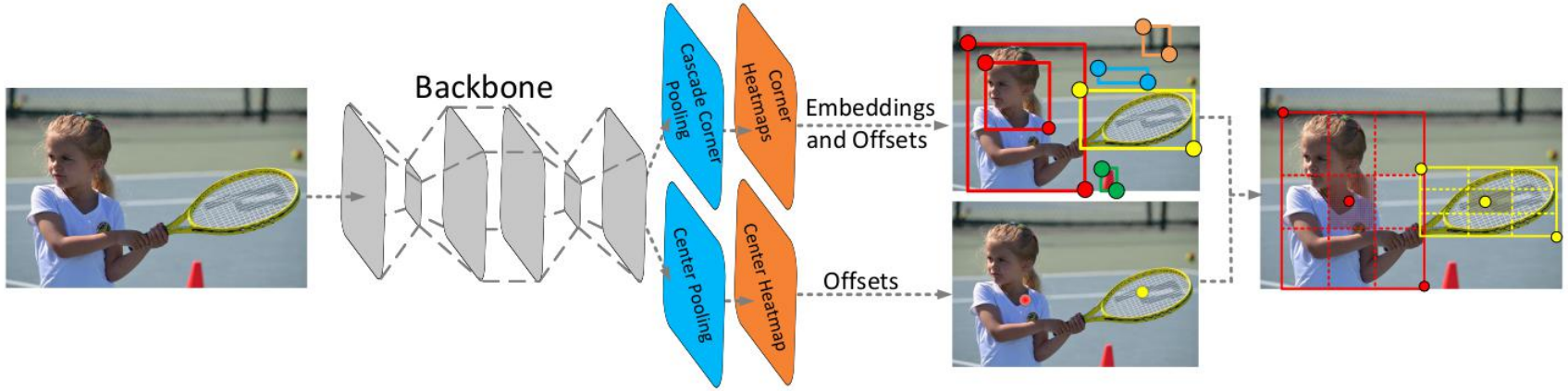
# CenterNet解读



# anchor-free和anchor-based

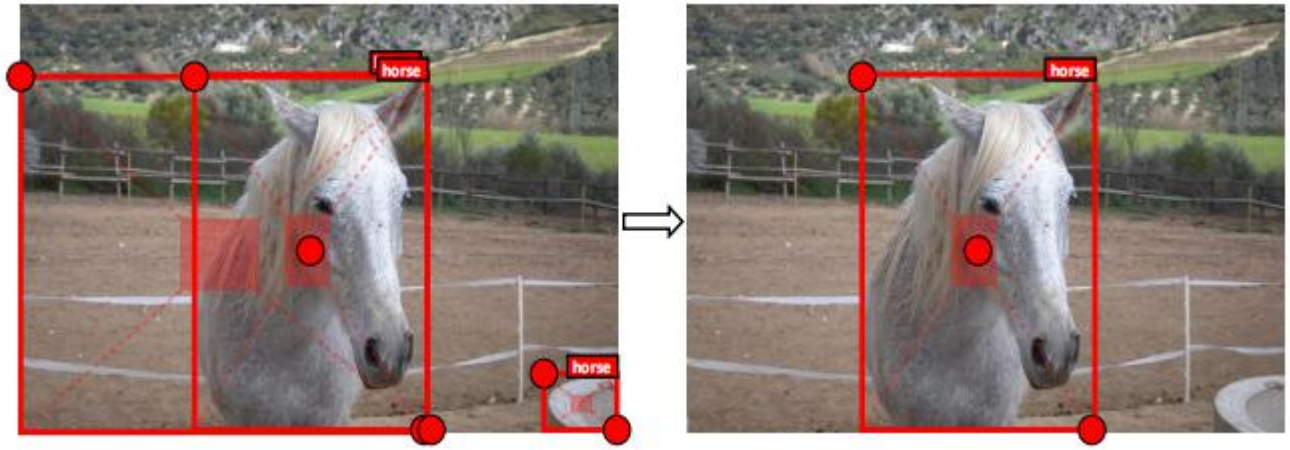
- anchor-based是指使用anchor boxes解决检测问题，anchor-free指未使用anchor boxes，通过另外一种手段来解决检测问题。
- anchor boxes广泛用于one stage检测器（SSD，DSSD，yolov2，RetinaNet），它可以获得与two stage检测器(Faster r-cnn,R-CNN,Fast r-cnn,Mask r-cnn)有竞争力的结果，同时效率更高。
- one stage检测器将anchor boxes密集地分布在图像上，通过对anchor boxes进行评分，并通过回归来改进其坐标来生成最终的边界框预测。
- 但anchor boxes的使用有两个缺点。
- 首先，我们通常需要一组非常大的anchor boxes，例如： 在DSSD中超过4万，在RetinaNet中超过10万， 这是因为训练器被训练以分类每个anchor boxes是否与ground truth充分重叠，并且需要大量anchor boxes以确保与大多数ground truth充分重叠。 结果，只有一小部分anchor boxes与ground truth重叠; 这在正负样本之间造成了巨大的不平衡，减慢了训练速度（Focal Loss）。
- 其次，anchor boxes的使用引入了许多超参数和设计选择。 这些包括多少个box，大小和宽高比。 这些选择主要是通过手工设计的，并且当与多尺度架构相结合时可能会变得更加复杂，其中单个网络在多个分辨率下进行单独预测，每个尺度使用不同的特征和它自己的一组anchor boxes（SSD，DSSD，RetinaNet）。

为了克服anchor-based方法的缺点，有一种使用关键点检测目标的方法，这是一种anchor-free方法。

基于关键点的目标检测方法例如 角点网络（CornerNet），是通过检测物体的左上角点和右下角点来确定目标，但在确定目标的过程中，无法有效利用物体的内部的特征，即无法感知物体内部的信息，从而导致该类方法产生了很多误检 (错误目标框)。（蓝框为真实框，红框为检测框）（下图使用网络为角点网络(CornerNet)）

- 中心网络（CenterNet）对角点网络（CornerNet）进行了改进。
- 中心网络（CenterNet）在角点网络（CornerNet）基础上赋予了网络能感知每个候选区域内部信息的能力，使其能判断边框的正确性。利用一个额外的关键点来获取候选区域的中心部分，该中心区域与几何中心很近。
- 中心网络（CenterNet）基于上述原因，推断出：如果目标框是准确的，那么在其中心区域能够检测到目标中心点的概率就会很高，反之亦然。
- 简单地说，就是在推理时，通过一对角点关键点得到一个候选区域，通过检测是否有同种类别的中心关键点出现在候选区域的中心区域 来判断这个候选区域是否真的是目标。

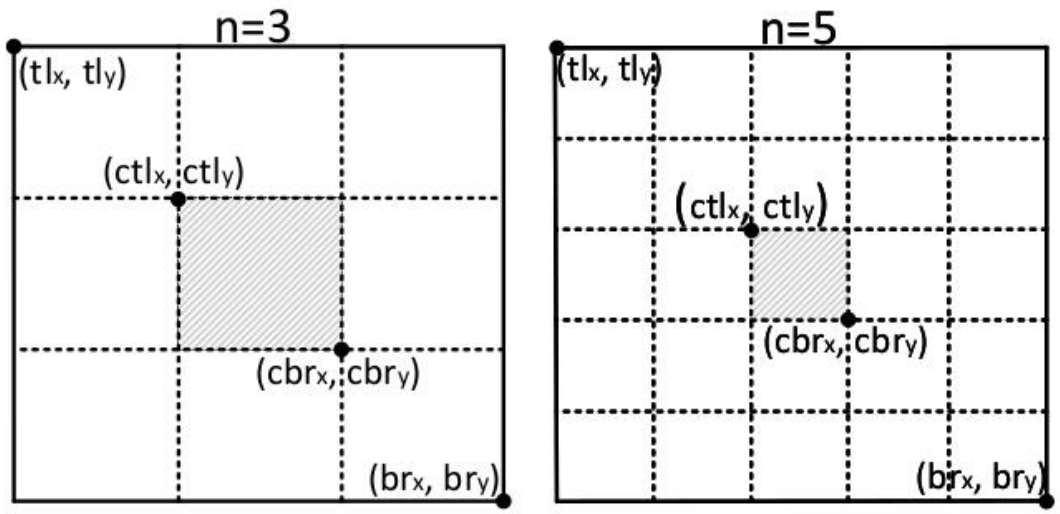
- 因此，首先利用左上和右下两个角点生成初始目标框，对每个预测框定义一个中心区域，然后判断每个目标框的中心区域是否含有中心点，若有则保留该目标框，若无则删除该目标框。如下图所示




- 用三个关键点，而不是两个关键点，来表示一个物体。
- 预测框中心区域的尺寸会影响检测效果。例如中心区域过小导致很多准确的小尺度的目标也会被去除（召回(recall)过低），而中心区域过大导致很多大尺度的错误目标框无法被去除（精度(precision)过低）。

- 中心网络（CenterNet）采用了一种尺寸感知（scale-aware）方法来自适应地确定中心区域大小。对于大目标则给予相对小的中心区域，对于小目标则给予相对大的中心区域。
- 假设我们需要判断一个边界框是否需要被保留，tlx,tly代表框左上角的点，brx,bry代表框右下角的点。定义一个中心区域，定义左上角的点的坐标为(ctlx,ctly),右下角点(cbrx,cbry)。需满足以下公式

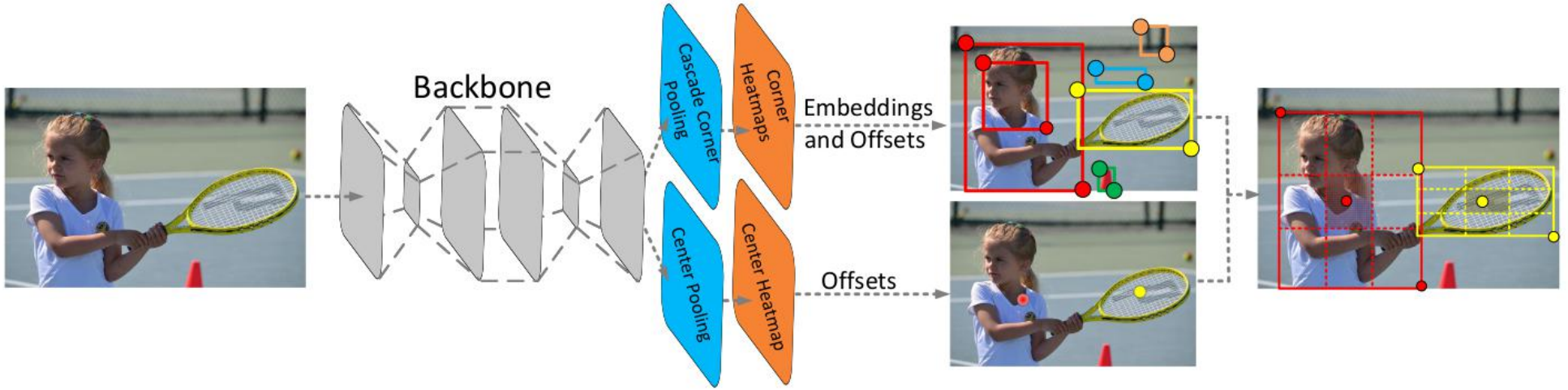
$$
\begin{cases}
\mathrm{ctl_x} = \dfrac{(n+1)\mathrm{tl_x} + (n-1)\mathrm{br_x}}{2n} \\
\mathrm{ctl_y} = \dfrac{(n+1)\mathrm{tl_y} + (n-1)\mathrm{br_y}}{2n} \\
\mathrm{cbr_x} = \dfrac{(n-1)\mathrm{tl_x} + (n+1)\mathrm{br_x}}{2n} \\
\mathrm{cbr_y} = \dfrac{(n-1)\mathrm{tl_y} + (n+1)\mathrm{br_y}}{2n}
\end{cases}
$$




- 其中n为奇数（奇数才有中心区域），在COCO数据集中，当bbox的size大于150时，n=5,小于150时，n=3。
- 该方法可以在预测框的尺度较大时定义一个相对较小的中心区域，在预测框的尺度较小时预测一个相对较大的中心区域。

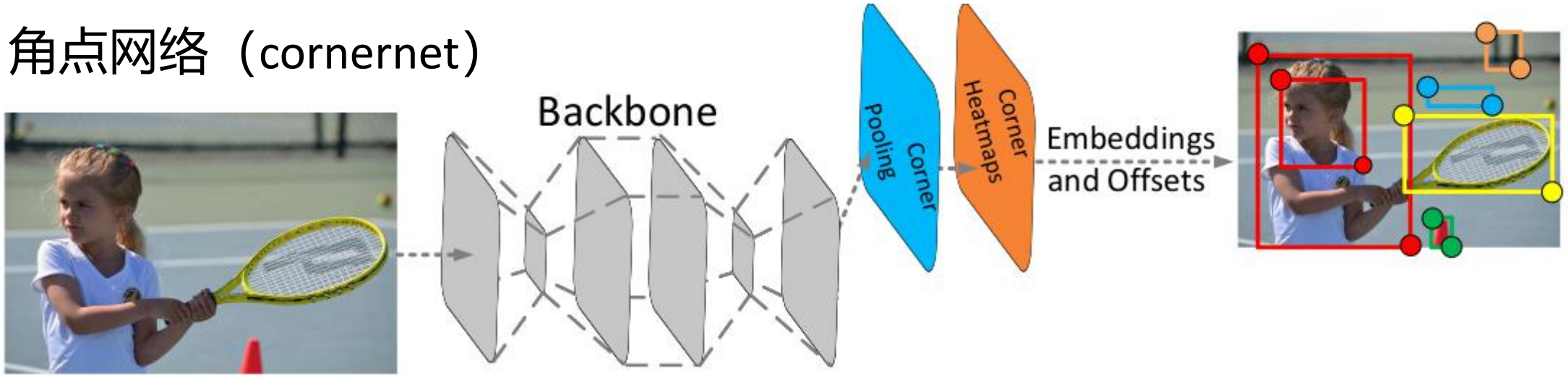
# 中心网络（centernet）两条分支

- 为了更好的检测中心的关键点和角点，通过两个方法来丰富中心点及角点的信息。
- （1）中心池化（center pooling）：用于预测中心关键点的分支，有利于中心点获得更多目标物的特征，进而更易感知候选区域（proposal）的中心区域。通过取中心位置横向与纵向响应值的和的最大值实现此方法。
- （2）级联角点池化（cascade corner pooling）：增加角点网络（cornernet）中的角点池化（corner pooling）感知内部信息的功能。结合了特征图（feature map）中目标物内部及边界方向的响应值和的最大值来预测角点。

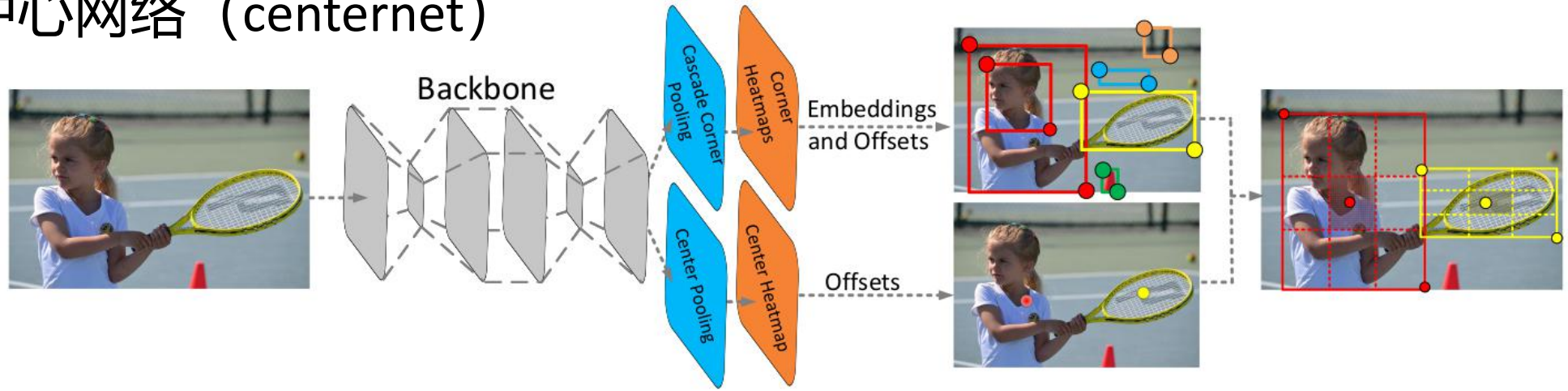


# 中心网络（centernet）和角点网络（cornernet）对比

## 角点网络（cornernet）




## 中心网络（centernet）

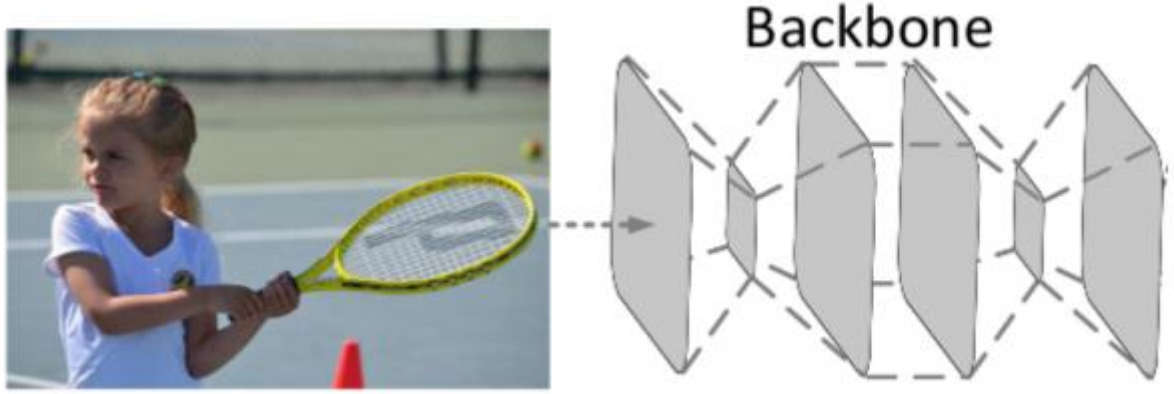


# Backbone-Hourglass

角点网络（cornernet）和中心网络（centernet）
都使用相同的沙漏网络（Hourglass Network）作为特征提取网络（backbone）。

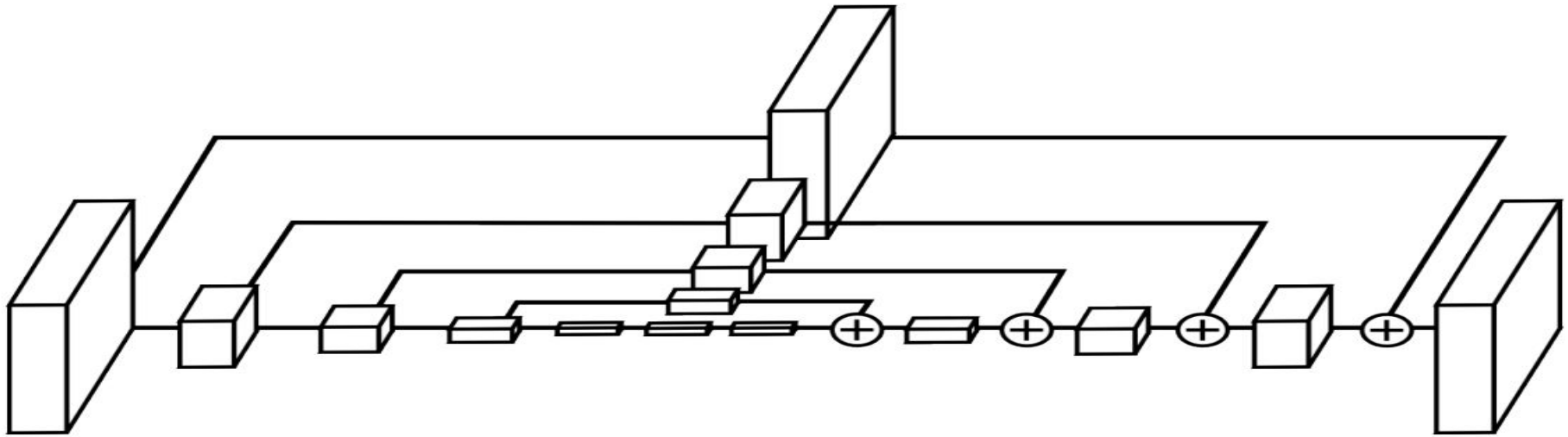
使用沙漏网络（Hourglass Network）作为特征提取网络是受到人体姿态估计的启发（论文Stacked Hourglass Networks for Human Pose Estimation）。

考虑到速度，图像在进入沙漏网络（Hourglass Network）前通过一个步长为2的7×7卷积网络和一个步长为2的残差块，将尺寸缩小了4倍。



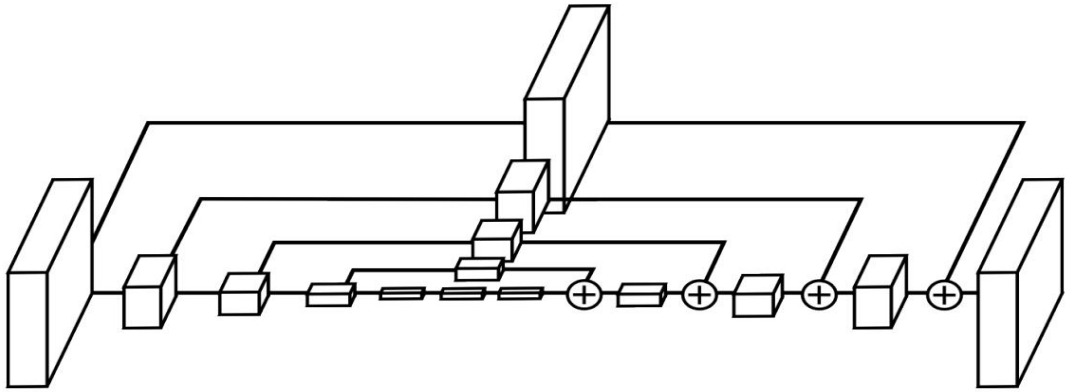
# Hourglass

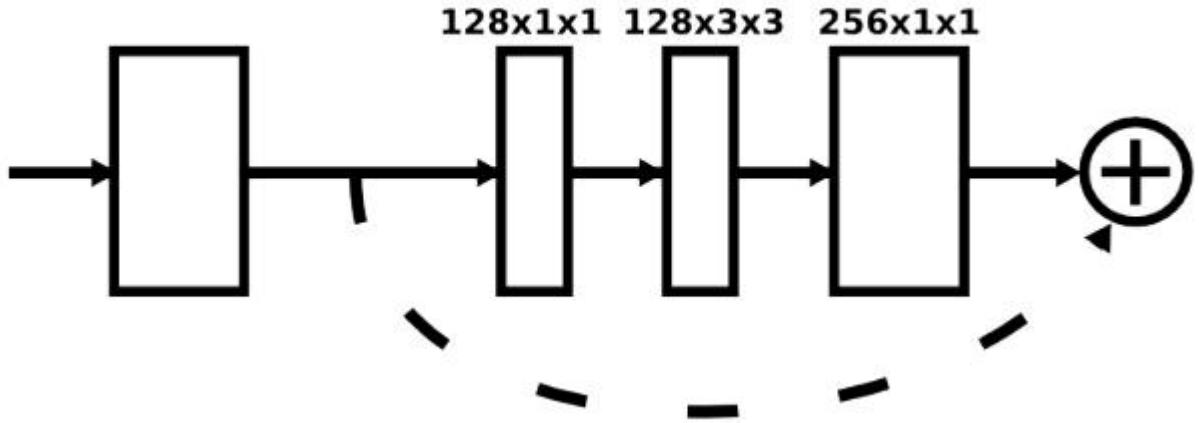
- 这个网络的形状非常像沙漏。
- 网络前半部分通过卷积池化处理，进行多次下采样操作，获得一些分辨率较低的特征，同时使计算复杂度降低。
- 下采样不断变小feature map，后面则通过上采样增大feature map恢复到输入图像大小，上采样操作使得图像的分辨率增高，同时更有能力预测物体的准确位置。
- 由于在下采样中丢失了详细信息，因此将添加残差层以将详细信息带回到上采样的特性中，因此才有将前后网络相加的操作。这样网络输出整合了所有尺度下的特征。



# Hourglass

- 沙漏网络中的白色方块表示的都是类似于 ResNet 中的残差模块，其作用是在保留原特征信息的同时进一步提取更深层次的特征，同时也能使得网络变得更深又不至于梯度消失。

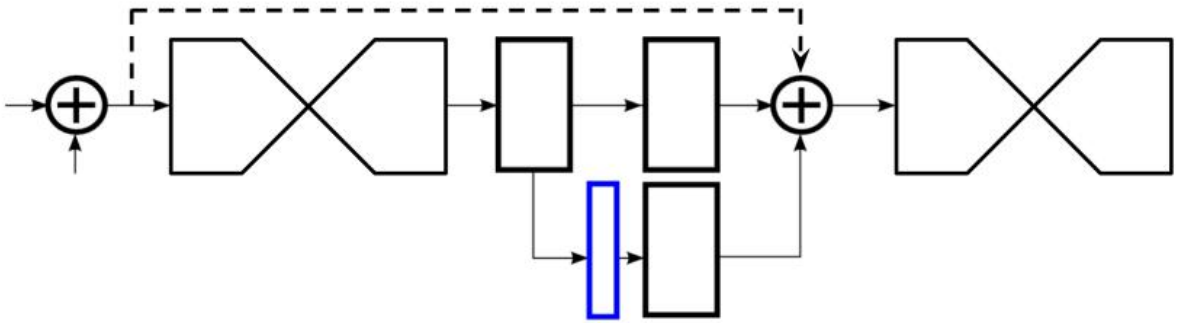




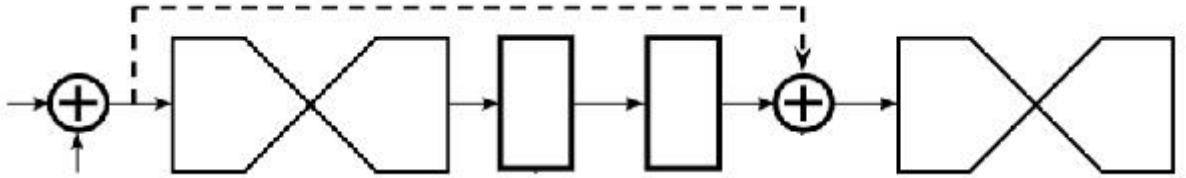
# Hourglass

- 像堆叠残差模块一样堆叠沙漏模块就得到了堆叠沙漏网络。当多个沙漏模块被堆放在网络中时，沙漏模块可以重新处理这些特性以捕获更高级别的信息。这些特性使得沙漏网络也成为目标检测的理想选择。
- 值得注意的是，沙漏网络的输入和输出大小可以是一样的，也就是说在每个沙漏模块之后都可以进行最终结果的预测并计算损失，起到中间监督作用。另外，上层模块的预测结果也可以作为下层模块的输入，从而更好的帮助下层模块进行预测，因此预测结果也可以通过 1*1 的卷积重新加入到原来的特征中，进行由粗糙到细致的估计。
- 角点网络（cornernet)没有使用中间监督，因为发现这会损害网络的表现。

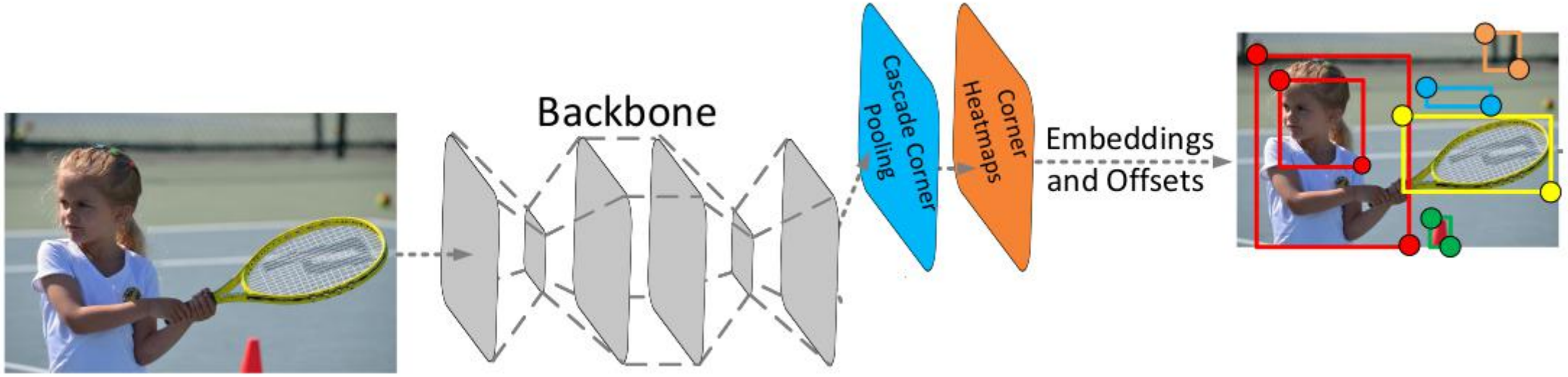
人体姿态估计



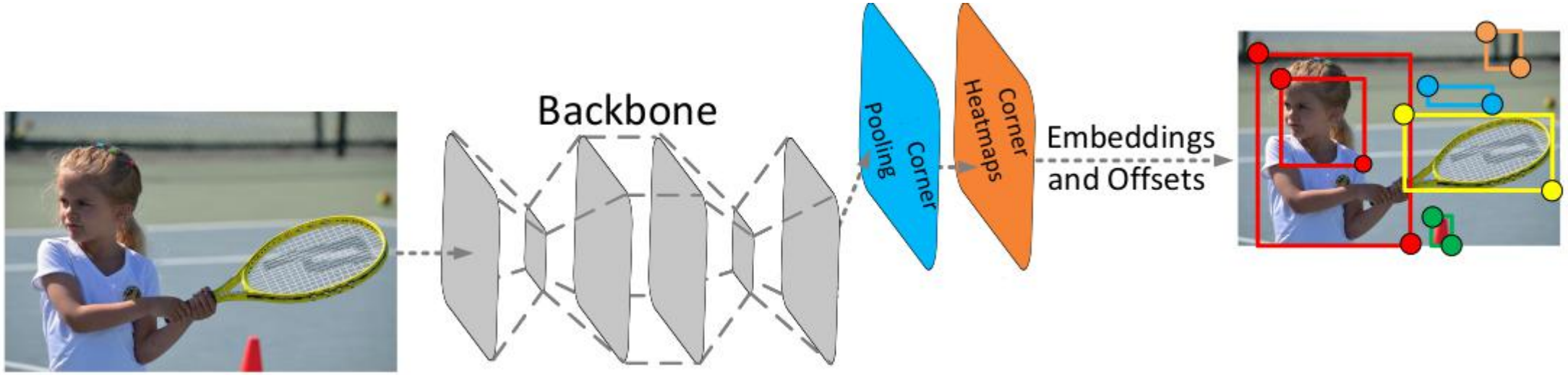
角点网络



- 特征提取网络后，中心网络（CenterNet）提出了一个分支级联角点池化（cascade corner pooling ）来提取角点的特征。

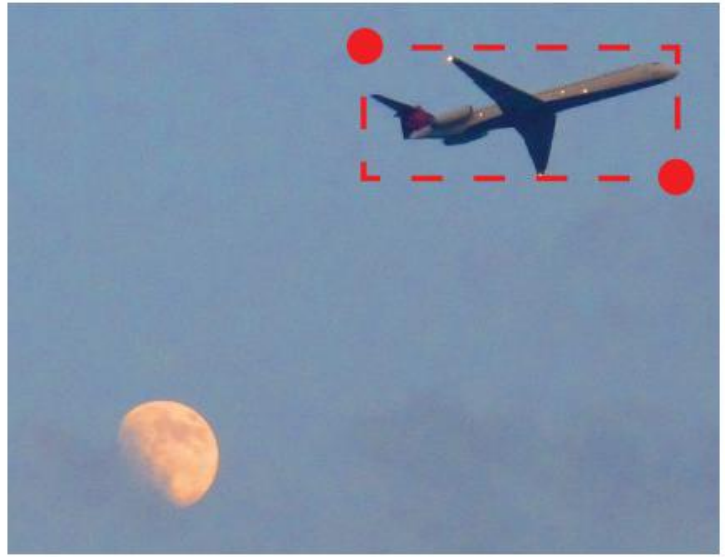



- 此池化从角点网络（cornernet）的角点池化（corner pooling ）改进而来。



# corner pooling

- 目标边框的角点经常在目标物外，如下图所示。（自然界的大部分目标是没有边界框也不会有矩形的顶点）



- 这种情况下，角点的位置不能通过局部特征来定位。角点处特征值表示不了目标。

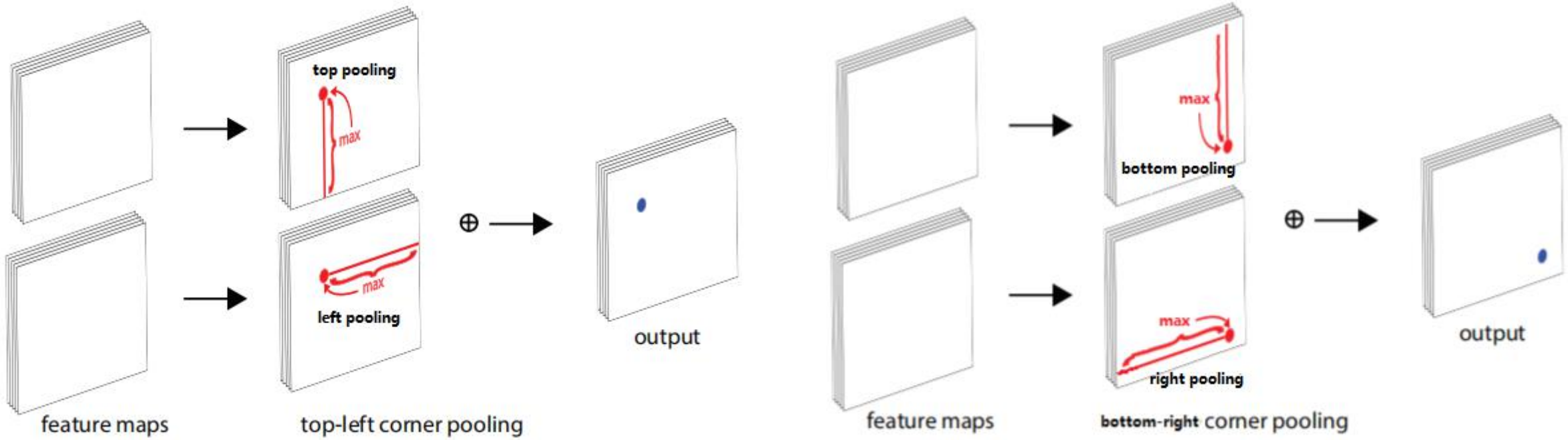
# corner pooling

- 为了查看一个像素点是否为左上点，我们需要从此点开始水平向右查看到物体的顶部，垂直向下查看到物体的左边。（右下点类似，从此点开始水平向左查看到物体的底部，垂直向上查看到物体的右边。）

- 以上图左上点为例，考虑到左上角角点的右边有目标顶端的特征信息（第一张图的头顶），左上角角点的下边有目标左侧的特征信息（第一张图的手），因此如果左上角角点经过池化操作后能有这两个信息，那么就有利于该点的预测。

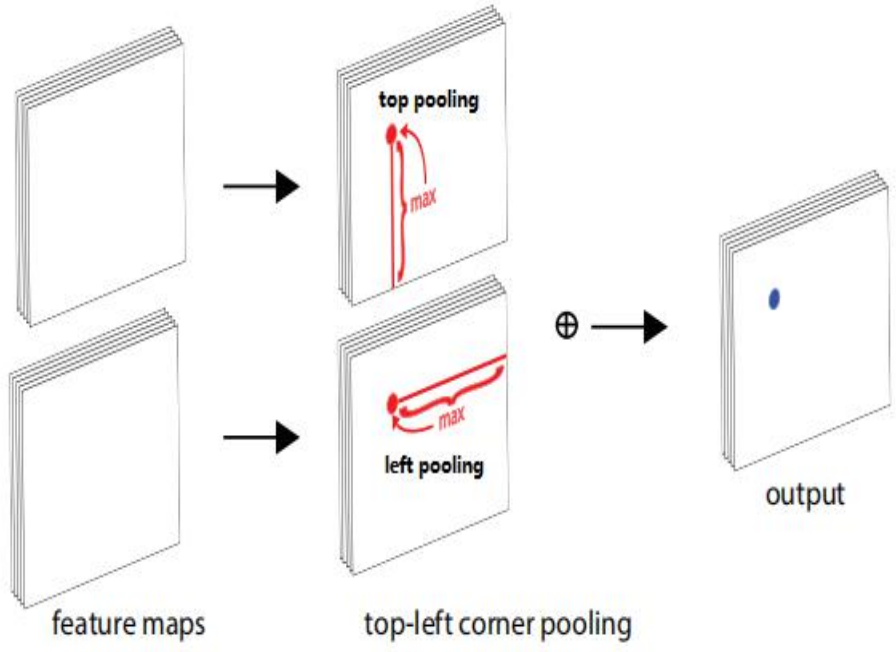
# corner pooling

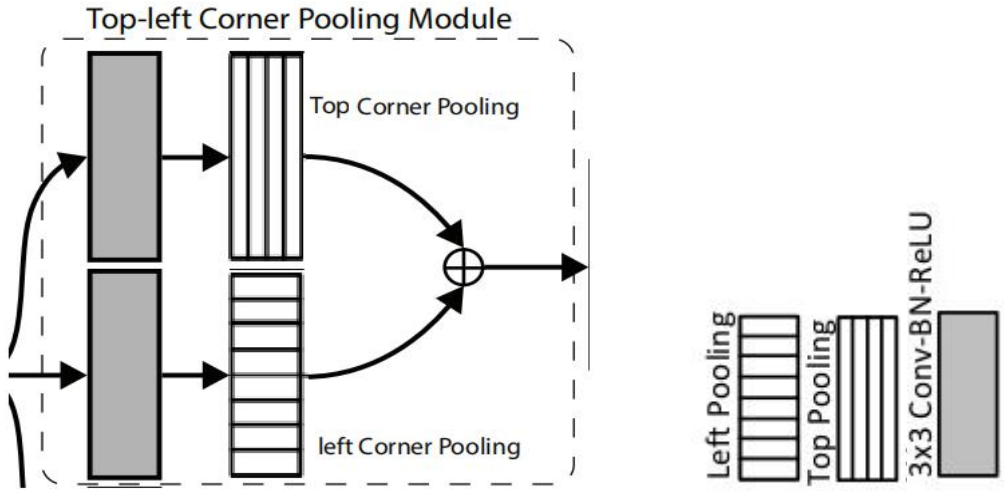
- 为了达到这个目的，角点网络（cornernet）设计了一种新型的池化层（pooling），角点池化（corner pooling）。
- Corner pooling示例图如下



# corner pooling

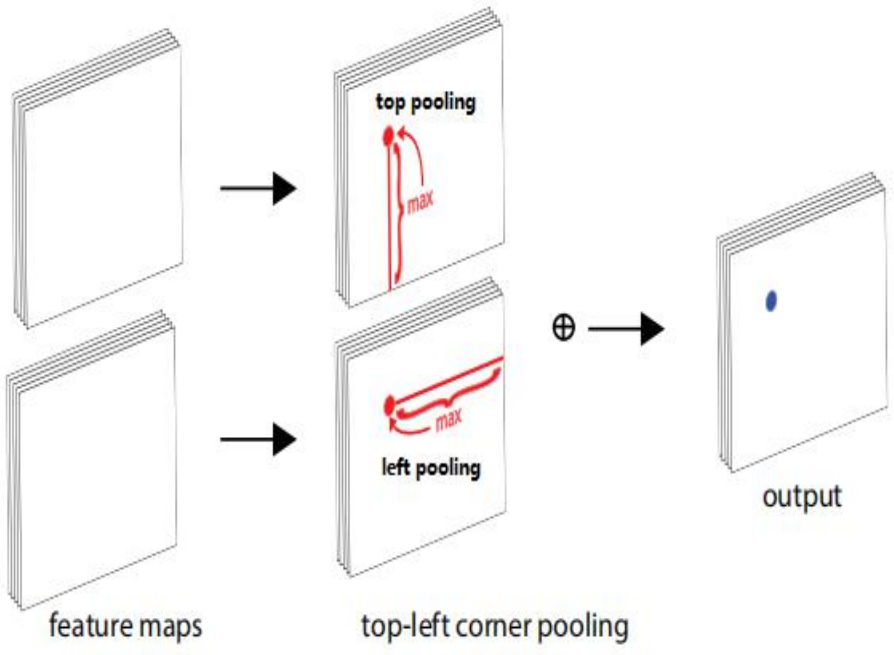
- 以左上角点池化为例，输入两个feature maps，对第一个Feature map，每一个像素点位置的下方进行max pool操作，对第二个feature map，每一个像素点位置的右侧进行max pool，最后将二者的结果进行相加处理（特征融合），得到该点位置的特征值。如下示例图和网络结构图



# corner pooling

- 具体地说，假设设输入特征图为$f_t$和$f_l$，$f_{tij}$和$f_{lij}$分别表示在$f_t$和$f_l$中位置为(i,j)的特征值，特征图的宽高分别用W和H表示，接下来要确定图中红色点（坐标假设是(i,j)）是否为角点，那么池化层就要计算(i,j)到(i,H)的最大值$t_{ij}$（对应top pooling），类似于找到左侧手信息；同时计算(i,j)到(W,j)的最大值$l_{ij}$（对应left pooling），类似于找到头顶信息，然后将这两个最大值相加得到(i,j)点的值（对应蓝色点）。（右下角点的corner pooling操作类似，只不过计算最大值变成从(0,j)到(i,j)和从(i,0)到(i,j)）。图片公式如下




$$t_{ij} = \begin{cases} \max(f_{t_{ij}}, t_{i(j+1)}) & \text{if } j < H \\ f_{t_{iH}} & otherwise \end{cases}$$

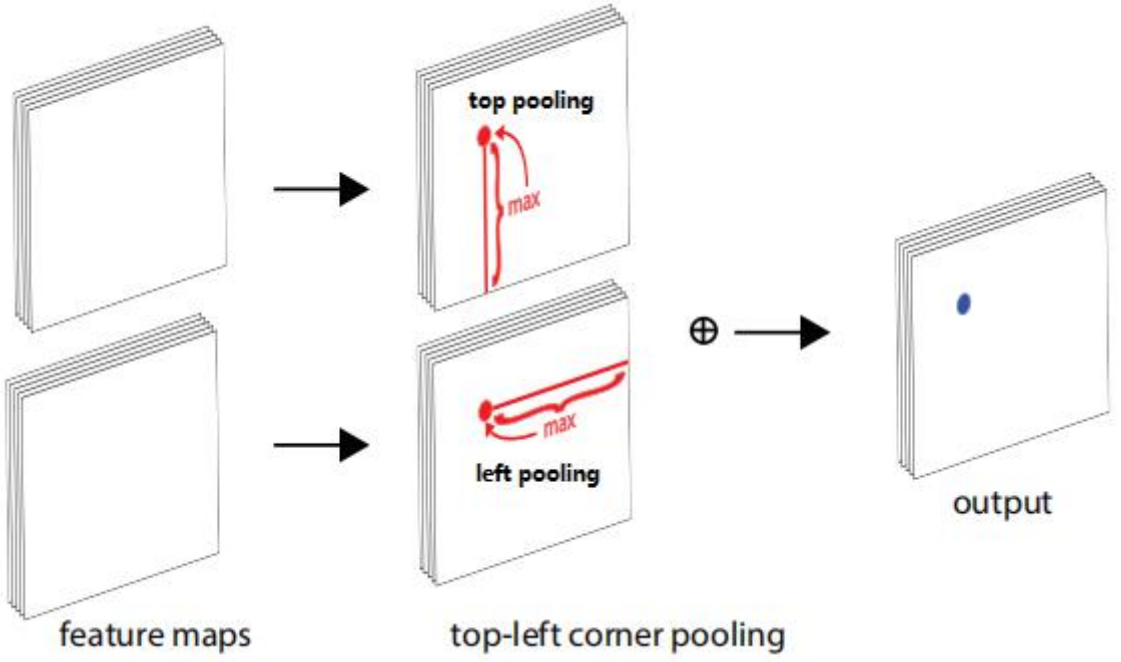
$$t_{ij} = \begin{cases} \max(f_{t_{ij}}, t_{i(j+1)}) & \text{if } j < H \\ f_{t_{iH}} & otherwise \end{cases}$$

# corner pooling

- 左上角池化计算公式可以表示为

$$l_{ij} = \begin{cases} \max(f_{l_{ij}}, l_{(i+1)j}) & \text{if } i < W \\ f_{l_{Wj}} & otherwise \end{cases}$$

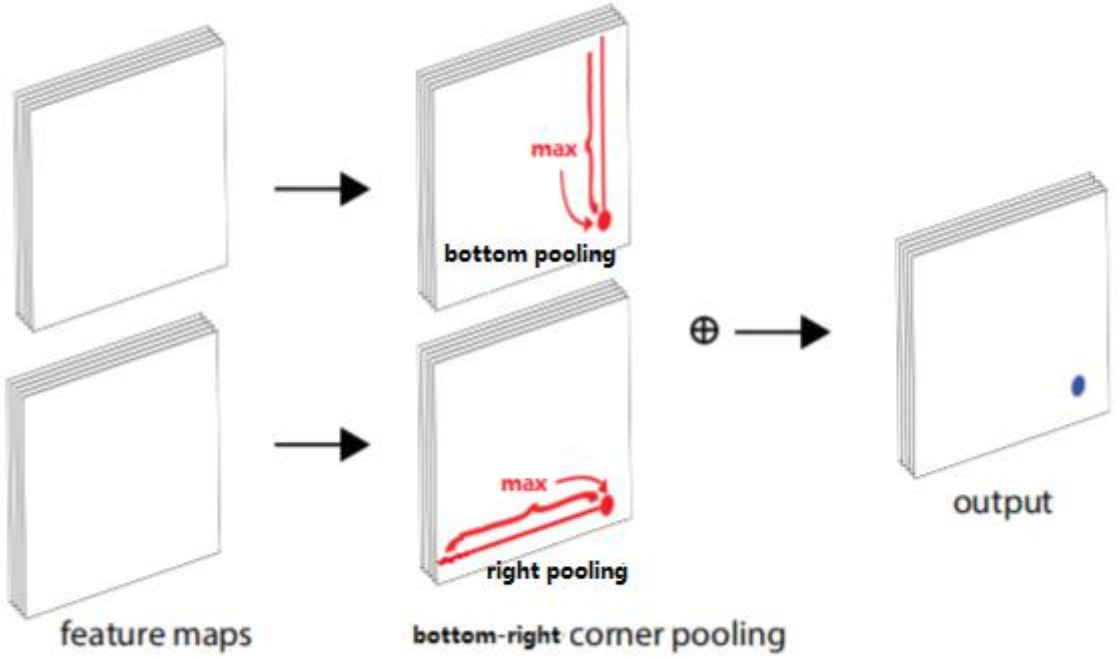
$$t_{ij} = \begin{cases} \max(f_{t_{ij}}, t_{i(j+1)}) & \text{if } j < H \\ f_{t_{iH}} & otherwise \end{cases}$$




- 同理右下角池化计算公式类似，只不过计算最大值变成从(0,j)到(i,j)和从(i,0)到(i,j)。

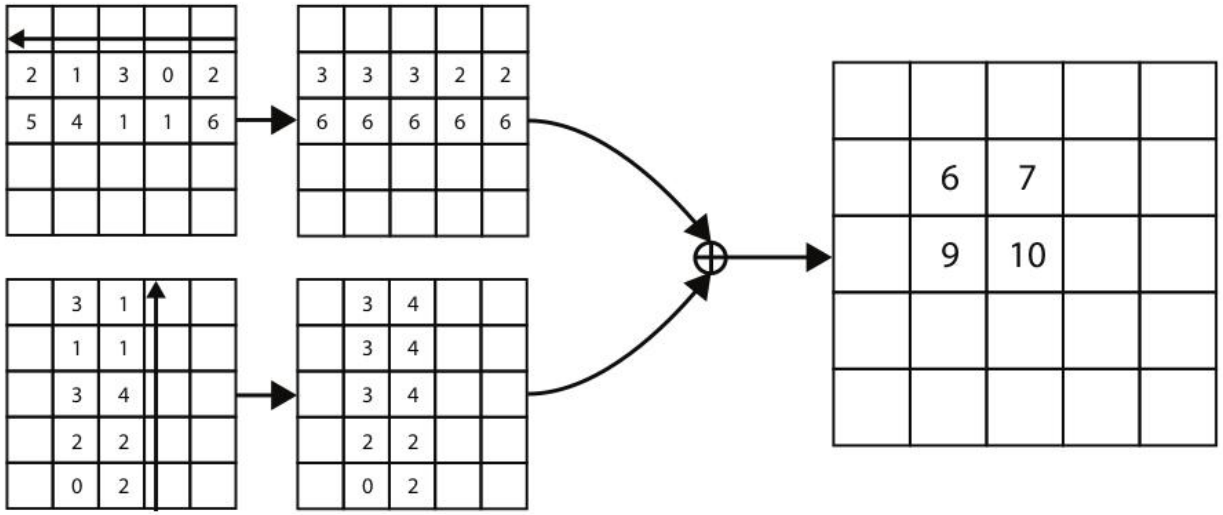
$$r_{ij} = \begin{cases} \max(f_{r_{ij}}, r_{(i-1)j}) & \text{if } i > 0 \\ f_{r_{0j}} & otherwise \end{cases}$$

$$b_{ij} = \begin{cases} \max(f_{b_{ij}}, b_{i(j-1)}) & \text{if } j > 0 \\ f_{b_{i0}} & otherwise \end{cases}$$



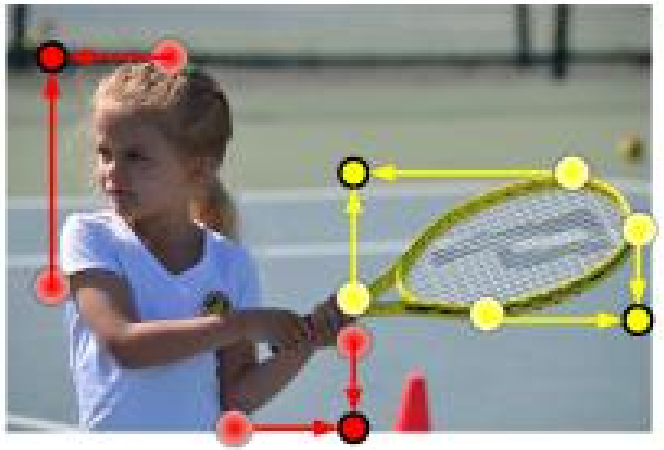
# corner pooling

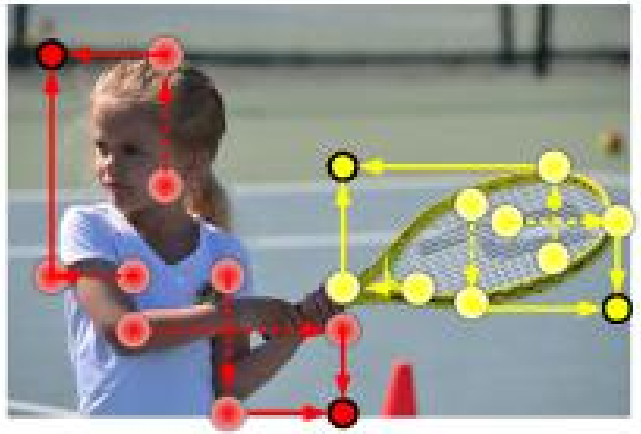
- 下图是一个左上点具体的实例，说明了左上角点池化（Top-left Corner pooling）四个点之后的结果。



# cascade corner pooling

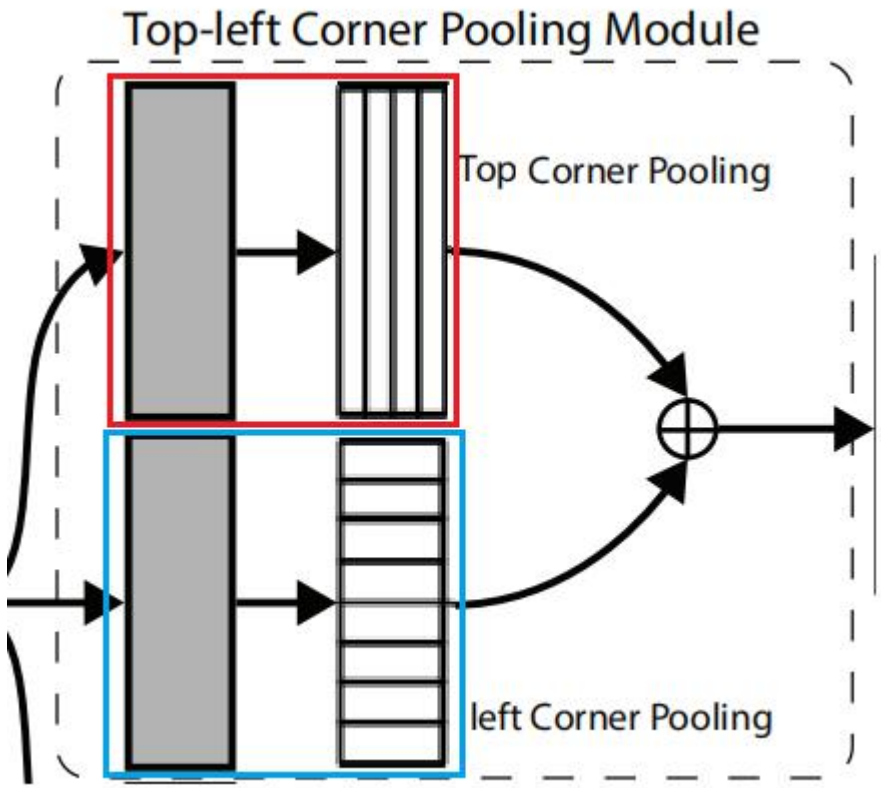
- 由上可知Corner pooling提取物体边界最大值并相加，该方法只能提供关联物体边缘语义信息，使其对边缘敏感，对于更加丰富的物体内部语义信息则很难提取到。参见下左图。
- Cascade corner pooling解决了这个问题。首先和Corner Pooling一样，首先沿边界查找边界最大值，然后沿边界最大值的位置向内部看查找内部最大值，最后将两个最大值相加，参见下右图。通过这种方式，使得角点可以同时获得物体的边界信息和内部视觉信息。Cascade corner pooling可以通过结合不同方向的corner pooling得到。

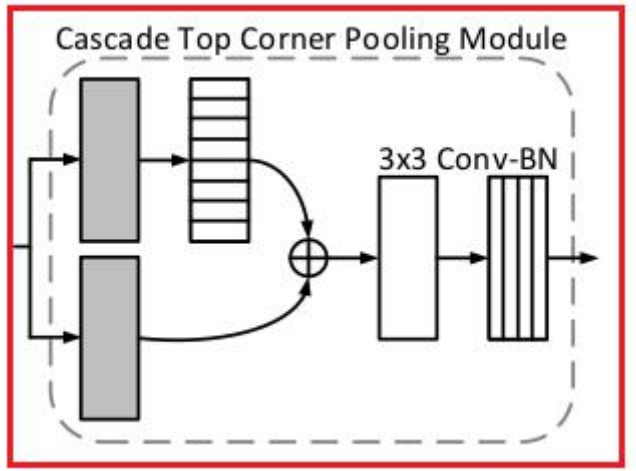


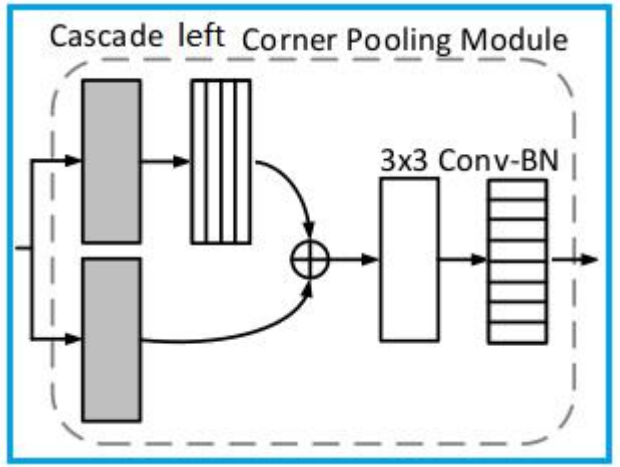


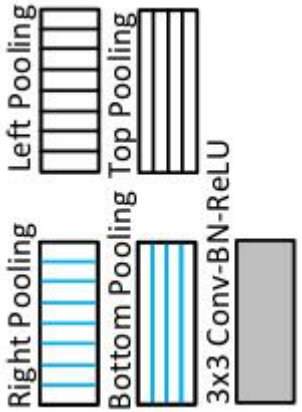
# cascade corner pooling

- 将上角点池化模块（top corner pooling）替换为级联上角点池化模块（cascade top cprner pooling），图中红框所示。
- 将左角点池化模块（left corner pooling）替换为级联左角点池化模块（cascade left cprner pooling），图中蓝框所示。
- 就将左上角点池化（top-left corner pooling）改进为级联左上角点池化（cascade top-left corner pooling）。（右下角点同理）



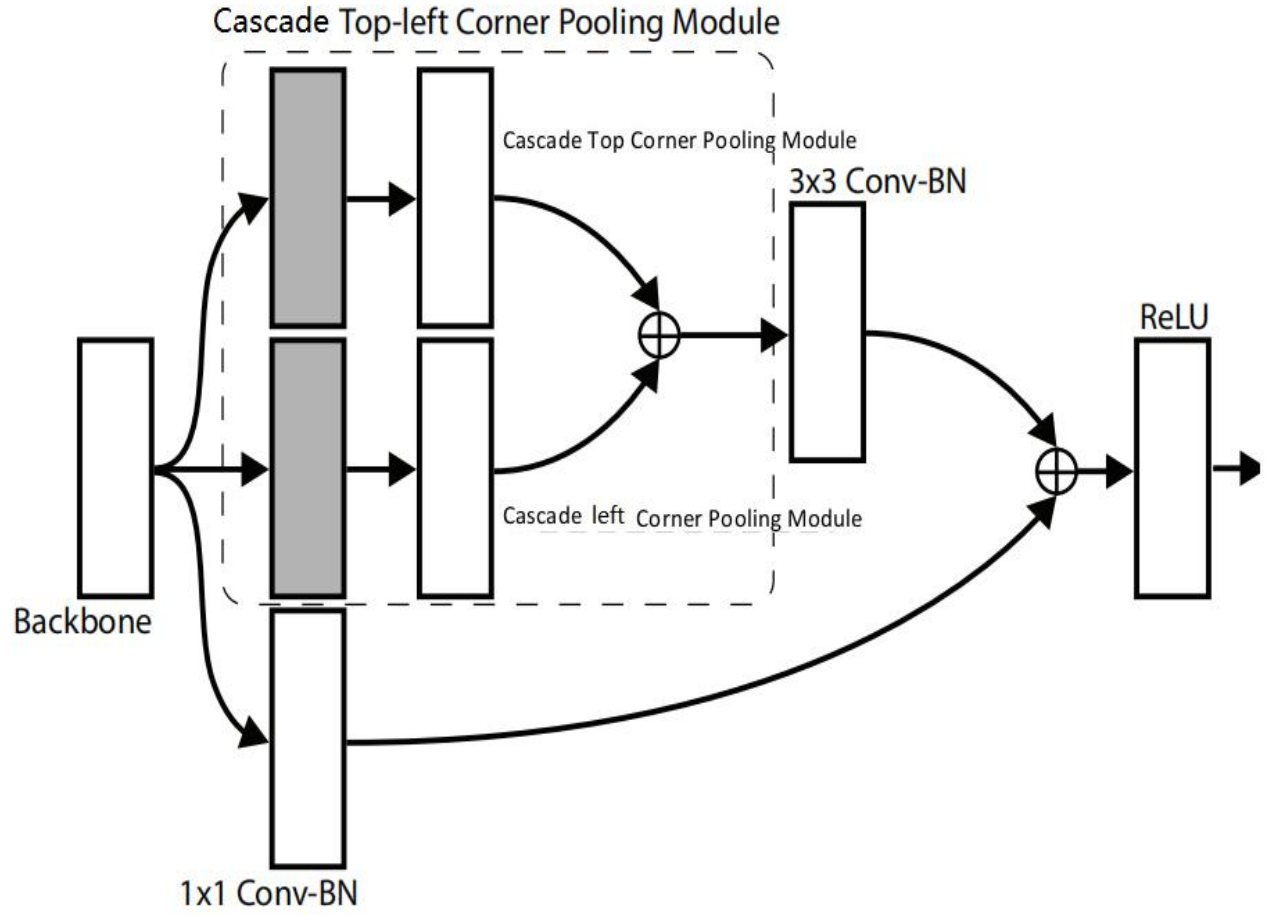



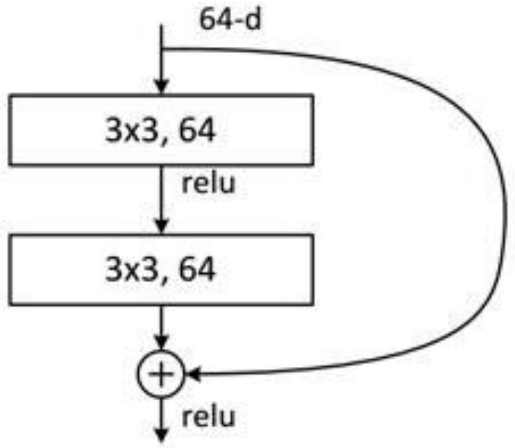



# cascade corner pooling

- 为了使预测的效果更好，还将级联角点池化（cascade corner pooling）和残差块相结合。将第一个3×3的卷积层替换为级联角点池化模块（cascade corner pooling module）

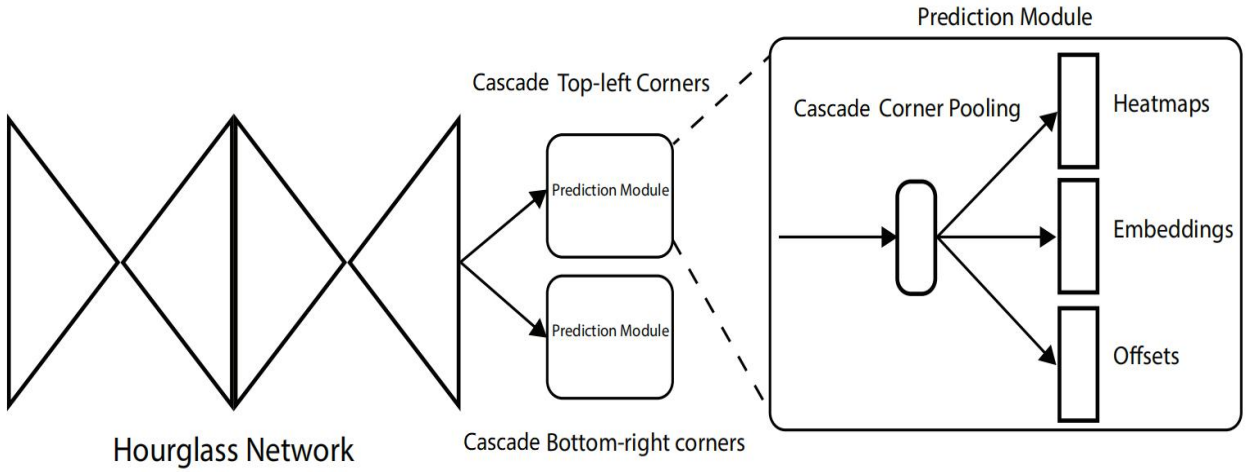


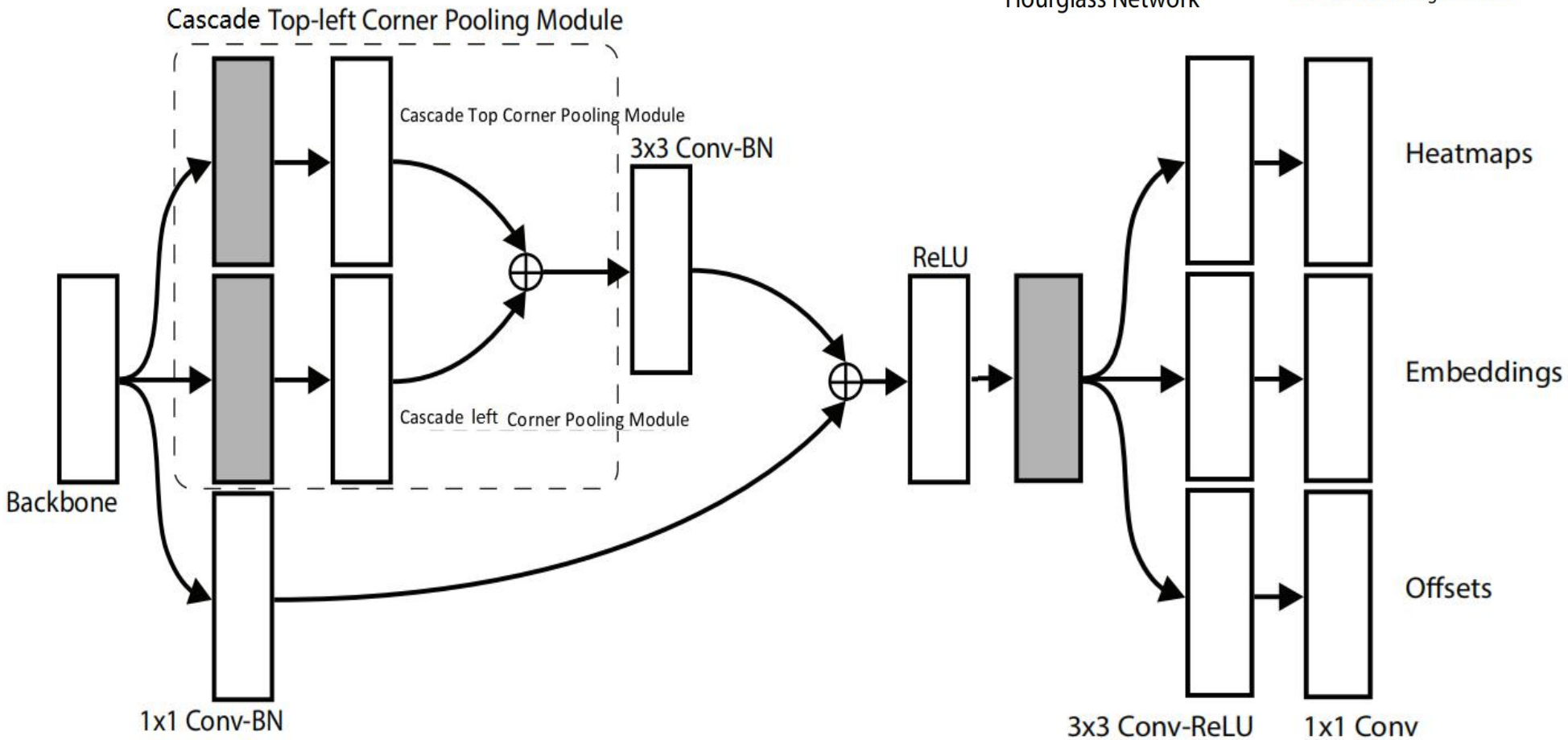

残差块



# cascade corner pooling

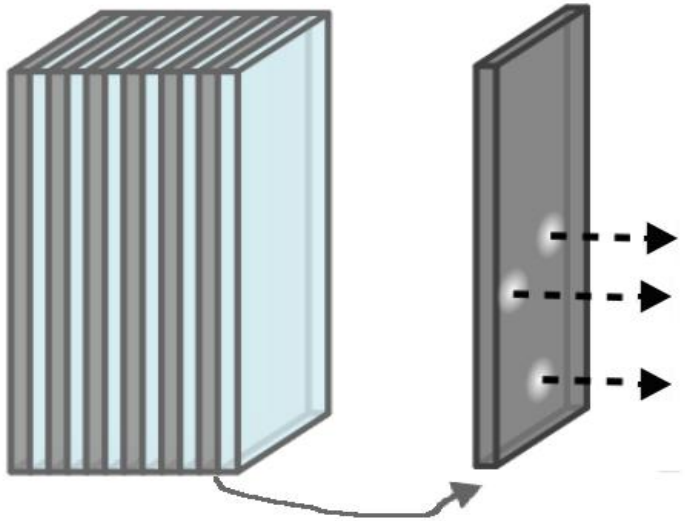
- 特征图经过级联角点池化（cascade corner pooling）后，输出三条分支，
- 热点图（heatmaps）,嵌入编码（embeddings）和偏移（offsets），
- 体现了模块化设计网络的思路。

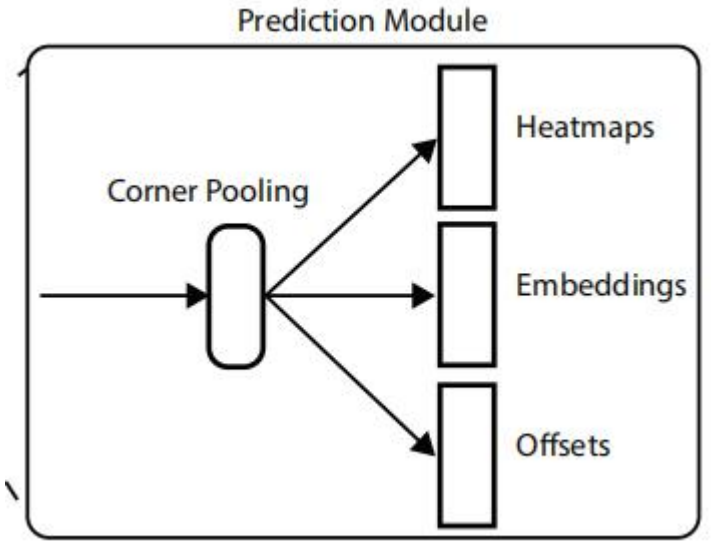


# heatmaps

- Heatmaps的大小是128*128，通道为80（coco数据，80类），所以Heatmaps是一个大小为128*128*80的张量。
- 以Top-left corners模块中的Heatmaps为例，其每个通道代表coco数据中的80个类别，每个通道的128*128网格中的每个点的输出表示该点是一个左上角点的概率。并且该通道预测出来的所有左上角点对应的所有bbox的物体类别都一样。
- 举个例子，假设在通道编号为20（假设物体类别是羊）的128*128热图里面，预测出了3个左上角点，也就是有3个bbox，那么这3个bbox的物体类别都是羊。

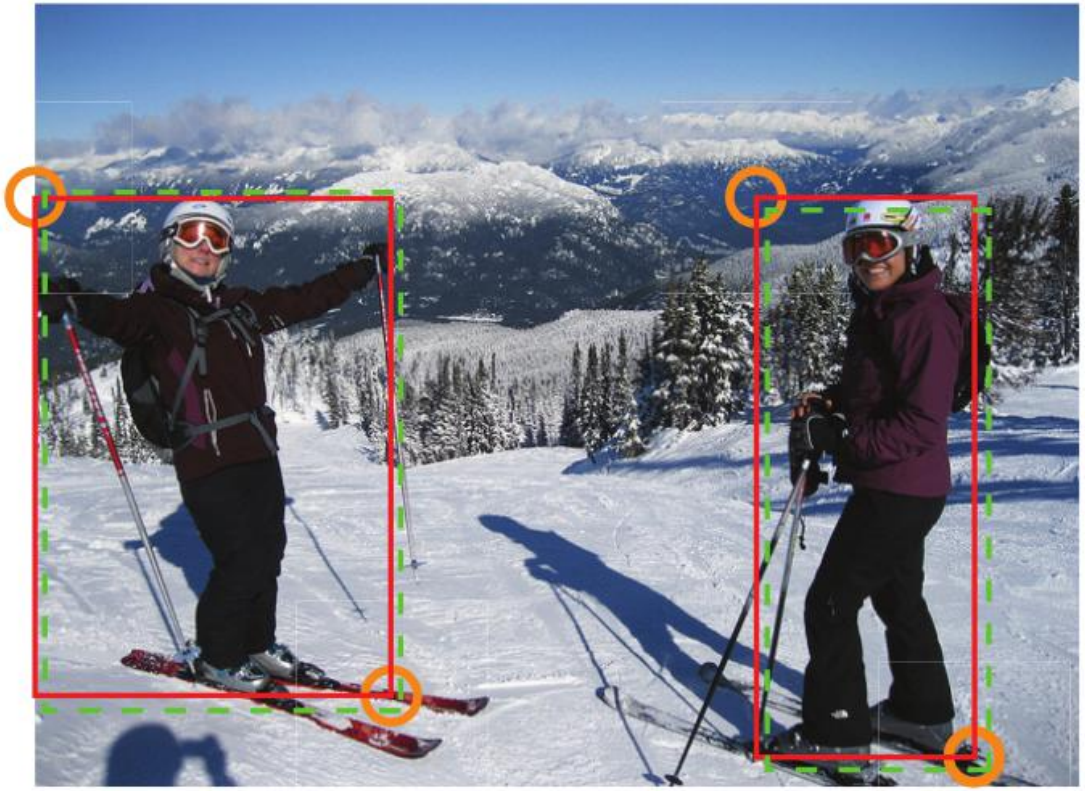




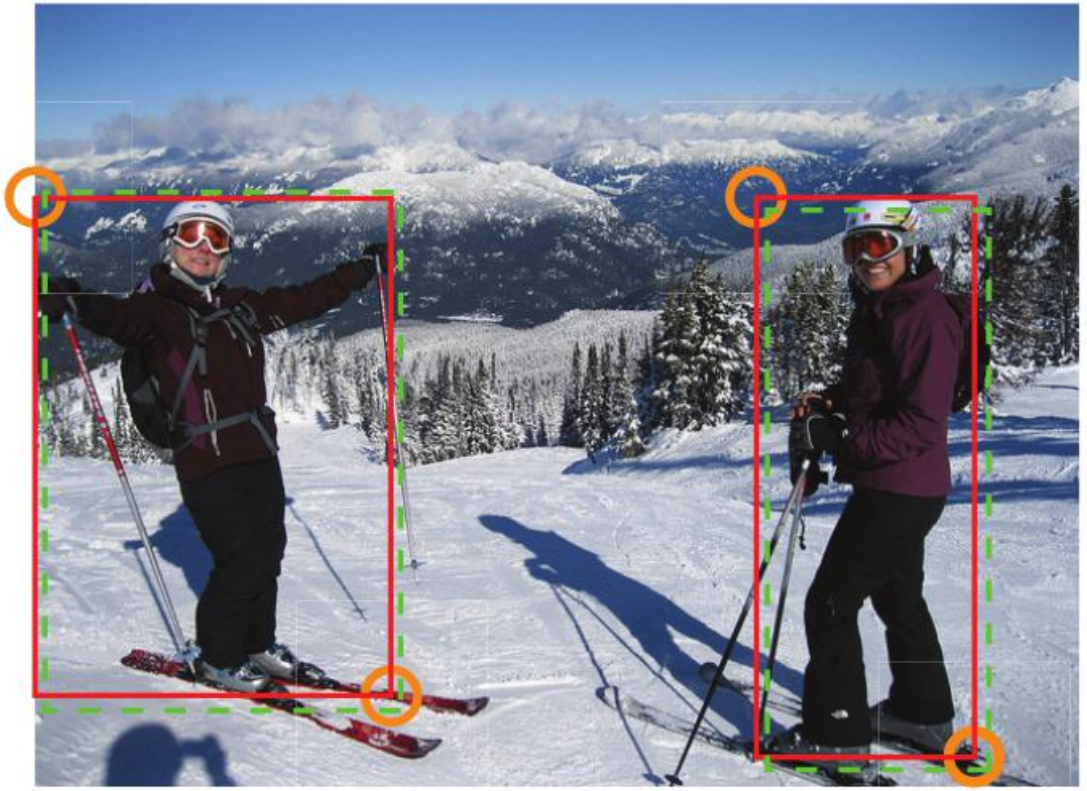
# heatmaps

- 对于每个顶点，只有一个真实值（ground truth），其他位置都是负样本。
- 靠近真实值（ground truth）的角点位置的误检顶点形成的预测框和ground truth的标注框仍然有较大的重叠区域。如下图
- 图中红色实线框是真实值，绿色虚线

是一个预测框，可以看出这个预测框的
两个角点和真实值并不重合，
但是该预测框基本框住了目标，
因此是有用的预测框，
所以要有一定权重的损失返回。
结论：对于不同的负样本点，
损失函数应该采用不同的权重值。



- 具体做法是:在每个真实值顶点设定半径r区域内都是正样本，这是因为落在半径r区域内的顶点依然可以生成有效的边界定位框，橘色圆圈就是根据真实值的左上角顶点、右下角顶点和设定的半径值画出来的，半径是根据圆圈内的角点组成的框和真实值的IOU值大于0.3而设定的，圆圈内的点的数值是以圆心往外呈二维的高斯分布 $e^{-\frac{x^2+y^2}{2\sigma^2}}$（σ为三分之一的半径值）设置的，其中，圆心坐标是标注的角点位置。



- Heatmaps的标准focal loss损失函数

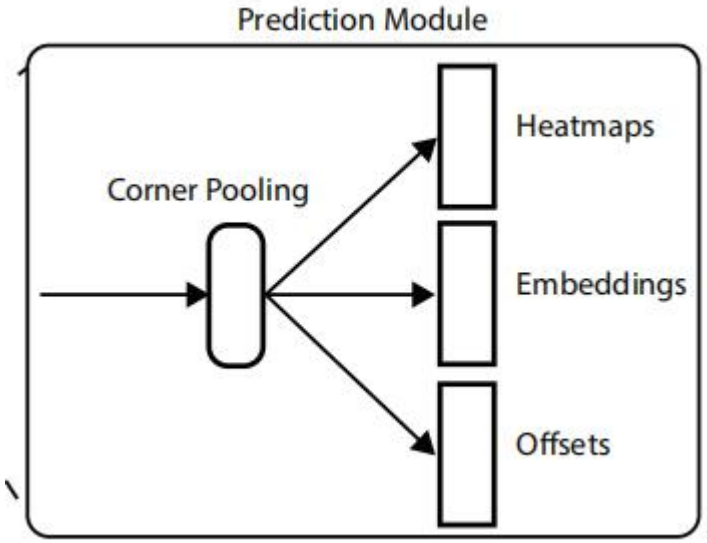
$$L_{det} = \frac{-1}{N} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \begin{cases} (1-p_{cij})^{\alpha} \log(p_{cij}) & \text{if } y_{cij} = 1 \\ (p_{cij})^{\alpha} \log(1-p_{cij}) & \text{otherwise} \end{cases}$$

- 标准focal loss未考虑较大重叠区域的误检框
- Heatmaps的基于focal loss修改的损失函数

$$L_{det} = \frac{-1}{N} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \begin{cases} (1-p_{cij})^{\alpha} \log(p_{cij}) & \text{if } y_{cij} = 1 \\ (1-y_{cij})^{\beta} (p_{cij})^{\alpha} \log(1-p_{cij}) & \text{otherwise} \end{cases}$$

- N是物体个数。
- α，β是控制每个点贡献的超参数（α = 2 ，β = 4）。
- C是通道数。
- H是高度。
- W是宽度。
- Pcij是网络预测类别为c在位置（i，j）是角点的概率值。
- Ycij是类别为c在位置（i，j）的真实值。
- 其中
- Ycij=1时表示位置（i，j）是类别为C物体的角点。
- Ycij=0时表示位置（i，j）表示不是类别为C物体的角点。
- 此时Ycij=0时表示在圆圈外。
- 当点在圆圈内且不在圆心时，Ycij在0到1之间，根据高斯函数 $e^{-\frac{x^2+y^2}{2\sigma^2}}$ 计算 。
- 达到降低对正样本附近圆周位置的负样本的惩罚的效果。

- Offsets大小为128*128，通道为2（分别为边角点x，y的偏移量）。
- 由于用热图来表示某一点的xy坐标，会损失精度。因此对每个边角点预测其真实位置的偏移量。举个例子，假设通过热图预测出某一左上角点的坐标是（50，50），该坐标的对应尺度是热图的128*128，而真实的坐标从网络的输入尺度511*511，映射到128*128时是（50.2,50.6）这样就损失了精度了。现在为（50，50）预测了偏移量为（0.3，0.5），那么可以计算出网络最终的预测坐标为（50.3，50.5）。尽管比真正的坐标（50.2，50.6）差了一点，但比（50，50）还是精确了许多。



- offsets损失函数

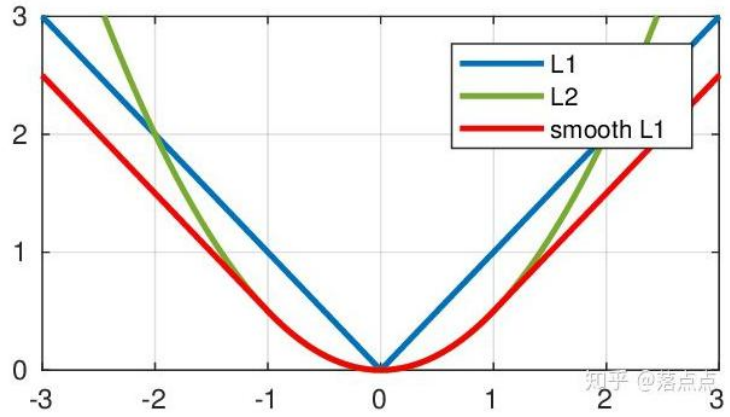
$$L_{off} = \frac{1}{N}\sum_{k=1}^{N} \text{SmoothL1Loss}\left(\boldsymbol{o}_k, \hat{\boldsymbol{o}}_k\right)$$

- N是物体个数。
- 其中$o_k$ 是相对原图坐标的偏移量。

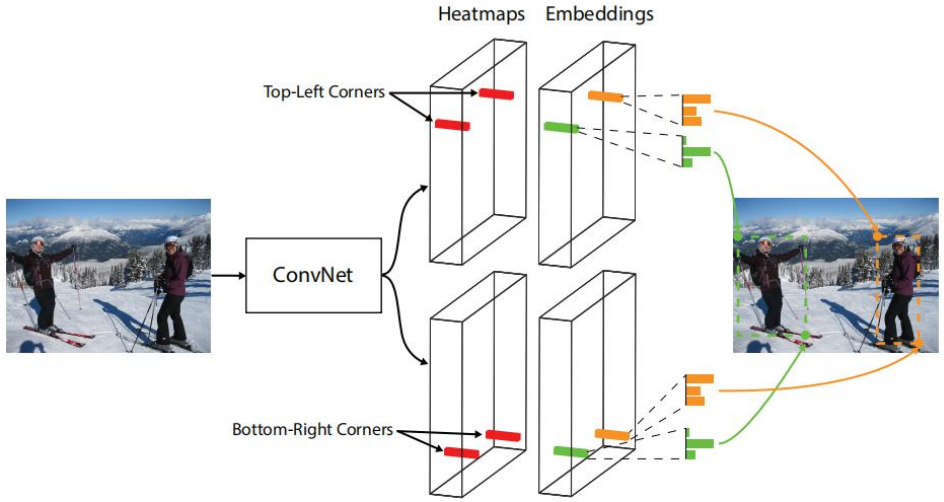
$$\boldsymbol{o}_k = \left(\frac{x_k}{n} - \left\lfloor \frac{x_k}{n} \right\rfloor, \frac{y_k}{n} - \left\lfloor \frac{y_k}{n} \right\rfloor\right)$$

- $x_k$和 $y_k$是角点k的在原图的坐标。

- 使用SmoothL1Loss损失函数
- L2值较大时，梯度值过大。
- L1值较小时，梯度值不够小。



- Embedding大小为128*128，通道为1。
- 假设两个预测模块已经分别预测出了100个左上角点，和100个右下角点。
- 使用Embedding来对左上角点和右上角点进行配对，两两组合。
- Top-left预测模块和Bottom-right预测模块都有一个128*128的Embedding，这两个Embedding中的每个值相当于一个标签。假设在Top-left预测模块里的Embedding的（50，50）位置的值是10（这个值是多少无所谓），那么其表示Top-left位置为（50，50）的左上角点的标签是10。如果，在Bottom-right预测模块里的Embedding的（100，100）位置的值也是10。通过Embedding的值之差，那么就认为左上角坐标（50，50）和右下角坐标（100，100）属于同一个bbox。在实际应用中，属于同一个bbox的左上角和右下角的embedding值一般不会完全一样，所以论文在判断左上角点和右上角点是否属于同一bbox的时是通过判断两个点所对应的embedding值的距离。



- Embedding损失函数

$$L_{pull} = \frac{1}{N}\sum_{k=1}^{N}\left[(e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2\right],$$

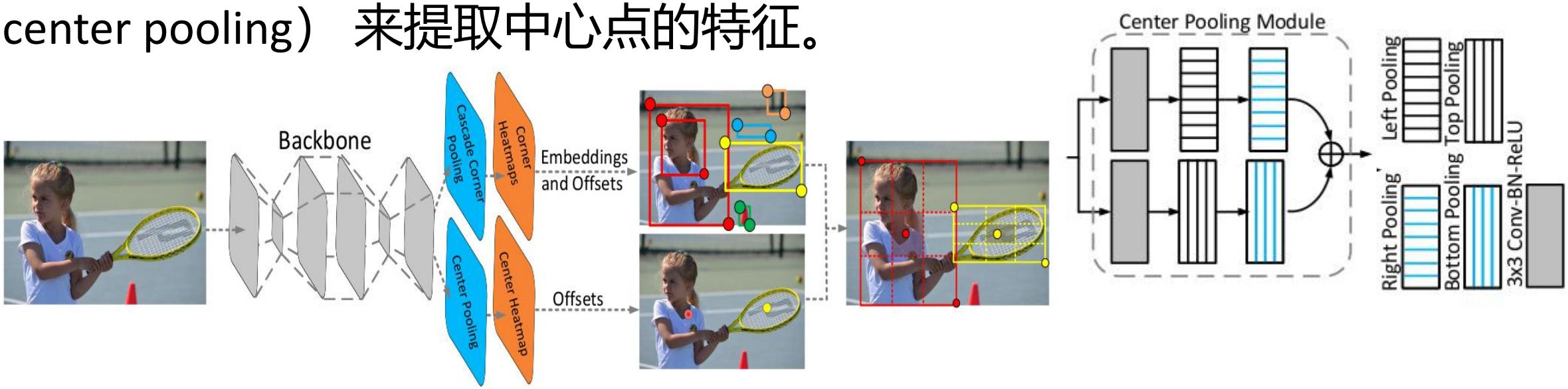
$$L_{push} = \frac{1}{N(N-1)}\sum_{k=1}^{N}\sum_{\substack{j=1 \\ j\neq k}}^{N}\max\left(0, \Delta - |e_k - e_j|\right),$$

- $e_{tk}$：物体k左上角点的编码
- $e_{bk}$：物体k右下角点的编码
- $e_k$:$e_{tk}$和$e_{bk}$的平均值
- Δ： margin,这里取1
- Lpull用来缩小属于同一个目标（k类目标）的两个角点的编码。
- Lpush用来扩大不属于同一个目标的两个角点的编码。
- Lpull损失函数使同一目标的顶点进行分组，Lpush损失函数用于分离不同目标的顶点。
- 训练模型使Lpull和Lpush趋向0。使相同类别的编码都趋近其平均数以使其相等，将不同类别的编码之间距离都大于1。

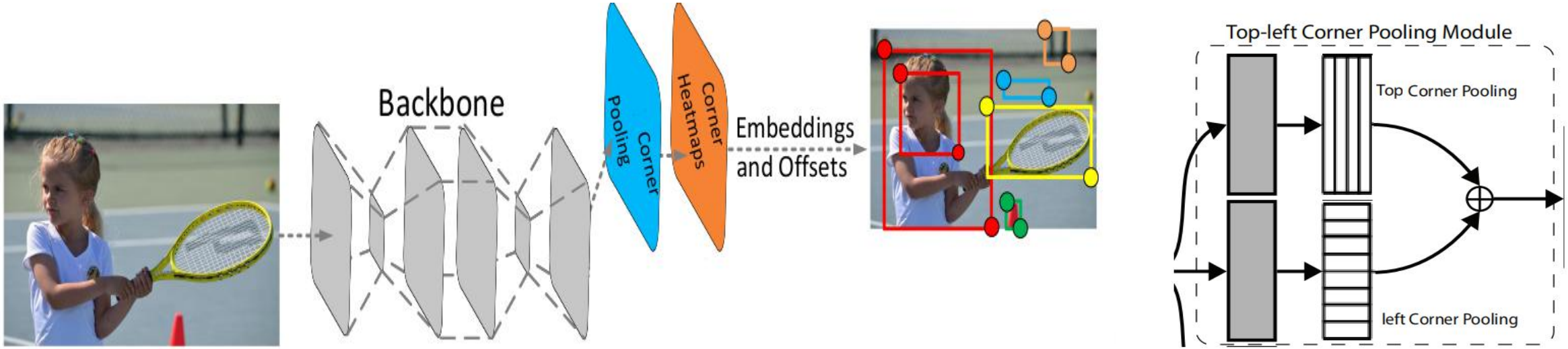
# 使用heatmaps,embeddings 和offsets得到目标框步骤

- （1）在heatmaps上做NMS,也就是3x3的max poolings,选择top 100的左上角点和top 100的右下角点
- （2）通过预测的offsets调整在原图中的位置
- （3）计算左上角点和右下角点的编码距离，通过筛掉编码距离大于0.5的以及不属于同一类的组合成矩形框
- （4）两个角点得分相加取平均得到最终矩形框的得分

- 特征提取网络后，中心网络（CenterNet）提出了另一个分支 中心池化（center pooling） 来提取中心点的特征。

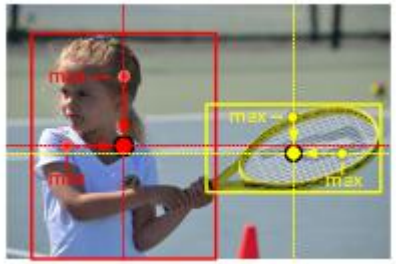


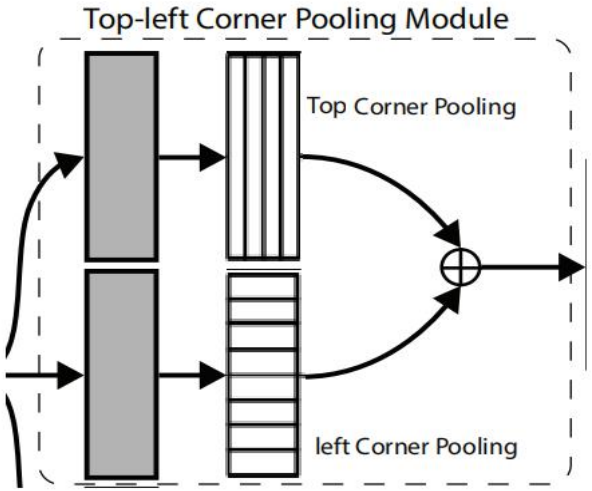

- 此池化由角点网络（cornernet）的角点池化（corner pooling ）启发而来。

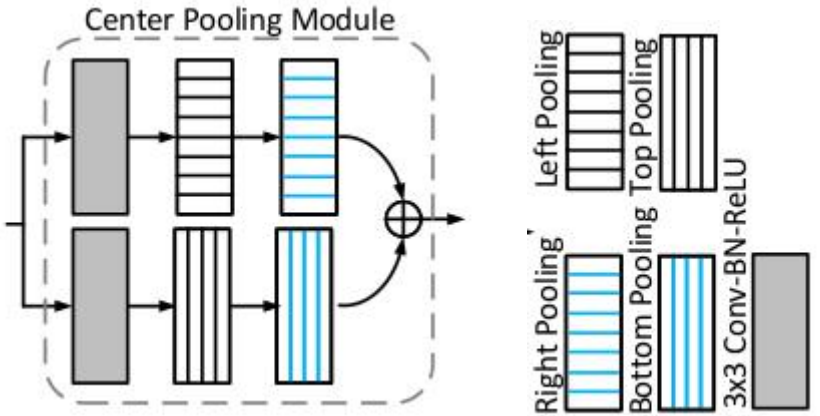


# center pooling

- 一个物体的中心并不一定含有很强的，易于区分于其他类别的语义信息。例如，一个人的头部含有很强的，易于区分于其他类别的语义信息，但是其中心往往位于身体的中部（如右图红框所示）。
- 为了解决中心位置特征对物体的表征能力不够的问题，centernet提取中心点水平方向和垂直方向最大值并相加。
- Center pooling 通过不同方向上的 corner pooling 的组合实现。一个水平方向上的取最大值操作可由 left pooling 和 right pooling通过串联实现，同理，一个垂直方向上的取最大值操作可由 top pooling 和 bottom pooling通过串联实现。



# 使用中心点过滤预测框步骤

- 根据中心点的heatmap的得分，选取top-k个center keypoints。
- 使用对应的offsets将点微调还原到原图上，得到精度更高的中心点位置。
- 根据left-top和right-bottom点得到的所有bounding box。为每一个bbox定义出一个中心区域。
- 判断中心区域是否存在中心点，且类别是否相同。如果存在且相同，则保留bbox，分数为三点分数平均数，反之，移除该bbox。

# 损失函数

$$\mathrm{L} = \mathrm{L}_{\mathrm{det}}^{\mathrm{co}} + \mathrm{L}_{\mathrm{det}}^{\mathrm{ce}} + \alpha \mathrm{L}_{\mathrm{pull}}^{\mathrm{co}} + \beta \mathrm{L}_{\mathrm{push}}^{\mathrm{co}} + \gamma \left( \mathrm{L}_{\mathrm{off}}^{\mathrm{co}} + \mathrm{L}_{\mathrm{off}}^{\mathrm{ce}} \right)$$

co是corner points（角点）。ce是center point（中心点）。

权重α=β=0.1,γ=1。中心点不用配对，所以没有 $\alpha L_{pull}^{ce} + \beta L_{push}^{ce}$

$\mathrm{L}_{\mathrm{det}}^{\mathrm{co}}$和$\mathrm{L}_{\mathrm{det}}^{\mathrm{ce}}$是角点和中心点的 focal loss损失

$$L_{det} = \frac{-1}{N} \sum_{c=1}^{C} \sum_{i=1}^{H} \sum_{j=1}^{W} \begin{cases} (1 - p_{cij})^{\alpha} \log (p_{cij}) & \text{if } y_{cij} = 1 \\ (1 - y_{cij})^{\beta} (p_{cij})^{\alpha} \log (1 - p_{cij}) & \text{otherwise} \end{cases}$$

$L_{pull}^{co}$是用来缩来缩小属于同一标两个角点的编码

$L_{push}^{co}$是用来扩来扩大不属于同目标标的两个角点的编码

$$L_{pull} = \frac{1}{N} \sum_{k=1}^{N} \left[ (e_{t_k} - e_k)^2 + (e_{b_k} - e_k)^2 \right],$$

$$L_{push} = \frac{1}{N(N-1)} \sum_{k=1}^{N} \sum_{\substack{j=1 \\ j \neq k}}^{N} \max (0, \Delta - |e_k - e_j|),$$

$\mathrm{L}_{\mathrm{off}}^{\mathrm{co}}$和$\mathrm{L}_{\mathrm{off}}^{\mathrm{ce}}$是角点和中心点 在特征图上的坐标相对于原图的偏移

$$\boldsymbol{o}_k = \left( \frac{x_k}{n} - \left\lfloor \frac{x_k}{n} \right\rfloor, \frac{y_k}{n} - \left\lfloor \frac{y_k}{n} \right\rfloor \right)$$

$$L_{off} = \frac{1}{N} \sum_{k=1}^{N} \mathrm{SmoothL1Loss} (\boldsymbol{o}_k, \hat{\boldsymbol{o}}_k)$$

# CenterNet与主流检测网络对比表

CenterNet511表示输入图像分辨率 511x511

| Method | Backbone | Train input | Test input | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $AR_1$ | $AR_{10}$ | $AR_{100}$ | $AR_S$ | $AR_M$ | $AR_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Two-stage:** | | | | | | | | | | | | | | | |
| DeNet [40] | ResNet-101 [14] | 512×512 | 512×512 | 33.8 | 53.4 | 36.1 | 12.3 | 36.1 | 50.8 | 29.6 | 42.6 | 43.5 | 19.2 | 46.9 | 64.3 |
| CoupleNet [47] | ResNet-101 | ori. | ori. | 34.4 | 54.8 | 37.2 | 13.4 | 38.1 | 50.8 | 30.0 | 45.0 | 46.4 | 20.7 | 53.1 | 68.5 |
| Faster R-CNN by G-RMI [16] | Inception-ResNet-v2 [39] | ∼ 1000×600 | ∼ 1000×600 | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 | - | - | - | - | - | - |
| Faster R-CNN +++ [14] | ResNet-101 | ∼ 1000×600 | ∼ 1000×600 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 | - | - | - | - | - | - |
| Faster R-CNN w/ FPN [23] | ResNet-101 | ∼ 1000×600 | ∼ 1000×600 | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 | - | - | - | - | - | - |
| Faster R-CNN w/ TDM [37] | Inception-ResNet-v2 | - | - | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | 52.1 | **31.6** | **49.3** | **51.9** | **28.1** | **56.6** | **71.1** |
| D-FCN [7] | Aligned-Inception-ResNet | ∼ 1000×600 | ∼ 1000×600 | 37.5 | 58.0 | - | 19.4 | 40.1 | 52.5 | - | - | - | - | - | - |
| Regionlets [43] | ResNet-101 | ∼ 1000×600 | ∼ 1000×600 | 39.3 | 59.8 | - | 21.7 | 43.7 | 50.9 | - | - | - | - | - | - |
| Mask R-CNN [12] | ResNeXt-101 | ∼ 1300×800 | ∼ 1300×800 | 39.8 | 62.3 | 43.4 | 22.1 | 43.2 | 51.2 | - | - | - | - | - | - |
| Soft-NMS [2] | Aligned-Inception-ResNet | ∼ 1300×800 | ∼ 1300×800 | 40.9 | 62.8 | - | 23.3 | 43.6 | 53.3 | - | - | - | - | - | - |
| Fitness R-CNN [41] | ResNet-101 | 512×512 | 1024×1024 | 41.8 | 60.9 | 44.9 | 21.5 | 45.0 | 57.5 | - | - | - | - | - | - |
| Cascade R-CNN [4] | ResNet-101 | - | - | 42.8 | 62.1 | 46.3 | 23.7 | 45.5 | 55.2 | - | - | - | - | - | - |
| Grid R-CNN w/ FPN [28] | ResNeXt-101 | ∼ 1300×800 | ∼ 1300×800 | 43.2 | 63.0 | 46.6 | 25.1 | 46.5 | 55.2 | - | - | - | - | - | - |
| D-RFCN + SNIP (multi-scale) [38] | DPN-98 [5] | ∼ 2000×1200 | ∼ 2000×1200 | 45.7 | **67.3** | 51.1 | 29.3 | 48.8 | 57.1 | - | - | - | - | - | - |
| PANet (multi-scale) [26] | ResNeXt-101 | ∼ 1400×840 | ∼ 1400×840 | **47.4** | 67.2 | **51.8** | **30.1** | **51.7** | **60.0** | - | - | - | - | - | - |
| **One-stage:** | | | | | | | | | | | | | | | |
| YOLOv2 [32] | DarkNet-19 | 544×544 | 544×544 | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 | 20.7 | 31.6 | 33.3 | 9.8 | 36.5 | 54.4 |
| DSOD300 [34] | DS/64-192-48-1 | 300×300 | 300×300 | 29.3 | 47.3 | 30.6 | 9.4 | 31.5 | 47.0 | 27.3 | 40.7 | 43.0 | 16.7 | 47.1 | 65.0 |
| GRP-DSOD320 [35] | DS/64-192-48-1 | 320×320 | 320×320 | 30.0 | 47.9 | 31.8 | 10.9 | 33.6 | 46.3 | 28.0 | 42.1 | 44.5 | 18.8 | 49.1 | 65.0 |
| SSD513 [27] | ResNet-101 | 513×513 | 513×513 | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 | 28.3 | 42.1 | 44.4 | 17.6 | 49.2 | 65.8 |
| DSSD513 [8] | ResNet-101 | 513×513 | 513×513 | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 | 28.9 | 43.5 | 46.2 | 21.8 | 49.1 | 66.4 |
| RefineDet512 (single-scale) [45] | ResNet-101 | 512×512 | 512×512 | 36.4 | 57.5 | 39.5 | 16.6 | 39.9 | 51.4 | - | - | - | - | - | - |
| CornerNet511 (single-scale) [20] | Hourglass-52 | 511×511 | ori. | 37.8 | 53.7 | 40.1 | 17.0 | 39.0 | 50.5 | 33.9 | 52.3 | 57.0 | 35.0 | 59.3 | 74.7 |
| RetinaNet800 [24] | ResNet-101 | 800×800 | 800×800 | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 | - | - | - | - | - | - |
| CornerNet511 (multi-scale) [20] | Hourglass-52 | 511×511 | ≤1.5× | 39.4 | 54.9 | 42.3 | 18.9 | 41.2 | 52.7 | 35.0 | 53.5 | 57.7 | 36.1 | 60.1 | 75.1 |
| CornerNet511 (single-scale) [20] | Hourglass-104 | 511×511 | ori. | 40.5 | 56.5 | 43.1 | 19.4 | 42.7 | 53.9 | 35.3 | 54.3 | 59.1 | 37.4 | 61.9 | 76.9 |
| RefineDet512 (multi-scale) [45] | ResNet-101 | 512×512 | ≤2.25× | 41.8 | 62.9 | 45.7 | 25.6 | 45.1 | 54.1 | | | | | | |
| CornerNet511 (multi-scale) [20] | Hourglass-104 | 511×511 | ≤1.5× | 42.1 | 57.8 | 45.3 | 20.8 | 44.8 | 56.7 | 36.4 | 55.7 | 60.0 | 38.5 | 62.7 | 77.4 |
| **CenterNet511** (single-scale) | Hourglass-52 | 511×511 | ori. | 41.6 | 59.4 | 44.2 | 22.5 | 43.1 | 54.1 | 34.8 | 55.7 | 60.1 | 38.6 | 63.3 | 76.9 |
| **CenterNet511** (single-scale) | Hourglass-104 | 511×511 | ori. | 44.9 | 62.4 | 48.1 | 25.6 | 47.4 | 57.4 | 36.1 | 58.4 | 63.3 | 41.3 | 67.1 | 80.2 |
| **CenterNet511** (multi-scale) | Hourglass-52 | 511×511 | ≤1.8× | 43.5 | 61.3 | 46.7 | 25.3 | 45.3 | 55.0 | 36.0 | 57.2 | 61.3 | 41.4 | 64.0 | 76.3 |
| **CenterNet511** (multi-scale) | Hourglass-104 | 511×511 | ≤1.8× | **47.0** | **64.5** | **50.7** | **28.9** | **49.9** | **58.9** | **37.5** | **60.3** | **64.8** | **45.1** | **68.3** | **79.7** |

Table 2: Performance comparison (%) with the state-of-the-art methods on the MS-COCO test-dev dataset. CenterNet outperforms all existing one-stage detectors by a large margin and ranks among the top of state-of-the-art two-stage detectors.

注：训练时，使用了训练集和验证集；测试时，使用0填充将图片长宽扩充到128的倍数，得到输入图片。将输入图片和水平翻转后的图片同时输入网络，使用soft-nms消除两者得到的冗余框，根据得分选取前一百个框作为最后结果。

| Method | Backbone | Train input | Test input | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $AR_1$ | $AR_{10}$ | $AR_{100}$ | $AR_S$ | $AR_M$ | $AR_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Two-stage:** | | | | | | | | | | | | | | | |
| DeNet [40] | ResNet-101 [14] | 512×512 | 512×512 | 33.8 | 53.4 | 36.1 | 12.3 | 36.1 | 50.8 | 29.6 | 42.6 | 43.5 | 19.2 | 46.9 | 64.3 |
| CoupleNet [47] | ResNet-101 | ori. | ori. | 34.4 | 54.8 | 37.2 | 13.4 | 38.1 | 50.8 | 30.0 | 45.0 | 46.4 | 20.7 | 53.1 | 68.5 |
| Faster R-CNN by G-RMI [16] | Inception-ResNet-v2 [39] | ~ 1000×600 | ~ 1000×600 | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 | - | - | - | - | - | - |
| Faster R-CNN +++ [14] | ResNet-101 | ~ 1000×600 | ~ 1000×600 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 | - | - | - | - | - | - |
| Faster R-CNN w/ FPN [23] | ResNet-101 | ~ 1000×600 | ~ 1000×600 | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 | - | - | - | - | - | - |
| Faster R-CNN w/ TDM [37] | Inception-ResNet-v2 | - | - | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | 52.1 | **31.6** | **49.3** | **51.9** | **28.1** | **56.6** | **71.1** |
| D-FCN [7] | Aligned-Inception-ResNet | ~ 1000×600 | ~ 1000×600 | 37.5 | 58.0 | - | 19.4 | 40.1 | 52.5 | - | - | - | - | - | - |
| Regionlets [43] | ResNet-101 | ~ 1000×600 | ~ 1000×600 | 39.3 | 59.8 | - | 21.7 | 43.7 | 50.9 | - | - | - | - | - | - |
| Mask R-CNN [12] | ResNeXt-101 | ~ 1300×800 | ~ 1300×800 | 39.8 | 62.3 | 43.4 | 22.1 | 43.2 | 51.2 | - | - | - | - | - | - |
| Soft-NMS [2] | Aligned-Inception-ResNet | ~ 1300×800 | ~ 1300×800 | 40.9 | 62.8 | - | 23.3 | 43.6 | 53.3 | - | - | - | - | - | - |
| Fitness R-CNN [41] | ResNet-101 | 512×512 | 1024×1024 | 41.8 | 60.9 | 44.9 | 21.5 | 45.0 | 57.5 | - | - | - | - | - | - |
| Cascade R-CNN [4] | ResNet-101 | - | - | 42.8 | 62.1 | 46.3 | 23.7 | 45.5 | 55.2 | - | - | - | - | - | - |
| Grid R-CNN w/ FPN [28] | ResNeXt-101 | ~ 1300×800 | ~ 1300×800 | 43.2 | 63.0 | 46.6 | 25.1 | 46.5 | 55.2 | - | - | - | - | - | - |
| D-RFCN + SNIP (multi-scale) [38] | DPN-98 [5] | ~ 2000×1200 | ~ 2000×1200 | 45.7 | **67.3** | 51.1 | 29.3 | 48.8 | 57.1 | - | - | - | - | - | - |
| PANet (multi-scale) [26] | ResNeXt-101 | ~ 1400×840 | ~ 1400×840 | **47.4** | 67.2 | **51.8** | **30.1** | **51.7** | **60.0** | - | - | - | - | - | - |
| **One-stage:** | | | | | | | | | | | | | | | |
| YOLOv2 [32] | DarkNet-19 | 544×544 | 544×544 | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 | 20.7 | 31.6 | 33.3 | 9.8 | 36.5 | 54.4 |
| DSOD300 [34] | DS/64-192-48-1 | 300×300 | 300×300 | 29.3 | 47.3 | 30.6 | 9.4 | 31.5 | 47.0 | 27.3 | 40.7 | 43.0 | 16.7 | 47.1 | 65.0 |
| GRP-DSOD320 [35] | DS/64-192-48-1 | 320×320 | 320×320 | 30.0 | 47.9 | 31.8 | 10.9 | 33.6 | 46.3 | 28.0 | 42.1 | 44.5 | 18.8 | 49.1 | 65.0 |
| SSD513 [27] | ResNet-101 | 513×513 | 513×513 | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 | 28.3 | 42.1 | 44.4 | 17.6 | 49.2 | 65.8 |
| DSSD513 [8] | ResNet-101 | 513×513 | 513×513 | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 | 28.9 | 43.5 | 46.2 | 21.8 | 49.1 | 66.4 |
| RefineDet512 (single-scale) [45] | ResNet-101 | 512×512 | 512×512 | 36.4 | 57.5 | 39.5 | 16.6 | 39.9 | 51.4 | - | - | - | - | - | - |
| CornerNet511 (single-scale) [20] | Hourglass-52 | 511×511 | ori. | 37.8 | 53.7 | 40.1 | 17.0 | 39.0 | 50.5 | 33.9 | 52.3 | 57.0 | 35.0 | 59.3 | 74.7 |
| RetinaNet800 [24] | ResNet-101 | 800×800 | 800×800 | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 | - | - | - | - | - | - |
| CornerNet511 (multi-scale) [20] | Hourglass-52 | 511×511 | ≤1.5× | 39.4 | 54.9 | 42.3 | 18.9 | 41.2 | 52.7 | 35.0 | 53.5 | 57.7 | 36.1 | 60.1 | 75.1 |
| CornerNet511 (single-scale) [20] | Hourglass-104 | 511×511 | ori. | 40.5 | 56.5 | 43.1 | 19.4 | 42.7 | 53.9 | 35.3 | 54.3 | 59.1 | 37.4 | 61.9 | 76.9 |
| RefineDet512 (multi-scale) [45] | ResNet-101 | 512×512 | ≤2.25× | 41.8 | 62.9 | 45.7 | 25.6 | 45.1 | 54.1 | | | | | | |
| CornerNet511 (multi-scale) [20] | Hourglass-104 | 511×511 | ≤1.5× | 42.1 | 57.8 | 45.3 | 20.8 | 44.8 | 56.7 | 36.4 | 55.7 | 60.0 | 38.5 | 62.7 | 77.4 |
| **CenterNet511** (single-scale) | Hourglass-52 | 511×511 | ori. | 41.6 | 59.4 | 44.2 | 22.5 | 43.1 | 54.1 | 34.8 | 55.7 | 60.1 | 38.6 | 63.3 | 76.9 |
| **CenterNet511** (single-scale) | Hourglass-104 | 511×511 | ori. | 44.9 | 62.4 | 48.1 | 25.6 | 47.4 | 57.4 | 36.1 | 58.4 | 63.3 | 41.3 | 67.1 | 80.2 |
| **CenterNet511** (multi-scale) | Hourglass-52 | 511×511 | ≤1.8× | 43.5 | 61.3 | 46.7 | 25.3 | 45.3 | 55.0 | 36.0 | 57.2 | 61.3 | 41.4 | 64.0 | 76.3 |
| **CenterNet511** (multi-scale) | Hourglass-104 | 511×511 | ≤1.8× | **47.0** | **64.5** | **50.7** | **28.9** | **49.9** | **58.9** | **37.5** | **60.3** | **64.8** | **45.1** | **68.3** | **79.7** |

CenterNet511-52表示输入图像分辨率511x511，52表示Hourglass-52的basebone

- CenterNet511-52在单一尺寸下测试AP值41.6%，比CornerNet511-52的37.8%提升了3.8%，多尺寸的AP值为43.5%，比CornerNet511-52的39.4%提升了4.1%，当基础网络使用Hourglass-104时，在单尺寸和多尺寸上比CornerNet提升了4.4%（从40.5%到44.9%）和4.9%（42.1%到47.0%）。
- 在小目标上CenterNet511-52的AP提升5.5%（单尺寸）和6.4%(多尺寸)，CenterNet511-104的AP提升6.2%（单尺寸）和8.1%(多尺寸)。
- 说明误检框越小，中心区域存在中心关键点的可能越低。CenterNet能有效抑制小目标误检框。
- AR在多尺寸下有良好表现，因为CenterNet移除了大量误检框，相当于提升了那些定位准确但是得分较低的预测框置信度。
- two-satge方法通常使用更高分辨率的图片作为输入，这会显著提高准确率，尤其在小目标的检测上。但是单尺寸的CenterNet511-52的AP值媲美Fitness R-CNN(41.6%vs41.8%)，CenterNet511-104媲美D-RFCN和SNIP(44.9%vs45.7%)。多尺寸的CenterNet511-104的AP值47.0%接近PANet（47.4%）。

# 与CornerNet对比

| Method | FD | $FD_5$ | $FD_{25}$ | $FD_{50}$ | $FD_S$ | $FD_M$ | $FD_L$ |
|---|---|---|---|---|---|---|---|
| CornerNet511-52 | 40.4 | 35.2 | 39.4 | 46.7 | 62.5 | 36.9 | 28.0 |
| CenterNet511-52 | **35.1** | **30.7** | **34.2** | **40.8** | **53.0** | **31.3** | **24.4** |
| CornerNet511-104 | 37.8 | 32.7 | 36.8 | 43.8 | 60.3 | 33.2 | 25.1 |
| CenterNet511-104 | **32.4** | **28.2** | **31.6** | **37.5** | **50.7** | **27.1** | **23.0** |

- AP反应一个网络能预测多少高质量目标框（通常IOU>=0.5）,而不能直接反应多少不正确的目标框（通常IOU<<0.5)。
- FD=1-AP，使用FD来直观衡量误检框。
- 在IoU=0.05时，CornerNet511-52和CornerNet511-104有较高的FD值35.2%和32.7%,和中目标和大目标对比，在小目标上有显著提高的FD值62.5%和60.3%。
- CenterNet在这些值上都得到了降低。
- 结果表明 CenterNet 相对于CornerNet去除了大量的错误目标框，尤其是小尺度的错误目标框。
- 使用NVIDIA Tesla P100 GPU,CornerNet511-104速度300ms/pic，CenterNet511-104速度340ms/pic,CenterNet511-52速度270ms/pic。

# 消融实验（Ablation Study）

| CRE | CTP | CCP | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | $AR_1$ | $AR_{10}$ | $AR_{100}$ | $AR_S$ | $AR_M$ | $AR_L$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 37.6 | 53.3 | 40.0 | 18.5 | 39.6 | 52.2 | 33.7 | 52.2 | 56.7 | 37.2 | 60.0 | 74.0 |
| | | ✓ | 38.3 | 54.2 | 40.5 | 18.6 | 40.5 | 52.2 | 34.0 | 53.0 | 57.9 | 36.6 | 60.8 | 75.8 |
| ✓ | | | 39.9 | 57.7 | 42.3 | 23.1 | 42.3 | 52.3 | 33.8 | 54.2 | 58.5 | 38.7 | 62.4 | 74.4 |
| ✓ | ✓ | | 40.8 | 58.6 | 43.6 | 23.6 | 43.6 | 53.6 | 33.9 | 54.5 | 59.0 | 39.0 | 63.2 | 74.7 |
| ✓ | ✓ | ✓ | **41.3** | **59.2** | **43.9** | **23.6** | **43.8** | **55.8** | **34.5** | **55.0** | **59.2** | **39.1** | **63.5** | **75.1** |

Table 4: Ablation study on the major components of CenterNet511-52 on the MS-COCO validation dataset. The CRE denotes central region exploration, the CTP denotes center pooling, and the CCP denotes cascade corner pooling.

- 基础网络：CornerNet511-52
- CRE(中心区域扩展)：增加一个输出中心点热力图的分支，使用常规卷积层而不是中心点池化（center pooling）预测中心点，使用三点预测目标框。AP值提升了2.3%(37.6%到39.9%)。对小目标提升大（4.6%），大目标几乎无提升（52.2%到52.3%）。这可能因为小目标中心点比大目标更好定位。
- CTP（中心点池化）：使用CTP后，AP值提升0.9%（39.9%到40.8%）。对大目标的AP值提升1.4%（52.2%到53.6%），效果比常规卷积层明显（1.4%vs0.1%）。说明中心池化对检测目标中心关键点是有效的，尤其是大目标。中心池化会获取更丰富的图像内部信息，而大目标能获得的内部信息更多。
- CCP（级联角点池化）：CenterNet511-52使用cascade corner pooling比使用corner pooling，AP提升0.5%（40.8%到41.3%）。CornerNet511-52使用cascade corner pooling替代corner pooling,AP提升0.7%（37.6%到38.3%），大目标AP值无提升（52.2%和52.2%），但是AR从74.0%提升到75.8%，表明使用CCP能看到更丰富的内部信息，但是过于丰富的内部信息会干扰边界判断，产生许多误检框。使用CenterNet可以有效去除误检框，大目标AP提升了2.2%（53.6%到55.8%）。

# 误差分析（Error Analysis）

| Method | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|
| CenterNet511-52 w/o GT | 41.3 | 59.2 | 43.9 | 23.6 | 43.8 | 55.8 |
| CenterNet511-52 w/ GT | **56.5** | **78.3** | **61.4** | **39.1** | **60.3** | **70.3** |
| CenterNet511-104 w/o GT | 44.8 | 62.4 | 48.2 | 25.9 | 48.9 | 58.8 |
| CenterNet511-104 w/ GT | **58.1** | **78.4** | **63.9** | **40.4** | **63.0** | **72.1** |

- 通过使用真实的中心点替代预测的中心点，
- CenterNet511-52的AP值从41.3%提升到56.5%，
- CenterNet511-104的AP值从44.8%提升到58.1%；
- CenterNet511-52对小目标，中目标和大目标的AP值分别提升了15.5%，16.5%，14.5%，
- CenterNet511-104对小目标，中目标和大目标的AP值分别提升了14.5%，14.1%，13.3%。
- 结果表明中心点的检测准确度还有很大的提升空间。center pooling并没有在候选区域的中心区域里很好的把中心点预测出来。

# 谢谢聆听