging. The most common detection datasets contain thousands to hundreds of thousands of images with dozens to hundreds of tags [3] [10] [2]. Classification datasets have millions of images with tens or hundreds of thousands of categories [20] [2].

与分类和标记等其他任务的数据集相比，目前目标检测数据集是有限的。最常见的检测数据集包含成千上万到数十万张具有成百上千个标签的图像[3][10][2]。分类数据集有数以百万计的图像，数十或数十万个类别[20][2]。

We would like detection to scale to level of object classification. However, labelling images for detection is far more expensive than labelling for classification or tagging (tags are often user-supplied for free).

Thus we are unlikely to see detection datasets on the same scale as classification datasets in the near future.

我们希望检测能够扩展到目标分类的级别。但是，标注检测图像要代价比标注分类或贴标签要大得多（标签通常是用户免费提供的）。因此，我们不太可能在近期内看到与分类数据集相同规模的检测数据集。

We propose a new method to harness the large amount of classification data we already have and use it to expand the scope of current detection systems. Our method uses a hierarchical view of object classification that allows us to combine distinct datasets together.

我们提出了一种新的方法来利用我们已经拥有的大量分类数据，并用它来扩大当前检测系统的范围。我们的方法使用目标分类的分层视图，允许我们将不同的数据集组合在一起。

We also propose a joint training algorithm that allows us to train object detectors on both detection and classification data. Our method leverages labeled detection images to learn to precisely localize objects while it uses classification images to increase its vocabulary and robustness.

我们还提出了一种联合训练算法，使我们能够在检测和分类数据上训练目标检测器。我们的方法利用有标签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词表和鲁棒性。

Using this method we train YOLO9000, a real-time object detector that can detect over 9000 different object categories. First we improve upon the base YOLO detection system to produce YOLOv2, a state-of-the-art, real-time detector. Then we use our dataset combination method and joint training algorithm to train a model on more than 9000 classes from ImageNet as well as detection data from COCO.

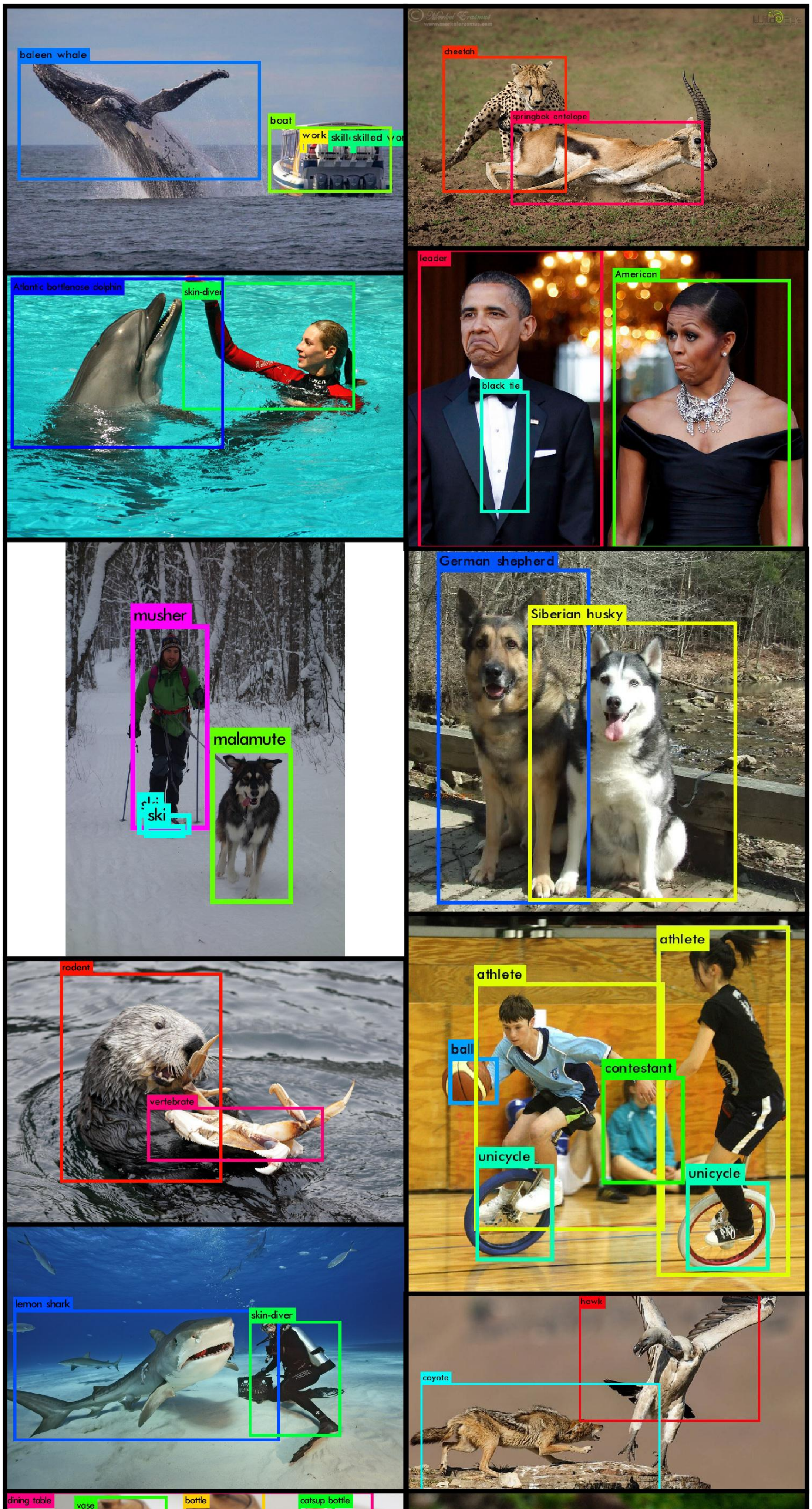

baleen whale
boat
work
skilled wo
cheetah
springbok antelope
Atlantic bottlenose dolphin
skin-diver
leader
American
black tie
musher
malamute
ski
ski
German shepherd
Siberian husky
athlete
athlete
ball
contestant
unicycle
unicycle
rodent
vertebrate
lemon shark
skin-diver
hawk
coyote
dining table
vase
bottle
catsup bottle

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

使用这种方法我们训练 YOLO9000，一个实时的目标检测器，可以检测超过 9000 种不同的目标类别。首先，我们改进 YOLO 基础检测系统，产生最先进的实时检测器 YOLOv2。然后利用我们的数据集组合方法和联合训练算法对来自 ImageNet 的 9000 多个类别以及 COCO 的检测数据训练了一个模型。

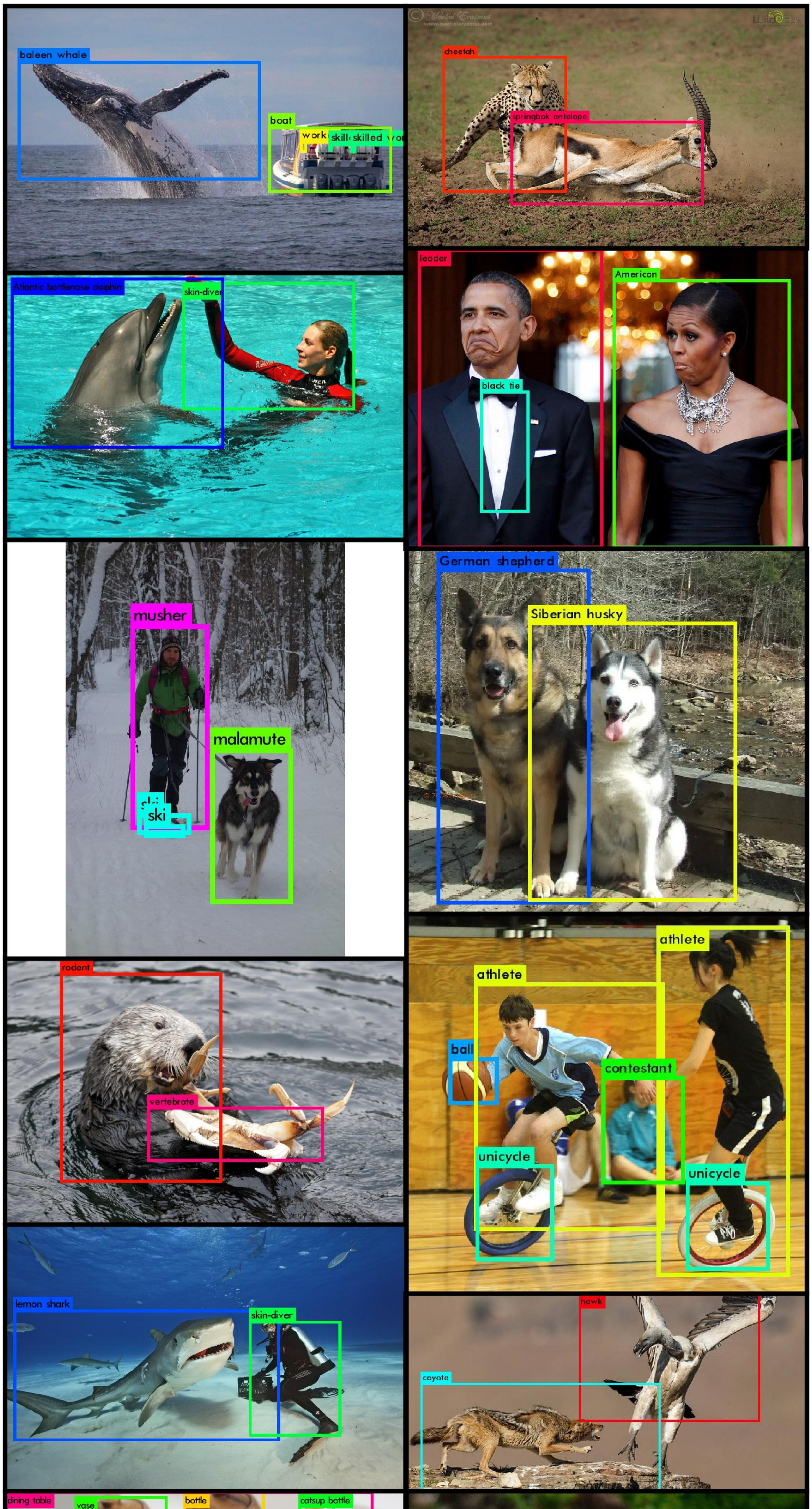

baleen whale
boat
work
skilled
cheetah
springbok antelope
Atlantic bottlenose dolphin
skin-diver
leader
American
black tie
musher
malamute
ski
German shepherd
Siberian husky
rodent
vertebrate
athlete
athlete
ball
contestant
unicycle
unicycle
lemon shark
skin-diver
hawk
coyote
dining table
vase
bottle
catsup bottle

图 1：YOLO9000。YOLO9000 可以实时检测许多目标类别。

All of our code and pre-trained models are available online at http://pjreddie.com/yolo9000/.

我们的所有代码和预训练模型都可在线获得：http://pjreddie.com/yolo9000/。

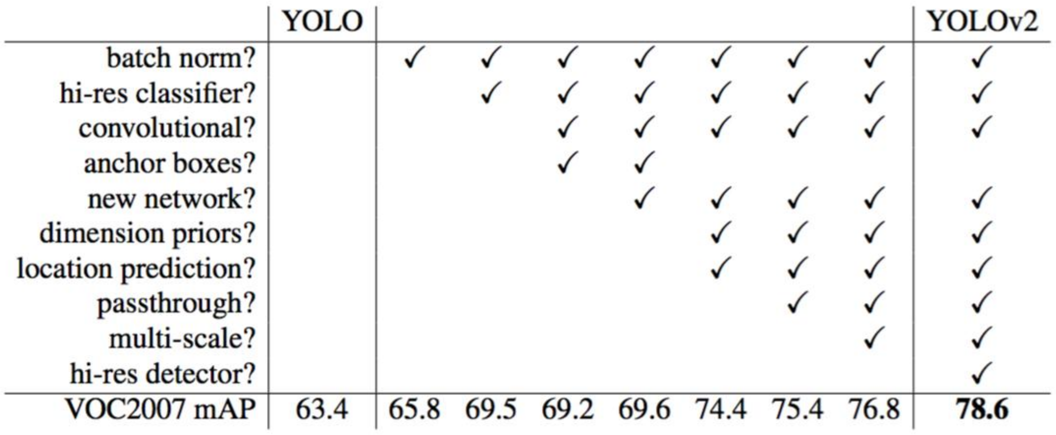
2. Better

YOLO suffers from a variety of shortcomings relative to state-of-the-art detection systems. Error analysis of YOLO compared to Fast R-CNN shows that YOLO makes a significant number of localization errors. Furthermore, YOLO has relatively low recall compared to region proposal-based methods. Thus we focus mainly on improving recall and localization while maintaining classification accuracy.

2. 更好

与最先进的检测系统相比，YOLO 有许多缺点。YOLO 与 Fast R-CNN 的误差分析比较表明，YOLO 存在大量的定位误差。此外，与基于 region proposal 的方法相比，YOLO 召回率相对较低。因此，我们主要侧重于提高召回率和改进目标精确定位，同时保持分类准确性。

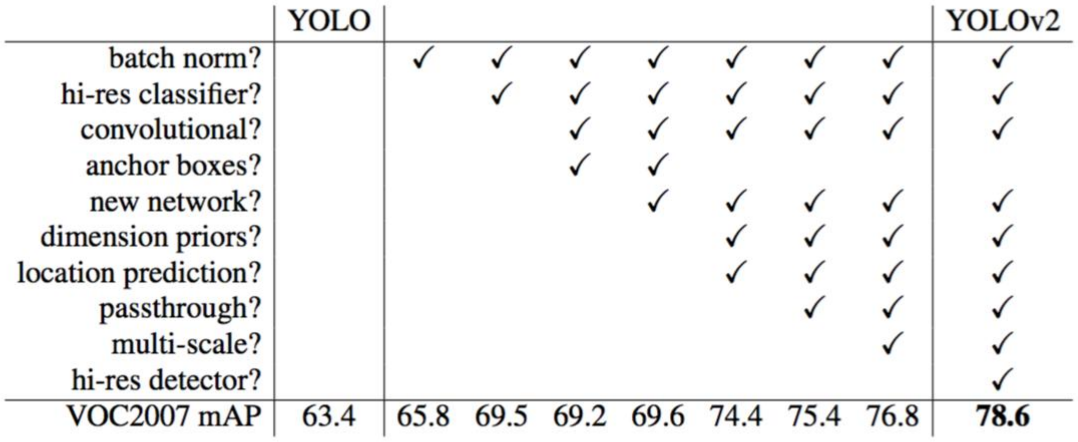
Computer vision generally trends towards larger, deeper networks [6] [18] [17]. Better performance often hinges on training larger networks or ensembling multiple models together. However, with YOLOv2 we want a more accurate detector that is still fast. Instead of scaling up our network, we simplify the network and then make the representation easier to learn. We pool a variety of ideas from past work with our own novel concepts to improve YOLO’s performance. A summary of results can be found in Table 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**

Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

计算机视觉一般趋向于更大、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组合在一起。但是，在 YOLOv2 中，我们需要一个更精确的检测器，而且需要它仍然很快。我们不是扩大我们的网络，而是简化网络，然后让表示更容易学习。我们将过去的工作与我们自己的新概念汇集起来，以提高 YOLO 的性能。表 2 列出了结果总结。



	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 的过程。列出的大部分设计决定都会获得 mAP 的显著增加。有两个例外是切换到具有 anchor 框的一个全卷积网络和使用新网络。切换到 anchor 框形式的方法增加了召回率，然而没有改变 mAP，但可以使新网络会削减 33%的计算量。

Batch Normalization. Batch normalization leads to significant improvements in convergence while eliminating the need for other forms of regularization [7]. By adding batch normalization on all of the convolutional layers in YOLO we get more than 2% improvement in mAP. Batch normalization also helps regularize the model. With batch normalization we can remove dropout from the model without overfitting.

批归一化。批归一化会获得收敛性的显著改善，同时消除了对其他形式正则化的需求[7]。通过在 YOLO 的所有卷积层上添加批归一化，我们在 mAP 中获得了超过 2%的改进。批归一化也有助于模型正则化。通过批归一化，我们可以从模型中删除 dropout 而不会过拟合。

High Resolution Classifier. All state-of-the-art detection methods use classifier pre-trained on ImageNet [16]. Starting with AlexNet most classifiers operate on input images smaller than 256 × 256 [8]. The original YOLO trains the classifier network at 224 × 224 and increases the resolution to 448 for detection. This means the network has to simultaneously switch to learning object detection and adjust to the new input resolution.

高分辨率分类器。所有最先进的检测方法都使用在 ImageNet[16]上预训练的分类器。从 AlexNet 开始，大多数分类器对小于 256×256[8]的输入图像进行操作。YOLO 初始版本以 224×224 分辨率的

图像训练分类器网络，并在检测时将分辨率提高到 448。这意味着网络必须同时切换到学习目标检测和调整到新的输入分辨率。

For YOLOv2 we first fine tune the classification network at the full 448 × 448 resolution for 10 epochs on ImageNet. This gives the network time to adjust its filters to work better on higher resolution input. We then fine tune the resulting network on detection. This high resolution classification network gives us an increase of almost 4%4% mAP.

对于 YOLOv2，我们首先在 ImageNet 上以 448×448 的分辨率对分类网络进行 10 个迭代周期的 fine tune。这使得网络来调整其卷积核以便更好地处理更高分辨率的输入。然后我们对得到的网络进行 fine tune 并用于检测任务。这个高分辨率分类网络使我们增加了近 4% 的 mAP。

Convolutional With Anchor Boxes. YOLO predicts the coordinates of bounding boxes directly using fully connected layers on top of the convolutional feature extractor. Instead of predicting coordinates directly Faster R-CNN predicts bounding boxes using hand-picked priors [15]. Using only convolutional layers the region proposal network (RPN) in Faster R-CNN predicts offsets and confidences for anchor boxes. Since the prediction layer is convolutional, the RPN predicts these offsets at every location in a feature map. Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn.

具有 Anchor 框的卷积。YOLO 直接使用卷积特征提取器顶部的全连接层来预测边界框的坐标。Faster R-CNN 使用手动选择的先验来预测边界框而不是直接预测坐标[15]。Faster R-CNN 中的 region proposal 网络（RPN）仅使用卷积层来预测 Anchor 框的偏移和置信度。由于预测层是卷积类型的层，所以 RPN 在特征图的每个位置上预测这些偏移。预测偏移而不是坐标简化了问题，并且使网络更容易学习。

We remove the fully connected layers from YOLO and use anchor boxes to predict bounding boxes. First we eliminate one pooling layer to make the output of the network’s convolutional layers higher resolution. We also shrink the network to operate on 416 input images instead of 448 ×448. We do this because we want an odd number of locations in our feature map so there is a single center cell. Objects, especially large objects, tend to occupy the center of the image so it’s good to have a single location right at the center to predict these objects instead of four locations that are all nearby. YOLO’s convolutional layers downsample the image by a factor of 32 so by using an input image of 416 we get an output feature map of 13 × 13.

我们从 YOLO 中移除全连接层，并使用 Anchor 框来预测边界框。首先，我们去除了一个池化层，使网络卷积层输出具有更高的分辨率。我们还缩小了网络，操作 416×416 的输入图像而不是 448×448。我们这样做是因为我们要在我们的特征图中位置个数是奇数，所以只会

有一个中心格子。目标，特别是大目标，往往占据图像的中心，所以在中心有一个单独的位置来预测这些目标的很好的，而不是四个都相邻的位置。YOLO 的卷积层将图像下采样 32 倍，所以通过使用 416 的输入图像，我们得到了 13×13 的输出特征图。

When we move to anchor boxes we also decouple the class prediction mechanism from the spatial location and instead predict class and objectness for every anchor box. Following YOLO, the objectness prediction still predicts the IOU of the ground truth and the proposed box and the class predictions predict the conditional probability of that class given that there is an object.

当我们移动到 Anchor 框时，我们也将类预测机制与空间位置分离，预测每个 Anchor 框的类别和目标。与 YOLO 类似，是否为目标的预测仍然预测了真值和 proposal 的边界框的 IOU，并且类别预测预测了当存在目标时该类别的条件概率。

Using anchor boxes we get a small decrease in accuracy. YOLO only predicts 98 boxes per image but with anchor boxes our model predicts more than a thousand. Without anchor boxes our intermediate model gets 69.5 mAP with a recall of 81%. With anchor boxes our model gets 69.2 mAP with a recall of 88%. Even though the mAP decreases, the increase in recall means that our model has more room to improve.

使用 Anchor 框，我们的精度发生了一些小的下降。YOLO 对每张图像只预测 98 个边界框，但是使用 Anchor 框我们的模型预测超过

一千个。如果不使用 Anchor 框，我们的中间模型将获得 69.5 的 mAP，召回率为 81%。使用 Anchor 框的模型得到了 69.2 mAP，召回率为 88%。尽管 mAP 下降了一点，但召回率的上升意味着我们的模型有更大的改进空间。

Dimension Clusters. We encounter two issues with anchor boxes when using them with YOLO. The first is that the box dimensions are hand picked. The network can learn to adjust the boxes appropriately but if we pick better priors for the network to start with we can make it easier for the network to learn to predict good detections.

维度聚类。当 Anchor 框与 YOLO 一起使用时，我们遇到了两个问题。首先是边界框尺寸是手工挑选的。网络可以学习到如何适当调整边界框，但如果我们为网络选择更好的先验，我们可以使网络更容易学习它以便获得更好的检测结果。

Instead of choosing priors by hand, we run k-means clustering on the training set bounding boxes to automatically find good priors. If we use standard k-means with Euclidean distance larger boxes generate more error than smaller boxes. However, what we really want are priors that lead to good IOU scores, which is independent of the size of the box. Thus for our distance metric we use:

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

We run k-means for various values of k and plot the average IOU with closest centroid, see Figure 2. We choose k=5 as a good tradeoff between model complexity and high recall. The cluster centroids are significantly different than hand-picked anchor boxes. There are fewer short, wide boxes and more tall, thin boxes.

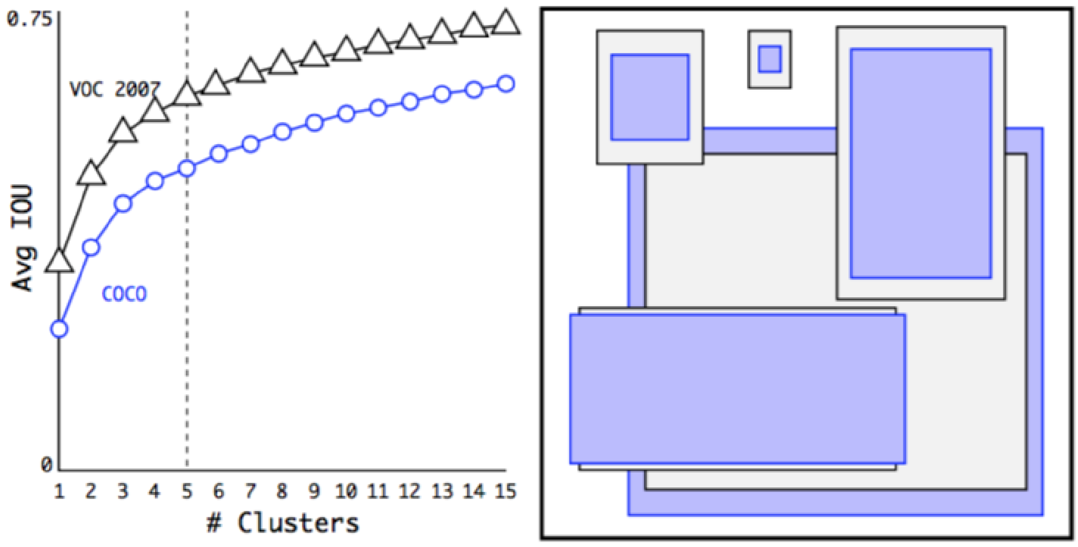


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that k=5 gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. Both sets of priors favor thinner, taller boxes while COCO has greater variation in size than VOC.

我们不用手工选择先验，而是在训练集边界框上运行 k-means 聚类，自动找到好的先验。如果我们使用欧式距离的标准 k-means，那么较大的边界框比较小的边界框产生更多的误差。然而，我们真正想要的是产生好的 IOU 分数的先验，这是独立于边界框大小的。因此，对于我们的距离度量，我们使用：

d(box,centroid)=1−IOU(box,centroid)

如图 2 所示，我们运行不同 k 值的 k-means，并画出平均 IOU 与最接近的几何中心的关系图。我们选择 k=5 时模型复杂性和高召回率

之间的具有良好的权衡。聚类中心与手工挑选的 Anchor 框明显不同。聚类结果有更短更宽的边界框，也有更高更细的边界框。

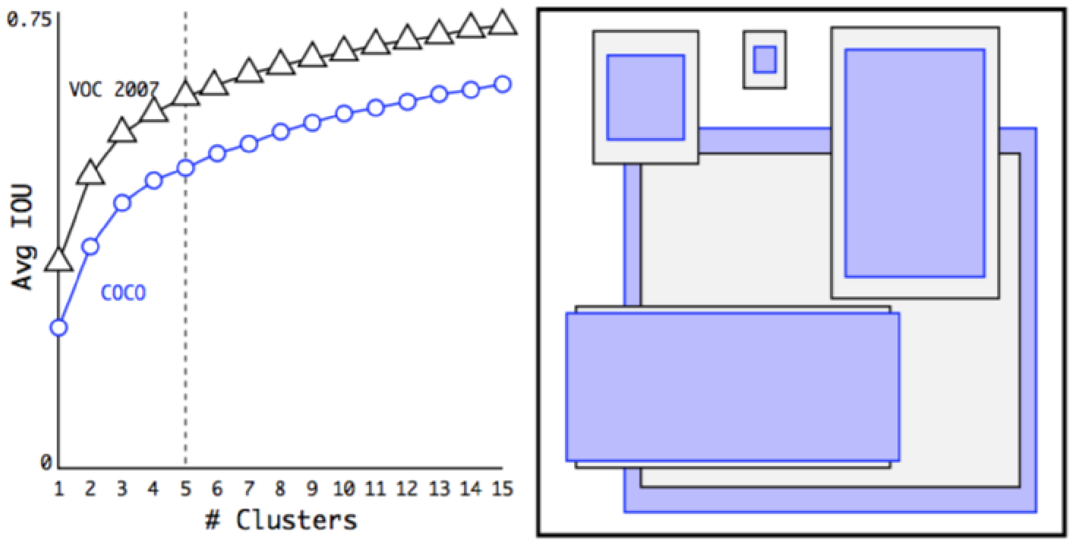


图 2：VOC 和 COCO 的聚类边界框尺寸。我们对边界框的维度进行 k-means 聚类，以获得我们模型的良好先验。左图显示了我们通过对 k 的各种选择得到的平均 IOU。我们发现 k=5 给出了一个很好的召回率与模型复杂度的权衡。右图显示了 VOC 和 COCO 的相对中心。这两种先验集合都具有更瘦更高的边界框，而 COCO 比 VOC 在尺寸上有更大的变化。

We compare the average IOU to closest prior of our clustering strategy and the hand-picked anchor boxes in Table 1. At only 5 priors the centroids perform similarly to 9 anchor boxes with an average IOU of 61.0 compared to 60.9. If we use 9 centroids we see a much higher average IOU. This indicates that using k-means to generate our bounding box starts the model off with a better representation and makes the task easier to learn.

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

Table 1: Average IOU of boxes to closest priors on VOC 2007. The average IOU of objects on VOC 2007 to their closest, unmodified prior using different generation methods. Clustering gives much better results than using hand-picked priors.

在表 1 中我们将平均 IOU 与我们聚类策略中最接近的先验以及手工选取的 Anchor 框进行了比较。仅有 5 个先验中心的平均 IOU 为 61.0，其性能类似于 9 个 Anchor 框的 60.9。如果我们使用 9 个中心，我们会看到更高的平均 IOU。这表明使用 k-means 来生成我们的边界框会以更好的表示开始训练模型，并使得任务更容易学习。

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 上最接近先验的边界框平均 IOU。VOC 2007 上目标的平均 IOU 与其最接近的，使用不同生成方法之前未经修改的平均值。聚类结果比使用手工选择的先验结果要更好。

Direct location prediction. When using anchor boxes with YOLO we encounter a second issue: model instability, especially during early iterations. Most of the instability comes from predicting the (x,y) locations for the box. In region proposal networks the network predicts values tx and ty and the (x,y) center coordinates are calculated as:

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

直接定位预测。当 YOLO 使用 Anchor 框时，我们会遇到第二个问题：模型不稳定，特别是在早期的迭代过程中。大部分的不稳定来自预测边界框的(x,y)位置。在 region proposal 网络中，网络预测值 tx 和 ty，(x, y)中心坐标计算如下：

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

For example, a prediction of tx=1 would shift the box to the right by the width of the anchor box, a prediction of tx=−1 would shift it to the left by the same amount.

例如，预测 tx=1 会将边界框向右移动 Anchor 框的宽度，预测 tx=−1 会将其向左移动相同的宽度。

This formulation is unconstrained so any anchor box can end up at any point in the image, regardless of what location predicted the box. With random initialization the model takes a long time to stabilize to predicting sensible offsets.

这个公式是不受限制的，所以任何 Anchor 框都可以在图像任一点结束，而不管在哪个位置预测该边界框。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量。

Instead of predicting offsets we follow the approach of YOLO and predict location coordinates relative to the location of the grid cell. This bounds the ground truth to fall between 0 and 1. We use a logistic activation to constrain the network’s predictions to fall in this range.

我们没有预测偏移量，而是按照 YOLO 的方法预测相对于网格单元位置的位置坐标。这使得真值落到了 0 和 1 之间。我们使用 logistic 激活函数来限制网络的预测值落在这个范围内。

The network predicts 5 bounding boxes at each cell in the output feature map. The network predicts 5 coordinates for each bounding box, tx, ty, tw, th, and to. 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:

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

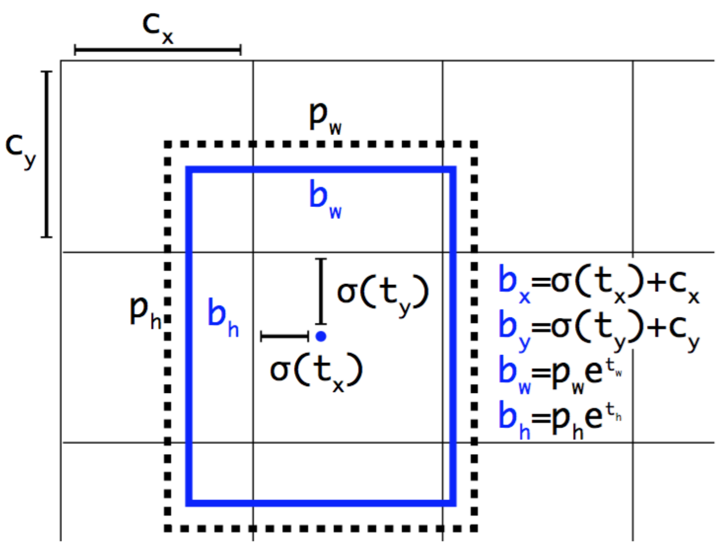


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

网络预测输出特征图中每个格子的 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。如果格子相对于图像的左上角偏移量为(cx, cy)，边界框先验的宽度和高度为 pw, ph，那么预测结果对应为：

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

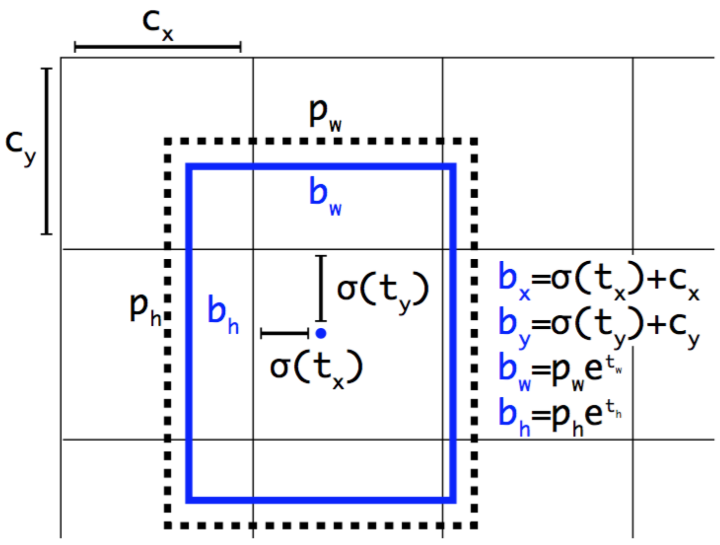


图 3：维度先验和位置预测的边界框示意图。我们预测边界框的宽度和高度作为聚类中心的偏移量。我们使用 sigmoid 函数预测边界框相对于卷积核应用位置的中心坐标。

Since we constrain the location prediction the parametrization is easier to learn, making the network more stable. Using dimension clusters along with directly predicting the bounding box center location improves YOLO by almost 5% over the version with anchor boxes.

由于我们限制位置预测参数化更容易学习，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的方式比使用 Anchor 框的版本将 YOLO 提高了近 5%。

Fine-Grained Features. This modified YOLO predicts detections on a 13 × 13 feature map. While this is sufficient for large objects, it may benefit from finer grained features for localizing smaller objects. Faster R-

CNN and SSD both run their proposal networks at various feature maps in the network to get a range of resolutions. We take a different approach, simply adding a passthrough layer that brings features from an earlier layer at 26 × 26 resolution.

细粒度特征。这个修改后的 YOLO 在 13×13 特征图上预测检测结果。虽然这对于大型目标来说已经足够了，但它通过更细粒度的特征定位出更小的目标。Faster R-CNN 和 SSD 都在网络的各种特征图上运行他们提出的网络，以获得一系列的分辨率。我们采用不同的方法，仅仅添加一个 passthrough 层，从 26x26 分辨率的更早层中提取特征。

The passthrough layer concatenates the higher resolution features with the low resolution features by stacking adjacent features into different channels instead of spatial locations, similar to the identity mappings in ResNet. This turns the 26 × 26 × 512 feature map into a 13 × 13 × 2048 feature map, which can be concatenated with the original features. Our detector runs on top of this expanded feature map so that it has access to fine grained features. This gives a modest 1% performance increase.

passthrough 层通过将相邻特征堆叠到不同的通道而不是空间位置来连接较高分辨率特征和较低分辨率特征，类似于 ResNet 中的恒等映射。将 26×26×512 特征图变成 13×13×2048 特征图（译者注：如何将 26×26×512 变成 13×13×2048？26×26×512 首先变成 4 个 13×13×512，然后在通道方向上将 4 个拼接在一起就成了 13×13×

2048），其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，以便它可以访问细粒度的特征。这会使性能提高 1%。

Multi-Scale Training. The original YOLO uses an input resolution of 448 × 448. With the addition of anchor boxes we changed the resolution to 416×416. However, since our model only uses convolutional and pooling layers it can be resized on the fly. We want YOLOv2 to be robust to running on images of different sizes so we train this into the model.

多尺度训练。原来的 YOLO 使用 448×448 的输入分辨率。通过添加 Anchor 框，我们将分辨率更改为 416×416。但是，由于我们的模型只使用卷积层和池化层，因此它可以动态调整大小。我们希望 YOLOv2 能够鲁棒地运行在不同大小的图像上，因此我们可以将该特性训练到模型中。

Instead of fixing the input image size we change the network every few iterations. Every 10 batches our network randomly chooses a new image dimension size. Since our model downsamples by a factor of 32, we pull from the following multiples of 32: {320, 352, ⋯, 608}. Thus the smallest option is 320 × 320 and the largest is 608 × 608. We resize the network to that dimension and continue training.

我们没有固定输入图像的大小，而是每隔几次迭代就改变网络。每隔 10 个批次我们的网络会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，我们从下面的 32 的倍数中选择：

{320,352，…，608}。因此最小的是 320×320，最大的是 608×608。我们将网络调整到这些尺寸并继续训练。

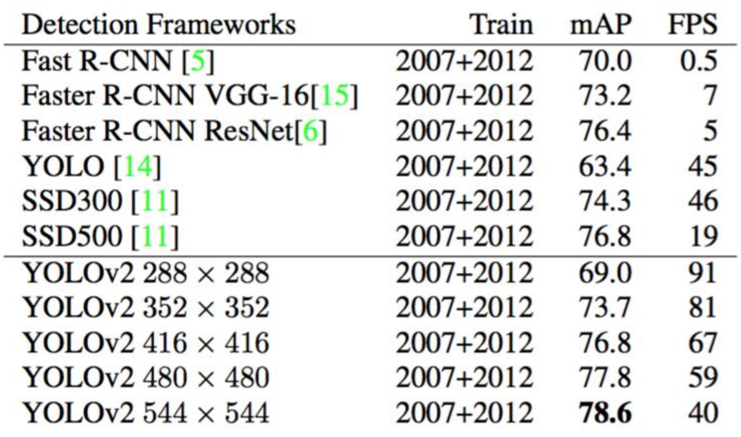
This regime forces the network to learn to predict well across a variety of input dimensions. This means the same network can predict detections at different resolutions. The network runs faster at smaller sizes so YOLOv2 offers an easy tradeoff between speed and accuracy.

这个模型架构迫使网络学习如何在各种输入维度上完成较好的预测。这意味着相同的网络可以预测不同分辨率下的检测结果。在更小尺寸上网络运行速度更快，因此 YOLOv2 在速度和准确性之间得到了一个简单的折衷。

At low resolutions YOLOv2 operates as a cheap, fairly accurate detector. At 288 × 288 it runs at more than 90 FPS with mAP almost as good as Fast R-CNN. This makes it ideal for smaller GPUs, high framerate video, or multiple video streams.

分辨率较低时 YOLOv2 可以作为一个低成本、相当准确的检测器。在 288×288 时，其运行速度超过 90FPS，mAP 与 Fast R-CNN 差不多。这使其成为小型 GPU、高帧率视频或多视频流的理想选择。

At high resolution YOLOv2 is a state-of-the-art detector with 78.6 mAP on VOC 2007 while still operating above real-time speeds. See Table 3 for a comparison of YOLOv2 with other frameworks on VOC 2007. Figure 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

Table 3: Detection frameworks on PASCAL VOC 2007. YOLOv2 is faster and more accurate than prior detection methods. It can also run at different resolutions for an easy tradeoff between speed and accuracy. Each YOLOv2 entry is actually the same trained model with the same weights, just evaluated at a different size. All timing information is on a Geforce GTX Titan X (original, not Pascal model).

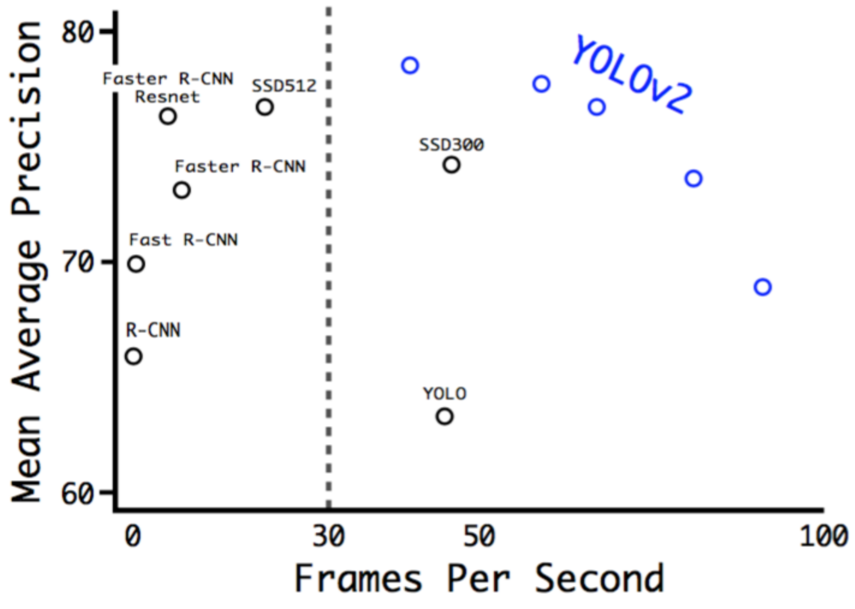


Figure 4: Accuracy and speed on VOC 2007.

在高分辨率下，YOLOv2 是 VOC 2007 上最先进的检测器，mAP 达到了 78.6，同时能够保持实时检测的速度要求。如表 3 所示为 YOLOv2 与其他框架在 VOC 2007 上的比较。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 上的检测框架。**YOLOv2** 比先前的检测方法更快、更准确。它也可以在不同的分辨率下运行，以便在速度和准确性之间进行简单折衷。每条 **YOLOv2** 结果实际上是具有相同权重的相同训练模型，只是以不同的输入大小进行评估。所有的时间信息都是在 **Geforce GTX Titan X**（原始的，而不是 **Pascal** 模型）上测得的。

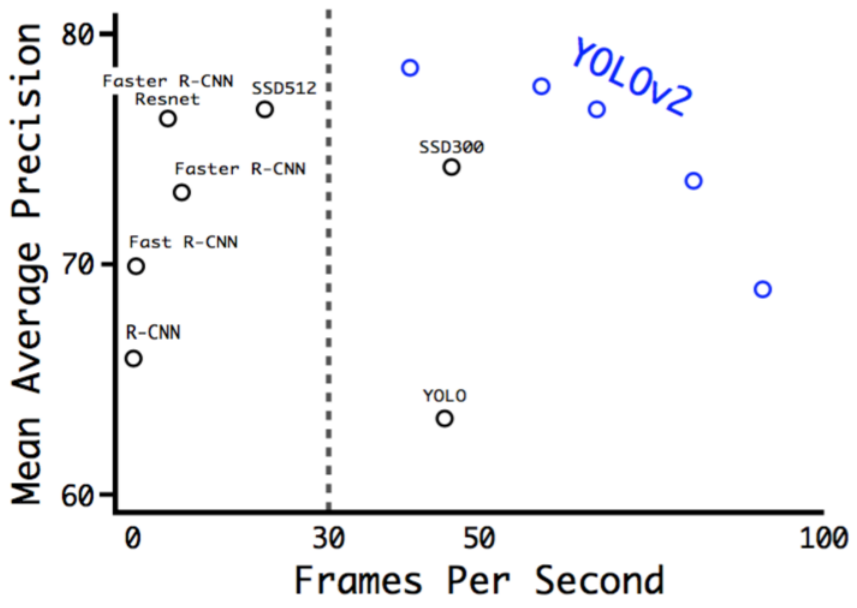
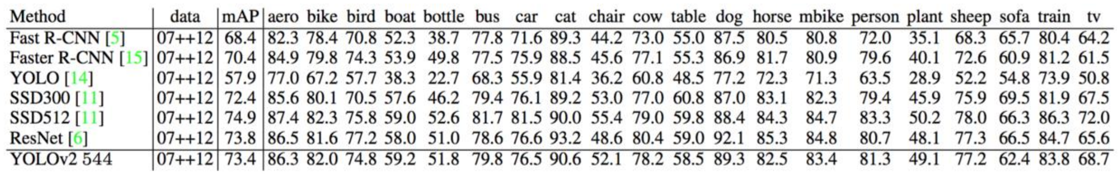


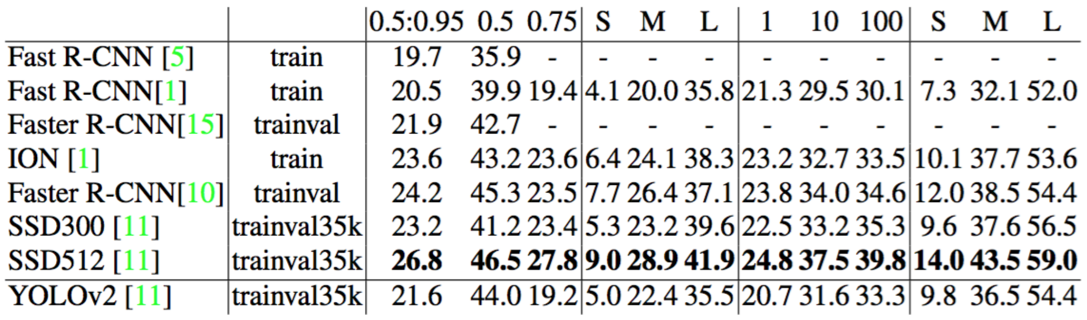
图 4：VOC 2007 上的准确性与速度。

Further Experiments. We train YOLOv2 for detection on VOC 2012. Table 4 shows the comparative performance of YOLOv2 versus other state-of-the-art detection systems. YOLOv2 achieves 73.4 mAP while running far faster than competing methods. We also train on COCO and compare to other methods in Table 5. On the VOC metric (IOU = .5) YOLOv2 gets 44.0 mAP, comparable to SSD and Faster R-CNN.



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

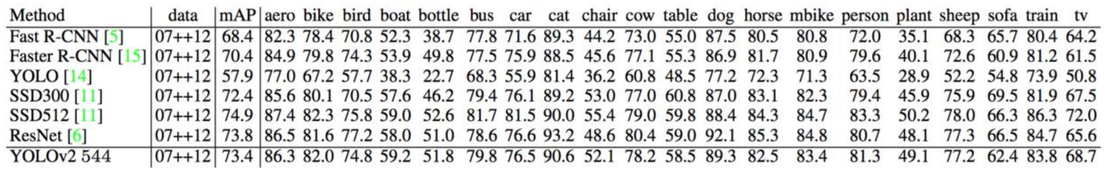
Table 4: PASCAL VOC2012 test detection results. YOLOv2 performs on par with state-of-the-art detectors like Faster R-CNN with ResNet and SSD512 and is 2−10× faster.



		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

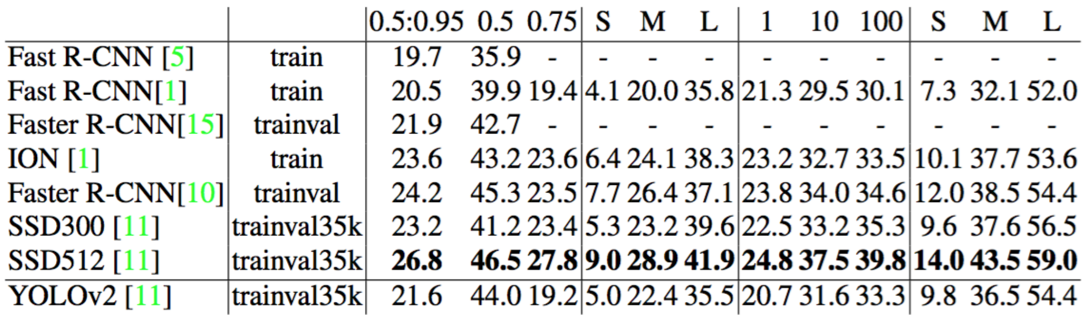
Table 5: Results on COCO test-dev2015. Table adapted from [11]

进一步实验。我们在 VOC 2012 上训练 YOLOv2 检测模型。表 4 所显为 YOLOv2 与其他最先进的检测系统性能比较的结果。YOLOv2 取得了 73.4 mAP 的同时运行速度比比对方法快的多。我们在 COCO 上进行了训练，并与表 5 中其他方法进行比较。在 VOC 指标（IOU = 0.5）上，YOLOv2 得到 44.0 mAP，与 SSD 和 Faster 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 VOC2012 测试集上的检测结果。YOLOv2 与最先进的检测器如具有 ResNet 的 Faster R-CNN、SSD512 在标准数据集上运行，YOLOv2 比它们快 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：在 COCO test-dev2015 数据集上的结果。表改编自[11]

3. Faster

We want detection to be accurate but we also want it to be fast. Most applications for detection, like robotics or self-driving cars, rely on low latency predictions. In order to maximize performance we design YOLOv2 to be fast from the ground up.

3. 更快

我们不仅希望检测是准确的，而且我们还希望它速度也快。大多数检测应用（如机器人或自动驾驶机车）依赖于低延迟预测。为了最大限度提高性能，我们从头开始设计 YOLOv2。

Most detection frameworks rely on VGG-16 as the base feature extractor [17]. VGG-16 is a powerful, accurate classification network but it is needlessly complex. The convolutional layers of VGG-16 require 30.69 billion floating point operations for a single pass over a single image at 224 × 224 resolution.

大多数检测框架依赖于 VGG-16 作为的基础特征提取器[17]。VGG-16 是一个强大的、准确的分类网络，但它有些过于复杂。在单

张图像 224×224 分辨率的情况下，VGG-16 的卷积层运行一次前向传播需要 306.90 亿次浮点运算。

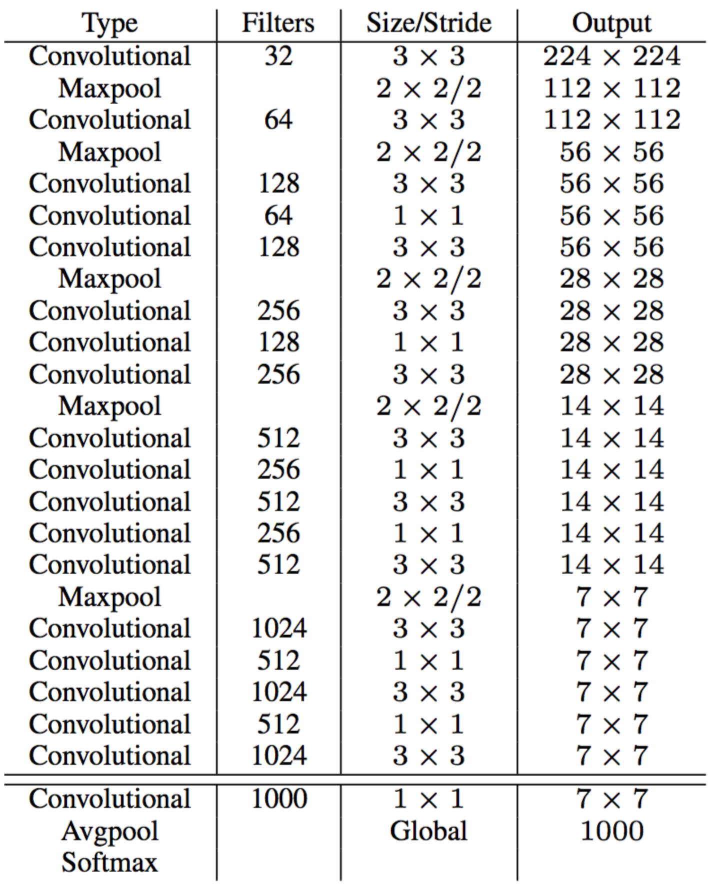
The YOLO framework uses a custom network based on the Googlenet architecture [19]. This network is faster than VGG-16, only using 8.52 billion operations for a forward pass. However, it's accuracy is slightly worse than VGG-16. For single-crop, top-5 accuracy at 224 × 224, YOLO's custom model gets 88.0% ImageNet compared to 90.0% for VGG-16.

YOLO 框架使用基于 GoogLeNet 架构[19]的自定义网络。这个网络比 VGG-16 更快，一次前向传播只有 85.2 亿次的计算操作。然而，它的准确性比 VGG-16 略差（译者注：ILSVRC2014 竞赛中 GoogLeNet 获得分类任务第一名，VGG 第二名，但是在定位任务中 VGG 是第一名）。在 ImageNet 上，对于单张裁剪图像，224×224 分辨率下的 top-5 准确率，YOLO 的自定义模型获得了 88.0%，而 VGG-16 则为 90.0%。

Darknet-19. We propose a new classification model to be used as the base of YOLOv2. Our model builds off of prior work on network design as well as common knowledge in the field. Similar to the VGG models we use mostly 3 × 3 filters and double the number of channels after every pooling step [17]. Following the work on Network in Network (NIN) we use global average pooling to make predictions as well as 1 × 1 filters to compress the feature representation between 3 × 3 convolutions [9]. We use batch normalization to stabilize training, speed up convergence, and regularize the model [7].

Darknet-19。我们提出了一个新的分类模型作为 YOLOv2 的基础。我们的模型建立在网络设计先前工作以及该领域常识的基础上。与 VGG 模型类似，我们大多使用 3×3 卷积核，并在每个池化步骤之后使得通道数量加倍[17]。按照 Network in Network（NIN）的工作，我们使用全局平均池化的结果做预测，并且使用 1×1 卷积核来压缩 3×3 卷积之间的特征表示[9]。我们使用批归一化来稳定训练、加速收敛，并正则化模型[7]。

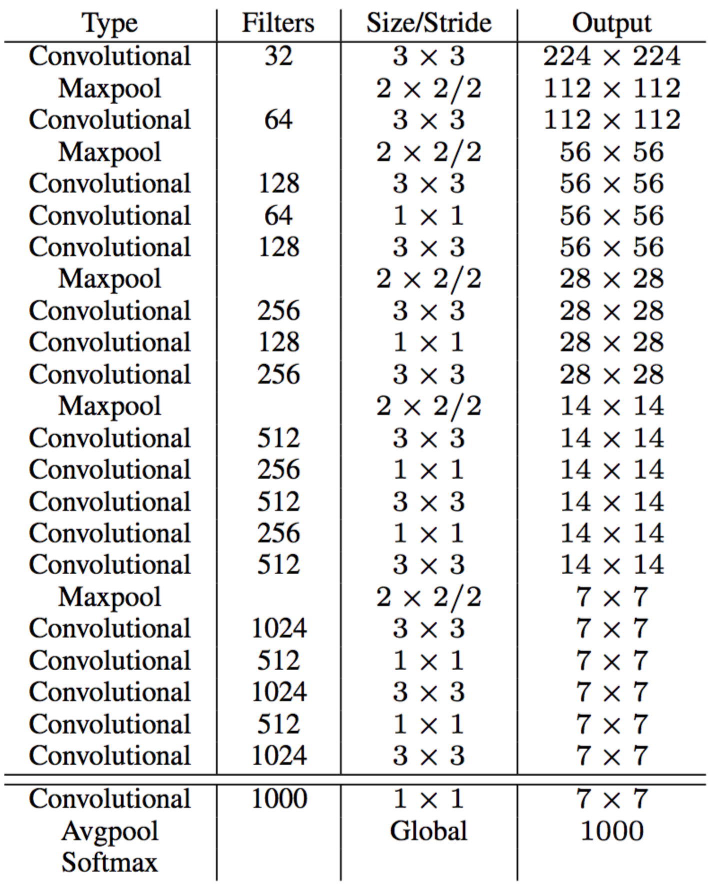
Our final model, called Darknet-19, has 19 convolutional layers and 5 maxpooling layers. For a full description see Table 6. Darknet-19 only requires 5.58 billion operations to process an image yet achieves 72.9% top-1 accuracy and 91.2% top-5 accuracy on ImageNet.



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			

Table 6: Darknet-19.

我们的最终模型叫做 Darknet-19，它有 19 个卷积层和 5 个最大池化层。完整描述请看表 6。Darknet-19 只需要 55.8 亿次运算来处理图像，但在 ImageNet 上却达到了 72.9%的 top-1 准确率和 91.2%的 top-5 准确率。



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。

Training for classification. We train the network on the standard ImageNet 1000 class classification dataset for 160 epochs using stochastic gradient descent with a starting learning rate of 0.1, polynomial rate decay with a power of 4, weight decay of 0.0005 and momentum of 0.9 using the Darknet neural network framework [13]. During training we use standard data augmentation tricks including random crops, rotations, and hue, saturation, and exposure shifts.

分类训练。我们使用 Darknet 神经网络结构，使用随机梯度下降、初始学习率为 0.1、学习率多项式衰减系数为 4、权重衰减为 0.0005、动量为 0.9，在标准 ImageNet 1000 类分类数据集上训练网络 160 个

迭代周期[13]。在训练过程中，我们使用标准的数据增强技巧，包括随机裁剪、旋转、以及色调、饱和度和曝光的改变。

As discussed above, after our initial training on images at 224 × 224 we fine tune our network at a larger size, 448. For this fine tuning we train with the above parameters but for only 10 epochs and starting at a learning rate of 10^{-3}. At this higher resolution our network achieves a top-1 accuracy of 76.5% and a top-5 accuracy of 93.3%.

如上所述，在我们对 224×224 的图像进行初始训练之后，我们对网络在更大的尺寸 448 上进行了 fine tune。对于这种 fine tune，我们使用上述参数进行训练，但是只有 10 个迭代周期，并且以 10^{-3} 的学习率开始（译者注：fine-tune 时通常会使用较低的学习率）。在这种更高的分辨率下，我们的网络达到了 76.5%的 top-1 准确率和 93.3%的 top-5 准确率。

Training for detection. We modify this network for detection by removing the last convolutional layer and instead adding on three 3 × 3 convolutional layers with 1024 filters each followed by a final 1 × 1 convolutional layer with the number of outputs we need for detection. For VOC we predict 5 boxes with 5 coordinates each and 20 classes per box so 125 filters. We also add a passthrough layer from the final 3 × 3 × 512 layer to the second to last convolutional layer so that our model can use fine grain features.

检测训练。我们修改这个网络使得可以用于检测任务，删除了最后一个卷积层，加上了三层具有 1024 个卷积核的 3×3 卷积层，每层后面接 1×1 卷积层，卷积核数量与我们检测输出数量一致。对于 VOC，我们预测 5 个边界框，每个边界框有 5 个坐标和 20 个类别，所以有 125 个卷积核。我们还添加了从最后的 3×3×512 层到倒数第二层卷积层的直通层，以便我们的模型可以使用细粒度特征。

We train the network for 160 epochs with a starting learning rate of 10^{-3}, dividing it by 10 at 60 and 90 epochs. We use a weight decay of 0.0005 and momentum of 0.9. We use a similar data augmentation to YOLO and SSD with random crops, color shifting, etc. We use the same training strategy on COCO and VOC.

我们训练网络 160 个迭代周期，初始学习率为 10^{-3}，在 60 个和 90 个迭代周期时将学习率除以 10。我们使用 0.0005 的权重衰减和 0.9 的动量。我们对 YOLO 和 SSD 进行类似的数据增强：随机裁剪、色彩改变等。我们对 COCO 和 VOC 使用相同的训练策略。

4. Stronger

We propose a mechanism for jointly training on classification and detection data. Our method uses images labelled for detection to learn detection-specific information like bounding box coordinate prediction and objectness as well as how to classify common objects. It uses images with only class labels to expand the number of categories it can detect.

4. 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。

During training we mix images from both detection and classification datasets. When our network sees an image labelled for detection we can backpropagate based on the full YOLOv2 loss function. When it sees a classification image we only backpropagate loss from the classification-specific parts of the architecture.

在训练期间，我们混合来自检测数据集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，我们只能从该架构特定的分类部分反向传播损失。

This approach presents a few challenges. Detection datasets have only common objects and general labels, like dog or boat. Classification datasets have a much wider and deeper range of labels. ImageNet has more than a hundred breeds of dog, including Norfolk terrier, Yorkshire terrier, and Bedlington terrier. If we want to train on both datasets we need a coherent way to merge these labels.

这种方法存在一些挑战。检测数据集只有常见目标和通用标签，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。

Most approaches to classification use a softmax layer across all the possible categories to compute the final probability distribution. Using a softmax assumes the classes are mutually exclusive. This presents problems for combining datasets, for example you would not want to combine ImageNet and COCO using this model because the classes “Norfolk terrierand” and “dog” are not mutually exclusive.

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 假定这些类是互斥的。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 不是互斥的。

We could instead use a multi-label model to combine the datasets which does not assume mutual exclusion. This approach ignores all the structure we do know about the data, for example that all of the COCO classes are mutually exclusive.

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. ImageNet labels are pulled from WordNet, a language database that structures concepts and how they relate [12]. In WordNet, Norfolk terrier and Yorkshire terrier are both hyponyms of terrier which is a type of hunting dog, which is a type of dog, which is a canine, etc. Most approaches to classification assume a flat structure to the labels however for combining datasets, structure is exactly what we need.

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[12]。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。分类的大多数方法假设标签是一个扁平结构，但是对于数据集的组合，结构正是我们所需要的。

WordNet is structured as a directed graph, not a tree, because language is complex. For example a dog is both a type of canine and a type of domestic animal which are both synsets in WordNet. Instead of using the full graph structure, we simplify the problem by building a hierarchical tree from the concepts in ImageNet.

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（犬），也是一种 domestic animal（家畜），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。

To build this tree we examine the visual nouns in ImageNet and look at their paths through the WordNet graph to the root node, in this case “physical object”. Many synsets only have one path through the graph so first we add all of those paths to our tree. Then we iteratively examine the concepts we have left and add the paths that grow the tree by as little as possible. So if a concept has two paths to the root and one path would add three edges to our tree and the other would only add one edge, we choose the shorter path.

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。

The final result is WordTree, a hierarchical model of visual concepts. To perform classification with WordTree we predict conditional probabilities at every node for the probability of each hyponym of that synset given that synset. For example, at the terrier node we predict:

Pr(Norfolk terrier|terrier)

Pr(Yorkshire terrier|terrier)

Pr(Bedlington terrier|terrier)

...

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：

Pr(Norfolk terrier|terrier)

Pr(Yorkshire terrier|terrier)

Pr(Bedlington terrier|terrier)

...

If we want to compute the absolute probability for a particular node we simply follow the path through the tree to the root node and multiply to conditional probabilities. So if we want to know if a picture is of a Norfolk terrier we compute:

Pr(Norfolk terrier)=Pr(Norfolk terrier|terrier)

*Pr(terrier|hunting dog)

…

*Pr(mammal|Pr(animal)

*Pr(animal|physical object)

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：

Pr(Norfolk terrier)=Pr(Norfolk terrier|terrier)

*Pr(terrier|hunting dog)

…

*Pr(mammal|Pr(animal)

*Pr(animal|physical object)

For classification purposes we assume that the the image contains an object: Pr(physical object)=1.

对于分类任务，我们假定图像只包含一个目标：Pr(physical object)=1。

To validate this approach we train the Darknet-19 model on WordTree built using the 1000 class ImageNet. To build WordTree1k we add in all of the intermediate nodes which expands the label space from 1000 to 1369. During training we propagate ground truth labels up the tree so that if an image is labelled as a Norfolk terrier it also gets labelled as a dog and a mammal, etc. To compute the conditional probabilities our model predicts a vector of 1369 values and we compute the softmax over all sysnsets that are hyponyms of the same concept, see Figure 5.

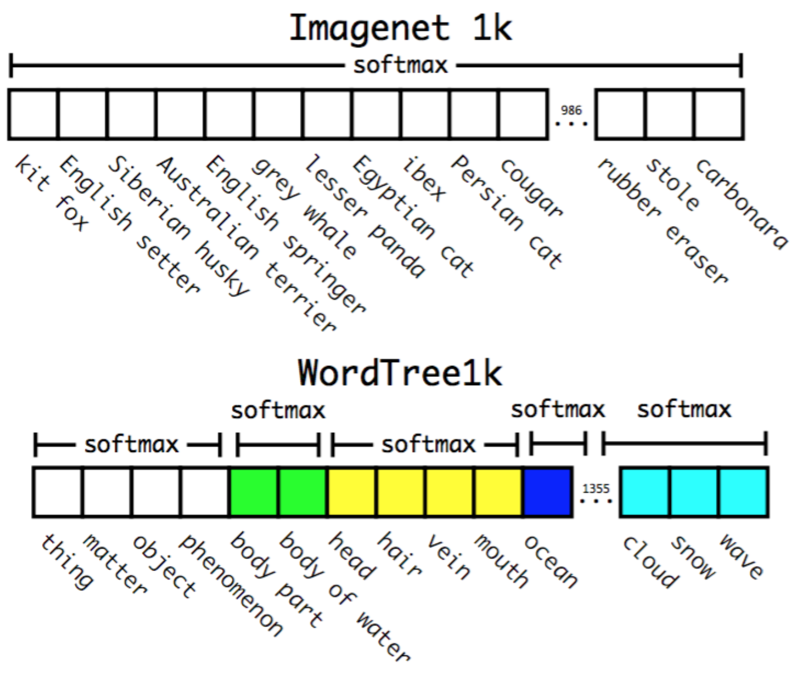


Figure 5: Prediction on ImageNet vs WordTree. Most ImageNet models use one large softmax to predict a probability distribution. Using WordTree we perform multiple softmax operations over co-hyponyms.

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。

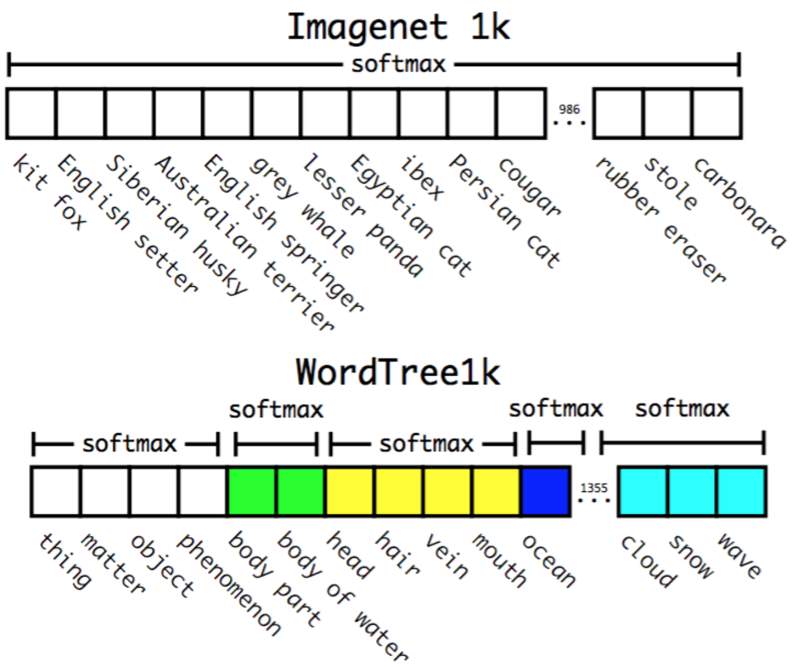


图 5：在 ImageNet 与 WordTree 上的预测。大多数 ImageNet 模型使用一个较大的 softmax 来预测概率分布。使用 WordTree，我们可以在共同的下义词上执行多次 softmax 操作。

Using the same training parameters as before, our hierarchical Darknet-19 achieves 71.9% top-1 accuracy and 90.4% top-5 accuracy. Despite adding 369 additional concepts and having our network predict a tree structure our accuracy only drops marginally. Performing classification in this manner also has some benefits. Performance degrades gracefully on new or unknown object categories. For example, if the network sees a picture of a dog but is uncertain what type of dog it is, it will still predict dog with high confidence but have lower confidences spread out among the hyponyms.

使用与之前相同的训练参数，我们的分层 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不会下降太多。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”，但是扩展到下义词后可能有更低的置信度。

This formulation also works for detection. Now, instead of assuming every image has an object, we use YOLOv2’s objectness predictor to give us the value of Pr(physical object). The detector predicts a bounding box and the tree of probabilities. We traverse the tree down, taking the highest confidence path at every split until we reach some threshold and we predict that object class.

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(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。

Dataset combination with WordTree. We can use WordTree to combine multiple datasets together in a sensible fashion. We simply map the categories in the datasets to synsets in the tree. Figure 6 shows an example of using WordTree to combine the labels from ImageNet and COCO. WordNet is extremely diverse so we can use this technique with most datasets.

使用 WordTree 组合数据集。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同义词集（synsets）即可。图 6 显示了使用 WordTree 来组合来自 ImageNet 和 COCO 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。

Joint classification and detection. Now that we can combine datasets using WordTree we can train our joint model on classification and detection. We want to train an extremely large scale detector so we create our combined dataset using the COCO detection dataset and the top 9000 classes from the full ImageNet release. We also need to evaluate our method so we add in any classes from the ImageNet detection challenge that were not already included. The corresponding WordTree for this dataset has 9418 classes. ImageNet is a much larger dataset so we balance the dataset by oversampling COCO so that ImageNet is only larger by a factor of 4:1.

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，因此还添加了 ImageNet 检测挑战中未包含的类。该数据集的对应的 WordTree 有 9418 个类别。ImageNet 相比于 COCO 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。

Using this dataset we train YOLO9000. We use the base YOLOv2 architecture but only 3 priors instead of 5 to limit the output size. When our network sees a detection image we backpropagate loss as normal. For classification loss, we only backpropagate loss at or above the corresponding level of the label. For example, if the label is dog we do assign any error to predictions further down in the tree, German Shepherd versus Golden Retriever, because we do not have that information.

使用这个数据集我们训练 YOLO9000。我们使用基础的 YOLOv2 架构，但只有 3 个先验（priors）而不是 5 个来限制输出大小。当我们的网络看到一个检测图像时，我们正常地对损失进行反向传播。对于分类损失，我们仅在等于或高于标签对应的层对损失进行反向传播。

例如，如果标签是“狗”，我们将沿着树向下进一步预测“德国牧羊犬”与“金毛猎犬”之间的差异，因为我们没有这些信息。

When it sees a classification image we only backpropagate classification loss. To do this we simply find the bounding box that predicts the highest probability for that class and we compute the loss on just its predicted tree. We also assume that the predicted box overlaps what would be the ground truth label by at least 0.3 IOU and we backpropagate objectness loss based on this assumption.

当它看到分类图像时，我们只能反向传播分类损失。要做到这一点，我们只需找到预测该类别最高概率的边界框，然后计算其预测树上的损失。我们还假设预测边界框与真实标签重叠至少 0.3 的 IOU，并且基于这个假设反向传播目标损失。

Using this joint training, YOLO9000 learns to find objects in images using the detection data in COCO and it learns to classify a wide variety of these objects using data from ImageNet.

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。

We evaluate YOLO9000 on the ImageNet detection task. The detection task for ImageNet shares on 44 object categories with COCO which means that YOLO9000 has only seen classification data for the majority of the test images, not detection data. YOLO9000 gets 19.7 mAP overall with 16.0 mAP on the disjoint 156 object classes that it has never seen any labelled detection data for. This mAP is higher than results achieved by DPM but YOLO9000 is trained on different datasets with only partial supervision [4]. It also is simultaneously detecting 9000 other object categories, all in real-time.

我们在 ImageNet 检测任务上评估了 YOLO9000。ImageNet 的检测任务与 COCO 共有的目标类别有 44 个，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[4]。它也同时检测 9000 个其他目标类别，所有的都是实时的。

When we analyze YOLO9000’s performance on ImageNet we see it learns new species of animals well but struggles with learning categories like clothing and equipment. New animals are easier to learn because the objectness predictions generalize well from the animals in COCO. Conversely, COCO does not have bounding box label for any type of clothing, only for person, so YOLO9000 struggles to model categories like “sunglasses” or “swimming trunks”.

当我们分析 YOLO9000 在 ImageNet 上的性能时，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效果不好。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，只有针对人的检测标签，因此 YOLO9000 很难建模好“墨镜”或“泳裤”等类别。

5. Conclusion

We introduce YOLOv2 and YOLO9000, real-time detection systems. YOLOv2 is state-of-the-art and faster than other detection systems across a variety of detection datasets. Furthermore, it can be run at a variety of image sizes to provide a smooth tradeoff between speed and accuracy.

5. 结论

我们介绍了 YOLOv2 和 YOLO9000，两个实时检测系统。YOLOv2 在各种检测数据集上都是最先进的，也比其他检测系统更快。此外，它可以运行在各种图像大小，以提供速度和准确性之间的平滑折衷。

YOLO9000 is a real-time framework for detection more than 9000 object categories by jointly optimizing detection and classification. We use WordTree to combine data from various sources and our joint optimization technique to train simultaneously on ImageNet and COCO. YOLO9000 is a strong step towards closing the dataset size gap between detection and classification.

YOLO9000 是一个通过联合优化检测和分类来检测 9000 多个目标类别的实时框架。我们使用 WordTree 将各种来源的数据和我们的

联合优化技术相结合，在 ImageNet 和 COCO 上同时进行训练。YOLO9000 是在检测和分类之间缩小数据集大小差距的重要一步。

Many of our techniques generalize outside of object detection. Our WordTree representation of ImageNet offers a richer, more detailed output space for image classification. Dataset combination using hierarchical classification would be useful in the classification and segmentation domains. Training techniques like multi-scale training could provide benefit across a variety of visual tasks.

我们的许多技术都可以泛化到目标检测之外。我们对 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领域将是有用的。像多尺度训练这样的训练技术可以为各种视觉任务提供益处。

For future work we hope to use similar techniques for weakly supervised image segmentation. We also plan to improve our detection results using more powerful matching strategies for assigning weak labels to classification data during training. Computer vision is blessed with an enormous amount of labelled data. We will continue looking for ways to bring different sources and structures of data together to make stronger models of the visual world.

对于未来的工作，我们希望使用类似的技术来进行弱监督的图像分割。我们还计划使用更强大的匹配策略来改善我们的检测结果，以在训练期间将弱标签分配给分类数据。计算机视觉需要大量标记的数

据。我们将继续寻找方法，将不同来源和数据结构的数据整合起来，形成更强大的视觉世界模型。

References

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) challenge. 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 learning 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. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. In European Conference on Computer 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: Towards 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