论文： YOLO9000：Better，Faster，Stronger

论文链接：https://arxiv.org/abs/1612.08242

YOLO9000是CVPR2017的最佳论文提名。首先讲一下这篇文章一共介绍了YOLO v2和YOLO9000两个模型，二者略有不同。前者主要是YOLO的升级版（关于YOLO v1的介绍可以参考：YOLO v1算法详解），后者的主要检测网络也是YOLO v2，同时对数据集做了融合，使得模型可以检测9000多类物体。而提出YOLO9000的原因主要是目前检测的数据集数据量较小，因此利用数量较大的分类数据集来帮助训练检测模型。

接下来基本上按照文章的顺序来解读一下算法，这样读起来也比较清晰。主要包括三个部分：Better，Faster，Stronger，其中前面两部分基本上讲的是YOLO v2，最后一部分讲的是YOLO9000。

Better

这部分细节很多，想要详细了解的话建议还是看源码。

很明显，本篇论文是YOLO作者为了改进原有的YOLO算法所写的。**YOLO有两个缺点：一个缺点在于定位不准确，另一个缺点在于和基于region proposal的方法相比召回率较低。**因此YOLOv2主要是要在这两方面做提升。另外YOLOv2并不是通过加深或加宽网络达到效果提升，反而是简化了网络。大概看一下YOLOv2的表现：**YOLOv2算法在VOC 2007数据集上的表现为67 FPS时，MAP为76.8，在40FPS时，MAP为78.6.**

1、Batch Normalization

BN（Batch Normalization）层简单讲就是对网络的每一层的输入都做了归一化，这样网络就不需要每层都去学数据的分布，收敛会快点。原来的YOLO算法（采用的是GoogleNet网络提取特征）是没有BN层的，因此在YOLOv2中作者为每个卷积层都添加了BN层。另外由于BN可以规范模型，所以本文加入BN后就把dropout去掉了。实验证明添加了BN层可以提高2%的mAP。

2、High Resolution Classifier

首先fine-tuning的作用不言而喻，现在基本跑个classification或detection的模型都不会从随机初始化所有参数开始，所以一般都是用预训练的网络来finetuning自己的网络，而且预训练的网络基本上都是在ImageNet数据集上跑的，一方面数据量大，另一方面训练时间久，而且这样的网络都可以在相应的github上找到。

原来的YOLO网络在预训练的时候采用的是224*224的输入（这是因为一般预训练的分类模型都是在ImageNet数据集上进行的），然后在detection的时候采用448*448的输入，这会导致从分类模型切换到检测模型的时候，模型还要适应图像分辨率的改变。而YOLOv2则将预训练分成两步：先用224*224的输入从头开始训练网络，大概160个epoch（表示将所有训练数据循环跑160次），然后再将输入调整到448*448，再训练10个epoch。注意这两步都是在ImageNet数据集上操作。最后再在检测的数据集上fine-tuning，也就是detection的时候用448*448的图像作为输入就可以顺利过渡了。作者的实验表明这样可以提高几乎4%的MAP。

3、Convolutional With Anchor Boxes

原来的YOLO是利用全连接层直接预测bounding box的坐标，而YOLOv2借鉴了Faster R-CNN的思想，引入anchor。首先将原网络的全连接层和最后一个pooling层去掉，使得最后的卷积层可以有更高分辨率的特征；然后缩减网络，用416*416大小的输入代替原来448*448。这样做的原因在于希望得到的特征图都有奇数大小的宽和高，奇数大小的宽和高会使得每个特征图在划分cell的时候就只有一个center cell（比如可以划分成7*7或9*9个cell，center cell只有一个，如果划分成8*8或10*10的，center cell就有4个）。为什么希望只有一个center cell呢？因为大的object一般会占据图像的中心，所以希望用一个center cell去预测，而不是4个center cell去预测。网络最终将416*416的输入变成13*13大小的feature map输出，也就是缩小比例为32。

我们知道原来的YOLO算法将输入图像分成7*7的网格，每个网格预测两个bounding box，因此一共只有98个box，但是在YOLOv2通过引入anchor boxes，

预测的box数量超过了1千（以输出feature map大小为13*13为例，每个grid cell有9个anchor box的话，一共就是13*13*9=1521个，当然由后面第4点可知，最终每个grid cell选择5个anchor box）。顺便提一下在Faster RCNN在输入大小为1000*600时的boxes数量大概是6000，在SSD300中boxes数量是8732。显然增加box数量是为了提高object的定位准确率。

作者的实验证明：虽然加入anchor使得MAP值下降了一点（69.5降到69.2），但是提高了recall（81%提高到88%）。

4、Dimension Clusters

我们知道在Faster R-CNN中anchor box的大小和比例是按经验设定的，然后网络会在训练过程中调整anchor box的尺寸。**但是如果一开始就能选择到合适尺寸的anchor box，那肯定可以帮助网络越好地预测detection。**所以作者采用k-means**的方式对训练集的**bounding boxes**做聚类**，试图找到合适的anchor box。

另外作者发现如果采用标准的k-means（即用欧式距离来衡量差异），在box的尺寸比较大的时候其误差也更大，而我们希望的是误差和box的尺寸没有太大关系。所以通过IOU定义了如下的距离函数，使得误差和box的大小无关：

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

如下图Figure2，左边是聚类的簇个数核IOU的关系，两条曲线分别代表两个不同的数据集。在分析了聚类的结果并平衡了模型复杂度与recall值，作者选择了K=5，这也就是Figure2中右边的示意图是选出来的5个box的大小，这里紫色和黑色也是分别表示两个不同的数据集，可以看出其基本形状是类似的。而且发现聚类的结果和手动设置的anchor box大小差别显著。**聚类的结果中多是高瘦的box，而矮胖的box数量较少。**

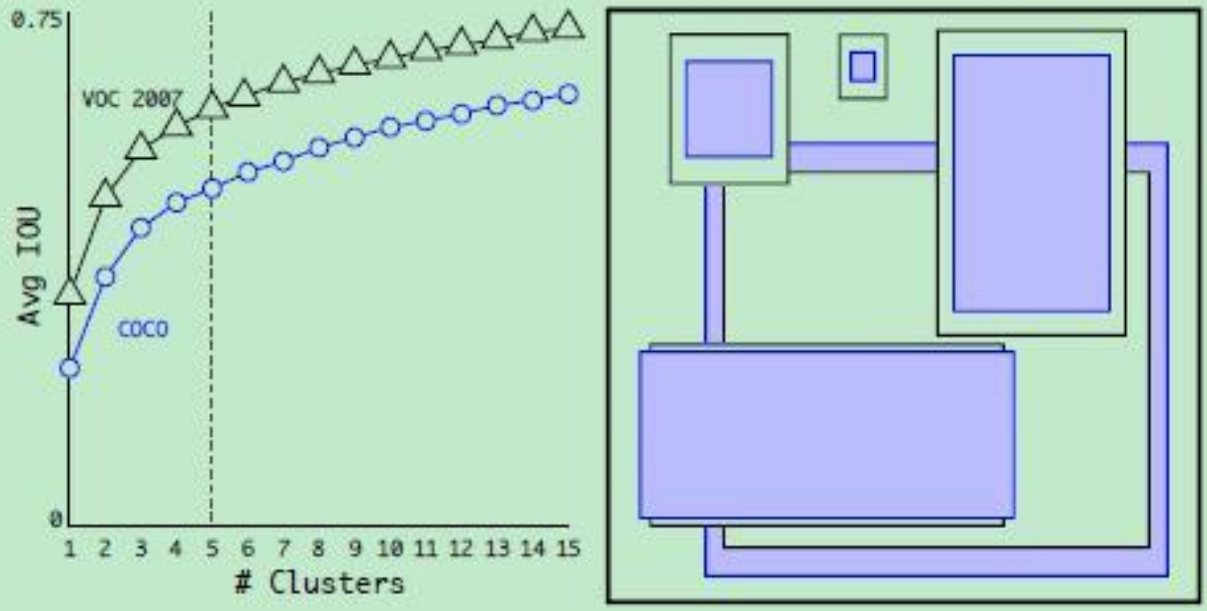

Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that $k = 5$ gives a good tradeoff for recall vs. complexity of the model. 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.

Table1中作者采用的5种anchor（Cluster IOU）的Avg IOU是61，而采用9种Anchor Boxes的Faster RCNN的Avg IOU是60.9，也就是说本文仅选取5种box就能达到Faster RCNN的9中box的效果。

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.

5、Direct Location prediction

作者在引入anchor box的时候遇到的第二个问题：模型不稳定，尤其是在训练刚开始的时候。作者认为这种不稳定主要来自预测box的(x,y)值。**我们知道在基于region proposal的object detection算法中，是通过预测下图中的tx和ty来得到(x,y)值，也就是预测的是offset。**另外关于文中的这个公式，个人认为应该把后面的减号改成加号，这样才能符合公式下面的example。**这里xa和ya是anchor的坐标，wa和ha是anchor的size，x和y是坐标的预测值，tx和ty是偏移量。**文中还特地举了一个例子：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.

region proposal networks the network predicts values t_x and t_y and the (x, y) center coordinates are calculated as:

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

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

这里贴一下Faster R-CNN里面的公式，和上面这个公式将减号变成加号是一致的。

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a,$$

在这里作者并没有采用直接预测offset的方法，还是沿用了YOLO算法中直接预测相对于grid cell的坐标位置的方式。

前面提到网络在最后一个卷积层输出13*13大小的feature map，然后每个cell预测5个bounding box，然后每个bounding box预测5个值：tx，ty，tw，th和to（这里的to类似YOLOv1中的confidence）。看下图，tx和ty经过sigmoid函数处理后范围在0到1之间，这样的归一化处理也使得模型训练更加稳定；cx和cy表示一个cell和图像左上角的横纵距离；pw和ph表示bounding box的宽高，这样bx和by就是cx和cy这个cell附近的anchor来预测tx和ty得到的结果。

the output feature map. The network predicts 5 coordinates for each bounding box, t_x, t_y, t_w, t_h, and t_o. If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w, p_h, then the predictions correspond to:

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

如果对上面的公式不理解，可以看Figure3，首先是cx和cy，表示grid cell与图像左上角的横纵坐标距离，黑色虚线框是bounding box，蓝色矩形框就是预测的结果。

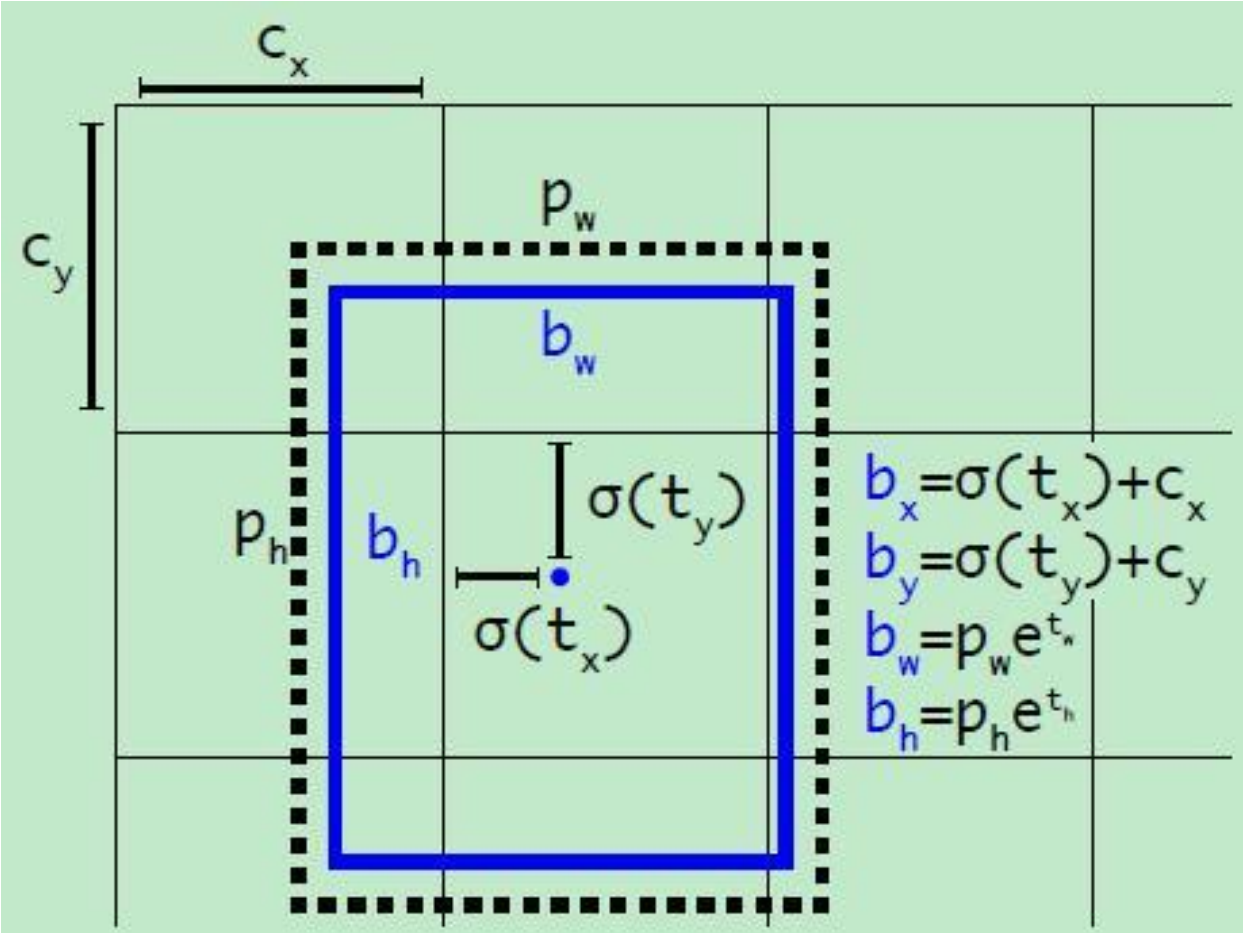

Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

6、Fine-Grained Features

这里主要是添加了一个层：passthrough layer。这个层的作用就是将前面一层的26*26的feature map和本层的13*13的feature map进行连接，有点像ResNet。这样做的原因在于虽然13*13的feature map对于预测大的object以及足够了，但是对于预测小的object就不一定有效。也容易理解，越小的object，经过层层卷积和pooling，可能到最后都不见了，所以通过合并前一层的size大一点的feature map，可以有效检测小的object。

7、Multi-Scale Training

为了让YOLOv2模型更加robust，作者引入了Muinti-Scale Training，简单讲就是在训练时输入图像的size是动态变化的，**注意这一步是在检测数据集上fine tune时候采用的，不要跟前面在Imagenet数据集上的两步预训练分类模型混淆，本文细节确实很多**。具体来讲，在训练网络时，每训练10个batch（文中是10个batch，个人认为会不会是笔误，不应该是10个epoch？），网络就会随机选择另一种size的输入。那么输入图像的size的变化范围要怎么定呢？前面我们知道本文网络本来的输入是416*416，最后会输出13*13的feature map，也就是说downsample的factor是32，因此作者采用32的倍数作为输入的size，具体来讲文中作者采用从{320,352,…,608}的输入尺寸。

这种网络训练方式使得相同网络可以对不同分辨率的图像做detection。虽然在输入size较大时，训练速度较慢，但同时在输入size较小时，训练速度较快，而multi-scale training又可以提高准确率，因此算是准确率和速度都取得一个不错的平衡。

Table3就是在检测时，不同输入size情况下的YOLOv2和其他object detection算法的对比。可以看出通过multi-scale training的检测模型，在测试的时候，输入图像在尺寸变化范围较大的情况下也能取得mAP和FPS的平衡。不过同时也可以看出SSD算法的表现也十分抢眼。

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).

总的看下这些技巧对mAP的贡献：

	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%.

High Resolution Classifier的提升非常明显（近4%），另外通过结合dimension prior+localtion prediction这两种方式引入anchor也能带来近5%mAP的提升。

Faster

在YOLO v1中，作者采用的训练网络是基于GooleNet，这里作者将GooleNet和VGG16做了简单的对比，**GooleNet在计算复杂度上要优于VGG16（8.25 billion operation VS 30.69 billion operation），但是前者在ImageNet上的top-5准确率要稍低于后者（88% VS 90%）**。而在YOLO v2中，作者采用了新的分类模型作为基础网络，那就是Darknet-19。

1、Darknet-19

Table6是最后的网络结构：Darknet-19只需要5.58 billion operation。这个网络包含19个卷积层和5个max pooling层，而在YOLO v1中采用的GooleNet，包含24个卷积层和2个全连接层，因此Darknet-19整体上卷积卷积操作比YOLO v1中用的GoogleNet要少，这是计算量减少的关键。最后用average pooling层代替全连接层进行预测。**这个网络在ImageNet上取得了top-5的91.2%的准确率**。

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			

Table 6: Darknet-19.

2、Training for Classification

这里的2和3部分在前面有提到，就是训练处理的小trick。这里的training for classification都是在ImageNet上进行预训练，主要分两步：1、从头开始训练

Darknet-19，数据集是ImageNet，训练160个epoch，输入图像的大小是224*224，初始学习率为0.1。另外在训练的时候采用了标准的数据增加方式比如随机裁剪，旋转以及色度，亮度的调整等。2、再fine-tuning 网络，这时候采用448*448的输入，参数的除了epoch和learning rate改变外，其他都没变，这里learning rate改为0.001，并训练10个epoch。**结果表明fine-tuning后的top-1准确率为76.5%，top-5准确率为93.3%，而如果按照原来的训练方式，Darknet-19的top-1准确率是72.9%，top-5准确率为91.2%。因此可以看出第1,2两步分别从网络结构和训练方式两方面入手提高了主网络的分类准确率。**

3、Training for Detection

在前面第2步之后，就开始把网络移植到detection，并开始基于检测的数据再进行fine-tuning。**首先把最后一个卷积层去掉，然后添加3个3*3的卷积层，每个卷积层有1024个filter，而且每个后面都连接一个1*1的卷积层，1*1卷积的filter个数根据需要检测的类来定。**比如对于VOC数据，由于每个grid cell我们需要预测5个box，每个box有5个坐标值和20个类别值，所以每个grid cell有125个filter（**与YOLOv1不同，在YOLOv1中每个grid cell有30个filter，还记得那个7*7*30的矩阵吗，而且在YOLOv1中，类别概率是由grid cell来预测的，也就是说一个grid cell对应的两个box的类别概率是一样的，但是在YOLOv2中，类别概率是属于box的，每个box对应一个类别概率，而不是由grid cell决定，因此这边每个box对应25个预测值（5个坐标加20个类别值），而在YOLOv1中一个grid cell的两个box的20个类别值是一样的**）。另外作者还提到将最后一个3*3*512的卷积层和倒数第二个卷积层相连。最后作者在检测数据集上fine tune这个预训练模型160个epoch，学习率采用0.001，并且在第60和90epoch的时候将学习率除以10，weight decay采用0.0005。

Stronger

这部分看得不是很懂，简单介绍，欢迎指正。带标注的检测数据集量比较少，而带标注的分类数据集量比较大，因此YOLO9000主要通过结合分类和检测数据集使得训练得到的检测模型可以检测约9000类物体，那么这是怎么做到的呢？一方面要构造数据集，另一方面要解决模型训练问题，前者采用WordTree解决，后者采用Joint classification and detection。

先来介绍下怎么做据集的融合，我们知道分类和检测数据集存在较大差别：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”.

ImageNet的标签来自WordNet，这种结构不是tree，举个例子，狗这个label即属于犬科也属于家畜，这就很复杂了，完全是个图结构，因此作者采用WordTree来解决分类和检测数据集的标签问题，也就是说在这个树里面任意一个节点只能属于唯一一个节点（跟WordNet是有区别的）。

这样的话，在WordTree的某个节点上就可以计算该节点的一些条件概率值，比如在terrier这个节点，可以得到如下条件概率值：

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

如果要预测一个节点的概率，可以根据WordTree将该节点到根节点的条件概率依次相乘得到，如下式：

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

另外：

$$Pr(\text{physical object}) = 1$$

按照这种方式，就可以得到Figure5，也就是在Imagenet1k数据集上采用WordTree方式得到的类别（主要是增加了369个中间节点，比如dog等）。然后作者分别在Figure5的两个数据集上用相同训练方法训练Darknet-19模型，最后在Imagenet1k数据集上的top-1 accuracy为72.9%，top-5 accuracy为91.2%；在WordTree1k数据集上的top-1 accuracy为71.9%，top-5 accuracy为90.4%。在WordTree1k数据集上的准确率要稍低一点，主要是因为那些新的类别的影响。

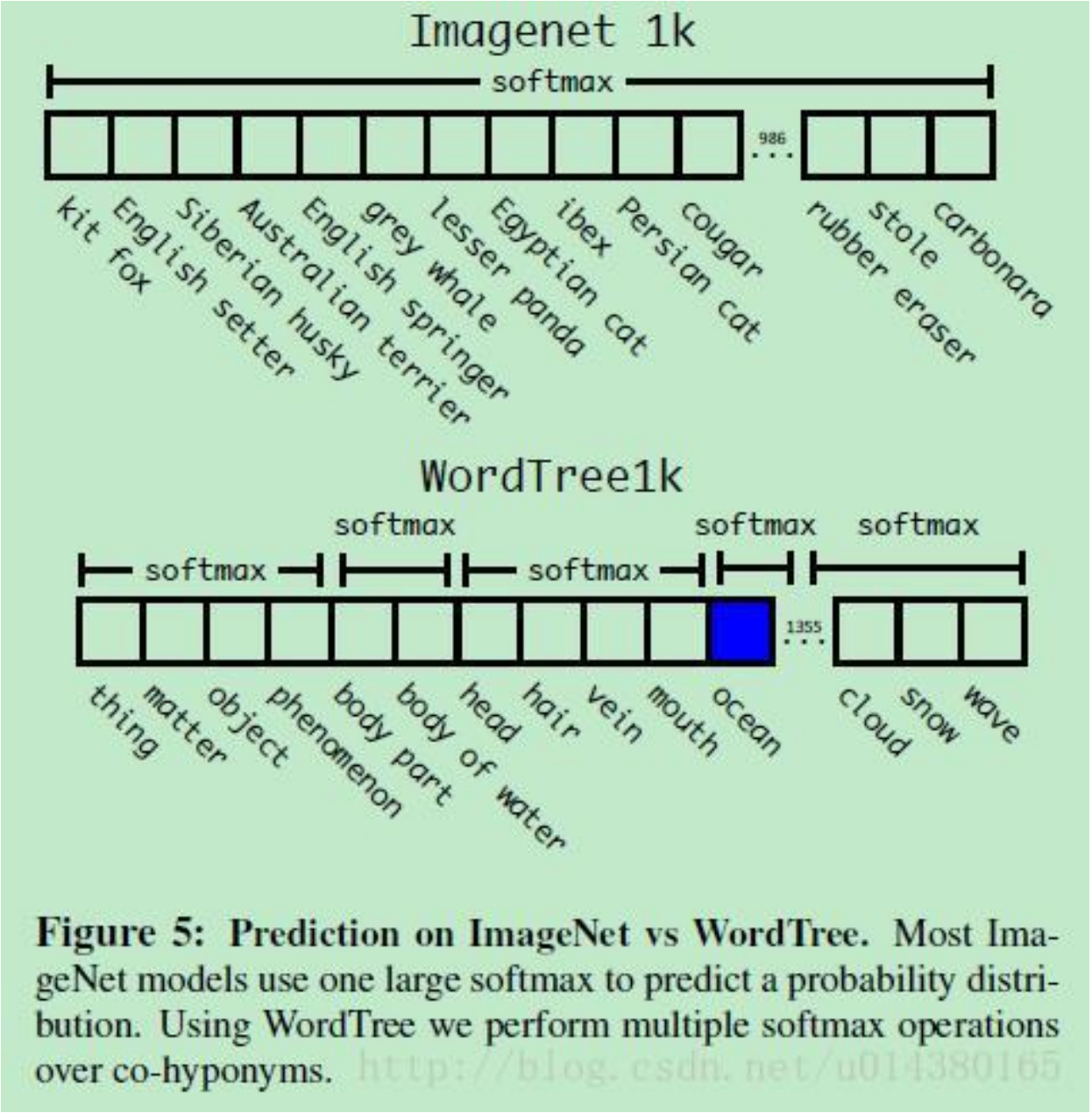

Figure 5: Prediction on ImageNet vs WordTree. Most ImageNet models use one large softmax to predict a probability distribution. Using WordTree we perform multiple softmax operations over co-hyponyms.

介绍完了WordTree的原理，就可以用WordTree来融合分类和检测的数据集了。Figure6是COCO和ImageNet数据集以及WordTree的示意图。在WordTree中用颜色区分了COCO数据集和ImageNet数据集的label节点。

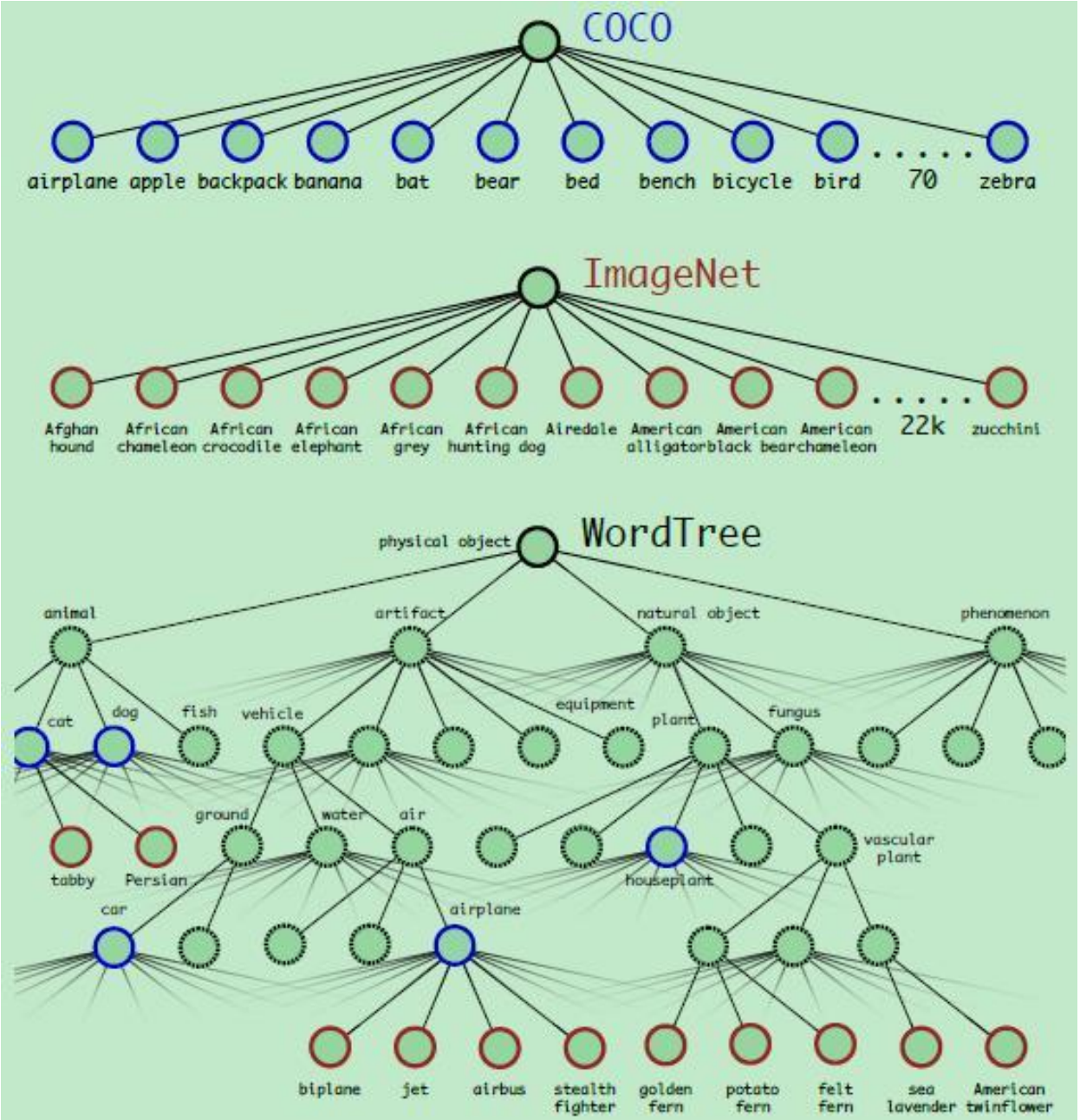

Figure 6: Combining datasets using WordTree hierarchy. Using the WordNet concept graph we build a hierarchical tree of visual concepts. Then we can merge datasets together by mapping the classes in the dataset to synsets in the tree. This is a simplified view of WordTree for illustration purposes.

前面介绍完了如何融合分类和检测的数据集，接下来就是训练的问题了，文中采用的是Joint classification and detection，什么意思呢？原文表述如下：

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.

这个joint classification and detection的直观解释就是：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.

另外YOLO9000的主网络基本和YOLOv2类似，只不过每个grid cell只采用3个box prior。