简介

利用一个网络找到center point，然后回归其他需要的东西。
用了三个backbone去测试COCO bbox ap：

1. Res18 + upConv AP 28.1% 142fps
2. DLA-32 37.4% 52fps
3. Hourglass-104 + multiscale test 45.1% AP 1.4FPS

可以发现其中第二个和第三个backbone其实是专门针对keypoint detection做的网络

对比

- 主要跟CornetNet以及ExtremeNet相同点是用的基础网络基本都是kp detection network
- 不同在于不需要后处理,比如CornerNet的group操作,所以速度会快一些
- 每个object只有一个positive sample,不需要anchor
- 输出层的feature map比一般的object dection大很多,centerNet output stride 4,而rcnn最后一般是16

细节

1. 输出是keypoint heatmap, $[0,1]^{\frac{W}{R}\times\frac{H}{R}\times C}$ 其中R代表最后的stride大小,文章里最后取4,这跟hourglass这些文章有关系。
2. 预测值为0到1之间，讲ground truth p先映射到feature map上，然后以这个位置为中心生成一个高斯的范围热图 $Y_{xyc}=\exp\left(-\frac{(x-\tilde{p}_x)^2+(y-\tilde{p}_y)^2}{2\sigma_p^2}\right)$，loss function用的是focal loss的思路，正样本和负样本的gmma值不太一样，这里的思路基本都是借鉴文献[30]

$$L_k=\frac{-1}{N}\sum_{xyc}\begin{cases}(1-\hat{Y}_{xyc})^{\alpha}\log(\hat{Y}_{xyc}) & \text{if } Y_{xyc}=1\\ (1-Y_{xyc})^{\beta}(\hat{Y}_{xyc})^{\alpha} & \text{otherwise}\\ \log(1-\hat{Y}_{xyc}) & \end{cases} \quad (1)$$

3. 一个简单但值得思考的模块，为了恢复中心点映射到feature map带来的量化error，对每个加了一个local offset $\hat{O}$ 生成的loss，all classes C share the same offset prediction，这里不太懂，如果每个feature map上都是置信度不同的center point，那么偏置项的生成是针对所有的offsets的，还是只是ground truth，如果是ground truth，测试时怎么操作

$$L_{off}=\frac{1}{N}\sum_{p}\left|\hat{O}_{\tilde{p}}-\left(\frac{p}{R}-\tilde{p}\right)\right|. \quad (2)$$

object detection

训练

接下是如何生成bbox的预测： 网络在预测中心的同时预测bbox，其中bbox的预测目标是bbox的size，也就是长和宽，使用简单的l1 loss。这里不用一个单独的branch去做回归，而是给输出的channel加上两个维度。

$$L_{size} = \frac{1}{N}\sum_{k=1}^{N}\left|\hat{S}_{p_k} - s_k\right|. \qquad (3)$$

所以最后的loss是三个loss的和，但是两个l1 loss的尺度不太一样，作者并没有直接在coord里进行归一化而是认为设定了两个权重系数如下所示：

$$L_{det} = L_k + \lambda_{size}L_{size} + \lambda_{off}L_{off}. \qquad (4)$$

分别取0.1和1
最后channel变为C + 4，这里多出来的两个维度是上面的local offset和这里要预测的object size。network结构在基础的backbone上均有一些调整。

测试

测试的时候找到每个catgroy的heatmap peak，peak的定义是其值大于或等于其八个邻居的位置（其实就是局部最大点），提取前100个peak，找到peak的位置，就可以知道coarse location, local offset和bbox size，最后的预测结果为：

$$(\hat{x}_i + \delta\hat{x}_i - \hat{w}_i/2,\ \ \hat{y}_i + \delta\hat{y}_i - \hat{h}_i/2,$$
$$\hat{x}_i + \delta\hat{x}_i + \hat{w}_i/2,\ \ \hat{y}_i + \delta\hat{y}_i + \hat{h}_i/2),$$

这里不需要nms因为找peak extraction就足够了，其实等价于一个3x3 max pooling。

backbone和实验

三个backbone，这里只说下resnet，参照文献[55]，给resnet加了三个上采样的up-conv层使得其最后的stride变为4，这三个channel分别为256,128,64，然会在每个up-conv层前加相同channel的3x3 sep conv。其中up-conv层的kernel采用bilinear inerpolation初始化。

实验

作者用的是titan Xp去测试。
和RetinaNet采用相同的resNet-101 backbone，相同准确率达到了两倍速，其中DLA-34是比较均衡的。问题是他咋不跟800分辨率的retinaNet比呢（40.8）,而且他这个还没有用到fpn，只是在一个scale上输出的。