深度学习框架下的目标检测

15352019 曾何萌

目前主流的目标检测算法主要是基于深度学习模型，其可以分成两大类：

1. **two-stage检测算法**：其将检测问题划分为两个阶段，首先产生候选区域（region proposals），然后对候选区域分类（一般还需要对位置精修），这类算法的典型代表是基于region proposal的R-CNN系算法，如R-CNN，Fast R-CNN，Faster R-CNN等。
2. **one-stage检测算法**：其不需要region proposal阶段，直接产生物体的类别概率和位置坐标值，是一种端到端的检测方法，比较典型的算法如YOLO和SSD。目标检测模型的主要性能指标是检测准确度和速度，对于准确度，目标检测要考虑物体的定位准确性，而不单单是分类准确度。一般情况下，two-stage算法在准确度上有优势，而one-stage算法在速度上有优势。不过，随着研究的发展，两类算法都在两个方面做改进 。

   对于**端到端（end-to-end）**的理解：其实就是从一端（输入，原始数据）到另一端（输出，结果）的意思。也就是说，端到端意味着，缩减人工预处理和后续处理，尽可能是模型从原始输出到最终我们想要的输出结构，给模型更多可以根据数据自动调节的空间，增加模型契合度。 对于一个问题，如果分阶段学习的话，第一阶段的最优解不能保证第二阶段的问题达到最优。end-to-end把他们堆在一起来优化，确保最后阶段的解达到最优。

目标检测模型的性能指标：

1. **IoU**：交并比，表示预测框与真实框（ground-truth box）之间的重叠程度，用来衡量定位精度，。
2. **mAp**：各个类别AP的平均值（Mean Average Precision ）。多个类别物体检测中，每一个类别都可以根据recall和precision绘制一条曲线，AP就是该曲线下的面积。
3. **速度**：常用指标相同硬件环境下的FPS。

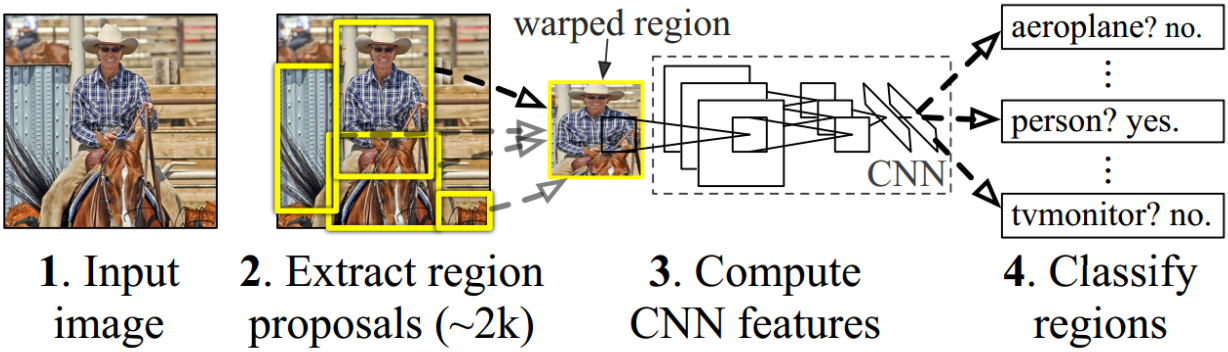
RCNN系列

RCNN系列方法的基本思路都是：

1. 得到候选区域ROI
2. 利用CNN网络进行特征提取和分类器对ROI进行分类预测C。
3. 对有物体的ROI进行位置精修得到最终结果bbox。

也可以先利用CNN提取特征，再提取ROI，最后都是要根据ROI的特征对其进行分类预测。

RCNN



- 检测流程如下：

    1. 输入图像。
    2. 提取候选区域：使用SS算法，该算法有利于直接提取可能是物体的候选区域，而非利用滑动窗口获取候选区域。再将候选区域送入CNN网络前进行预处理，归一化得到的~2K个候选区域，变形为某指定大小（缩放或填充，不同方式对结果影响较大）。
    3. CNN特征提取。（关于参数训练：大样本集下预训练一个CNN分类器，像是迁移学习，特定样本集下对这个分类器的参数进行微调，毕竟特定样本[比如带物体类别及位置标签的图像]的数量有限，不适合直接用来训练多层的RNN，最后把训练好的CNN分类器的参数直接拿过来用）。
    4. 分类器SVM：CNN计算出来的多维特征作为分类器的输入（相比训练cnn时，对对候选框与Ground truth 的IoU阈值的要求较严格），输出为候选区域的对于不同类别物体的评分结果，并舍去得分比较低的候选区域(可能是背景)，对剩下的候选区域进行5，6步处理。
    5. 非极大值抑制NMS：剔除重叠框，保留同类别下局部概率最高的候选框。
    6. 回归器修正得到bbox，提高定位精度。

- Q&A

    1. 为什么候选区域要resize为固定大小?

        CNN中的卷积操作对输入的大下（尺寸）没有硬性要求，输出大小与输入大小有关。但是CNN网络之后是全连接层，全连接层的输入需要是固定大小的，而CNN输出的特征图大小与ROI的大小有关，该特征图会作为全连接层的输入，所以需要把ROI resize为固定大小，才能保证最后的特征图大小固定。

        SPP-net是继RCNN之后提出的另一种目标检测算法，它在CNN与全连接层之间加入了一个SPP层，并且直接以整张图片作为CNN的输入而无需resize,只需做一次CNN前向计算。然后通过空间映射方式计算得到每一个proposal相应的CNN特征，并通过pooling的方式得到不同候选区域的固定维度的特征图 作为全连接层的输入。 对于不同大小的特征图，为什么SPP层能通过pooling得到固定大下的输出呢？因为pooling的参数比如stride,windowSize是动态的，是根据当前的输入尺寸和固定输出尺寸计算出来的，以保证不管输入尺寸是什么，pooling之后都能得到固定的输出尺寸。

Fast-RCNN

和上面介绍的SPP-net的思想一样，是把整张图作为输入，在CNN网络得到特征图上提取候选区域的特征图块，不会重复计算特征 ，具体模型有以下不同：

1. 使用ROI pooling 层而不是SPP层。事实上，ROI Pooling 层是SPP层的简化形式。SPPc层是空间金字塔Pooling层，包括不同的尺度；ROI pooling 层只包含一种尺度，如论文中所述 $7*7$。这样对于ROI pooling层的输入（r,c,h,w），ROI pooling 层首先产生$7*7$个

$r*c*(h/7)*(w/7)$的Block(块)，然后用Max-Pool方式求出每一个Block的最大值，这样ROI pooling层的输出是$r*c*7*7$.

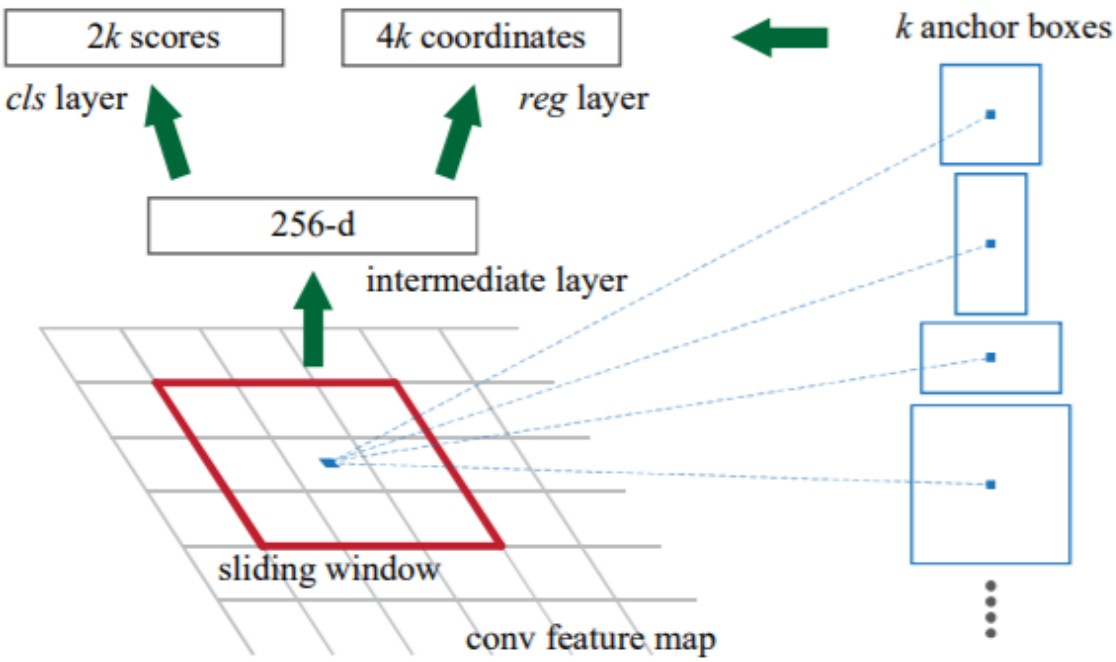
2. 用softMax进行分类预测而不是SVM分类器。SVM计算类别得分（SVM是用于做二分类的，所以需要为不同的类别训练不同的分类器），softMax计算类别概率，而且RCNN在训练CNN网络的时候也是用的softMax，之后去掉softMax层，加入了SVM分类器（该分类器是单独训练的）。所以可能直接用softMax进行分类效果会比较好。

Faster-RCNN

RCNN，SPP-net，Fast-RCNN候选区域都是基于SS算法的，速度比较慢。

Faster-RCNN 在Fast-RCNN的基础上采用RPN网络直接提取 ROI 。RPN是浅层的CNN带有全连接层，用前一个CNN提取的特征作为输入，RPN卷积时（卷积核滑动时），每个窗口位置一般设置k个不同大小或比例的先验框（anchors, default bounding boxes），这意味着每个位置预测k个候选区域（region proposals）， 输出是候选区域坐标与有和无目标得分,选择目标得分高的候选区域 （RPN采用的是二分类，仅区分背景与物体，但是不预测物体的类别）。

- RPN网络结构如下：



conv feature map是前面的CNN网络提取出来的特征图。RPN在卷积操作时，当前滑动窗口（红色框部分）的的中心位置所对应的原始图像的位置（对应到原图应该是一个矩形区域，取其中心点坐标）即为anchor(锚点)，即为得到的候选区域bbox的中心位置，其bbox的宽高可以根据事前确定的k种anchor box的尺寸确定，一个anchor可以计算出k个ROI。

（2018-6-22补充下段：）

上图中对于当前滑动窗口得到的$256d$的向量，可以看作是$1*1*256d$的向量，cls-layer和reg-layer都是用$1*1$的卷积核对$1*1*256d$的输入进行卷积，相当于全连接，输出的2k和4k的结果分别是2k个目标得分和4k个坐标偏移(t_x, t_y, t_w, t_h)，预测输出与k个bbox的坐标的关系如下公式：

$$
\begin{aligned}
t_{\mathrm{x}} &= (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t_{\mathrm{w}} &= \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}), \\
t^*_{\mathrm{x}} &= (x^* - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t^*_{\mathrm{y}} = (y^* - y_{\mathrm{a}})/h_{\mathrm{a}}, \\
t^*_{\mathrm{w}} &= \log(w^*/w_{\mathrm{a}}), \quad t^*_{\mathrm{h}} = \log(h^*/h_{\mathrm{a}}),
\end{aligned}
$$

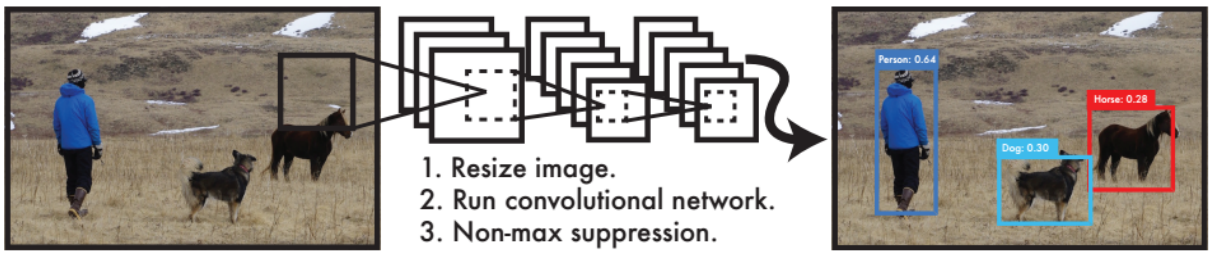
上面式子中的t_x表示RPN预测的坐标偏移，x_a表示anchor在原始图像中预设的坐标，x^*表示物体真实坐标。第三四行公式是训练阶段要用到的真实标签的计算$(t^*_x, t^*_y, t^*_w, t^*_h)$。因此最终RPN的输出坐标要进行一下变换，得到$(x, y, w, h)$。

因为RPN利用上述滑动窗口确定的大量ROI有很多是大面积重叠的，因此，RPN在最终输出ROI前会进行一次NMS（根据cls分数）.

Yolo系列

YoloV1

Yolo是一个可以一次性预测多个box位置和类别的卷积神经网络，能够实现端到端的目标检测和识别，其最大的优势就是速度快，可以做到实时检测。Yolo将图像分成S*S个网格，若某个物体Ground truth的中心位置的坐标落入该某个网格内，该网格就负责检测出这个物体。



- 检测过程如下（共有C类物体）：
  1. 将原始图片尺寸resize为$448 * 448$。
  2. 将图片划分为$S * S$个网格，每个网格预测B个bbox的位置（中心坐标(x, y)和宽高w, h）和置信度，同时每个网格预测C个条件类别概率。最后的输出尺寸为为$S * S * (B * 5 + C)$
  3. 去掉得分较低的bbox，并进行NMS。

- 关于置信度的理解：

  置信度反映是否包含物体以及包含物体情况下位置的准确性,定义为：

  $$Pr(Object) \times IOU^{truth}_{pred}, \text{其中} Pr(Object) \in \{0, 1\}$$

  当然以上只是人为定义的，（以下为个人理解，不知道对不对）通过Yolo网络计算得到的置信度说白了只是一个数字，训练的过程就是不断调整网络参数，使得这个数字不断趋近我们定义的置信度。在真正测试的时候计算出来的这个数字便具有参考意义，因为它代表了该bbox的置信度。

- 损失函数

损失函数可以分为三部分，因为bbox的坐标位置，置信度以及网格的分类结果都会影响模型的最终效果，所以训练时的损失函数需要同时考虑它们三个：

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

bbox坐标位置预测误差

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

bbox置信度预测误差

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

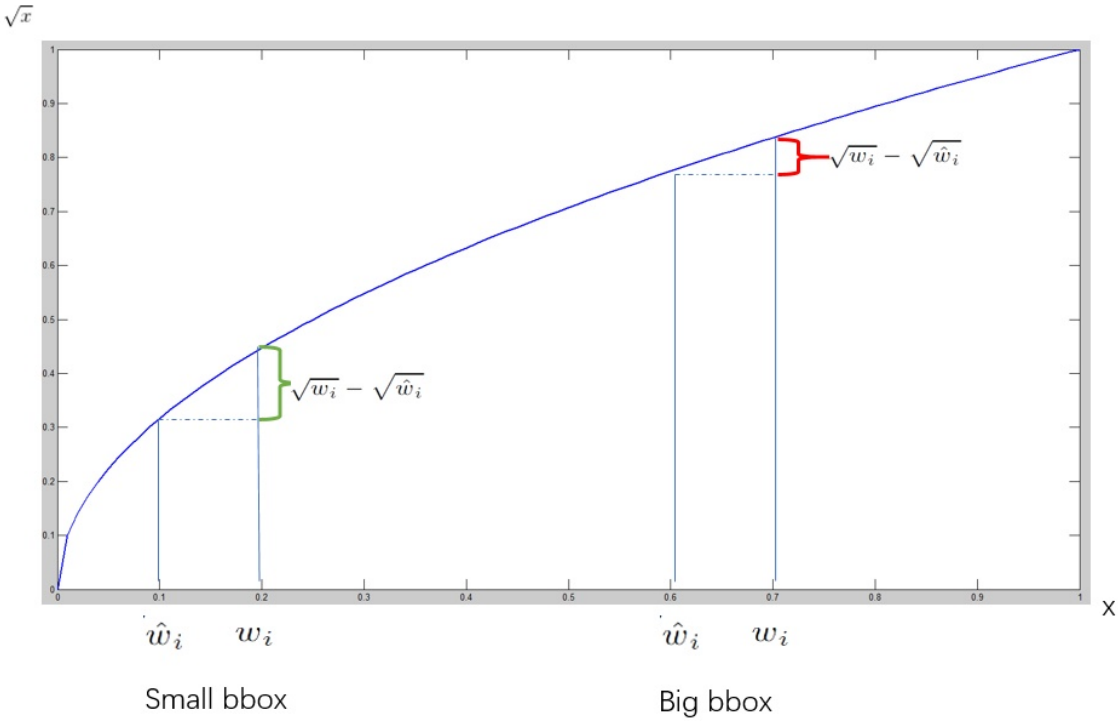
网格分类预测误差

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)$$

1. 不同的误差对结果的影响不同，所以它们前面都有系数，最终误差为所有误差项的加权和。至于权值怎么确定，看实现的人怎么想了。
2. $II_{ij}^{obj} = \{0, 1\}$表示网格i的第j个box有负责预测的物体，若该bbox与物体ground truth的IOU最大，则该bbox负责预测该物体。

   $II_{ij}^{noobj} = \{0, 1\}$表示网格$i$的第$j$个box没有负责预测的物体。

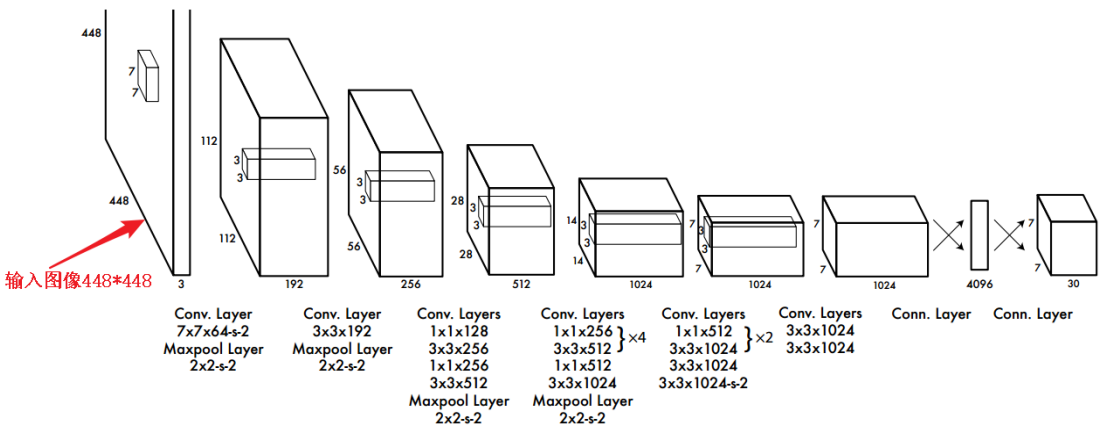
   $II_{i}^{obj} = \{0, 1\}$表示网格i有负责预测的物体，有物体的中心落在该网格内。
3. bbox的宽高误差计算时为什么取平方根？对于相等的误差值，大物体误差对检测的影响应小于小物体误差对检测的影响。这是因为，相同的位置偏差占大物体的比例远小于同等偏差占小物体的比例 。取根号可以缓解这个问题，这与平方根函数的性质有关：



4. 对于负责物体预测的bbox的置信度误差与不负责物体的bbox的置信度误差对最终误差的贡献不一样。我们最终的目的是为了检测出物体，而真正负责预测物体的bbox远少于不负责预测物体的bbox，如果同等看待的话，可能会导致梯度消失但却未能达到目的，所以需要提高前者的置信度误差的贡献，加快模型收敛速度。

- Q&A

1. yolo的网络结构如下图所示（共20个物体类别），在该网络中为什么$7*7*30$的输出分别对应原始图片$(448*448)$划分的$7*7$个网格？甚至从该流程中并没有看到原始图片有被划分为$7*7$的网格。



又要开始吹水了，神经网络的神奇之处在于：在一个网络结构较为理想的前提下，对于给定的数据集（有输入和标签），只要经过足够的训练，该网络便可以拟合我们的目的（对于给定的输入，经过网络可以得到想要的输出）。yolo网络在训练的时候，它的训练集的标签应该就是$7*7$个网格负责预测的物体Ground truth的box位置信息，置信度，该网格的分类结果概率（如果该网格没有负责预测的物体，也有box等信息，但置信度为0）。所以，可以认为最终的yolo模型得到的结果可以对应到原始图片的$7*7$网格中。

YoloV1的一些优点：

1. 检测物体非常快：标准版本的Yolo在Titan X 的 GPU 上能达到45 FPS。更快的Fast YOLO检测速度可以达到155 FPS。而且，Yolo的mAP是之前其他实时物体检测系统的两倍以上。
2. Yolo可以很好的避免背景错误：不像其他物体检测系统使用了sliding window或region proposal，分类器只能得到图像的局部信息。Yolo在训练和测试时都能够看到一整张图像的信息，因此Yolo在检测物体时能很好的利用上下文信息(context)，从而不容易在背景上预测出错误的物体信息。
3. Yolo可以学到物体的泛化特征：当Yolo在自然图像上做训练，在艺术作品上做测试时，Yolo表现的性能比之前的物体检测系统要好很多。因为Yolo可以学习到高度泛化的特征，从而迁移到其他领域。

YoloV1的一些缺点：

1. Yolo的物体检测精度低于其他state-of-the-art（顶级的）的物体检测系统。
2. Yolo容易产生物体的定位错误。
3. 测试图像中，当同一类物体出现不常见的长宽比和其他特殊情况时，表现不好，泛化能力还有待提高。
4. Yolo对小物体的检测效果不好（尤其是密集的小物体，因为一个网格只能预测1个物体）。

YoloV2

YoloV2（better-提高mAp,recalll等指标，faster-训练时加快模型收敛，测试时提升检测速度FPS，stronger-Yolo9000能预测更多类别）提出了新的分类模型DarkNet-19，针对以上YoloV1的缺点进行了改进，有以下几个方面：

	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**

1. 在激活层之后引入**Batch normalization**，且取消了V1中用的dropout。BN本质是对于每个隐层神经元，把逐渐向非线性函数映射后向取值区间极限饱和区靠拢的输入分布强制拉回到均值为0方差为1的比较标准的正态分布，使得非线性变换函数的输入值落入对输入比较敏感的区域，以此避免梯度消失问题。因为梯度一直都能保持比较大的状态，所以很明显对神经网络的参数调整效率比较高，变动大，就是说向损失函数最优值迈动的步子大，也就是收敛地快 。关于BN更详细的解释，请参考Batch normalization
2. YoLo v2受到Faster-RCNN的启发，引入了anchor。它使用了K-Means方法，在大量的训练数据下对anchor box数量和尺寸进行聚类分析，得到合适的boxes。**anchors boxes** 是在yolo层实现的（网络的最后一层）。

   传统的K-means聚类方法使用的是欧氏距离函数来度量样本相似度，也就意味着较大的boxes会比较小的boxes产生更多的error，聚类结果可能会偏离。为此，作者采用的评判标准是IOU得分，这样的话，error就和box的尺度无关了，最终的距离函数为：

$$d(box;centroid) = 1 - IOU(box;centroid)$$

最后作者选定的是$K=5$，即5种anchor box。

3. **细粒度特征**：把浅层特征图连接带深层特征图上（深度叠加，即扩展通道）,作者在构建网络是时候把它成为转移层（**passThrough layer**），类似与Res-net中的skip-connection，使模型有了细粒度特征。

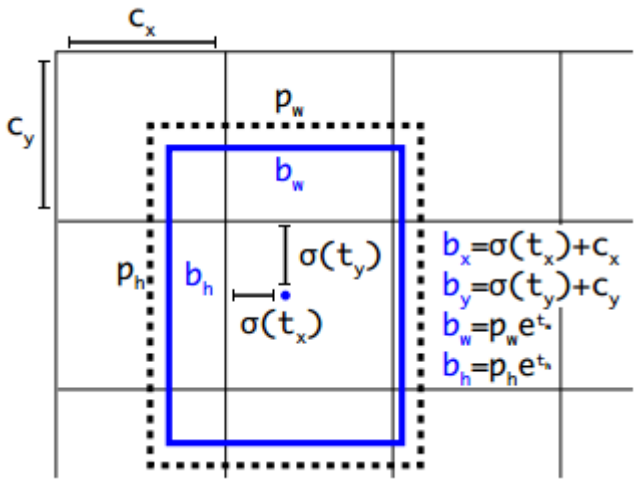
   用ship connection（或者说是 Residual block残差块 ）构建的ResNet，不仅可以在一定程度上保留细粒度特征，而且被证明不会随着网络层数的增加而效果变差。普通的CNN会因为层数的增加性能下降。

4. **Direct location prediction**。 作者引入anchor后，发现一个问题：模型变得不稳定，尤其是训练的早期迭代阶段，主要是因为bbox的中心坐标(x,y), 在Faster-RCNN中，RPN网络预测出的偏移量(t_x,t_y)是偏移量，bbox的中心坐标计算如下：

   $$x = (t_x * w_a) + x_a \ (y同理)$$

   模型不稳定是因为t_x,t_y的取值没有限制，若它们取值的绝对值很大，预测出来的bbox可能相对与anchor 有很大偏移（或许，该ahchors根本就没义务负责该bbox对应的区域，它只负责它附近区域），这会导致训练的时候模型动荡。因此限制t_x,t_y的取值更合理一点，比如$(-1,1)$，使得anchor值负责预测它周围临近物体。

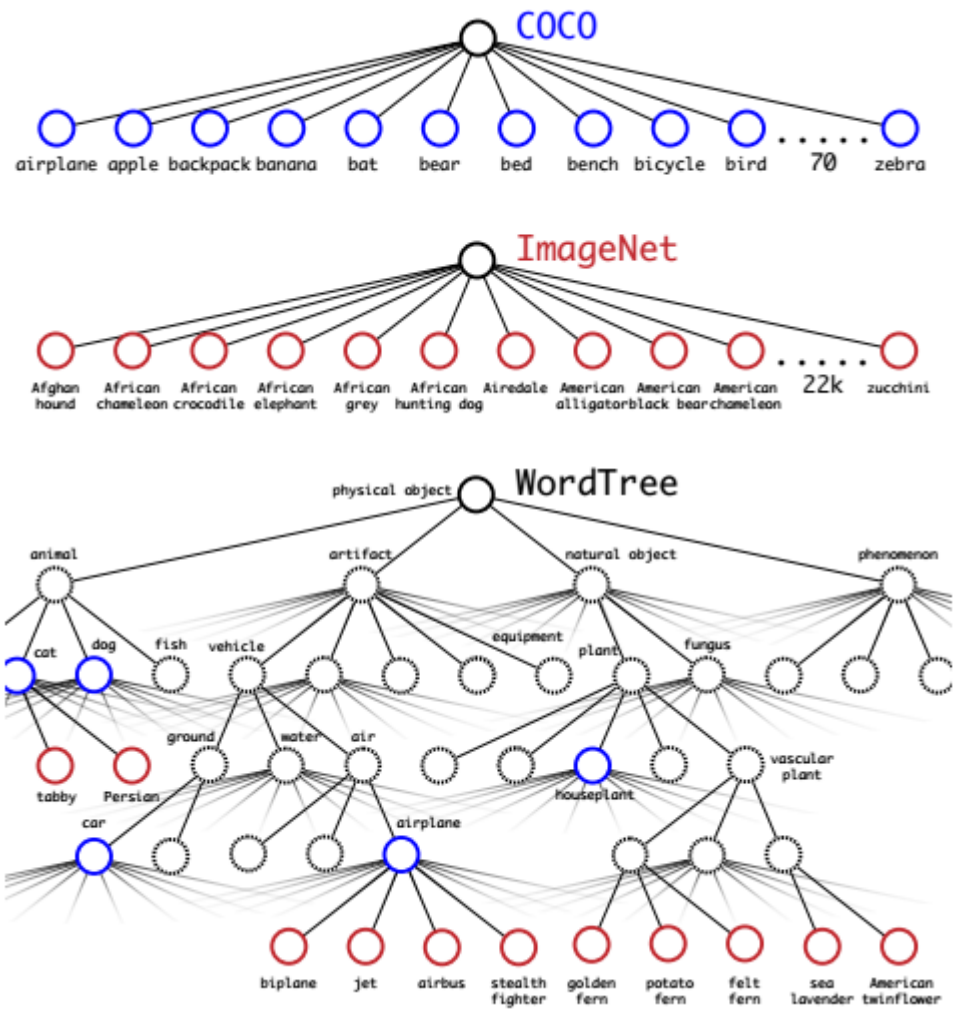
   作者最后没有用相对于anchor的偏移量，而是相对于当前网格左上角的坐标偏移，和宽高相对于anchor box宽高的偏移，并且使用logistic回归函数将偏移限制在$(0,1)$之间，之后根据预测出的偏移量计算bbox的公式如下：



5. 还有一些小的改进，比如提高输入分辨率并且去掉后面的一个池化层以确保输出的卷积特征图有更高的分辨率 (**hig-res**)、多尺度预测（在训练时会微调网络的输入尺寸以适应不同尺度的输入）。

作者还提出了一种**联合训练**机制，实现了在分类和检测数据集上的联合训练，基于YoloV2的网络结构训练得到新的模型Yolo-9000，可以检测9000多个目标类别。对于检测数据集，可以用来学习预测物体的边界框、置信度以及为物体分类，而对于分类数据集可以仅用来学习分类，但是其可以大大扩充模型所能检测的物体种类 。但是不同的数据集的类别并不是完全互斥的，比如检测数据集只有常用目标和常规标签，如“dog”或“boat”；分类数据集具有更宽和更深的标签范围，有各种品种的狗和船。因此作者提出了基于wordTree的分层分类 ，将多个不同的数据集组合在一起。YOLO9000同时使用COCO和ImageNet数据集训练模型，使用COCO中的检测数据学习找到图像

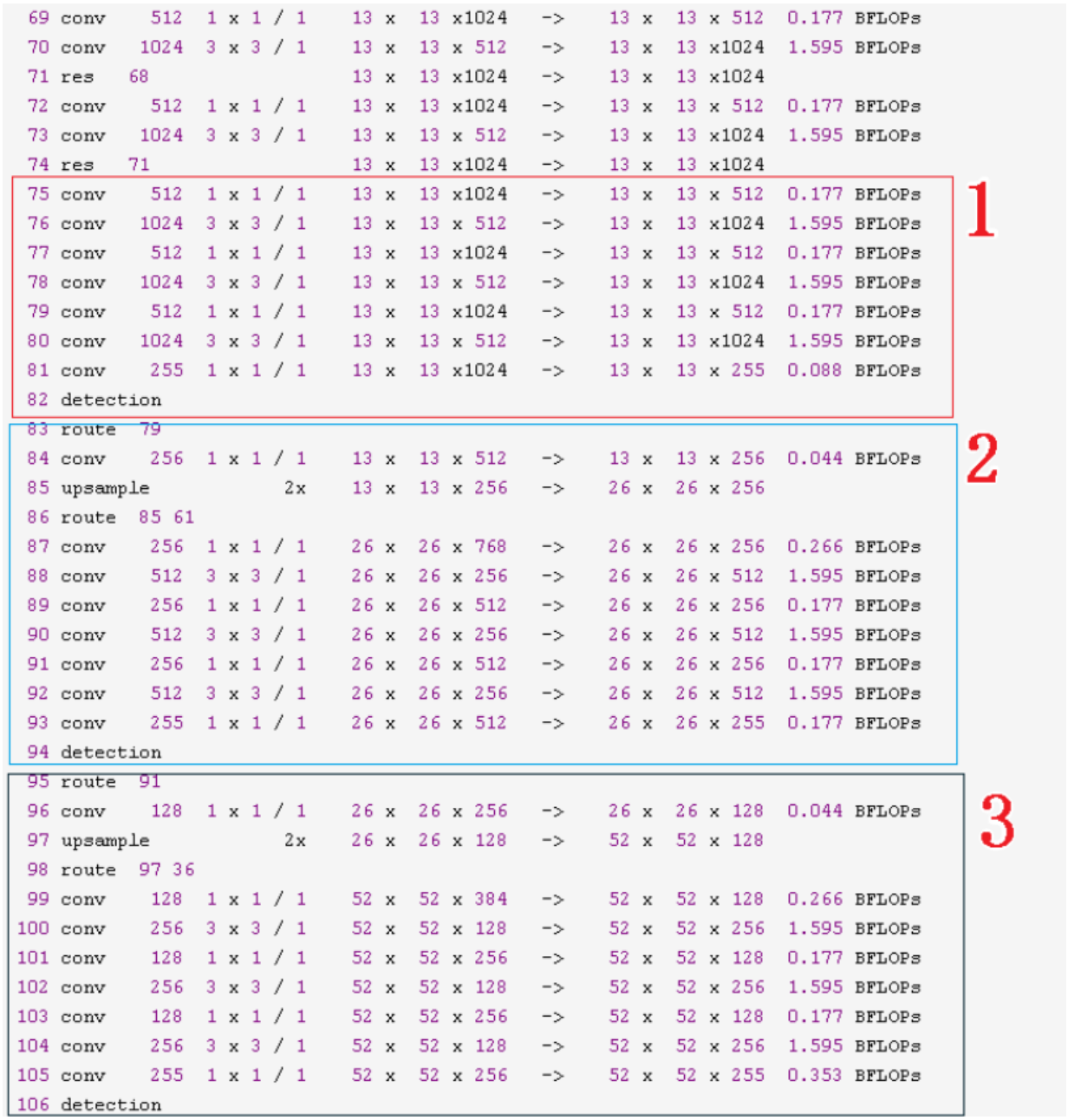
中的目标，并使用ImageNet中的数据学习分类各种各样的这些目标。训练时，当看到检测图片是像YoloV2一样反向传播loss,当看到分类图片时，仅反向传播分类错误（至于分类错误怎么算，应该是与树结构有关，真的是看不下去了。。）。



YoloV3

YoloV3的精度虽然没有明显提高，但光从速度上就碾压其他目标检测算法。比之前的模型复杂了不少，可以通过改变模型结构的大小来权衡速度与精度。改进如下：

1. 使用logistic分类器代替softmax。因为softmax分类器的原理默认类别之间是互斥的，即一个bbox只能预测一个类别，而为每个类别使用独立的logistic分类器得到的是属于该类别的概率，一个bbox可以预测多个类别。
2. 多尺度预测（刚开始看论文不太懂，后来看了作者实现的YoloV3的网络配置信息才明白他说的是啥意思）,3种尺度：



1. 对原网络的提取出来的特征，增加几个卷积层得到第一个特征图普做一次detect;
2. 在这个基础上，从后往前数倒数第三个卷积层的输出进行一次卷积并进行x2的上采样，将上采样结果与第43个卷积层进行~~element-wise合并（相加）~~是concat，不是相加，然后在经过几个卷积层之后，做第二次detect。这种方法使我们能够从上采样的特征和早期特征映射的细粒度信息中获得更有意义的语义信息 。
3. 继续从后向前获得倒数第3个卷积层的输出，进行一次卷积一次x2上采样，将上采样特征与第26个卷积特征连接，经过7个卷积得到第三个特征图谱，在这个特征图谱上做第三次detect。

多尺度预测对于小目标的检测效果提升是比较明显的。YoloV3中每个网格预测3个bounding box，看起来比YoloV2中每个网格预测5个bbox要少，其实不是这样的！因为YoloV3采用了多个scale的特征融合，从上面的网络结构中可以看到，对同一张图像进行了三次划分，分别划分成$13*13, 26*26, 52*52$的网格，所以bbox的数量要比之前多很多，以输入图像为416*416为例：$（13*13+26*26+52*52）*3$和$13*13*5$相比多得多。

3. 最终网络是Dark-net53（但这里应该不是53个卷积层，因为在多尺度预测在53层之后又增加了十几个卷积层），在Dark-net19的基础了增加了一些卷积层与short cut层（skip onnection）。Darknet-53可与state-of-the-art（顶级）分类器相媲美，但浮点运算更少，速度更快，可以实现每秒最高的测量浮点运算。这意味着网络结构可以更好地利用GPU，从而使其评估效率更高，速度更快。

Reference:

1. 以上算法的论文
2. 综述|基于深度学习的目标检测(一)
3. 图解yolo
4. Batch normalization