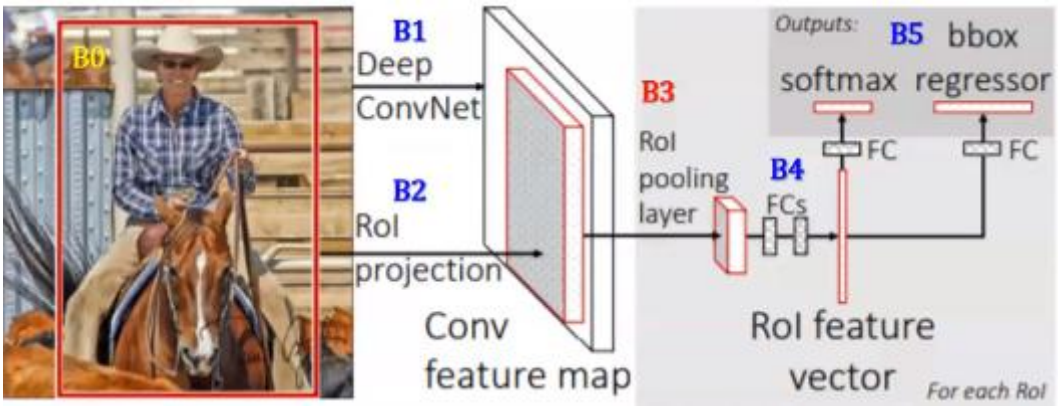
目标检测

RCNN

1. 首先通过 Selective Search 得到 2000 个框。非监督算法，通过计算每一个像素和周围像素的相似度，把图像分为 N 多块，然后把块合并或细分成 2000 个。
2. 然后把这 2000 个框从原图抠出来，resize 到同样大小，送进 CNN+FC 得到 feature vector。
3. 用 SVM 或其它算法将 feature vector 分类，打分。
4. 用 NMS 去重。

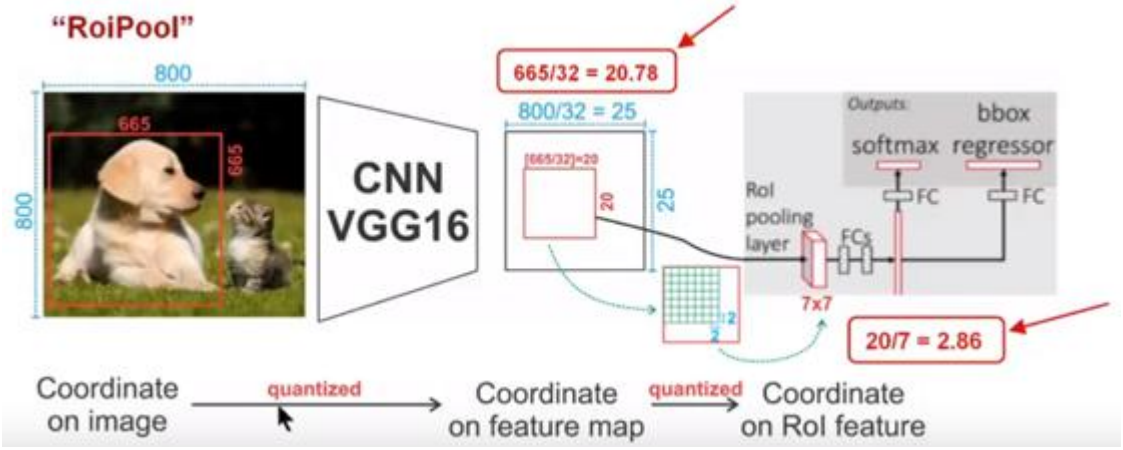
Fast-RCNN

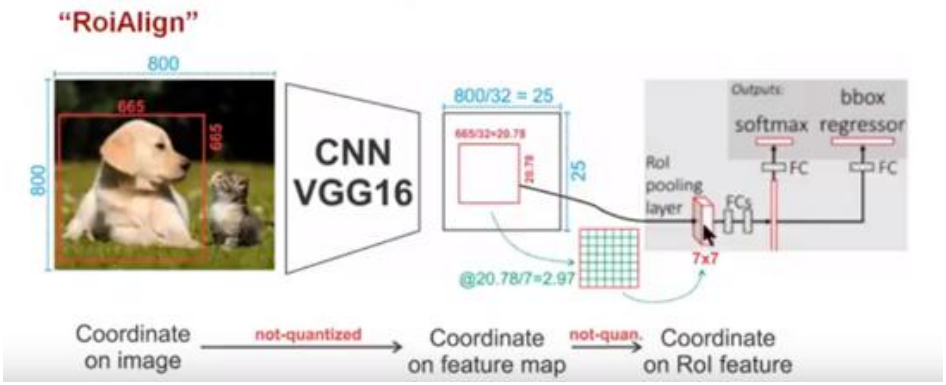


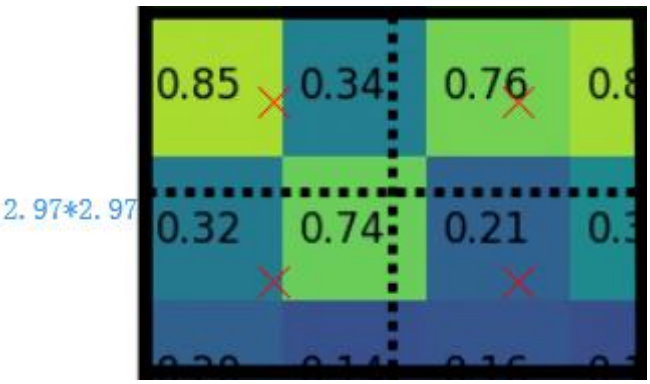
1. SS 和 RCNN 一致。
2. 然后把原图送进 CNN 得到一个 feature map。
3. 用 ROI Projection 得到 Proposal 每一个框的坐标点。简单的除法就行。
4. 用 ROI Pooling 将前一步得到的每个框都变成同一大小。
   (1) 为什么要做这一步，是因为后面跟了 fc，以后的算法用的全卷积的，就不需要这一步了。下采样后的每一个框的大小都是不一样的,怎么接 FC 呢,这里用了 SPP-NET 的思想，就是打格子。将下采样后的框打格子，比如打 7*7 个格子，然后在每个格子中 max-pool 一下，提取一个像素，这样就每个框都变成 7*7，就可以进 fc 了。但是这样会带来一个 Quantized 的问题。提取像素的框和原图的框之间有 offset，对于小图像来说，这是很致命的。比如一个 196*196 的图像，经过 8 的下采样，是 24.5，取整为 24，再打成 7*7 的格子，就是 3.43，取整为 3。这样就会有 20 个像素的偏移，如果小物体本身就只有 20 个像素大小，那错误就很大。

(2) 改进 ROI Pooling 为 ROI Align，在 projection 和打格子的过程中不取整，然后把 7*7 的格子也不取最大值，而是每个格子再分成四分，每份取中心点，中心点的值由周围的四个点插值得到。然后再在四个格子中取最大值。但是这样又没有用到格子内的所有信息。

(3) 再次改进 ROI Align 为 Precise ROI Pooling。还是在 projection 和打格子的过程中不取整，现在改为在 7*7 的格子中，对每个格子，计算有多少个点落在格子里，然后对这个格子，重新均匀打点。每个点按照周围四个点插值得到。然后求格子里面点的均值。

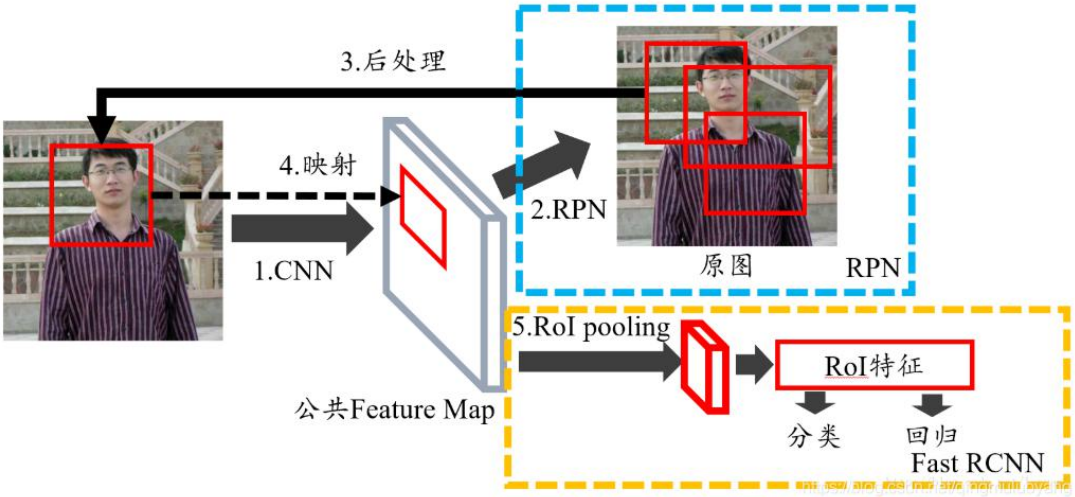




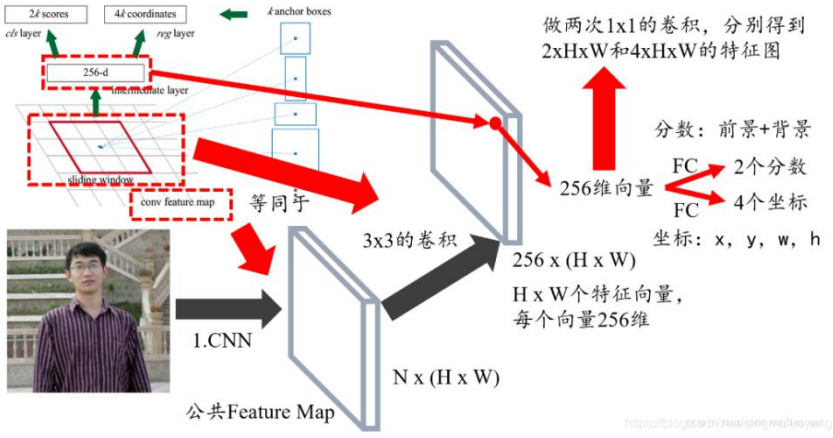


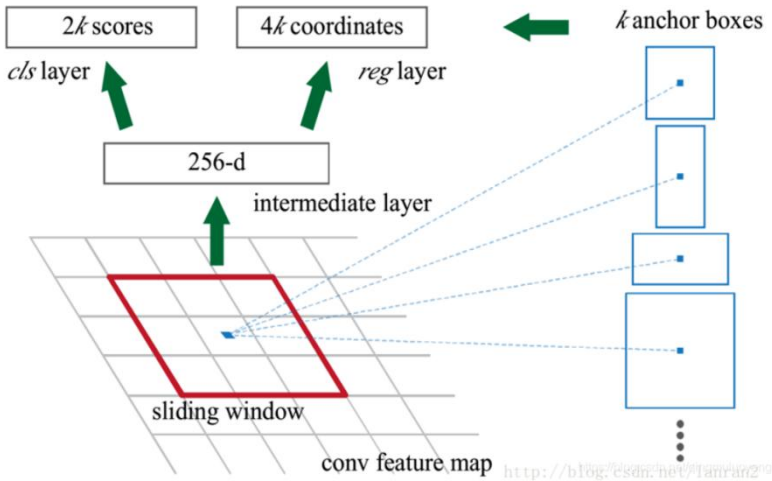
5. 送进 FC 得到 label 和 bbox（分类任务和回归任务）。

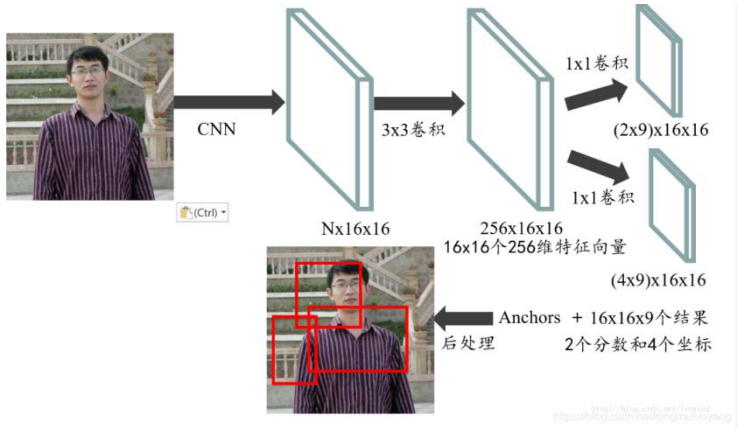
Faster-RCNN



1. 原图过 CNN 得到 feature map
2. feature map（H*W*C 维）过 RPN 得到 proposal。
   (1) 把 feature map 过 conv 得到同样大小 H*W*256 维的特征图。
   (2) 用 1*1 的卷积对每一个 256 维的特征点得到 H*W*2 个分数，H*W*4 个坐标。就是 H*W*6 维的特征图。
   (3) 2 个分数是前景/背景（分类是 ROI 后的事情，这里是得到 proposal）。
   (4) 4 个坐标点是 offset（x, y, w, h）。
   (5) 这里有个问题。取 feature map 的每一个点，对应原图是一个框。这个框就是下采样的倍数的大小。比如 8 倍下采样，那么框就是 8*8，这个明显不现实，因此预先设定 9 个 anchor boxes 来规定框的大小。
   (6) 1*1conv 后的特征图大小其实是 H*W*（2*9+4*9）维大小。







3. 在公共的 feature map 中取 proposal 对应的特征图
4. 用 ROI Pooling 将前一步得到的每个特征图都变成同一大小
5. 送进 FC 得到 label 和 bbox（分类任务和回归任务）

YOLO1

1. 基本流程

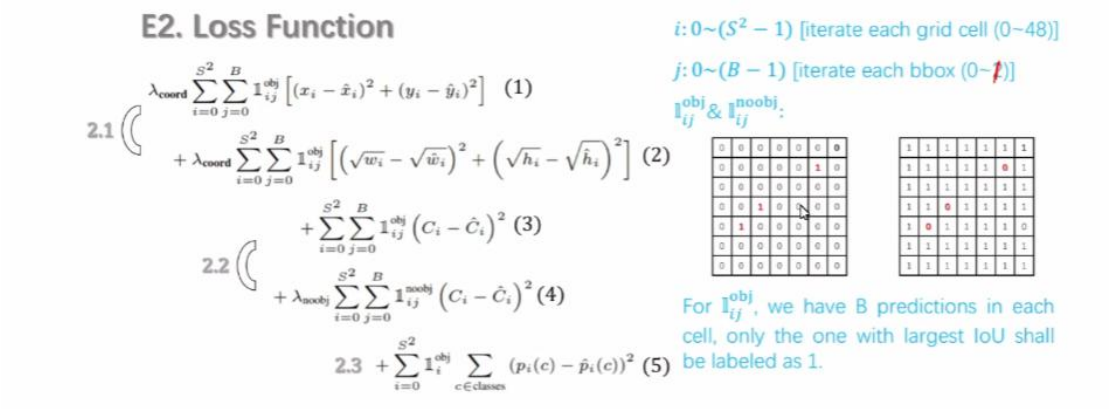
   (1) 原图过 CNN 得到 feature map。7*7*（N+（1+4）*2）维。

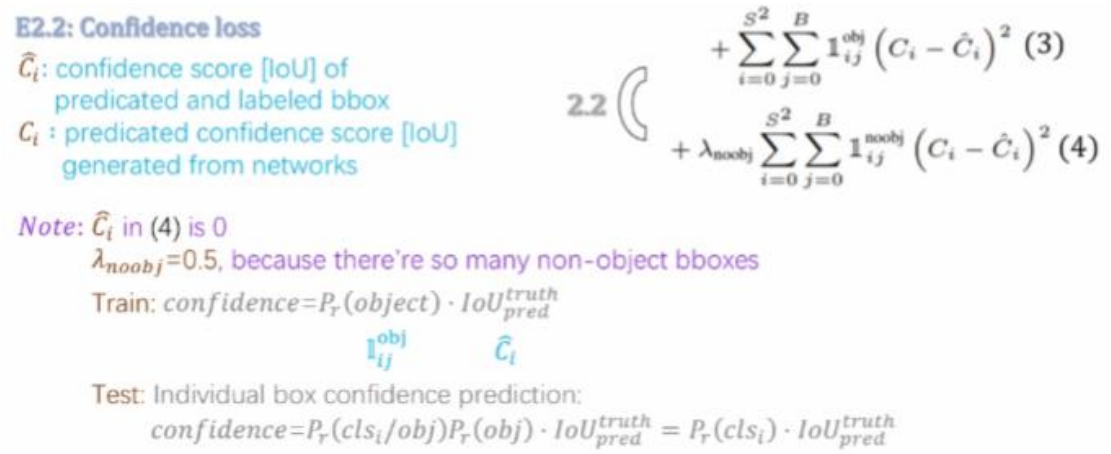
   (2) N 是类别数，对于 N，每一个 7*7*1 是一个 0/1 的特征图，如果为 1，表明这个类别的有一个目标的中心点落在了这个格子。Loss 值是预测这个格子有这个类别的物体的概率和真实值的 L2 距离。

   (3) 1+4 是一个 bbox，*2 是因为有两个 bbox。4 是（x1, y1, w1, h1）。x, y 是中心点，w, h 是宽高。Loss 是真实值和预测值的平方根的 L2 距离。

   (4) 1 是 confidence，是这个（x1, y1, w1, h1）对应的是不是物体的概率，以及这个（x1, y1, w1, h1）和 groundtruth 的 IoU 的值的乘积。Loss 是预测值和生成的

（x1,y1,w1,h1）和 groundtruth 的 IoU 的 L2 距离。





2. 优点是去掉了 Region Proposal。速度快。
3. 缺点是对小物体不友好，在算 bbox 的 loss 的时候直接用的 x,y,w,h 的值来算，大物体会占据 loss 中更大的比例，小物体得不到学习。中心点在同一个框的物体只能预测一个。稠密的物体不能预测。没有用 BN。对非特定形态的物体不好（带鱼）。

YOLO2

1. 基本流程
   (1) 原图过 CNN 得到 feature map。
   (2) 对 groundtruth 的（x,y,w,h）除以图像的高宽，归一化到[0,1]范围。然后*13，得到 13*13 大小下的坐标。然后（x,y）减去格子的 ID。（x,y）归一化到每个格子的[0,1]范围。然后（w,h）除以 anchor 的宽高。Anchor 选和 groundtruth 之间 IoU 最大的 anchor。网络输出的（x,y）要过 sigmoid，为了保证（x,y）在[0,1]之间。
   (3) 具体流程参见 YOLO3

2. 加了 BN
3. Backbone 用分辨率更高的图训练，224*224 的 ImageNet 训练时放大到 448*448。然后在目标检测的训练集上 finetune。
4. 从 7*7 提升到 13*13，feature map 变稠密。
5. 用了 anchors。anchor 是 Pre-defined virtual bboxes。和 target/label 一样都是已知数。最终的物体是由这个 anchors 优化而来。YOLO2 是用 k-means 聚出来 5 个不同大小和比例的 anchor boxes。一组 10 个数字[0.57273,0.677385,...,9.77052,9.16828]，是由原始 anchor 的宽除以图像的宽*13 得到的,高也是这样计算的。因为需要把 anchor 大小归一到 13*13 的特征图。
6. Fine-Grained Features。在网络的浅层学到的是物理信息。在深层学到的是语义信息。对于检测而言，判断类别深层信息重要。对于画框，物理信息比较重要。以前的网络都是用深层信息来判断，所以有不足之处。解决方案是把浅层的信息接根线到深层。但是会带来一个问题。浅层的特征图要大于深层。接不到一起。把大图拆成小图，如：4*4 的特征图。隔一个数取一个，拆成 4 个 2*2 的图，然后 concat 到一起，再和深层的小图 concat。但是只适合于偶数的特征图。这种方式只有 YOLO2 使用。
7. Multi-Scale Training。由于去掉了 fc 层，所以我们可以训练不同大小的图，在训练过程中，每隔 10 个 epochs，将图的大小 resize，比如 320，352，608 等等。

YOLO3

1. 基本流程
   (1) 原图过 CNN 得到 feature map。
   (2) 中间的 conv 有短接。
   (3) 得到（13*13*3+26*26*3+52*52*3）*（4+1+80）的特征图。
   (4) 在 82 层 13*13 特征图最先得到，然后跳回 79 层，upsample 后和 61 层 concat 再 conv 得到 26*26 的特征图，然后跳回去 91 层，upsample 后和 36 层 concat 得到 52*52 的特征图。13*13 对应的大 anchor，26*26 对应三个中 anchor，52*52 对应小 anchor。
   (5) 以 13*13 特征图 loss 计算为例，其它两个以此类推。
   (6) 13*13*3*（4+1+80）拿出来（3 是 3 个 anchor），前四层为预测的（x,y,w,h），其中（x,y）过 sigmoid。后一层为预测的 confidence，过 sigmoid，后 80 层为预

测的类概率，也过 sigmoid（为什么不用 softmax 见第 3 点）。

(7) 接着计算 label。

(8) 得到 13*13 的 grid 的 cell 的 ID，维度是 2*（1，1，13，13）。

(9) 然后将 anchor 的（x,y,w,h）归一化到 13*13，归一化后的 anchor 大小是 4*（1，3，1，1）。

(10)生成一个正样本 obj-mask（正样本为 1），一个负样本 noobj-mask（负样本为 1），一个 class-mask(cell 内有物体)，一个 IoU-scores(pred_boxes 和 ground_truth 的 IoU），tx，ty，tw，th 和 tcls。class-mask 和 IoU-scores 用于评价。其它用于计算 loss。除了 tcls 是（1,3,13,13,80），其它的都是（1,3,13,13）。

(11)Target 是（n_boxes,6)6 维分别是图片 ID，类别 ID，x,y,w,h。

(12)首先把 target 的（x,y,w,h）归一化到 13*13。

(13)然后把 target 的（w,h）和属于 13*13 的三个 anchor 算 IoU。取最大的 IoU，得到对应的 anchor 的 IoU 和 ID。这样我们就得到了应该用哪个 anchor 参与计算。

(14)接下来将归一化后的（x,y）取整为（xi，yi），得到中心点对应 grid 中哪一个 cell。

(15)然后根据图片 ID,看取整后的(xi,yi)落在哪个 cell 中,将参与计算的那个 anchor 的 obj-mask 中对应的 cell 置为 1，noobj-mask 中对应的 cell 置为 0。这里要注意一个问题，如果三个 anchor 的 IoU 都很大，但是我们最大的一个做为唯一的正样本，如果我们直接把其它的两个 anchor 定为非物体，这样标准就很冲突，就很双标，这样网络无法训练，因此，我们根据前面计算出来的 anchor 的 IoU，取定一个阈值，大于阈值的就不算是负样本也不算正样本。

(16)接着就可以算 loss 了，target 的（x,y）减去取整的（xi，yi）得到余数，target 的(w,h)除以对应的 anchor(w,h)。然后对预测的(x,y,w,h)和 target 的(x,y,w,h)计算 L2 距离，得到（x,y,w,h）的 loss。

(17)把 target 的类 ID 拿出来，one-hot 一下，赋值给 tcls，将预测值和 tcls 值做交叉熵，得到类的 loss。

(18)然后求 confidence 的 loss，把预测的 confidence 中 obj-mask 为 1 的值挑出来，和 1 做交叉熵（confidence=是物体的概率*IoU，对于 ground truth 来说，这个值就等于 1），把预测的 confidence 中 noobj-mask 为 1 的值挑出来，和 0 做交叉熵。然后乘以权重再相加，这里要注意，想象中的负样本权重应该小于正样本，但其实

由于负样本预测一般都对，所以 loss 较小，所以负样本反而权重大于正样本。然后就可以得到 confidence 的 loss。

(19)最后将所有 loss 相加。

2. 从 13*13 变成 13*13，26*26，52*52 三种大小，feature map 变更稠密。
3. 从独一类变成多类，multi-lable 分类。（司机，运动员，姚明）。Softmax 变 sigmoid。
4. 从单个 conv 层相叠变成多个 module 相叠。module 是从(x, 128)用 1*1*64 的 conv 降维成(x*64)然后用 3*3*128 的 conv 升维回去，再通过一个 relu。

YOLO4

1. 多图拼接的数据增强
2. 把网络更新取消，将 loss 传回来加到图像上，等于加上噪音，以此来进行数据增强。
3. 改进后的空间注意力机制。注意力机制是为了告诉网络哪一部分更重要，让网路更加注意这个部分的特征。有两种，空间和层级注意力，空间是一层特征图上不同位置之间的哪个位置更重要，层级是哪个层的信息更重要。空间是把 H*W*C 维的特征图进行一个全局最大池化和平均池化得到两个 H×W×1 的特征图，然后拼一起过一个 7*7 的卷积层，在过 sigmoid 得到权重系数，再用权重系数乘以所有层的点。层级注意力是把 H*W*C 维的特征图进行一个全局最大池化和平均池化得到两个 1*1*C 的通道描述。接着分别送入一个两层神经网络，第一层神经元个数为 C/r，Relu，然后第二层神经元个数为 C。最后将两个得到的 vector 相加后再过一个 sigmoid 得到权重系数。最后用权重每层的系数乘以特征图的每一层。CBAM 就是这样的注意力机制中间层。轻量级，可以搭配网络使用，效果有提升。YOLO4 改进是直接对 H*W*C 过 sigmoid，得到每个点的权重，再和原特征图相乘。
4. 两次 FPN，特征之间不相加，直接 concat。
5. 工程上的优化在模型优化中详细介绍。

FPN Net

1. 在目标检测中有一个问题，小物体的预测，在模型逐步降采样的过程中，小物体的信息会丢失。而且在网络的浅层学到的是物理信息。在深层学到的是语义信息。对于检测而言，判断类别深层信息重要。对于画框，物理信息比较重要。因此，提出了 featurized image pyramid 模型金字塔，来把深层和浅层学习到的信息结合起来。

2. Featurized Image Pyramid，最原始，手工提取信息，把图片缩放到不同大小，对不同分辨率的图片分别提取特征来预测，每次特征提取/预测互相之间都是独立的，模型之间也很难共享它们中间提取的特征。训练和预测都费时费力。
3. CNN，Single feature map，只有最后得出的深层特征图会被用于预测。小物体只有寄希望于它的特征还没有完全丢失，但是这显然是不切实际的。
4. Pyramidal feature hierarchy，实际上 SSD 就是这样用的，对 CNN 中不同层得到的不同的分辨率的特征图直接预测。但是它也不是从最底层的特征图开始的，而是从稍高一点，比如 3 或 4 层以后每一层特征图都预测一下。但是结果也不特别好，可能是因为虽然用了不同的信息，但是是分开利用的信息，没有把这些信息结合起来看。
5. FPN 网络，通过把浅层和深层的信息跳接在一起来解决分辨率和特征语义的问题。在语义分割问题中常用。将深/潜层特征融合与多分辨率预测结合了起来。原图输入后，通过五层的 Conv，每一层降维 2 倍。得到的最五层特征图通过简单的最近邻插值升维后和第四层特征图相加。然后再升维后和第三层特征图相加，以此类推。最后得到五个不同的特征图，对每个特征图都送进 head 进行预测。在特征图相加的过程中，维数不一致用 1*1 的卷积来降维或升维。

RetinaNet

1. Backbone 是 resnet
2. FPN
3. Prediction 的部分去掉 fc，只用全卷积
4. 提出了 FOCAL LOSS。因为 one-stage 没有 two-stage 好的原因是因为正负样本比极度不均衡，以及好学的样本的 loss 下降的更快，而且简单样本一般更多，困难的得不到训练。

CENTERNET

1. 基本流程
   (1) 通过 CNN backbone 和 neck 以后。得到 128*128*（80+2+2+n）层的特征图。
   (2) 80 是 Center Heatmap。不找一个点，而是找中心点的概率分布。对 ground truth 的中心点做高斯平滑，得到的图叫做热度图。物体的边框越大，分布越广，框越小，分布越小。如果框有重叠，那么谁概率大算谁的。热度图就是实际使用的 label。
   (3) 类的 label 是 channel，一个 channel 代表一类物体。80 层代表 80 类物体。Center

Heatmap 的 loss 是 Focal Loss，因为真正有用的就是中心点。

(4) 最后用 3*3 的 max pool 来提取最大值，然后用阈值过滤，得到唯一的一个中心点。这样理论上就不用 NMS 了。实际操作过程中还是加了的，为了更准确。

(5) 接下来 2 层是 offset，因为除数的过程中，取整会有精度的损失，造成偏差。Loss 是用原坐标除以下采样倍数，去掉整数和预测的 offset 算 L1 距离。

(6) 接下来两层是 Size，就是长和宽，原坐标相减得到长和宽，除以下采样倍数，和预测值算 L1 距离。

(7) 接下来的 n 可以做其它的预测，比如关键点，深度，等等等等。

(8) 如果要预测的类过多，层数就会很多，对有的任务不适合，比如 1000 多个类预测的任务，会造成网络过大，速度慢。

2. 快又准
3. 检测物体就是检测点
4. 多任务（中心点，框，关键点）
5. 不用后处理，即 NMS
6. 不同的网络 Learning Rate 初始值不同

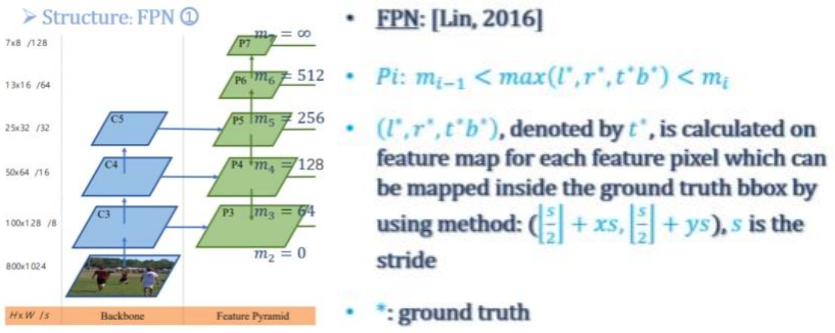
FCOS

1. 基本流程

(1) Backbone 是 FPN，最后一个 conv 出来后，再下采样两次，得到两个 feature map，上采样两次，分别和浅层的 conv 结合。得到两个 feature map。一共五个 feature map {p1, p2, p3, p4, p5}。

(2) Head 分为回归，分类和 Center-ness 三个功能。五个 feature map 都经过同一个 head。

(3) 计算 ground truth 的每个像素的到边框的距离{l, r, t, b}取最大值，落在哪个层，就用那一层来预测。m2=0, m3=64, 如果最大值落在这里，就用 P3 的 feature 来预测。



(4) 回归坐标这个功能，feature map 过 conv 卷积得到 H*W*4 的特征图，即对每个特征点，预测一个{x,y,w,h}值。然后用 IoU Loss。

(5) 分类这个功能,feature map 过 conv 卷积得到 H*W*C 的特征图,Loss 用 Focal Loss。

(6) Center-ness，希望找到的中心点尽量的中心。为了保证这一点，对于框中的每一个像素点，都计算一个值，这个值叫做中心性，用左右的最小值除以最大值，乘上，前后的最小值除以最大值，再开根号，只有最中心的点才能得到 1，用 BCE Loss。

$$centerness^* = \sqrt{\frac{\min(l^*,r^*)}{\max(l^*,r^*)} \times \frac{\min(t^*,b^*)}{\max(t^*,b^*)}}$$

2. 通过像素检测物体
3. FPN 是 backbone
4. multi-scale
5. one-stage
6. 需要 NMS