from url: https://github.com/ginobilinie/pytorch-retinanet (里面有我的注释，这样更好的理解原文)
from url: https://arxiv.org/abs/1708.02002
原来经常提到的赫赫有名retinanet便是focal loss for dense object detection。retinanet应该是fpn基础上发展起来的。这里我们重点不是讲述focal loss，而是讲retinanet的实现细节。这里我以pytorch为例。

我们要明白retinanet基本的知识。retinanet是在多resolution上去检测的，而且是有名的anchor-based one-stage detection method。然后，我们可以通过了解训练阶段和测试阶段的细节来理解retinanet。

首先，我们要理解这个retinanet的结构到底是啥?粗的看，就是unet拿来做detection。以前是unet的最后一层输出做bbox regression和object classification，或者如ssd那样多个resolution (scale)的concate到一起，然后在从这个concatenate的feature map里做bbox regression和object classification。而在retinanet里(不知道fpn是不是也这样)，是直接在不同resolution上做bbox regression和object classification，得到bboxes知乎，然后concate到一起，去和annotation的结果做supervised training。

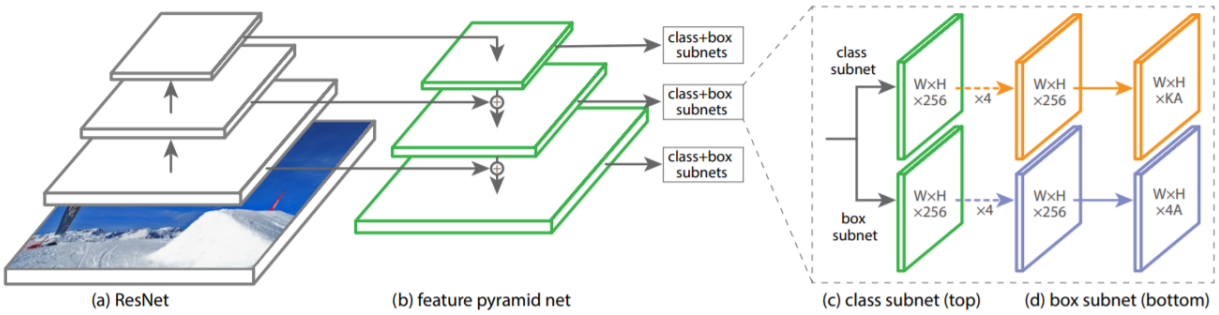

Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [20] backbone on top of a feedforward ResNet architecture [16] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN [20] while running at faster speeds.

其次，在训练的阶段，在上面提到的每个resolution上，我们regress出H*W*num_anchors (9) (这里可能会有个stride因素，如果不那么稠密的话，这样就是H/stride*W/stride*num_anchors)个anchor boxes，作为prediction的location(x1,y1,x2,y2)，但如果就拿这个直接去和annotation的bbox(毕竟annotatino的bbox肯定只有几个，而不会是那么稠密的)去做mse(或者mae)的话，那么annotation里的bbox究

竟是跟哪个predicted bbox做误差呢，又或者说跟几个predicted bboxes做误差呢？而且具体实现上这么来也不好做，毕竟不好选prediction对应的bboxes。
为此，作者发展出anchor boxes的概念，**其实anchor boxes的初衷是为了把annotation里的bbox映射到跟predicted bboxes一样稠密的空间里去，从而好去计算回归和分类loss。**想明白这点之后，其他就迎刃而解。
anchor boxe的个数: 为此，anchor boxes的个数肯定是H*W*num_anchors (注意可能有stride因素)，因为要与predicted boxes保持一致嘛。
如何将annotation里的bbox映射到anchor boxes空间里去: 而怎么将annotation里的bboxes映射到anchor boxes空间里来呢。就是将annotation bboxes和anchor boxes全部两两做IoU。然后选最大IoU，这样就根据对应的index(可以称为positive_index)，将每个annotation里的bbox映射到交叉率最大的那个anchor boxes里去。当然，为了保证映射质量，也得做取舍，比如, iou<0.4的，全部舍去，不做映射了(这些和前面所有不是最大值的anchor boxes设置为-1)。iou>0.5的，这些被认为是比较好的映射(记为1)。而0.4<iou<0.5，记作0。
计算分类loss: 注意这些-1, 0, 1之类的，对回归其实没啥影响，对分类loss计算有直接的影响，也就是说-1的区域，直接不计算loss。注意，这里的loss是bce loss，也就是说是到底是不是object(即object还是bg)的loss。
计算回归loss: 而对于regression loss计算，这样映射之后，就很明了，因为前面我们也说了，将annotation里的bbox映射到anchor boxes空间时，记下了positive_index，由于我们的prediction的bboxes空间也是相同的大小和序，这里我们直接找对应的第positive_index个predicted bbox就行，然后用这个predicted bbox和映射后的annotation bbox(也就是在anchor boxes空间里的第positive_index个)做mae或者是mse误差。这样就一目了然了。
由此，我们也可以看出其实retinanet是anchor-based的one-stage detection method。
最后，我们看下测试阶段。测试阶段主要是怎么得出最终的bbox和其对应分类呢?因为我们的regress和classify分支预测出的是H*W*9(可能有stride因素)个predicted bboxes，那么我们不可能说直接输出这么多个bboxes吧。
先将predicted bboxes的x1,y1,x2,y2位置确定，并且超出image范围的，clamp到边界。
然后再将这些predicted bboxes对应的classification score取max(对应的就是每个类分别会有一个max值)，用这个score做置信度。如果max_score<0.05的类，那这类应该是不会存在在图像上，直接取消处理。并根据这个max_scores来选取对应的predicted bbox(细节不表)。这样做出来，其实还是有可能一个objec它对应多个predicted bboxes，这时候nms就出来了，这样便得到了最终的bbox和类别。