本周主要对 faster-rcnn 和 SSD 两篇论文中提到的方法和名次进行了一定理解和消化。也花了一定时间在看代码。。然而代码的模块化将功能分的特别细，看不了多久就一片混乱，折腾中完全无法理解其思想。。

但是至少有一点，从代码的 train 中是可以看出代码的大致各个部分的大框架的。今天想到，不妨把代码中的各个部分分别全部抠出来，单独运行，先只提取其骨架，暂时舍弃掉其他提高程序兼容性的，乱七八糟的细节，理解其基本实现思想，这样理解应该会更有条理。

正负样本：

Proposal 挑选出来的候选框和 ground truth 的 IOU 大于某个阈值就把这个候选框是物体，标记为正样本，否则就认为是背景（负样本）。

Faster rcnn 的 RPN 中，其目的是为了从 20000+个 anchors 中选出 256 个 anchors 进行分类和位置回归 **对 RPN 网络进行训练**！，首先将每一个 ground truth 选择和它的 IOU 最高的一个 anchor 作为正样本，剩下的 anchors 中任意选择重叠度超过 0.7 的 anchors 作为正样本，正样本数目不超过 128 个。最后随机选择和 ground truth 重叠度小于 0.3 的 anchor 作为负样本，正负样本总数为 256。

RPN 给后续的网络（RoiHead）提供 Rois 作为训练样本，RPN 会产生约 2000 个 ROIs，但是只会选择 128 个用于训练，正负样本的比例为 1:3，正样本的 IOU>0.5,负样本的 IOU<0.1

SSD 则保持正负样本数目在 1:3 左右

难例挖掘：

经过反复理解认为难例挖掘这一操作简单来说，是找出所有 prior boxes 中错的最狠的 boxes（也就是 loss 最大）的一批 boxes，然后看它们原本是什么样本，在新的训练中就用这些错的最狠的样本重复再次进行训练（有点错题集的味道。）。比如，找出原本的正样本（IOU>0.7）中在 loss 最大的 boxes 中依旧在的，则留下来继续训练；找出原本的负样本中在 loss 最大的 boxes 中依旧在的，则留下来继续训练。

参考：https://www.cnblogs.com/xuanyuyt/p/7447111.html

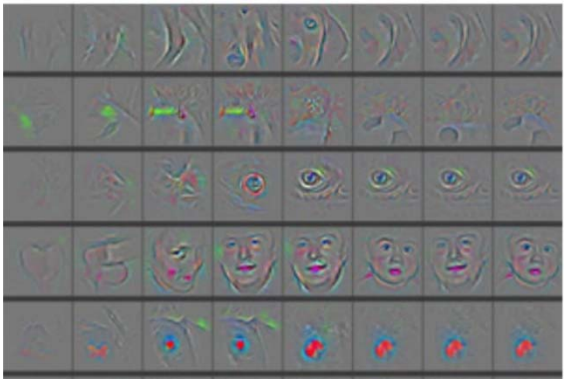
当然，认为参考的原文中说法有一定错误：

如果这P个候选正样本里有a个box不在这M个prior box里，将这a个box从候选正样本集中踢出去。如果这$8732-P$个候选负样本集中包含的$8732-P$有$M-a$个在这M个prior box，则将这$M-a$个候选负样本作为负样本。SSD算法中通过这种方式

fine-tuning：

finetune 就是直接从别人已经训练好的网络上拷贝参数,然后针对自己的数据训练新的模型。这时候需要比较小的 learning_rate，因为要在不破坏原有模型的情况下 fit 自己的数据

在大数据集上进行 pretrain 的目的之一是为了获得丰富、一般化的底层特征，换言之就是学到丰富的“基础几何形状”。有了这些丰富的基础几何形状，等过渡到小数据集上 finetune 的时候，就可以通过它们组合出上层具有强判别力的特征。

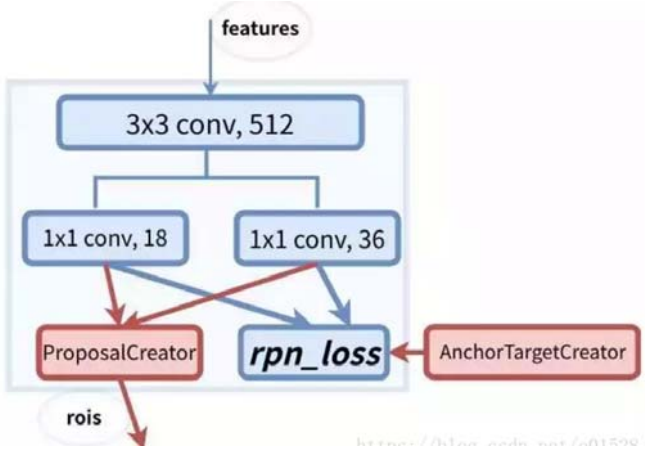


Faster rcnn 使用了全卷积的 RPN 网络预测目标物体的边界和分数。

Anchor 的总个数的计算方法：

3 种 scale 和 3 种 aspect ratio 共有 9 种，然后把这 9 不同尺寸的 anchor 分别在 feature map 的每一个像素点进行应用，就一共得到 width*height*9 个 anchor。

Anchors 的生成：给定最小的框的大小，默认为 16，以及所选择的待求大小和宽高比，其基本思路是先根据最小框的大小和宽高比确定最小的框的中心位置和对应于每个宽高比的尺寸得到一组 baseanchor，然后将其对应的宽和高对应乘上 scale 系数，根据所有 anchor 中心相同，生成对应的一系列 anchors



RPN 网络在 Feature map 上的每一个点分别应用 9 个 anchor 时，对于 score 网络部分可以得出 2*9 个 score，这是通过卷积来实现输出 18 个 channel，显然如果不进行训练，这个卷积网络不可能得出每一个点的得分，所以这一部分的训练是通过 AnchorTargetCreator 对 RPN 网络部分进行训练。提高 RPN 网络的 proposal 能力。

为了将物体分类，需要考虑物体在图中的位置以及这个物体的类别，所以有两个 loss，也就是分为定位和分类两个过程。

Faster rcnn 中使用了现有的 ImageNet 网络对图片的特征进行提取，其中的特征是 conv 网络中的一部分，根据 conv 网络的特点，conv 到达一定的深度后是将基础特征进行拼合了形成了宏观上的特征，如人脸。所以在如果将 ImageNet 从中间截断，就可以利用现有的 ImageNet 直接提取出图像的特征为分类和选定所用。

Faster rcnn 在得到特征图之后直接分别使用两个网络分别得出 anchor 的分类和定位结果。

Faster rcnn 在最后得到的 feature map 上考虑不同的尺度，但是 ssd 是在多个 feature map 上考虑多个尺度。可以认为这导致了 ssd 比 faster rcnn 准确率要高。

Faster RCNN Loss：

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*).$$

pi 和 ti 分别为预测的是否是物体和预测的 bounding box 的 x,y,w,h。

pi*和 ti*分别为 ground truth 的是否是物体和 ground truth 的 x,y,w,h。所以如果 ground truth 的区域如果不是 object，那么不关心其 bounding box 的 loss

$$L_{cls}(p_i,p_i^*) = -\log\left[p_i^* p_i + (1-p_i^*)(1-p_i)\right]$$

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

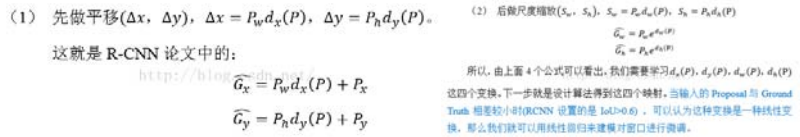
in which

Robust: $$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$ 这是 loss 中 Lreg 的 Robust 计算方法

位置的回归和精修：

Bounding-box regression 只对 proposal 的 x,y,w,h 进行微调，比如只对和 ground truth 很接近的地方，但是没有满足 IoU 条件，进行回归操作；因为如果 proposal 和 ground truth 距离远就会变成复杂非线性问题。。

而回归模型的建立中，考察的是坐标上的平移量和宽高上的尺度缩放，所以在程序中有 bbox_targets 作为该模型的回归目标，根据该模型，容易理解在默认状态下，最理想的状态是 dx(P),dy(P),dw(P),dh(P)这四个值均为 0。


（1） 先做平移$(\Delta x,\ \Delta y)$，$\Delta x = P_w d_x(P)$，$\Delta y = P_h d_y(P)$。

这就是 R-CNN 论文中的：

$$\widehat{G_x} = P_w d_x(P) + P_x$$

$$\widehat{G_y} = P_h d_y(P) + P_y$$

（2） 后做尺度缩放$(S_w,\ S_h)$，$S_w = P_w d_w(P)$，$S_h = P_h d_h(P)$

$$\widehat{G_w} = P_w e^{d_w(P)}$$

$$\widehat{G_h} = P_h e^{d_h(P)}$$

所以，由上面 4 个公式可以看出，我们需要学习$d_x(P)$，$d_y(P)$，$d_w(P)$，$d_h(P)$这四个变换。下一步就是设计算法得到这四个映射。当输入的 Proposal 与 Ground Truth 相差较小时(RCNN 设置的是 IoU>0.6)，可以认为这种变换是一种线性变换，那么我们就可以用线性回归来建模对窗口进行微调。

上式在论文中的写法为：其中，x，xa，w， w* 分别是预测输出的中心坐标 x，anchor 的位置 x 坐标，预测输出的宽 w，以及真实的（ground truth）w*。容易理解在这种情况下，我们是将预测输出的中心坐标和长宽向 ground truth 进行回归

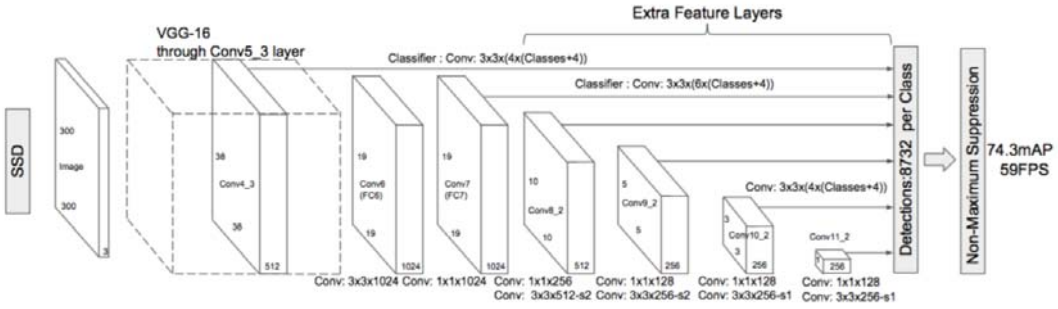
$$t_{\text{x}} = (x - x_{\text{a}})/w_{\text{a}}, \quad t_{\text{y}} = (y - y_{\text{a}})/h_{\text{a}},$$
$$t_{\text{w}} = \log(w/w_{\text{a}}), \quad t_{\text{h}} = \log(h/h_{\text{a}}),$$
$$t^*_{\text{x}} = (x^* - x_{\text{a}})/w_{\text{a}}, \quad t^*_{\text{y}} = (y^* - y_{\text{a}})/h_{\text{a}},$$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

在代码的实现中，注意到上面这一步回归是直接做到了 conv2d 中，conv2d 输出的 4*num_anchors 分别是 tx,ty,tw,th,利用上式反解得到预测的坐标。

VGG16 网络结构:

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

SSD：



38*38*4+19*19*6+10*10*6+5*5*6+3*3*4+4*1*1=8732

SSD 网络中，在所有 feature map 上总共得到 8732 个 anchor 进行计算，其来源是在 6 个 feature map 上分别对每一个点应用图中所标注的 anchor 个数，可以看出，SSD 网络需要处理的总 anchors 比 faster rcnn 要少很多。同时 SSD 没有了 proposal 过程，这使得 SSD 网络比 faster RCNN 要快很多。

训练过程中，prior box 是指 default box 实际在图中应用的时候，根据图像的不同位置而将 default box 进行了一定的裁剪后得到的实际 box，比如在边界处 default box 可能有超出的部分，这显然是需要进行处理得到实际的 prior box。

SSD Loss：

$$L(x,c,l,g)=\frac{1}{N}(L_{conf}(x,c)+\alpha L_{loc}(x,l,g))$$

The localization loss:

$$L_{loc}(x,l,g)=\sum_{i\in Pos}^{N}\sum_{m\in\{cx,cy,w,h\}}x_{ij}^{k}\text{smooth}_{\text{L1}}(l_i^m-\hat{g}_j^m)$$

$$\hat{g}_j^{cx}=(g_j^{cx}-d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy}=(g_j^{cy}-d_i^{cy})/d_i^h$$

$$\hat{g}_j^{w}=\log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^{h}=\log\left(\frac{g_j^h}{d_i^h}\right)$$

$x_{ij}^{p}=\{1,0\}$代表的是正负样本，正样本为 1，负样本为 0，l 则是预测的 bounding box parameter，g 为 ground truth parameter。以 i 为下标的为 default box 的参数，以 j 为下标的则是 ground truth 的参数，其 smoothL1 的计算与 faster rcnn 类似。

The confidence loss:

$$L_{conf}(x,c)=-\sum_{i\in Pos}^{N}x_{ij}^{p}log(\hat{c}_i^p)-\sum_{i\in Neg}log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p=\frac{\exp(c_i^p)}{\sum_p\exp(c_i^p)}$$

其中，c 是网络预测的 c 个类别的得分。经过 softmax 之后使用 log loss 即可得到 confidence loss。