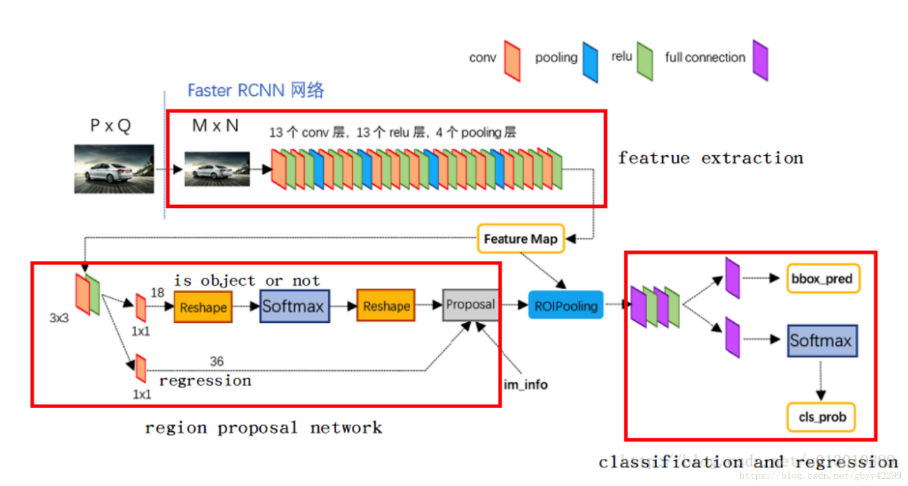
**计算机视觉之目标检测**

**1.Faster Rcnn**



**流程：**

(1)输入测试图像；

(2)将整张图片输入CNN，进行特征提取；

(3)用RPN生成建议窗口(proposals)，每张图片生成300个建议窗口；

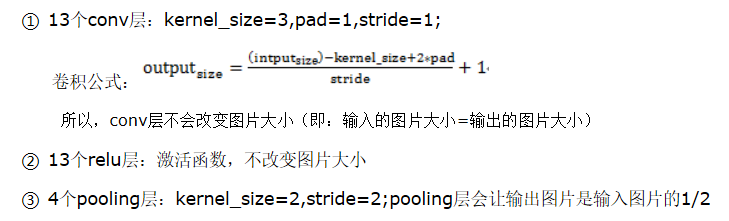
(4)把建议窗口映射到CNN的最后一层卷积feature map上；

(5)通过RoI pooling层使每个RoI生成固定尺寸的feature map；

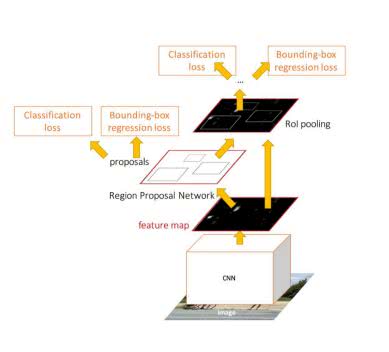
(6)利用Softmax Loss(探测分类概率) 和Smooth L1 Loss(探测边框回归)对分类概率和边框回归(Bounding box regression)联合训练.

**Conv layers**

Faster RCNN首先是支持输入任意大小的图片的，比如上图中输入的P\*Q，进入网络之前对图片进行了规整化尺度的设定，如可设定图像短边不超过600，图像长边不超过1000，我们可以假定M\*N=1000\*600（如果图片少于该尺寸，可以边缘补0，即图像会有黑色边缘）

****

C:\Users\12544\AppData\Roaming\Tencent\Users\1254468141\QQ\WinTemp\RichOle\OORAZ})O$N[)ED9ZA6{INPN.png



**RPN - Region Proposal Network**

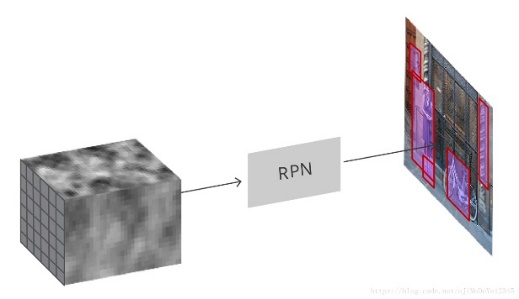


Figure 6. RPN 结构 - RPN 输入是卷积特征图，输出图片生成的 proposals.

RPN 利用所有的参考 boxes(anchors)，输出一系列 objecs 的良好的 proposals. 针对每个 anchor，都有两个不同的输出：

* anchor 内是某个 object 的概率.  
  RPN 不关注于 anchor 是某个 object class，只是确定其可能是一个 object (而不是背景background). 即： RPN 不关心 object 类别，只确定是 object 还是 background.  
  利用 object score，去滤出将要用于第二阶段的效果不佳的 boxes.
* anchor 边界框回归输出  
  边界框的输出用于调整 anchors 来更好的拟合预测的 object.

RPN 是全卷积(full conv) 网络，其采用基础网络输出的卷积特征图作为输入. 首先，采用 512 channel，3×3 kernel 的卷积层，然后是两个并行的 1×1 kernel 的卷积层，该卷积层的 channels 数量取决每个点的 anchors 的数量.

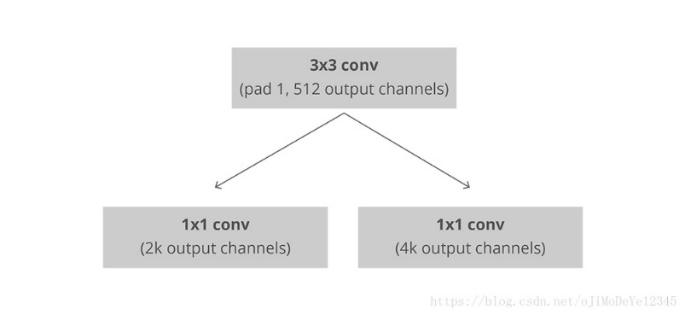


Figure 7. RPN 结构的全卷积实现，k 是 anchors 数量.

对于分类层，每个 anchor 输出两个预测值：anchor 是背景(background，非object)的 score 和 anchor 是前景(foreground，object) 的 score.

对于回归层，也可以叫边界框调整层，每个 anchor 输出 4 个预测值：Δxcenter,Δycenter,Δwidth,Δheight，即用于 anchors 来得到最终的 proposals.

根据最终的 proposal 坐标和其对应的 objectness score，即可得到良好的 objects proposals.

**RPN-训练、目标和损失函数**

RPN 有两种类型的预测值输出：二值分类和边界框回归调整.

训练时，对所有的 anchors 分类为两种类别. 与 ground-truth object 边界框的 [Intersection over Union](https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/) (IoU) 大于 0.5 的 anchors 作为 foreground；小于 0.1 的作为 background.

然后，随机采样 anchors 来生成batchsize=256 的 mini-batch，尽可能的保持 foreground 和 background anchors 的比例平衡.

RPN 对 mini-batch 内的所有 anchors 采用 [binary cross entropy](https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss) 来计算分类 loss. 然后，只对 mini-batch 内标记为 foreground 的 anchros 计算回归 loss. 为了计算回归的目标targets，根据 foreground anchor 和其最接近的 groundtruth object，计算将 anchor 变换到 object groundtruth 的偏移值 correctΔ.

[Faster R-CNN](http://arxiv.org/abs/1506.01497) 没有采用简单的 L1 或 L2 loss 用于回归误差，而是采用 Smooth L1 loss. Smooth L1 和 L1 基本相同，但是，当 L1 误差值非常小时，表示为一个确定值 σ ， 即认为是接近正确的，loss 就会以更快的速度消失.

采用动态 batches 是很有挑战性的. 即使已经尝试保持 background 和 foreground 的 anchors 的平衡比例，也不总是可行的. 根据图片中 groundtruth objects 和 anchors 的尺度与比例，很有可能得不到 foreground anchors. 这种情况时，将采用与 groundtruth boxes 具有最大 IoU 的 anchors. 这与理想情况相差很远，但实际中一般总能有 foreground 样本和要学习目标.

**RPN-非极大值抑制**

由于 Anchors 一般是有重叠的overlap，因此，相同 object 的 proposals 也存在重叠.为了解决重叠 proposals 问题，采用 NMS 算法处理，丢弃与一个score 更高的 proposal 间 IoU 大于预设阈值的 proposals.

虽然 NMS 看起来比较简单，但 IoU 阈值的预设需要谨慎处理. 如果 IoU 值太小，可能丢失 objetcs 的一些 proposals；如果 IoU 值过大，可能会导致 objects 出现很多 proposals. **IoU 典型值为 0.6**.

**PRN-Proposal 选择**

NMS 处理后，根据 score 对 topN个 proposals 排序. 在 [Faster R-CNN](http://arxiv.org/abs/1506.01497) 论文中 N=2000，其值也可以小一点，如 50，仍然能的高好的结果.

**单独应用 RPN**

RPN 可以独立使用，不用 2-stage 模型.

当处理的问题是，单个 object 类时，objectness 概率即可作为最终的类别概率. 此时，“foreground” = “single class”，“background”=“not single class”.

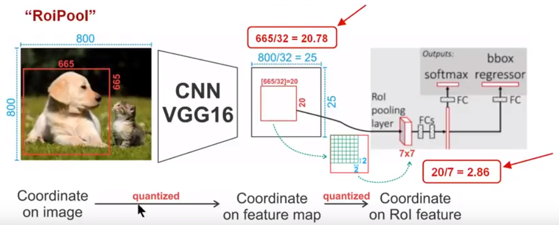
可以应用于人脸检测(face detection)，文字检测(text detection)，等.

仅单独采用 RPN 的优点在于，训练和测试速度较快. 由于 RPN 是仅有卷积层的简单网络，其预测效率比采用分类 base 网络的效率高.

**RoI Pooling**

这个可以在Faster RCNN中使用以便使生成的候选框region proposal映射产生固定大小的feature map

先贴出一张图，接着通过这图解释RoiPooling的工作原理.



针对上图

1)Conv layers使用的是VGG16，feat\_stride=32(即表示，经过网络层后图片缩小为原图的1/32),原图800\*800,最后一层特征图feature map大小:25\*25

2)假定原图中有一region proposal，大小为665\*665，这样，映射到特征图中的大小：665/32=20.78,即20.78\*20.78，如果你看过Caffe的Roi Pooling的C++源码，在计算的时候会进行取整操作，于是，进行所谓的第一次量化，即映射的特征图大小为20\*20

3)假定pooled\_w=7,pooled\_h=7,即pooling后固定成7\*7大小的特征图，所以，将上面在 feature map上映射的20\*20的 region  proposal划分成49个同等大小的小区域，每个小区域的大小20/7=2.86,即2.86\*2.86，此时，进行第二次量化，故小区域大小变成2\*2

4)每个2\*2的小区域里，取出其中最大的像素值，作为这一个区域的‘代表’，这样，49个小区域就输出49个像素值，组成7\*7大小的feature map

 总结，所以，通过上面可以看出，经过两次量化，即将浮点数取整，原本在特征图上映射的20\*20大小的region proposal，偏差成大小为7\*7的，这样的像素偏差势必会对后层的回归定位产生影响

 所以，产生了替代方案，RoiAlign.

**R-CNN – Region-based CNN**

R-CNN 是 [Faster R-CNN](http://arxiv.org/abs/1506.01497) 框架中的最后一个步骤.

* 计算图片的卷积特征图conv feature map；
* 然后采用 RPN 对卷积特征图处理，得到 object proposals；
* 再利用 RoI Pooling 对每个 proposal 提取特征；
* 最后，利用提取特征进行分类.

R-CNN 是模仿分类 CNNs 的最后一个阶段，采用全连接层来输出每个可能的 object 类别class 的score.

R-CNN 有两个不同的输出：

* 对每个 proposal 分类，其中类别包括一个 background 类(用于去除不良 proposals)；
* 根据预测的类别class，更好的调整 proposal 边界框.

在 [Faster R-CNN](http://arxiv.org/abs/1506.01497) 论文中，R-CNN 对每个 proposal 的特征图，拉平flatten，并采用 ReLU 和两个大小为 4096 维的全连接层进行处理.

然后，对每个不同 objects 采用两个不同的全连接层处理：

* 一个全连接层有 N+1 个神经单元，其中 N 是类别 class 的总数，包括 background class；
* 一个全连接层有 4N 个神经单元. 回归预测输出，得到 N 个可能的类别 classes 分别预测 Δcenterx,Δcentery,Δwidth,Δheight



Figure 9. R-CNN 结构

**训练和目标**

R-CNN 的目标基本上是与 RPN 目标的计算是一致的，但需要考虑不同的可能的 object 类别 classes.

根据 proposals 和 ground-truth boxes，计算其 IoU.

与任何一个 ground-truth box 的 IoU 大于 0.5 的 proposals 被设为正确的 boxes. IoU 在 0.1 到 0.5 之间时设为 background.

与 RPN 中目标组装相关，这里忽略没有任何交叉的 proposals. 这是因为，在此阶段，假设已经获得良好的 proposals，主要关注于解决难例. 当然，所有的这些超参数都是可以用于调整以更好的拟合 objects.

边界框回归的目标计算的是 proposal 与其对应的 ground-truth间的偏移量，只对基于 IoU 阈值设定类别class 后的 proposals 进行计算.

随机采用一个平衡化的 mini-batch=64，其中，25% 的 foreground proposals(具有类别class) 和 75% 的background proposals.

类似于 RPNs 的 losses，对于选定的 proposals，分类 loss 采用 multiclass entropy loss；对于 25% 的 foreground proposals 采用 SmoothL1 loss 计算其与 ground truth box 的匹配.

由于 R-CNN [全连接网络](https://www.baidu.com/s?wd=%E5%85%A8%E8%BF%9E%E6%8E%A5%E7%BD%91%E7%BB%9C&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)对每个类别class 仅输出一个预测值，当计算边框回归loss 时需谨慎. 当计算 loss 时，只需考虑正确的类别.

**后处理**

类似于 RPN，R-CNN 最终输出一堆带有类别 class 的objects，在返回结果前，再进一步进行处理.

为了调整边界框，需要考虑概率最大的类别的 proposals. 忽略概率最大值为 background class 的proposals.

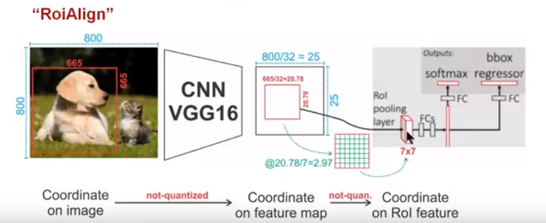
当得到最终的 objects 时，并忽略被预测为 background 的结果，采用 class-based NMS. 主要是通过对 objects 根据类别class 分组，然后根据概率排序，并对每个独立的分组采用 NMS 处理，最后再放在一起.

最终得到的 objects 列表，仍可继续通过设定概率阈值的方式，来限制每个类的 objects 数量.

**补充：[RoIAlign](https://www.cnblogs.com/wangyong/p/8523814.html)**

这个是在Mask RCNN中使用以便使生成的候选框region proposal映射产生固定大小的feature map时提出的。ROI Align 是在Mask-RCNN这篇论文里提出的一种区域特征聚集方式, 很好地解决了ROI Pooling操作中两次量化造成的区域不匹配(mis-alignment)的问题。实验显示，在检测测任务中将 ROI Pooling 替换为 ROI Align 可以提升检测模型的准确性。

 先贴出一张图，接着通过这图解释RoiAlign的工作原理



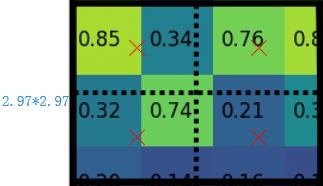
同样，针对上图，有着类似的映射

1)Conv layers使用的是VGG16，feat\_stride=32(即表示，经过网络层后图片缩小为原图的1/32),原图800\*800,最后一层特征图feature map大小:25\*25

2)假定原图中有一region proposal，大小为665\*665，这样，映射到特征图中的大小：665/32=20.78,即20.78\*20.78，此时，没有像RoiPooling那样就行取整操作，保留浮点数

3)假定pooled\_w=7,pooled\_h=7,即pooling后固定成7\*7大小的特征图，所以，将在 feature map上映射的20.78\*20.78的region proposal 划分成49个同等大小的小区域，每个小区域的大小20.78/7=2.97,即2.97\*2.97

4)假定采样点数为4，即表示，对于每个2.97\*2.97的小区域，平分四份，每一份取其中心点位置，而中心点位置的像素，采用双线性插值法进行计算，这样，就会得到四个点的像素值，如下图

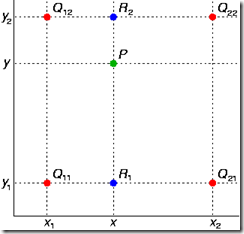


上图中，四个红色叉叉‘×’的像素值是通过双线性插值算法计算得到的

最后，取四个像素值中最大值作为这个小区域(即：2.97\*2.97大小的区域)的像素值，如此类推，同样是49个小区域得到49个像素值，组成7\*7大小的feature map

总结：知道了RoiPooling和RoiAlign实现原理，在以后的项目中可以根据实际情况进行方案的选择；对于检测图片中**大目标物体时，两种方案的差别不大**，而如果是**图片中有较多小目标物体需要检测，则优先选择RoiAlign**，更精准些。

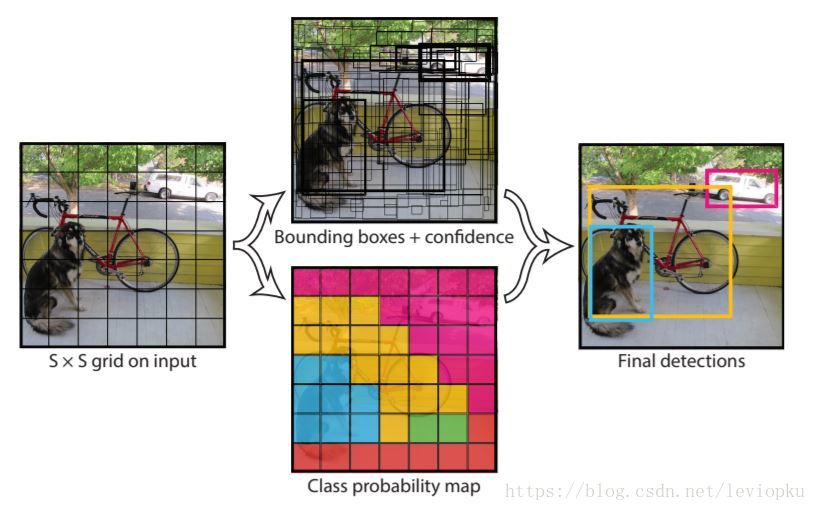
注：双线性插值，又称为双线性内插。在[数学](http://zh.wikipedia.org/wiki/%E6%95%B0%E5%AD%A6)上，双线性插值是有两个变量的[插值](http://zh.wikipedia.org/wiki/%E6%8F%92%E5%80%BC)函数的[线性插值](http://zh.wikipedia.org/wiki/%E7%BA%BF%E6%80%A7%E6%8F%92%E5%80%BC)扩展，其核心思想是在两个方向分别进行一次线性插值。



如图，已知Q12，Q22，Q11，Q21，但是要插值的点为P点，这就要用双线性插值了，首先在x轴方向上，对R1和R2两个点进行插值，这个很简单，然后根据R1和R2对P点进行插值，这就是所谓的双线性插值。

**2.yolo系列**

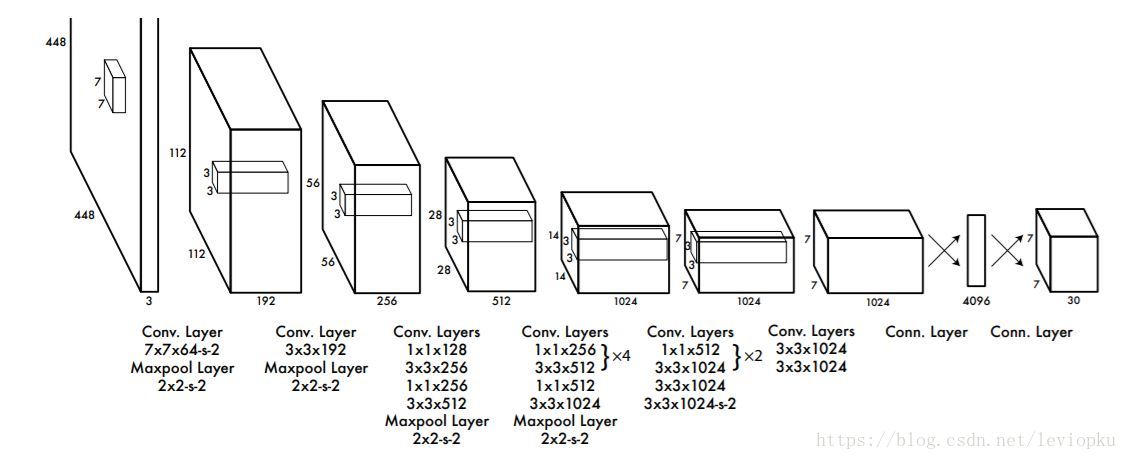
**yolo v1（24个conv 、2个fc）**



如上图所示，输入图片被划分为7x7个单元格，每个单元格独立作检测。

在这里很容易被误导：每个网格单元的视野有限而且很可能只有局部特征，这样就很难理解yolo为何能检测比grid\_cell大很多的物体。其实，yolo的做法并不是把每个单独的网格作为输入feed到模型，在inference的过程中，网格只是物体中心点位置的划分之用，并不是对图片进行切片，不会让网格脱离整体的关系。

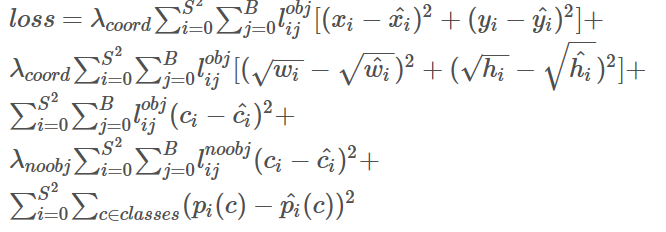
可以通过yolo\_v1的structure来进一步理解，相比faster r-cnn那种two-stage复杂的网络结构而言，yolo\_v1的网络结构显得亲民得多。基本思想是这样：预测框的位置、大小和物体分类都通过CNN暴力predict出来。

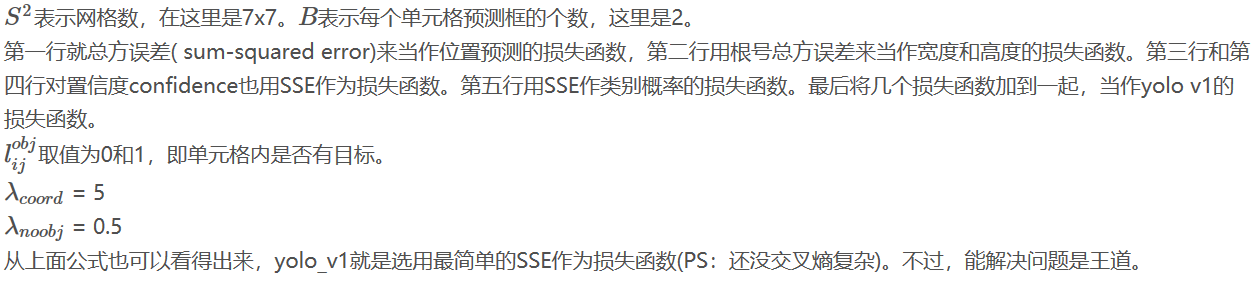


上面是结构图yolo\_v1结构图，通过结构图可以轻易知道前向传播的计算过程，是很便于读者理解的。v1的输出是一个7x7x30的张量，7x7表示把输入图片划分位7x7的网格，每一个小单元的另一个维度等于30。30=(2\*5+20)。代表能预测2个框的5个参数(x,y,w,h,score)和20个种类。

可以看出输出张量的深度影响yolo\_v1能检测目标的种类。v1的输出深度只有30，意味着每个单元格只能预测两个框(而且只认识20类物体)，**这对于密集型目标检测和小物体检测都不能很好适用**。

正如前面所说的，yolo是端到端训练，对于预测框的位置、size、种类、置信度(score)等信息的预测都通过一个损失函数来训练。

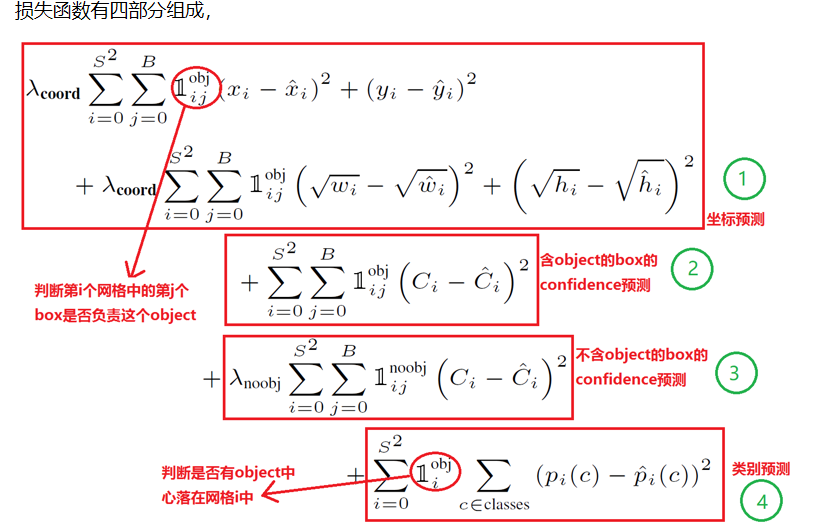




YOLO网络结构由24个卷积层与2个全连接层构成，网络入口为448x448(v2为416x416)，图片进入网络先经过resize，输出格式为：

C:\Users\12544\AppData\Roaming\Tencent\Users\1254468141\QQ\WinTemp\RichOle\(}@Z6PJ4I1Y1]Y89I67GBQ9.png

其中，S为划分网格数，B为每个网格负责目标个数，C为类别个数。B表示每个小格对应B组可能的框，5表示每个框的四个坐标和一个置信度，C表示类别，同时也说明B个框只能隶属于同一个类别。



**yolo v2**

**Batch Normalization**

BN（Batch Normalization）层简单讲就是对网络的每一层的输入都做了归一化，这样网络就不需要每层都去学数据的分布，收敛会快点。作者在YOLOv2种为每个卷积层都添加了BN层，由于BN可以规范模型，所以加入BN后就把dropout去掉了，实验证明添加了BN层可以提高2%的mAP。

**High Resolution Classifier**

YOLOv1在预训练的时候用的是224\*224的输入，一般预训练的分类模型都是在ImageNet数据集上进行的，然后在检测的时候采用448\*448的输入。这会导致从分类模型切换到检测模型的时候，模型还要适应图像分辨率的改变。

YOLOv2中将预训练分成两步：先用224\*224的输入从头开始训练网络，大概160个epoch，然后再将输入调整到448\*448，再训练10个epoch。\*\*注意这两步都是在ImageNet数据集上操作。\*\*最后再在检测的数据集上fine-tuning，也就是检测的时候用448\*448的图像作为输入就可以顺利过渡了。

**Convolutional With Anchor Boxes**

YOLOv1是利用全连接层直接预测bounding box的坐标。

YOLOv2则借鉴了Faster R-CNN的思想，引入anchor。

YOLOv2做了以下改变：

（1）删掉全连接层和最后一个pooling层，使得最后的卷积层可以有更高分辨率的特征；

（2）缩减网络，用416\*416大小的输入代替原来448\*448。这样做是希望希望得到的特征图都有奇数大小的宽和高，奇数大小的宽和高会使得每个特征图在划分cell的时候就只有一个中心cell。因为大的目标一般会占据图像的中心，所以希望用一个中心cell去预测，而不是4个中心cell。网络最终将416\*416的输入下采样32倍变为13\*13大小的feature map输出，查看.cfg文件可以看到有8个pooling层。

YOLOv1中将输入图像分成7\*7的网格，每个网格预测2个bounding box，一共只有7\*7\*2=98个box。

YOLOv2中引入anchor boxes，输出feature map大小为13\*13，每个cell有5个anchor box预测得到5个bounding box，一共有13\*13\*5=845个box。增加box数量是为了提高目标的定位准确率。

在这里作者并没有采用直接预测offset的方法，还是沿用了YOLO算法中直接预测相对于grid cell的坐标位置的方式。

**Yolov3**

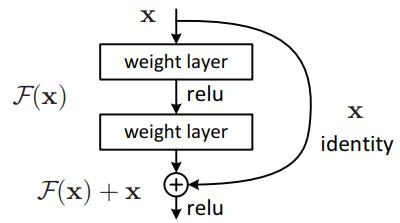
YOLO3主要的改进有：调整了网络结构；利用多尺度特征进行对象检测；对象分类用Logistic取代了softmax。

**新的网络结构Darknet-53**

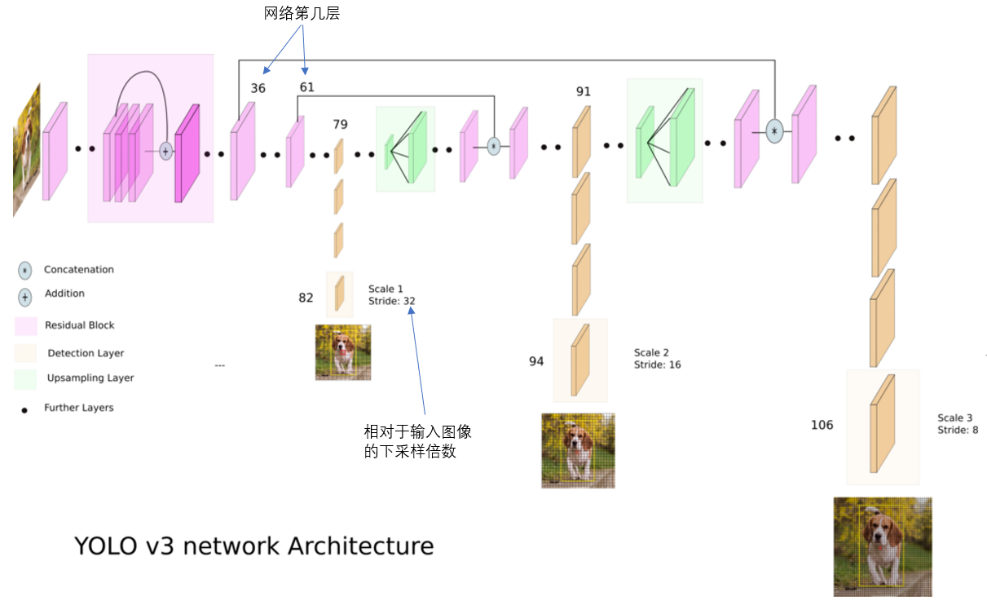
在基本的图像特征提取方面，YOLO3采用了称之为Darknet-53的网络结构（含有53个卷积层），它借鉴了残差网络residual network的做法，在一些层之间设置了快捷链路（shortcut connections）。



上图的Darknet-53网络采用256\*256\*3作为输入，最左侧那一列的1、2、8等数字表示多少个重复的残差组件。每个残差组件有两个卷积层和一个快捷链路，示意图如下：



**利用多尺度特征进行对象检测**



在YOLO3更进一步采用了3个不同尺度的特征图来进行对象检测。

结合上图看，卷积网络在79层后，经过下方几个黄色的卷积层得到一种尺度的检测结果。相比输入图像，这里用于检测的特征图有32倍的下采样。比如输入是416\*416的话，这里的特征图就是13\*13了。由于下采样倍数高，这里特征图的感受野比较大，因此适合检测图像中尺寸比较大的对象。

为了实现细粒度的检测，第79层的特征图又开始作上采样（从79层往右开始上采样卷积），然后与第61层特征图融合（Concatenation），这样得到第91层较细粒度的特征图，同样经过几个卷积层后得到相对输入图像16倍下采样的特征图。它具有中等尺度的感受野，适合检测中等尺度的对象。

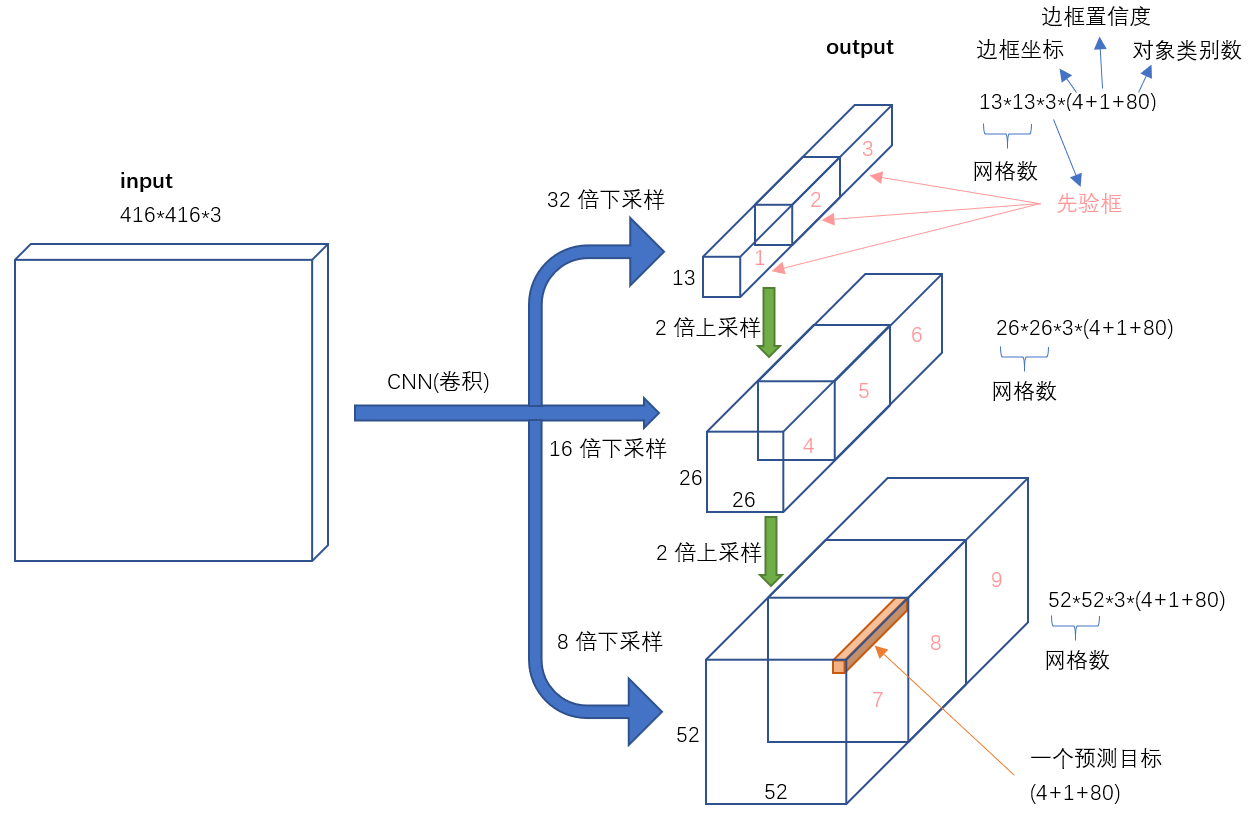
最后，第91层特征图再次上采样，并与第36层特征图融合（Concatenation），最后得到相对输入图像8倍下采样的特征图。它的感受野最小，适合检测小尺寸的对象。

YOLO3延续了这种方法，为每种下采样尺度设定3种先验框，总共聚类出9种尺寸的先验框。在COCO数据集这9个先验框是：(10x13)，(16x30)，(33x23)，(30x61)，(62x45)，(59x119)，(116x90)，(156x198)，(373x326)。

分配上，在最小的13\*13特征图上（有最大的感受野）应用较大的先验框(116x90)，(156x198)，(373x326)，适合检测较大的对象。中等的26\*26特征图上（中等感受野）应用中等的先验框(30x61)，(62x45)，(59x119)，适合检测中等大小的对象。较大的52\*52特征图上（较小的感受野）应用较小的先验框(10x13)，(16x30)，(33x23)，适合检测较小的对象。

**对象分类softmax改成logistic**

预测对象类别时不使用softmax，改成使用logistic的输出进行预测。这样能够支持多标签对象（比如一个人有Woman 和 Person两个标签）。

**输入映射到输出**  


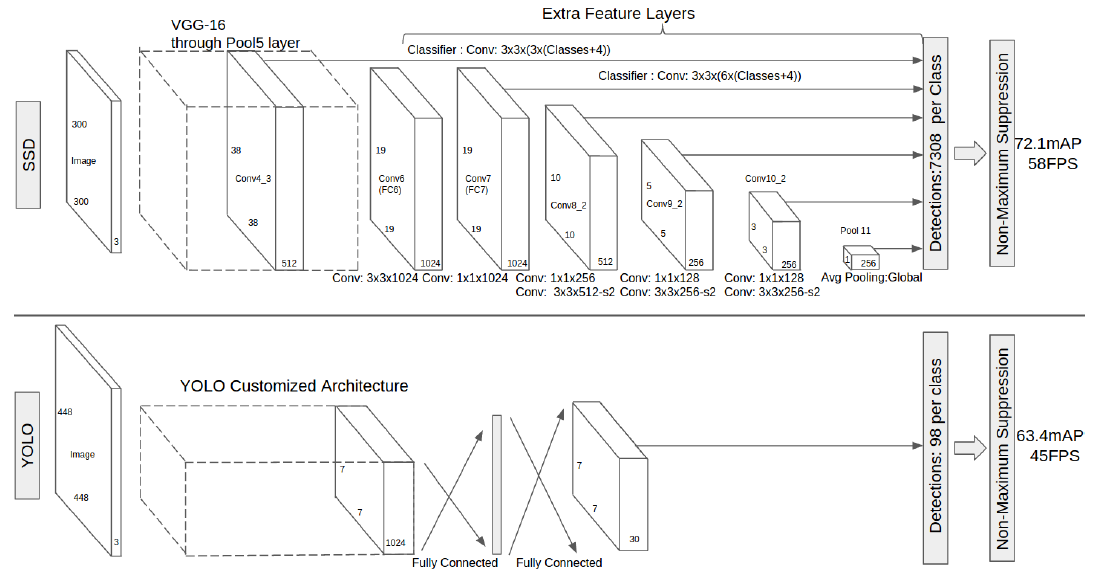
不考虑神经网络结构细节的话，总的来说，对于一个输入图像，YOLO3将其映射到3个尺度的输出张量，代表图像各个位置存在各种对象的概率。

我们看一下YOLO3共进行了多少个预测。对于一个416\*416的输入图像，在每个尺度的特征图的每个网格设置3个先验框，总共有 13\*13\*3 + 26\*26\*3 + 52\*52\*3 = 10647 个预测。每一个预测是一个(4+1+80)=85维向量，这个85维向量包含边框坐标（4个数值），边框置信度（1个数值），对象类别的概率（对于COCO数据集，有80种对象）。

对比一下，YOLO2采用13\*13\*5 = 845个预测，YOLO3的尝试预测边框数量增加了10多倍，而且是在不同分辨率上进行，所以mAP以及对小物体的检测效果有一定的提升。

除了w, h的损失函数依然采用总方误差之外，其他部分的损失函数用的是二值交叉熵

**3.SSD模型（比yolov1 快、精确，mAP可与faster rcnn相媲美 ）**



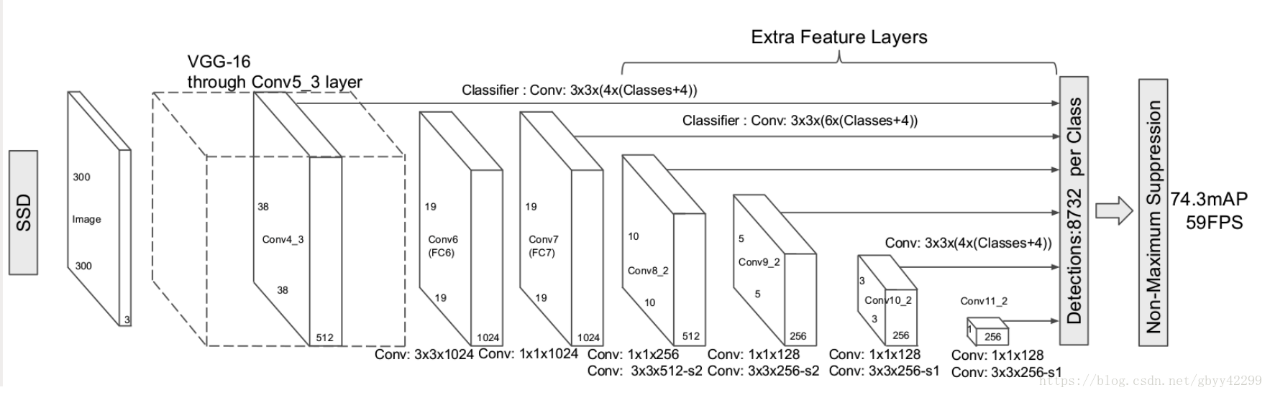
**概述**

SSD采用不用卷积层的feature map进行综合，将VGG16的最后两层全连接改为卷积层并额外增加四个卷积层来达到构造网络。对于这5个不同尺度的卷积层输出的feature map，作者分别采用不同的3x3的卷积核进行卷积，一个输出分类的置信度（condfidence），一个输出回归的localization。

作者文中多次提到的default boxes和aspect ratios，那么他们是什么意思呢？其实就是每一个box相对于其对应的 feature map cell 的位置是固定的。 在每一个 feature map cell 中，我们要 predict 得到的 box 与 default box 之间的 offsets，以及每一个 box 中包含物体的 score（每一个类别概率都要计算出）。

因此，对于一个位置上的 k 个boxes 中的每一个 box，我们需要计算出 c 个类，每一个类的 score，还有这个 box 相对于 它的默认 box 的 4 个偏移值（offsets）。于是，在 feature map 中的每一个 feature map cell 上，就需要有 (c+4)×k 个 filters。对于一张 m×n 大小的 feature map，即会产生 (c+4)×k×m×n个输出结果。

在训练的过程中，SSD与two stage的方法最大的区别是，SSD 训练图像中的 groundtruth 需要赋予到那些固定输出的 boxes 上。在前面也已经提到了，SSD 输出的是事先定义好的，一系列固定大小的 bounding boxes。



最后会得到（38\*38\*4 + 19\*19\*6 + 10\*10\*6 + 5\*5\*6 + 3\*3\*4 + 1\*1\*4）= 8732个default box。

训练中还有一个东西：prior box，是指实际中选择的default box（每一个feature map cell 不是k个default box都取）。也就是说default box是一种概念，prior box则是实际的选取。训练中一张完整的图片送进网络获得各个feature map，对于正样本训练来说，需要先将prior box与ground truth box做匹配，匹配成功说明这个prior box所包含的是个目标，但离完整目标的ground truth box还有段距离，训练的目的是保证default box的分类confidence的同时将prior box尽可能回归到ground truth box。 举个列子：假设一个训练样本中有2个ground truth box，所有的feature map中获取的prior box一共有8732个。那个可能分别有10、20个prior box能分别与这2个ground truth box匹配上。训练的损失包含分类损失和回归损失两部分。

**重点：**

一直很难理解上述的8732个框和每个feature map每个像素生成6个框，paper里面提到：ratio={1,2,3,1/2,1/3}，但是看了代码发现其实这5种ratio是作者计算出来的。

比如代码中的steps = [8, 16, 32, 64, 100, 300]

这一步要仔细理解，即计算卷积层产生的prior\_box距离原图的步长，先验框中心点的坐标会乘以step，相当于从feature map位置映射回原图位置，比如conv4\_3输出特征图大小为38\*38，而输入的图片为300\*300，所以38\*8约等于300，所以映射步长为8。这是针对300\*300的训练图片。

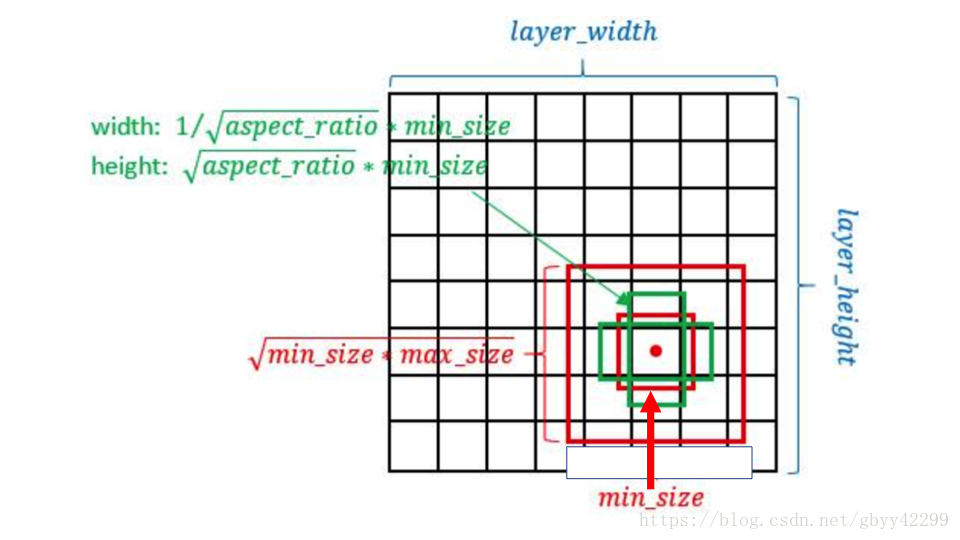
又如文中代码显示，给出的长宽比为

aspect\_ratios = [[2], [2, 3], [2, 3], [2, 3], [2], [2]]

这里并不是paper中所给出的ar={1,2,3,1/2,1/3}

在SSD中6层卷积层的每个特征图的每个中心点会产生2个不同大小的正方形默认框，另外每设置一个aspect\_ratio则会增加两个长方形默认框，而文中代码对于6层的aspect\_ratio个数分别为1、2、2、2、1、1，所以这也就是为什么会产生4、6、6、6、4、4个默认框了。例如conv4\_3默认生成两个不同大小的正方形默认框，另外又有一个aspect\_ratio=2产生了两个长方形默认框，所以总共有4个。再如fc7，默认生成两个正方形默认框，另外又有aspect\_ratio=[2,3]，所以又生成了4个不同的长方形默认框，共有6个不同大小的默认框。

接着我们再讲这些产生的默认框的大小计算。这里参考paper中的计算公式，我们可以知道，对于产生的正方形的默认框，一大一小共两个，其边长计算公式为：小边长=min\_size，而大边长=sqrt（min\_size\*max\_size）。对于产生的长方形默认框，我们需要计算它的高（height）和宽（width），其中，height=1/sqrt(aspect\_ratio)\*min\_size，width=sqrt(aspect\_ratio)\*min\_size，对其高和宽翻转后得到另一个面积相同但宽高相互置换的长方形。如图所示：



根据以上分析，我们可以计算6层中每个特征图的每个中心点所产生的默认框的大小，分别如下：

conv4\_3：小正方形边长=min\_size=30，大正方形边长=sqrt(min\_size\*max\_size)=sprt(30\*60)=42.42；长方形的宽=sqrt(aspect\_ratio)\*min\_size=sqrt(2)\*30，高=1/sqrt(aspect\_ratio)\*min\_size=30/sqrt(2)，宽高比刚好为2：1；将以上宽高旋转90度产生另一个长方形，宽高比变为1：2。

fc7：小正方形边长=min\_size=60，大正方形边长=sqrt(min\_size\*max\_size)=sprt(60\*111)=81.6；第1组长方形的宽=sqrt(aspect\_ratio)\*min\_size=sqrt(2)\*60，高=1/sqrt(aspect\_ratio)\*min\_size=60/sqrt(2)，宽高比刚好为2：1；将以上宽高旋转90度产生另一个长方形，宽高比变为1：2。

第2组长方形的宽=sqrt(aspect\_ratio)\*min\_size=sqrt(3)\*60，高=1/sqrt(aspect\_ratio)\*min\_size=60/sqrt(3)，宽高比刚好为3：1；将以上宽高旋转90度产生另一个长方形，宽高比变为1：3。

conv7\_2、conv8\_2、conv9\_2就不再计算了，这就是先验框的计算方式。

**Training**

在训练时，本文的 SSD 与那些用 region proposals + pooling 方法的区别是，SSD 训练图像中的 groundtruth 需要赋予到那些固定输出的 boxes 上。在前面也已经提到了，SSD 输出的是事先定义好的，一系列固定大小的 bounding boxes。

当这种将训练图像中的 groundtruth 与固定输出的 boxes 对应之后，就可以 end-to-end 的进行 loss function 的计算以及 back-propagation 的计算更新了。

**Matching strategy**

如何将 groundtruth boxes 与 default boxes 进行配对，以组成 label 呢？

在开始的时候，用 MultiBox 中的 best jaccard overlap 来匹配每一个 ground truth box 与 default box，这样就能保证每一个 groundtruth box 与唯一的一个 default box 对应起来。

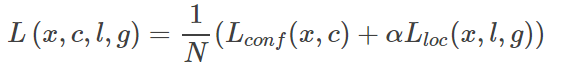
但是又不同于 MultiBox ，本文之后又将 default box 与任何的 groundtruth box 配对，只要两者之间的 jaccard overlap 大于一个阈值，这里本文的阈值为 0.5。

**Training objective**

SSD 训练的目标函数源自于 MultiBox 的目标函数，但是本文将其拓展，使其可以处理多个目标类别。用 C:\Users\12544\AppData\Roaming\Tencent\Users\1254468141\QQ\WinTemp\RichOle\H2VORENQ4L@0B4HPFYIX%O4.png表示 第 i个 default box 与 类别 p 的 第 j个 ground truth box 相匹配，否则若不匹配的话，则 C:\Users\12544\AppData\Roaming\Tencent\Users\1254468141\QQ\WinTemp\RichOle\6J$L_@0YA]E~WY]K(W6FNTC.png。

根据上面的匹配策略，一定有C:\Users\12544\AppData\Roaming\Tencent\Users\1254468141\QQ\WinTemp\RichOle\N10]O5[[(~$DBDCN{ORL928.png，意味着对于 第 j 个 ground truth box，有可能有多个 default box 与其相匹配。

总的目标损失函数（objective loss function）就由 localization loss（loc） 与 confidence loss（conf） 的加权求和：



其中：N是与 ground truth box 相匹配的 default boxes个数

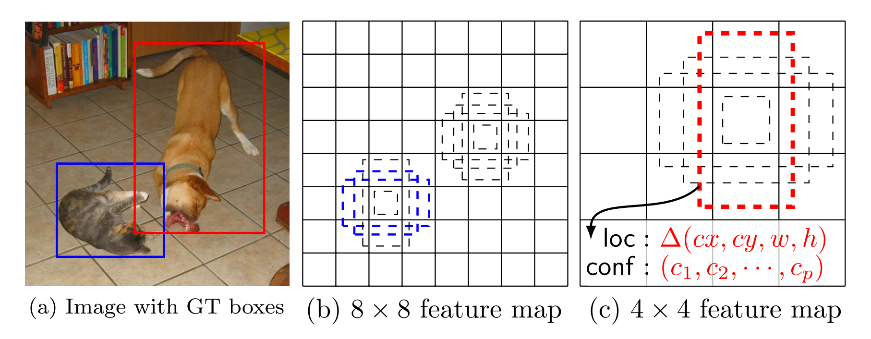
localization loss（loc） 是 Fast R-CNN 中 Smooth L1 Loss，用在 predict box（l） 与 ground truth box（g） 参数（即中心坐标位置，width、height）中，回归 bounding boxes 的中心位置，以及 width、height

confidence loss（conf） 是 Softmax Loss，输入为每一类的置信度 c

权重项 α，设置为 1

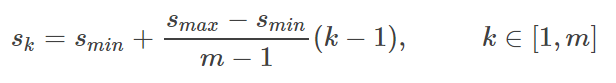
**Choosing scales and aspect ratios for default boxes：**

本文同时使用 lower feature maps、upper feature maps 来 predict detections。下图展示了本文中使用的两种不同尺度的 feature map，8×88×8 的feature map，以及 4×44×4 的 feature map：

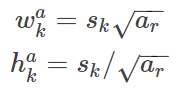


一般来说，一个 CNN 网络中不同的 layers 有着不同尺寸的 感受野（receptive fields）。这里的感受野，指的是输出的 feature map 上的一个节点，其对应输入图像上尺寸的大小。

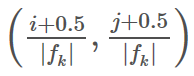
所幸的是，SSD 结构中，default boxes 不必要与每一层 layer 的 receptive fields 对应。本文的设计中，feature map 中特定的位置，来负责图像中特定的区域，以及物体特定的尺寸。假如我们用 m 个 feature maps 来做 predictions，每一个 feature map 中 default box 的尺寸大小计算如下：



其中，smin 取值 0.2，smax 取值 0.95，意味着最低层的尺度是 0.2，最高层的尺度是 0.95，再用不同 aspect ratio 的 default boxes，用 ar 来表示：ar={1,2,3,1/2,1/3}，则每一个 default boxes 的 width、height 就可以计算出来：



对于 aspect ratio 为 1 时，本文还增加了一个 default box，这个 box 的 scale 是C:\Users\12544\AppData\Roaming\Tencent\Users\1254468141\QQ\WinTemp\RichOle\H~X~1_8G]F@1{J7_$DNYZ%N.png。所以最终，在每个 feature map location 上，有 6 个 default boxes。

每一个 default box 的中心，设置为：。其中，|fk| 是第 k个 feature map 的大小，同时，i,j∈[0,|fk|)。

**Hard negative mining**

在生成一系列的 predictions 之后，会产生很多个符合 ground truth box 的 predictions boxes，但同时，不符合 ground truth boxes 也很多，而且这个 negative boxes，远多于 positive boxes。这会造成 negative boxes、positive boxes 之间的不均衡。训练时难以收敛。

因此，本文采取，先将每一个物体位置上对应 predictions（default boxes）是 negative 的 boxes 进行排序，按照 default boxes 的 confidence 的大小。 选择最高的几个，保证最后 negatives、positives 的比例在 3:13:1。

本文通过实验发现，这样的比例可以更快的优化，训练也更稳定。

**Data augmentation**

每一张训练图像，随机的进行如下几种选择：

* 使用原始的图像
* 采样一个 patch，与物体之间最小的 jaccard overlap 为：0.1，0.3，0.5，0.7 与 0.9
* 随机的采样一个 patch

采样的 patch 是原始图像大小比例的 [0.1，1]，aspect ratio 在 1/2 与 2 之间。

当 ground truth box 的 中心（center）在采样的 patch 中时，我们保留重叠部分。

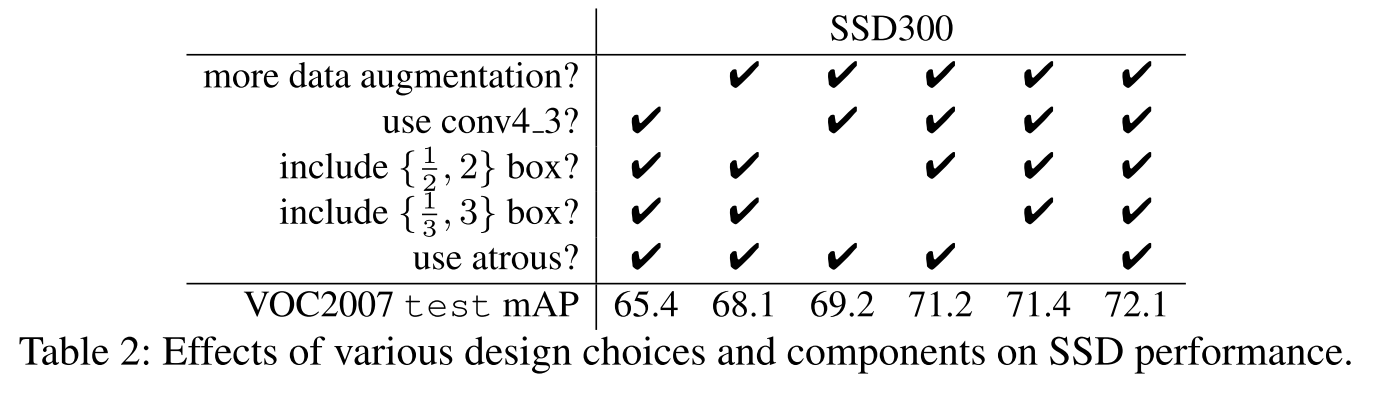
在这些采样步骤之后，每一个采样的 patch 被 resize 到固定的大小，并且以 0.5 的概率随机的 水平翻转（horizontally flipped）

SSD 模型对 bounding box 的 size 非常的敏感。也就是说，SSD 对小物体目标较为敏感，在检测小物体目标上表现较差。其实这也算情理之中，因为对于小目标而言，经过多层卷积之后，就没剩多少信息了。虽然提高输入图像的 size 可以提高对小目标的检测效果，但是对于小目标检测问题，还是有很多提升空间的。

同时，积极的看，SSD 对大目标检测效果非常好。同时，因为本文使用了不同 aspect ratios 的 default boxes，SSD 对于不同 aspect ratios 的物体检测效果也很好。

**Model analysis**

为了更好的理解 SSD，本文还使用控制变量法来验证 SSD 中的每一部分对最终结果性能的影响。测试如下表 Table 2 所示：



从上表可以看出一下几点：

* **数据增广（Data augmentation）**对于结果的提升非常明显

Fast R-CNN 与 Faster R-CNN 使用原始图像，以及 0.5 的概率对原始图像进行水平翻转（horizontal flip），进行训练。如上面写的，本文还使用了额外的 sampling 策略，YOLO 中还使用了 亮度扭曲（photometric distortions），但是本文中没有使用。 做了数据增广，将 mAP 从 65.4%提升到了 72.1%，提升了 6.7%。 我们还不清楚，本文的 sampling 策略会对 Fast R-CNN、Faster R-CNN 有多少好处。但是估计不会很多，因为 Fast R-CNN、Faster R-CNN 使用了 feature pooling，这比人为的对数据进行增广扩充，还要更 robust。

* **使用更多的 feature maps 对结果提升更大**

类似于 FCN，使用含图像信息更多的低 layer 来提升图像分割效果。我们也使用了 lower layer feature maps 来进行 predict bounding boxes。 我们比较了，当 SSD 不使用 conv4\_3 来 predict boxes 的结果。当不使用 conv4\_3，mAP 下降到了 68.1%。 可以看见，低层的 feature map 蕴含更多的信息，对于图像分割、物体检测性能提升帮助很大的。

* **使用更多的 default boxes，结果也越好**

如 Table 2 所示，SSD 中我们默认使用 6 个 default boxes（除了 conv4\_3 因为大小问题使用了 3 个 default boxes）。如果将 aspect ratios 为 1/3、3 的 boxes 移除，performance 下降了 0.9%。如果再进一步的，将 1/2、2 的 default boxes 移除，那么 performance 下降了近 2%。

* **Atrous 使得 SSD 又好又快**

如前面所描述，我们根据 ICLR 2015, DeepLab-LargeFOV，使用结合 atrous algorithm 的 VGG16 版本。 如果我们使用原始的 VGG16 版本，即保留 pool5 的参数为：2×2−s2，且不从 FC6，FC7 上采集 parameters，同时添加 conv5\_3 来做 prediction，结果反而会下降 0.7%。同时最关键的，速度慢了 50%。

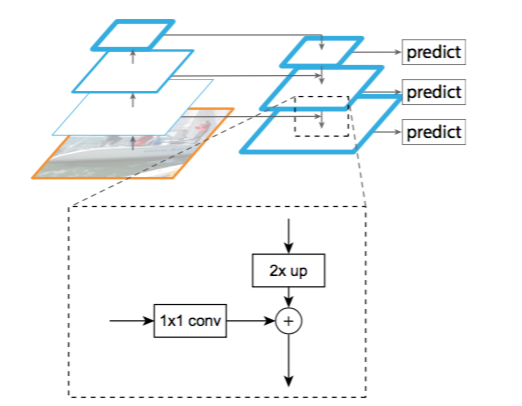
**Inference time**

本文的方法一开始会生成大量的 bounding boxes，所以有必要用 Non-maximum suppression（NMS）来去除大量重复的 boxes。

通过设置 confidence 的阈值为 0.01，我们可以过滤掉大多数的 boxes。

之后，我们再用 Thrust CUDA library 进行排序，用 GPU 版本的实现来计算剩下的 boxes 两两之间的 overlap。然后，进行 NMS，每一张图像保留 top 200 detections。这一步 SSD300 在 VOC 20 类的每张图像上，需要耗时 2.2 msec。

**4. FPN(特征金字塔)**



**一个自底向上的线路，一个自顶向下的线路，横向连接（lateral connection）**。图中放大的区域就是横向连接，这里1\*1的卷积核的主要作用是减少卷积核的个数，也就是减少了feature map的个数，并不改变feature map的尺寸大小。

自底向上其实就是网络的前向过程。在前向过程中，feature map的大小在经过某些层后会改变，而在经过其他一些层的时候不会改变，作者将不改变feature map大小的层归为一个stage，因此每次抽取的特征都是每个stage的最后一个层输出，这样就能构成特征金字塔。

自顶向下的过程采用上采样（upsampling）进行，而横向连接则是将上采样的结果和自底向上生成的相同大小的feature map进行融合（merge）。在融合之后还会再采用**3\*3的卷积核对每个融合结果进行卷积**，**目的是消除上采样的混叠效应**（aliasing effect）。并假设生成的feature map结果是P2，P3，P4，P5，和原来自底向上的卷积结果C2，C3，C4，C5一一对应。

**作者一方面将FPN放在RPN网络中用于生成proposal，**原来的RPN网络是以主网络的某个卷积层输出的feature map作为输入，简单讲就是只用这一个尺度的feature map。但是现在要将FPN嵌在RPN网络中，生成不同尺度特征并融合作为RPN网络的输入。在每一个scale层，都定义了不同大小的anchor，对于P2，P3，P4，P5，P6这些层，定义anchor的大小为32^2,64^2,128^2,256^2，512^2，另外每个scale层都有3个长宽对比度：1:2，1:1，2:1。所以整个特征金字塔有15种anchor。

**正负样本的界定和Faster RCNN差不多，**如果某个anchor和一个给定的ground truth有最高的IOU或者和任意一个Ground truth的IOU都大于0.7，则是正样本。如果一个anchor和任意一个ground truth的IOU都小于0.3，则为负样本。

作者提出的FPN（Feature Pyramid Network）算法同时利用低层特征高分辨率和高层特征的高语义信息，通过融合这些不同层的特征达到预测的效果。并且预测是在每个融合后的特征层上单独进行的，这和常规的特征融合方式不同。

**5. Relation Networks for Object Detection(cvpr 2018)**

当下主流的目标检测的方法还是对各个物体进行单独的检测，本文基于Attention，提出了一种object relation module，通过引入不同物体之间的外观和geometry特征做interaction，实现对物体之间relation的建模，提高检测效果，并且将**关系模块运用在duplicate remove**中，进行**可学习的NMS**（提出了一种特别的代替NMS的去重模块，可以避免NMS需要手动设置参数的问题），实现了第一个完全end-to-end的目标检测系统。

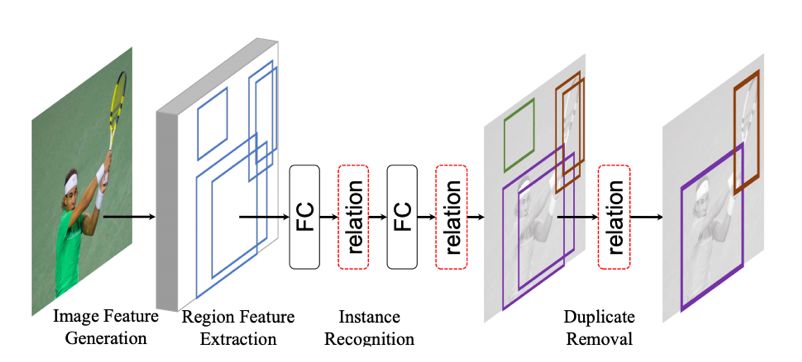


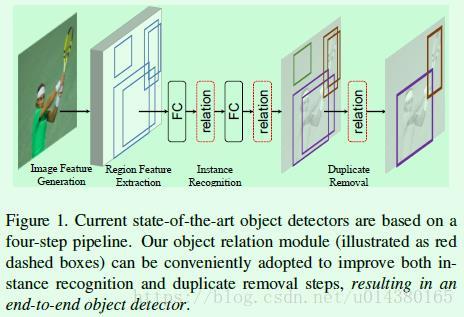
图 19 Relation Networks网络架构

这篇文章对关系的建模还有改进的空间，学出来的所谓“关系”并不清晰，（论文最后声明：our understanding of how relation module works is preliminary and left as future work.欲知后事如何，且听下回分解），更像是把Attention强行套入目标检测系统中。不过可学习的nms是很大的创新。

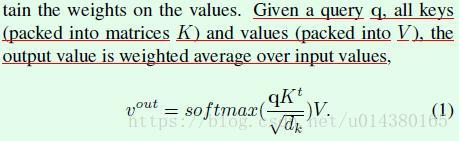
object relation module来描述object之间的关系，从而以attention的形式附加到原来的特征上最后进行回归和分类，另外这篇文章的一个亮点是同时将这种**attention机制引入nms操作**中，不仅实现了真正意义上的end-to-end训练，而且对于原本的检测网络也有提升。

值得注意的是object relation module和网络结构的耦合度非常低，同时输出的维度和输入的维度相同，因此可以非常方便地插入到其他网络结构中，而且可以叠加插入。

在这篇文章中，object relation module主要放在两个全连接层后面和NMS模块，如Figure1中的红色虚线框所示。在Figure1中，作者将目前目标检测算法分为4步：1、特征提取主网络；2、得到ROI及特征（RPN网络就包含在其中）；3、基于ROI做边框回归和object分类；4、NMS处理，去除重复框。从作者的分步情况和源码可以清晰地看出，这篇文章主要是基于Faster RCNN系列算法（Faster RCNN，FPN等）引入object relation module。



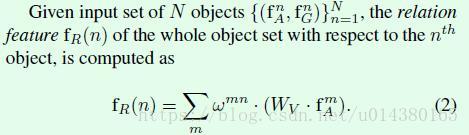
接下来介绍object relation module。因为这篇文章是借鉴了attention is all you need中的attention思想，因此先来了解下那篇文章中的主要公式。在attention is all you need这篇文章中介绍了一个基本的attention模块：scaled dot-product attention，如公式1所示。



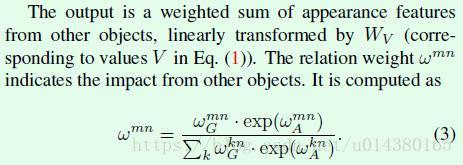
接下来定义两个重要的变量。假设输入中有N个object，那么N个object的两种特征集合如下所示，**fA是常规的图像特征（appearance feature），fG是位置特征（geometric feature）**。这两种特征是后续算法的基础。

这里写图片描述

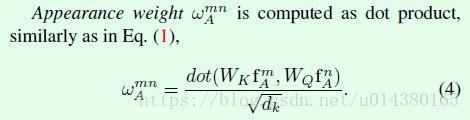
那么受公式1的启发，可以用公式2来刻画第n个object和所有object之间的关系特征（relation feature）。其中fAm表示第m个object的图像特征。Wv是一个线性变换操作，在代码中用1\*1的卷积层实现。wmn是用来描述不同object之间的关系权重（relation weight），该变量通过后面的一系列公式可以得到。计算得到的关系特征fR将和原有的图像特征fA融合并传递给下一层网络，这就完成了attention过程。那么公式2和公式1有什么联系呢？简单讲，公式2中的WV对应公式1中的V，公式2中的wmn对应公式1中的softmax()。



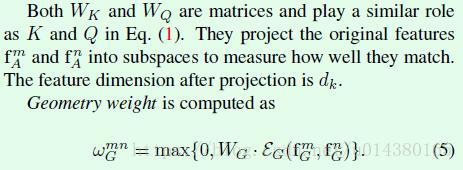
公式2中的wmn通过下面的公式3实现。从公式形式可以看出是一个归一化操作，因此在源码中也是通过softmax层实现（和公式1中的softmax操作对应），只不过对输入中的wG先做了一个log操作，然后将log(wG)+wA作为softmax的输入就能得到公式3。提前说一下该公式中的两个重要变量wG和wA分别表示object的位置特征权重（geometric weight）和object的图像特征权重（appearance weight），后面通过公式4和5分别得到。这里我直接用wG代替wGmn，用wA代替wAmn，一方面是因为实际代码实现中对这些变量的实现都是基于所有object统一得到，另一方面是书写简单。



**公式4是计算公式3中的wA变量，也就是图像特征权重**。在代码中，WKfA通过全连接层来实现，WK就是全连接层的参数，fA就是object特征；同理WQfA也是通过全连接层实现，WQ是全连接层参数。最后再做一个scale操作。



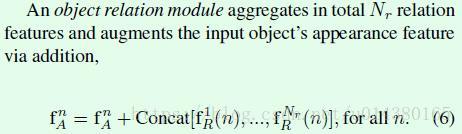
公式5是计算公式3中的wG变量，也就是位置特征权重。首先对fG做了坐标上的变换，如下面截图，主要进行尺度归一化和log操作，可以增加尺度不变性，不至于因为数值变化范围过大引起训练的发散。可以看出坐标变换公式和目标检测算法中的回归目标构造非常相似，最大的不同点在于对x和y做了log操作，原因在于这里要处理的xm与xn、ym与yn之间的距离要比目标检测算法中的距离远，因为目标检测算法中的距离是预测框和roi之间的距离，而这里是不同预测框（或者说是不同roi）之间的距离，因此加上log可以避免数值变化范围过大。EG操作主要是将4维的坐标信息embedding成高维的坐标信息（比如默认是64维），借鉴的是attention is all you need这篇文章的思想，主要由一些cosine函数和sine函数构成。接着，将WG和embedding特征相乘，这里WG也是通过全连接层实现的。最后的max操作类似relu层，主要目的在于对位置特征权重做一定的限制。wG的引入是本文和attention is all you need这篇文章较大的不同点。



fG的坐标变换公式：

这里写图片描述

在得到一个关系特征fR后，这篇文章最后的做法是融合多个（Nr个，默认是16）关系特征后再和图像特征fA做融合，如公式6所示。这里有个细节是关于关系特征的融合方式，这里采用的concat，主要原因在于计算量少，因为每个fR的通道维度是fA的1/Nr倍，concat后的维度和fA相同。假如用element-wise add的方式进行融合，那么每个fR的维度和fA都一样，这样的加法计算量太大。



综上，可以用下面的Algorithm 1来概括前面提到的公式算法，源码中的实现基本上和Algorithm 1相同，这在前面介绍公式的时候已经详细介绍了，这里就不再赘述。注意几个参数的默认值：dk默认是64，dg默认是64。

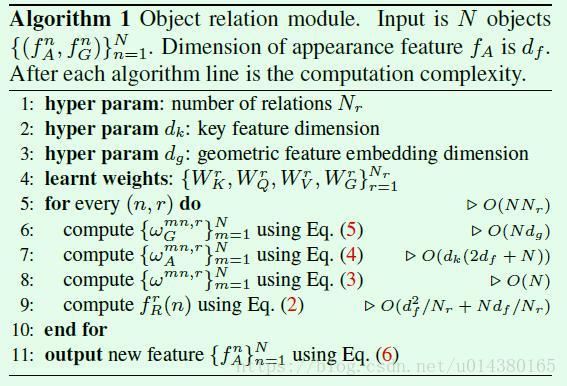
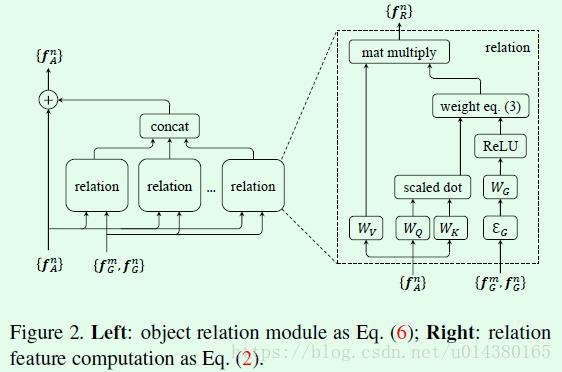
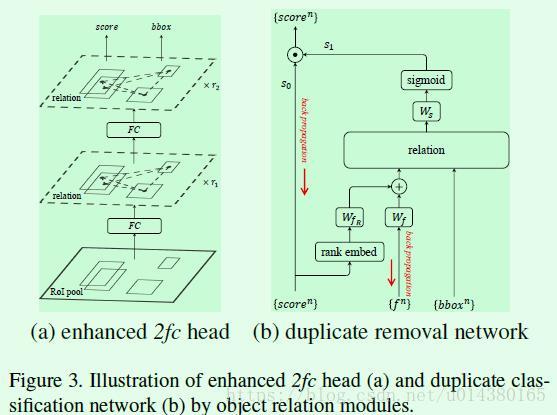
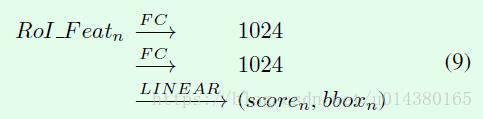


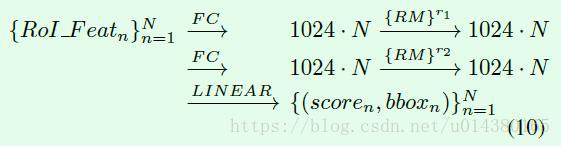
Figure2是用图表来描述Algorithm 1的算法过程。左图是整体上的attention模块和图像特征fA的融合；右图是attention模块的详细构建过程，和前面的公式2~5对应。



介绍完object relation module，接下来就是怎么应用在目标检测算法中了。Figure3是object relation module插入目标检测算法的示意图，左图是插入两个全连接层的情况，在全连接层之后会基于提取到的特征和roi的坐标构建attention，然后将attention加到特征中传递给下一个全连接层，再重复一次后就开始做框的坐标回归和分类。右图是插入NMS模块的情况，图像特征通过预测框得分的高低顺序和预测框特征的融合得到，然后将该融合特征与预测框坐标作为relation模块的输入得到attention结果，最后将NMS当作是一个二分类过程，并基于relation模块输出特征计算分类概率。



下面以object relation module插入全连接层为例，了解下这篇文章大致是怎么做的。首先是原有的全连接层形式如公式9所示：   


这篇文章直接在每个全连接层后面接RM（relation module），输出的维度和输入维度一致，如公式10所示。RM右上角的r1和r2表示数量，源码中默认是1，因为RM不改变输入的维度，所以RM的数量可以任意添加。   


**介绍完relation module在检测模块（全连接层）的嵌入后，另一部分重要的内容是relation module在NMS模块的嵌入。**

NMS的思想简单讲就是根据将和当前预测得分最高的框的IOU大于某个阈值的框过滤。因此这其实是一个二分类的问题，也就是针对每个ground truth，只有一个预测为该类的框是对的，其他预测为该类的框都是错的。

具体实现参考前面的Figure3(b)，其中s0是预测框的socre，这个socre对应公式9和10中最后一行的输出socre；s1是二分类的结果（1表示对的；0表示错的，要移除）。那么这个二分类怎么做呢?主要分成3步：1、先获取输入score的排序信息（index），然后对index做embedding，并将embedding后的index和图像特征（f）通过一些全连接层（维度从1024降为128）后进行融合，融合后的特征作为relation module的输入之一。2、预测框的坐标作为relation module的另一个输入，经过relation module后得到关系特征。3、关系特征经过一个线性变换（Figure3(b)中的Ws）后作为sigmoid的输入得到分类结果，这样就完成了预测框的二分类，且基于的特征引入了attention。

因此将NMS纳入end-to-end训练后，整体网络的损失函数不仅包含原来的坐标回归和分类损失函数，还包含NMS的二分类损失函数。

**6. 小目标检测：Improving Small Object Detection(2017)**

**(1) Motivation**

小目标的分辨率低，形状简单(轮廓粗糙)。一般的目标检测网络在大，中尺寸的目标上能取得较好的效果，但是在小目标的检测上性能不好。

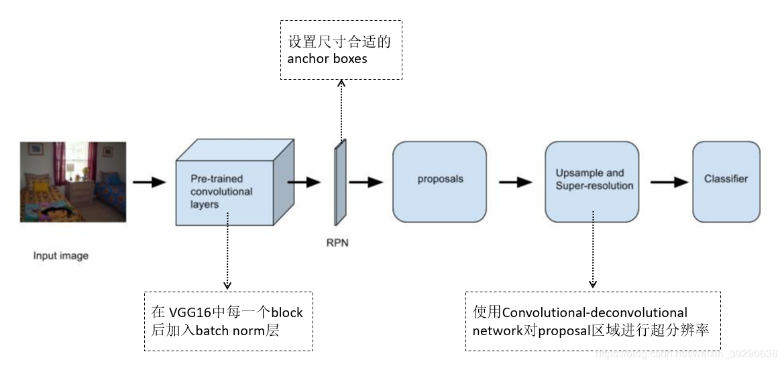
**(2) Contributions**

1. 通过数学推导，确定了[anchor](https://www.baidu.com/s?wd=anchor&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)s的合适尺寸，并进行了详细的试验以证明其选择的有效性。
2. 在网络中加入超分辨率技术，提高了网络的性能。

**(3) Framework**

本文是在Faster R-CNN的算法框架上针对小目标检测的的一些改进。核心思想有两点：

* RPN中anchor size的设置一定要合适，这样可以提高proposal的准确率。
* 对于分辨率低的小目标，我们可以对其所在的proposal进行超分辨率，提升小目标的特征质量，这样更有利于小目标的检测。

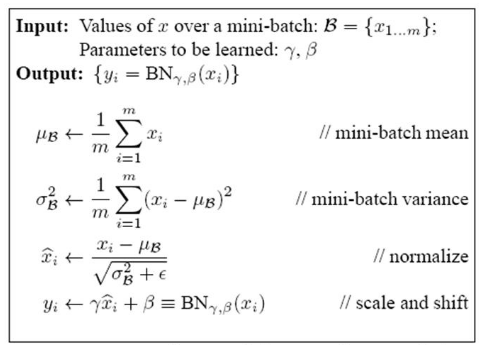


该算法框架和Faster R-CNN基本一致，改动出现在图中标注的3个虚线框处。

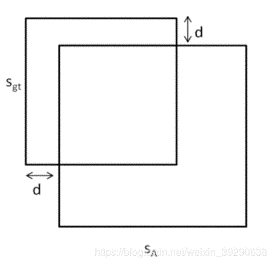
**（3.1） 在 VGG16中每一个block后加入batch norm层**

这里简单介绍下batch Normalization，网上资料很多，读者可自行查找。batch Normalization作为最近年来DL的重要成果，已经广泛被证明其有效性和重要性。它有以下几点好处：

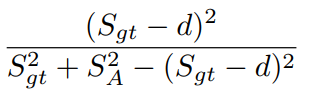
* 极大提升了训练速度和收敛过程。
* 提供了一种类似Dropout的防止过拟合的正则化表达方式，所以不用Dropout也能达到相当的效果；
* 简化调参过程，对于初始化要求没那么高，可以使用大的学习率。



**（3.2） 设置尺寸合适的anchor boxes**



如图3所示，我们假设anchor和ground truth的框是正方形。那么图中[Sgt](https://www.baidu.com/s?wd=Sgt&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)表示ground truth的边长，SA表示anchor的边长，d表示两框在非重合区域中边界的距离。我们知道anchor和ground truth的IoU表示为：



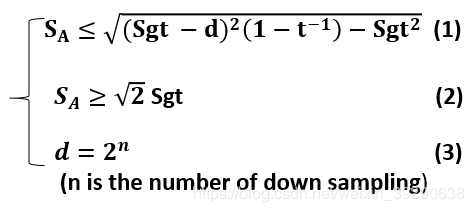
我们设置一个IOU的阈值为t，由min IoU ≥ t可得：

https://img-blog.csdnimg.cn/20181125140558939.png

接着，作者引用了论文([Improving small object proposals for company logo detection](https://arxiv.org/pdf/1704.08881v1.pdf) , ICMR 2017)中的结论：

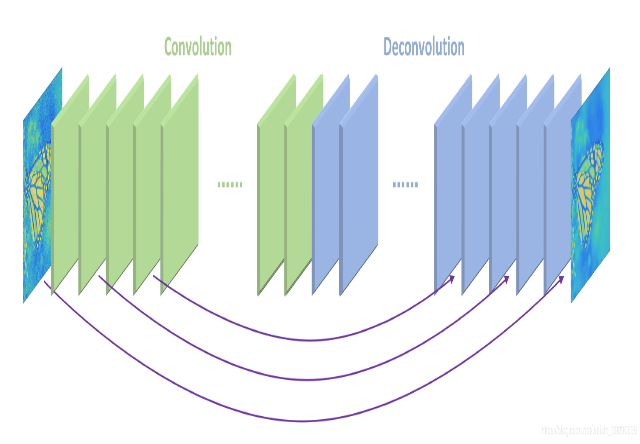
https://img-blog.csdnimg.cn/20181125140401967.png

最后，d最大值等于anchor的步进，而anchor的步进[可由](https://www.baidu.com/s?wd=%E5%8F%AF%E7%94%B1&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)主干网下采样的次数算出。我们可以得出：



**（3.3）使用convolutional-deconvolutional network对proposal区域进行超分辨率**

作者引用了论文(Image Restoration Using Convolutional Auto-encoders with Symmetric Skip Connections, NIPS2016)中的Convolutional-deconvolutional network对proposals进行超分辨率。



该网络的结构是对称的，每个卷积层都有对应的反卷积层。卷积层用来获取图像的抽象内容，反卷积层用来放大特征尺寸并且恢复图像细节。卷积层将输入图像尺寸减小后，再通过反卷积层上采样变大，使得输入输出的尺寸一样。每一组镜像对应的卷积层和反卷积层有着shortcut连接结构，将卷积层的特征和对应的反卷积层输出的特征做相加操作后再输入到下一个反卷积层。这样的结构类似ResNet的残差结构，卷积层和反卷积层学习的特征是目标图像和低质图像之间的残差，损失函数用的均方误差。该网络在ImageNet进行预训练。

超分辨后的proposal送入后续的Classifier进行分类(这里笔者认为还应该有对proposal里的小目标进行再一次regression的操作，但原作者并没有提到)。

**总结：**

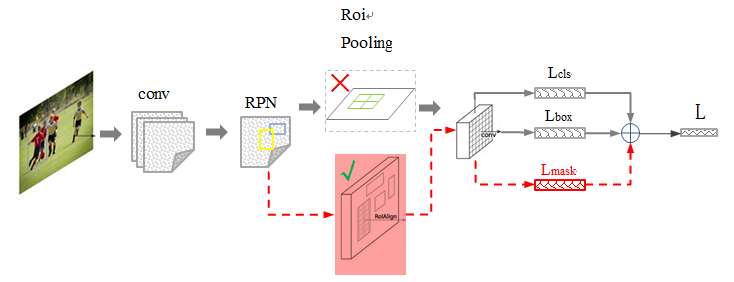
本文基于Faster R-CNN提出了一种改进小目标检测的方法。这种方法是从网络proposal部分入手，将模糊的小目标通过超分辨率清晰化后再进行分类和回归，提高了小目标检测的性能。

**7. Mask-RCNN**

**(1) Mask-RCNN 介绍**

 Mask-RCNN 来自于年轻有为的 Kaiming 大神，通过在 Faster-RCNN 的基础上添加一个分支网络，在实现目标检测的同时，把目标像素分割出来。

Mask-RCNN 的网络结构示意（在作者原图基础上修改了一下）：



其中 黑色部分为原来的 Faster-RCNN，红色部分为在 Faster网络上的修改：

1）将 Roi Pooling 层替换成了 RoiAlign

2）添加并列的FCN 层（mask 层）

先来概述一下 Mask-RCNN 的几个特点（来自于 Paper 的 Abstract）：

1）在边框识别的基础上添加分支网络，用于 语义Mask 识别；

2）训练简单，相对于 Faster 仅增加一个小的 Overhead，可以跑到 5FPS；

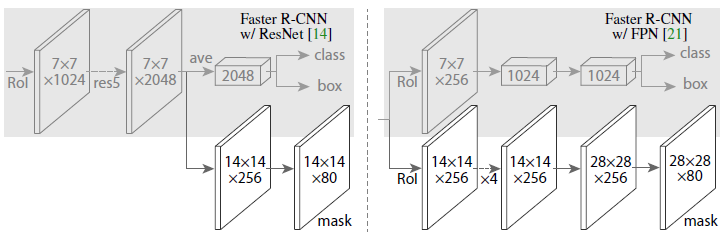
3）可以方便的扩展到其他任务，比如人的姿态估计 等；

4）不借助 Trick，在每个任务上，效果优于目前所有的 single-model entries；

包括 COCO 2016 的Winners。

**(2) RCNN行人检测框架**

来看下 后面两种 RCNN 方法与 Mask 结合的示意图（直接贴原图了）：



 图中灰色部分是 原来的 RCNN 结合 ResNet or FPN 的网络，下面黑色部分为新添加的并联 Mask层，这个图本身与上面的图也没有什么区别，旨在说明作者所提出的Mask RCNN 方法的泛化适应能力 - 可以和多种 RCNN框架结合，表现都不错。

**(3)Mask-RCNN 技术要点**

* 技术要点1 - 强化的基础网络

通过 ResNeXt-101+FPN 用作特征提取网络，达到 state-of-the-art 的效果。

* 技术要点2 - ROIAlign

采用 ROIAlign 替代 RoiPooling（改进池化操作）。引入了一个插值过程，先通过双线性插值到14\*14，再 pooling到7\*7，很大程度上解决了仅通过 Pooling 直接采样带来的 Misalignment 对齐问题。

PS： 虽然 Misalignment 在分类问题上影响并不大，但在 Pixel 级别的 Mask 上会存在较大误差。

后面我们把结果对比贴出来（Table2 c & d），能够看到 ROIAlign 带来较大的改进，可以看到，Stride 越大改进越明显。

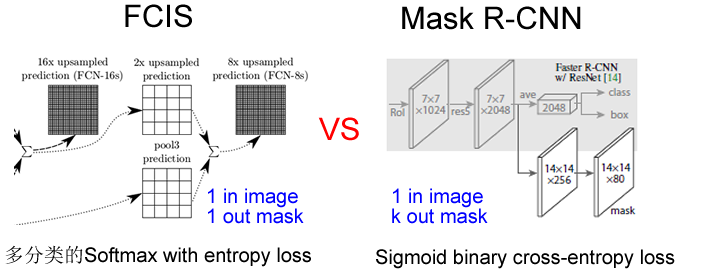
* 技术要点3 - Loss Function

每个 ROIAlign 对应 K \* m^2 维度的输出。K 对应类别个数，即输出 K 个mask，m对应 池化分辨率（7\*7）。Loss 函数定义：

Lmask(Cls\_k) = Sigmoid (Cls\_k)

平均二值交叉熵 （average binary cross-entropy）Loss，通过逐像素的 Sigmoid 计算得到。

Why K个mask？通过对每个 Class 对应一个 Mask 可以有效避免类间竞争（其他 Class 不贡献 Loss ）。



**8. VoxelNet：基于点云的三维物体检测的端到端学习**

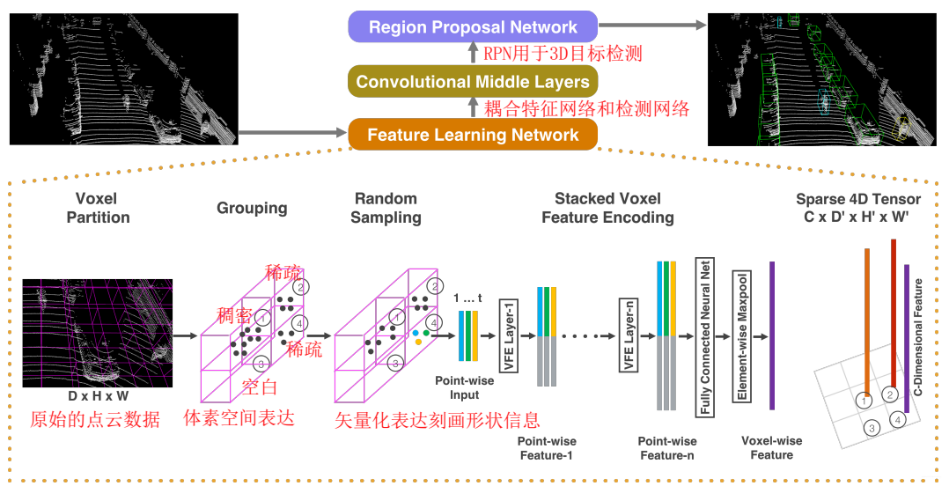
**(1)论文综述**

3D点云中目标的精准检测是很多应用场景的核心问题，如自动驾驶，家居机器人，虚拟/增强现实。为了将高度稀疏的LiDAR点云与区域候选网络连接在一起，很多研究学者将关注带你放在手工设计的特征表达，例如鸟瞰投影a bird's eye view projection. 这份工作中作者放弃了人工设计特征，直接提出了端到端的VoxelNet进行3D目标检测。具体来说，VoxelNet将点云划分为等间距的三维体素，并通过新引入的体素特征编码(VFE)层将每个体素内的一组点转换为统一的特征表示。这样，点云可以被编码成可以描述的体积表征，进而他被量接到区域候选网络进行目标检测。在KITTI数据集上，作者的方法取得了最好的结果。

* VoxelNet是一种基于点云的-可训练的-端到端的-3D目标识别框架，可以直接作用于系数的3D点云，避免了特征工程；
* 该框架融合了稀疏点云结构，并且充分利用了voxel grid的并行处理
* 该文也提出了一种有效的数据增强策略

**(2)主干和细节**

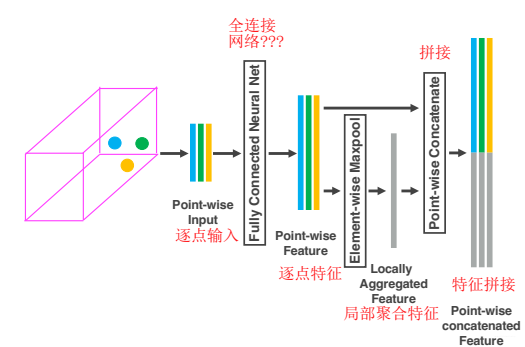
将三维点云划分为一定数量的Voxel，经过点的随机采样以及归一化后，对每一个非空Voxel使用若干个VFE(Voxel Feature Encoding)层进行局部特征提取，得到Voxel-wise Feature，然后经过3D Convolutional Middle Layers进一步抽象特征（增大感受野并学习几何空间表示，也就是我们经常说的浅层网络学习到的是基础视觉特征，深层网络学习到的是高级视觉语义特征），最后使用RPN(Region Proposal Network)对物体进行分类检测与位置回归。VoxelNet整个pipeline如下图所示：



VoxelNet网络结构. 特征学习网络直接输入原始的3D点云数据，然后将整个3D空间划分成独立的小voxel，每个voxel都采用特征提取网络进行特征提取，最后将各个特征按照原来的几何结构拼接在一起[这就是我们之前经常说的Global=Multi-Parts]。这样做的目的应该是为了抵抗形变。Middle Layer的目标在于增大感受野，提取更加高级的语义特征。最后交给3D-RPN网络进行BoundingBox回归。

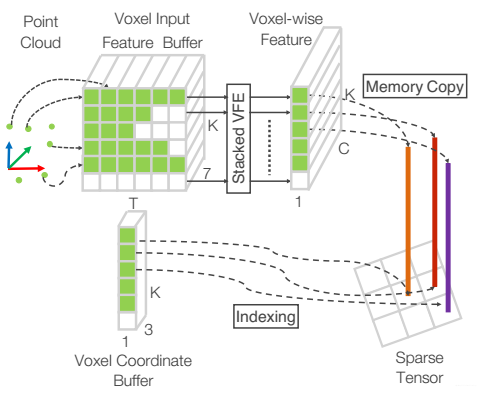
**(2.1)点云的多层次学习**

VoxelNet是对PointNet以及PointNet++这两项工作的拓展与改进，粗浅地说，是对点云划分后的Voxel使用"PointNet"。我们可以仔细看一下某一层VFE，如下图所示，一块Voxel中的三个点经过FCN抽象Point-wise feature，并使用MaxPool得到Locally Aggregated Feature(局部聚合特征)，然后将这个局部特征concatenate到每一个Point-wise feature上。



**(2.2)点云的高效查询**

此外，由于点云具有高度的稀疏性以及密度不均匀性，作者利用哈希表查询的方式，可以做到快速找到每一个Voxel中的点在三维点云中的具体位置，如下图所示。



**(3)论文总结**

在3D场景中，RGB信息对3D Detection不是特别重要。因为汽车、自行车、人这三类物体仅仅通过外形轮廓就能够区分出来，如果网络能够很好地学习到这些几何空间特征，那么只需要点云就能得到很好的效果。但是如果是针对3D Instance Segmentation这类任务，比如区分黄车与黑车，LiDAR data与RGB data之间进行fusion就很有必要了。

直接将两种data（或者两者对应的feature map）进行fusion操作，这种数据处理方式可能会使得神经网络不容易去学习更好的特征。比如人具备颜色信息感知与空间位置感知，但是这两种感知并不是混为一谈的，两者应该是并行且存在交互的关系，这种交互关系可能比目前先concatenate再通过若干层全连接层的fusion机制要更加高级抽象。

**Architecture**

* Feature learning network：实现从点云数据中提取特征并且三维像素化；
* Convolutional middle layers；将三维的feature map进行一系列的卷积操作，实现特征交互；
* Region proposal network；RPN在三维feature map中的实现，提取每一个anchor的概率值和回归值。（不太准确，后期再准确化）

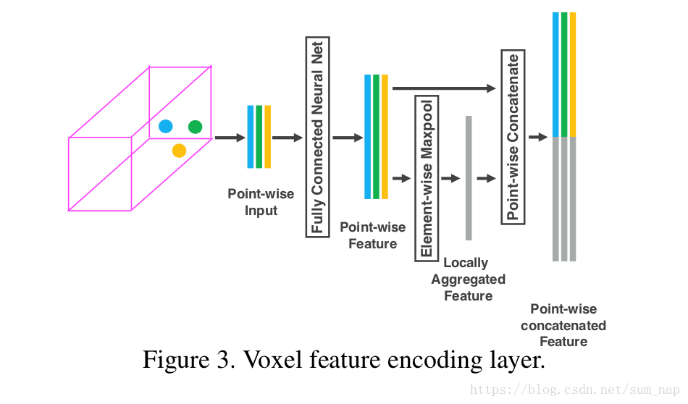
**Feature learning network**

voxel partition（空间像素化）：将点云数据在3D空间中进行像素化（equally）。在三维空间中，相应的正交坐标系的坐标为（X、Y、Z），激光雷达采集到的数据的信息的区间（range）分别为D（深度）、H（高度）、W（宽度），可以视为正交坐标的同等表达。假设像素化的过程中，每一个像素小格子的包含的区间为：d、h、w。则整个数据的三维像素化的结果在各个坐标上生成的像素小格子（grid）的个数为：D/d ；H/h；W/w；注：以下D、H、W为整体像素的的集合。

Grouping（像素格内点的成组）：因为点云数据中的点是完全离散的，所以像素化后每个像素格内有不同个数的点，将这些点特征可以通过一些方式的整理作为该像素格的初始特征表示。

Randon Sampling（像素格内随机选取点）：因为点云数据中，各个特征点的部分极为不均匀并且点的数量非常大（100k）；所以将每一个点都作为特征表示的初始信息一方面增加销量负担，另一方便点密度的差异也会给后续的检测工作带来偏差，所以，在一个像素格内选取部分不超过某个固定数量的点数就是一种比较合适的做法，选取的方式为随机选样。

Stacked Voxel Feature Encoding（栈式VFE处理）：The key innovation is the chain of VFE layers。这句话我觉得没毛病。VFE层作为特征学习层的主要元结构，通过一系列的VFE操作，将像素格中一系列点的特征encoding为像素的特征值。



我们设：

C:\Users\12544\AppData\Local\Temp\1553001743(1).png

为一个非空像素格，p为其内部选样后的点集。其中

C:\Users\12544\AppData\Local\Temp\1553001783(1).png

的四个坐标指分别为空间坐标和反射率。

第一步：点特征扩展，求出像素格的中心点C:\Users\12544\AppData\Local\Temp\1553001855(1).png

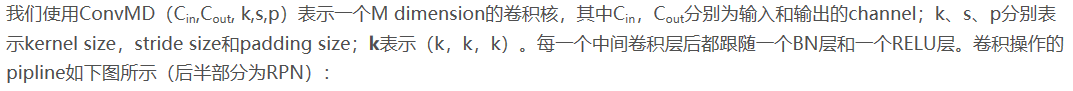
将每个点的特征扩展为：C:\Users\12544\AppData\Local\Temp\1553001885(1).png

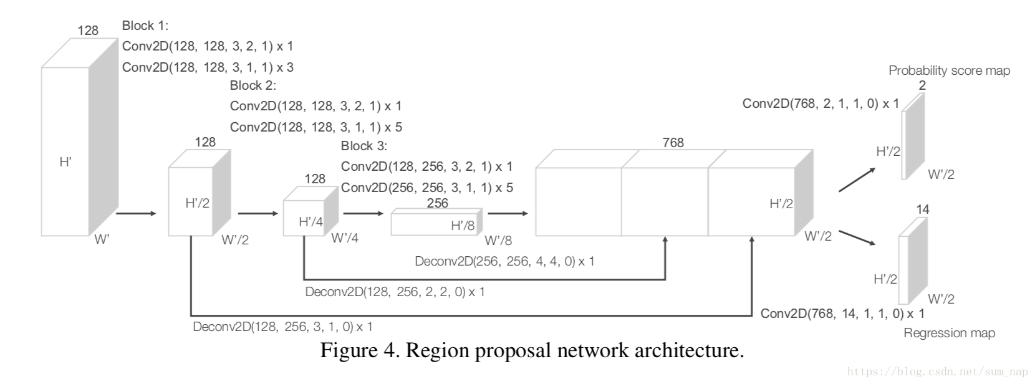
第二步，用一个全连接的操作（FCN）将点的特征扩展。扩展之后的特征设为f\*（扩展前特征f，维度=7）。扩展之后，再将不同点（最多T=35个点）进行maxpool，生成一个与f\*维度相等但是将T=35个点进行合并的特征向量f\*\*

第三步，然后将（T=35）个f\*与逐个maxpool后的f\*\*拼接，生成VFE层的输出。以上为VFE层的基本操作。

VFE中所有的全连接参数为共享参数，那么本层需要学习的参数量为c\_in\*c\_out/2个。

**Convolutional Middle Layers**

****

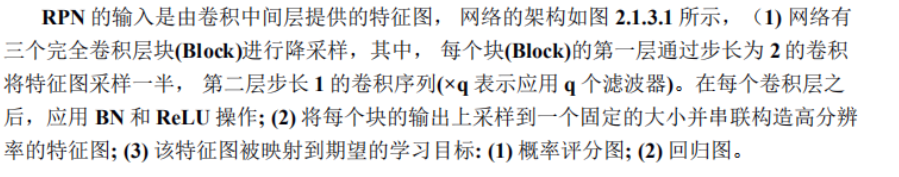


**Region Proposal Network**

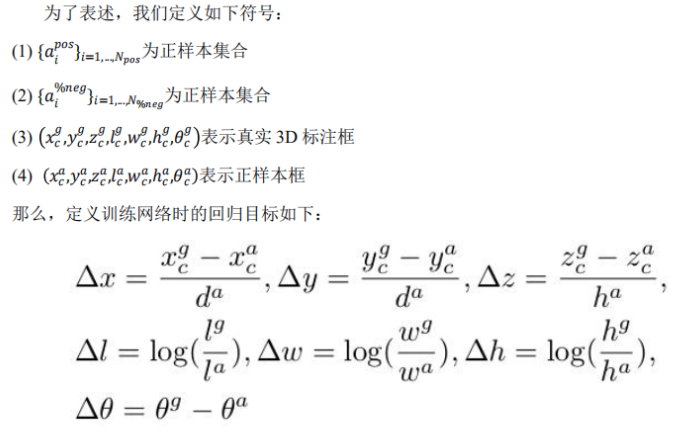
鉴于RPN在2D的detection 任务中取得比较好的效果，在这里，将RPN在3D的detection任务中与前述的卷积结果进行改造和结合。

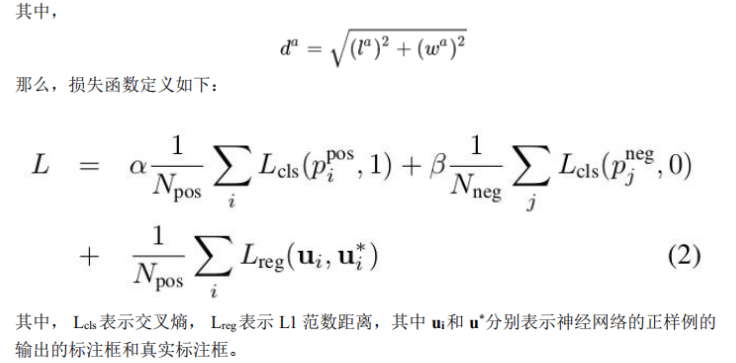
卷积层的最终输出以及中间结果进行一系列上采样或者下采样操作合并为相同size大小的特征，然后进行（通道叠加）合并。作为送入RPN的特征，然后用个固定大小的anchor进行object detection，因为主要检测的物体分为三类：Car、Pedestrian、Cyclist；所以不同的物体同不同大小的anchor，因为后两者大小比较类似，所以Pedestrian和Cyclist用的是同样大小的anchor。

个人理解：因为激光雷达有个好处，就是不会有多尺度的问题，所以，在生成的3D的voxel map中，这些物体几乎保持同样大小，所以每个物体只需要一个固定大小的anchor即可。

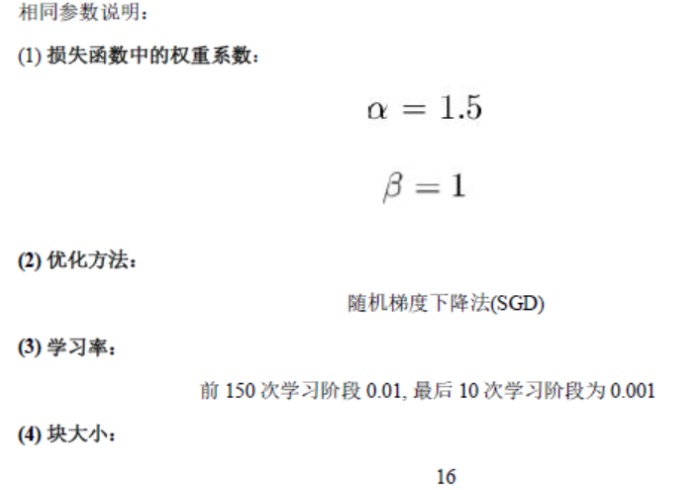


**Loss**

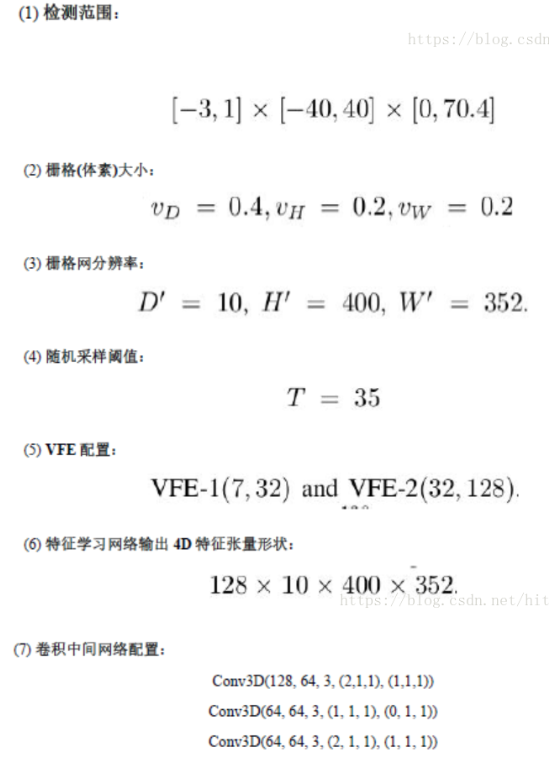
****

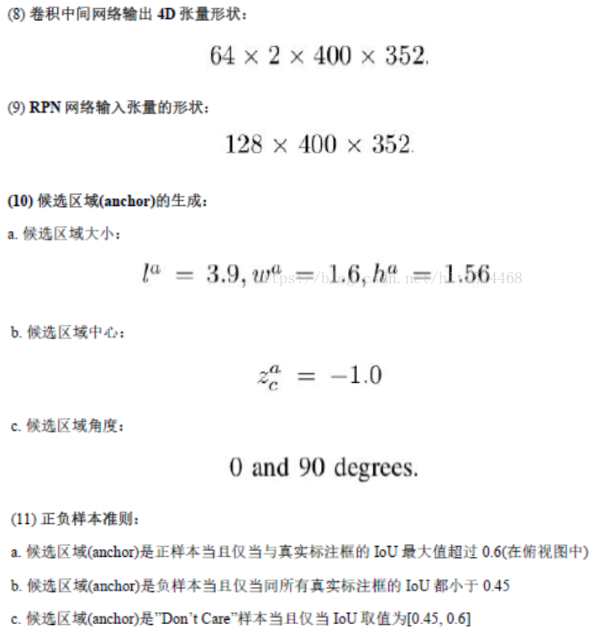
****

**训练**

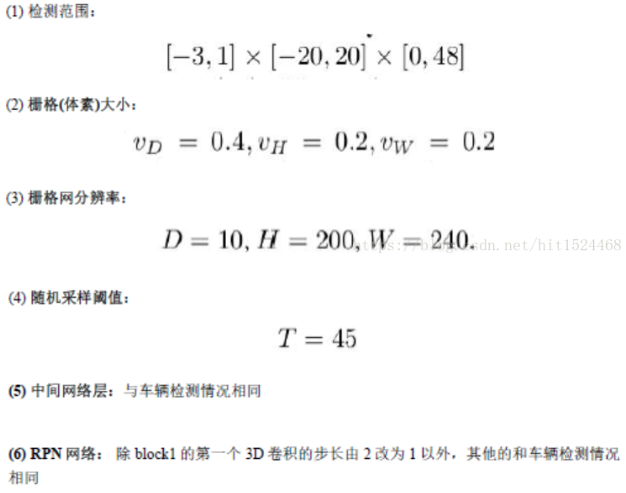
****

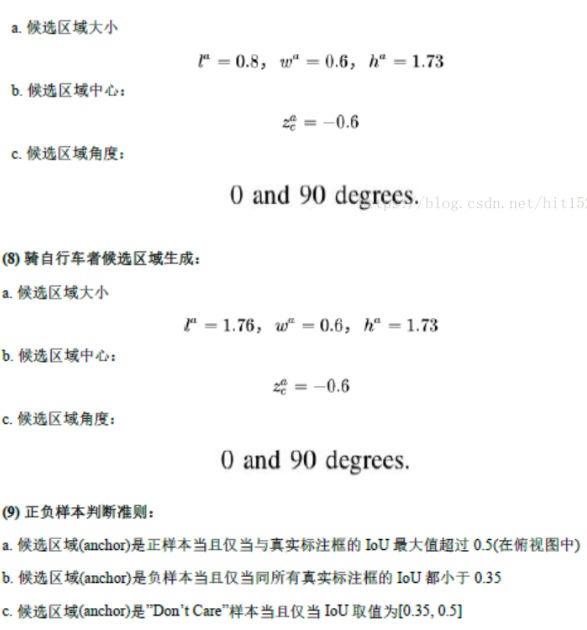
**车辆检测**

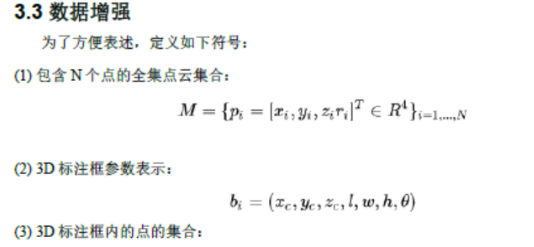
****

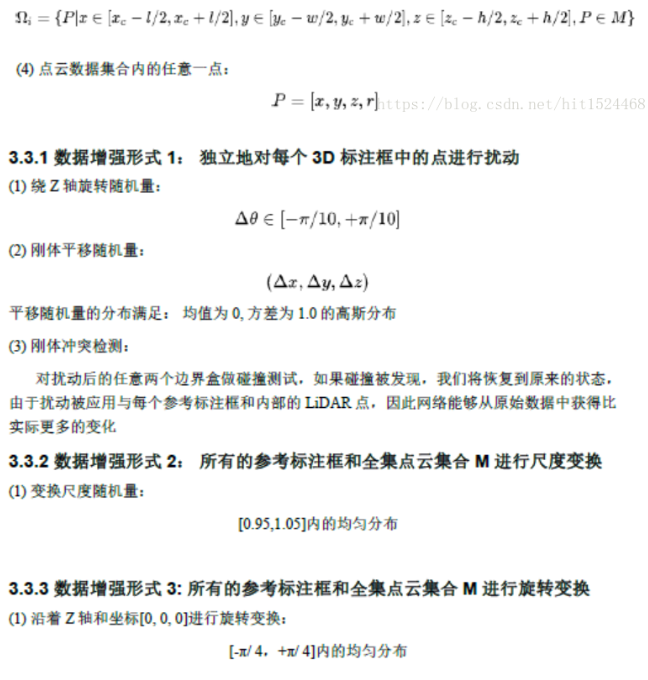
****

**行人检测**

****

****

****

****

**9. Acquisition of Localization Confidence for Accurate Object Detection（IoU-Net 让目标检测用上定位置信度）**

本文指出定位置信度与分类置信度不匹配的问题，并把定位问题从回归的方式改为基于优化的方式，从而提升定位精度。

**Motivation**

当前two-stage的目标检测任务流程为：

1. RPN找出一系列候选框，区分前景背景

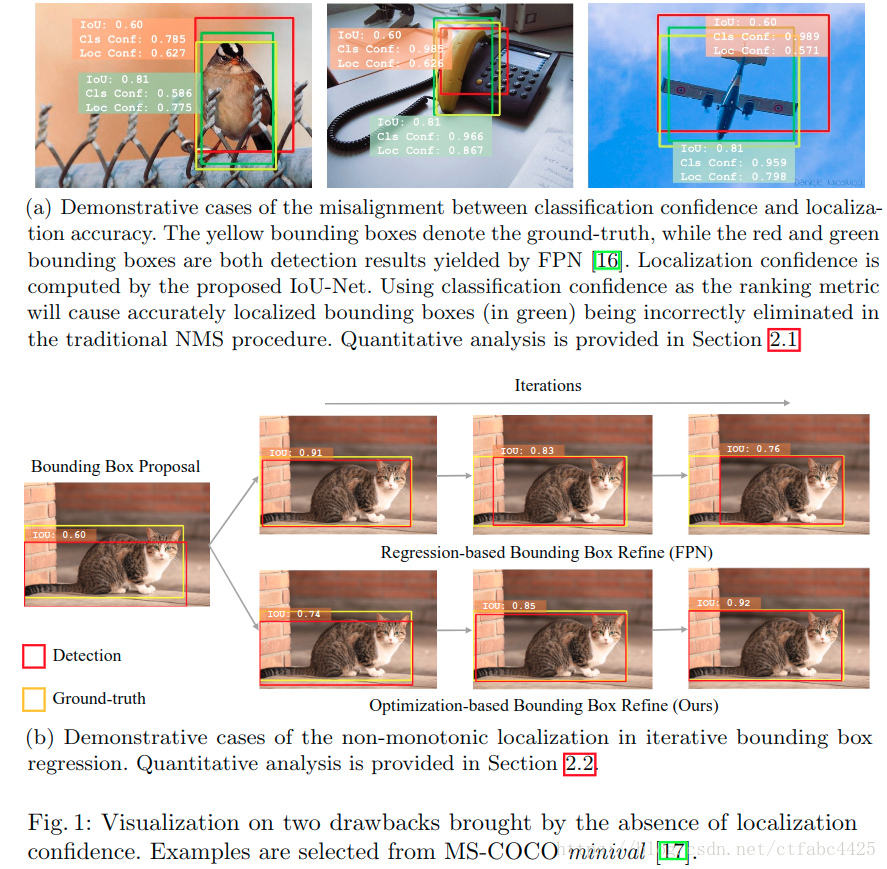
2. 对前景框进行分类和回归，得到每个框的分类置信度及回归坐标

3. 通过NMS，以分类置信度排序，过滤掉IoU大于一定阈值的框，得到最终的结果

可以看到，在这个过程中，对于定位置信度没有考虑。这样会出现两个问题：

（1）分类置信度高不代表IoU大，如上图(a)所示，以分类置信度当做这个框的置信度做NMS， 会把与ground-truth的IoU大，但是分类分数低的框给抑制

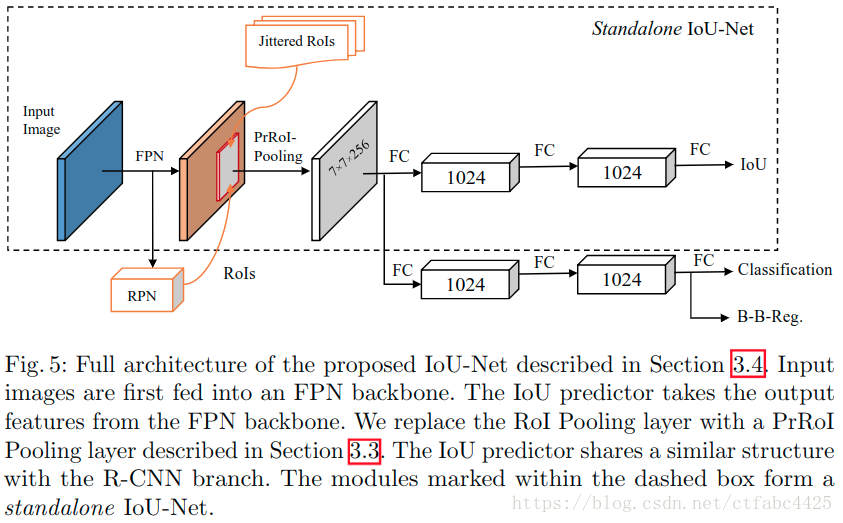
（2）边界框回归算法的可解释性较差，在Cascade RCNN论文中指出：边界框回归不是单调的，如上图(b)所示，如果边界框回归次数过多，结果可能会退化。



基于上述观察，本文探索将IoU置信度加入到目标检测的流程中，并以一种新的思路：基于优化的方法来进行边界框修正

**Contribution**

**IoU-Net**



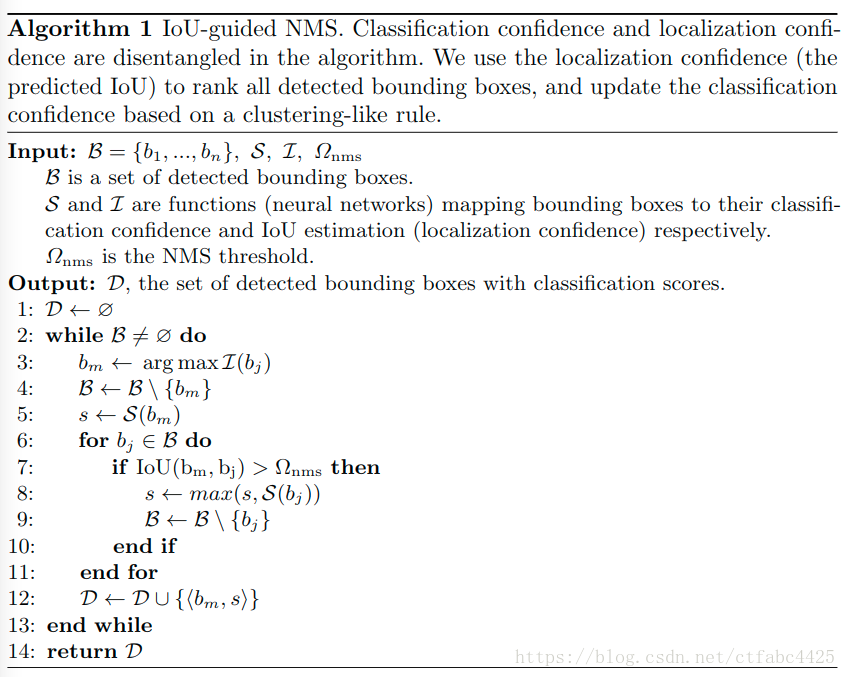
IoU-Net网络结构如上图所示，与Faster RCNN的主要区别在RoI-Pooling 换成了PrRoI-Pooling, 加入了一个IoU预测器。

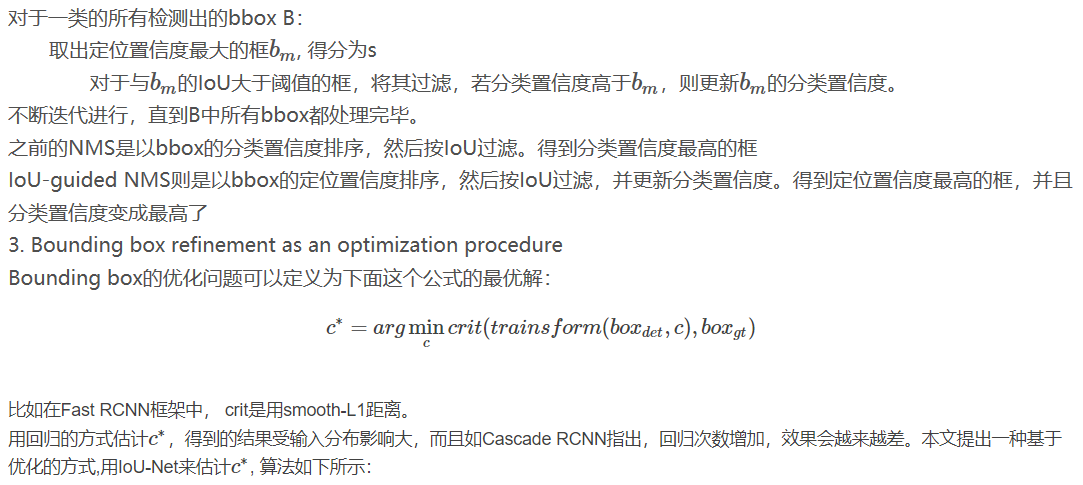
1. IoU predictor

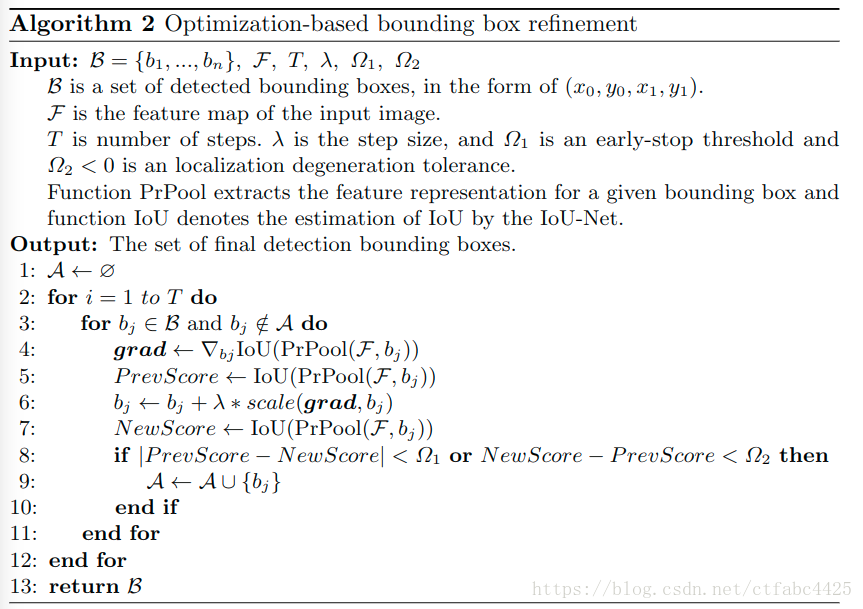
本文IoU预测器的输入不是用RPN产生的RoIs，而是对Ground Truth框随机扰动，得到Jittered RoIs，过滤掉与GT的IoU<0.5的框，剩下的作为输入，经过PrRoI Pooling给IoU predictor。

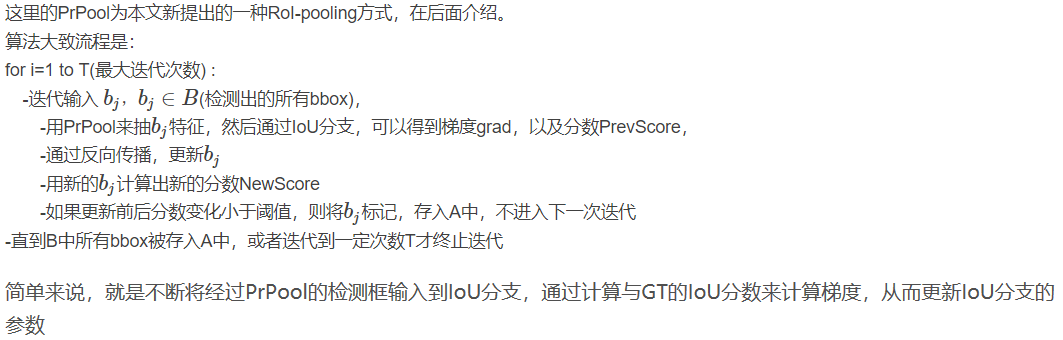
2. IoU-guided NMS

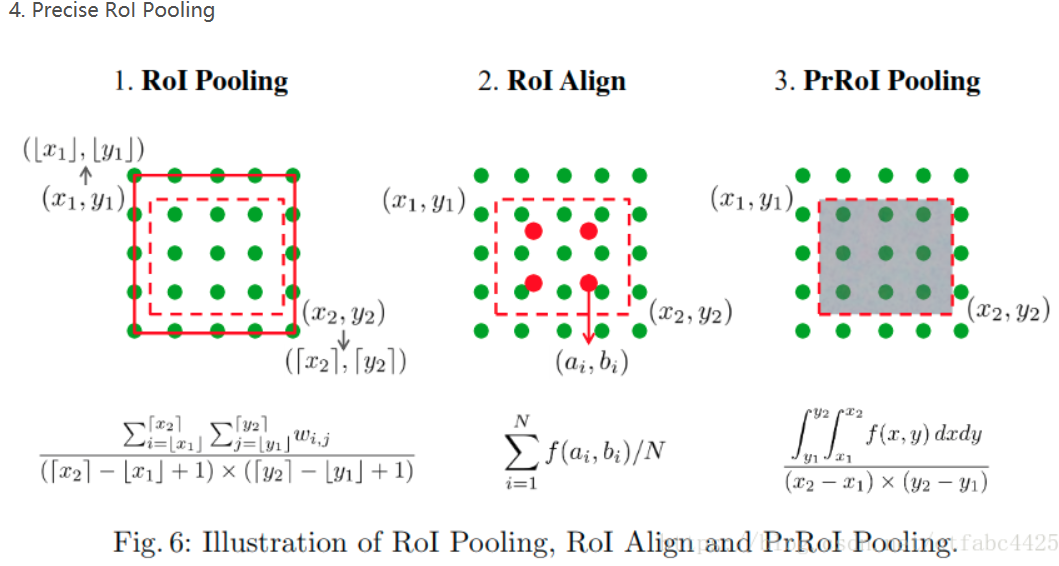
分类置信度与定位置信度不是对齐的。本文提出了IoU-guided NMS，将IoU与分类置信度结合在一起。算法如下图所示。

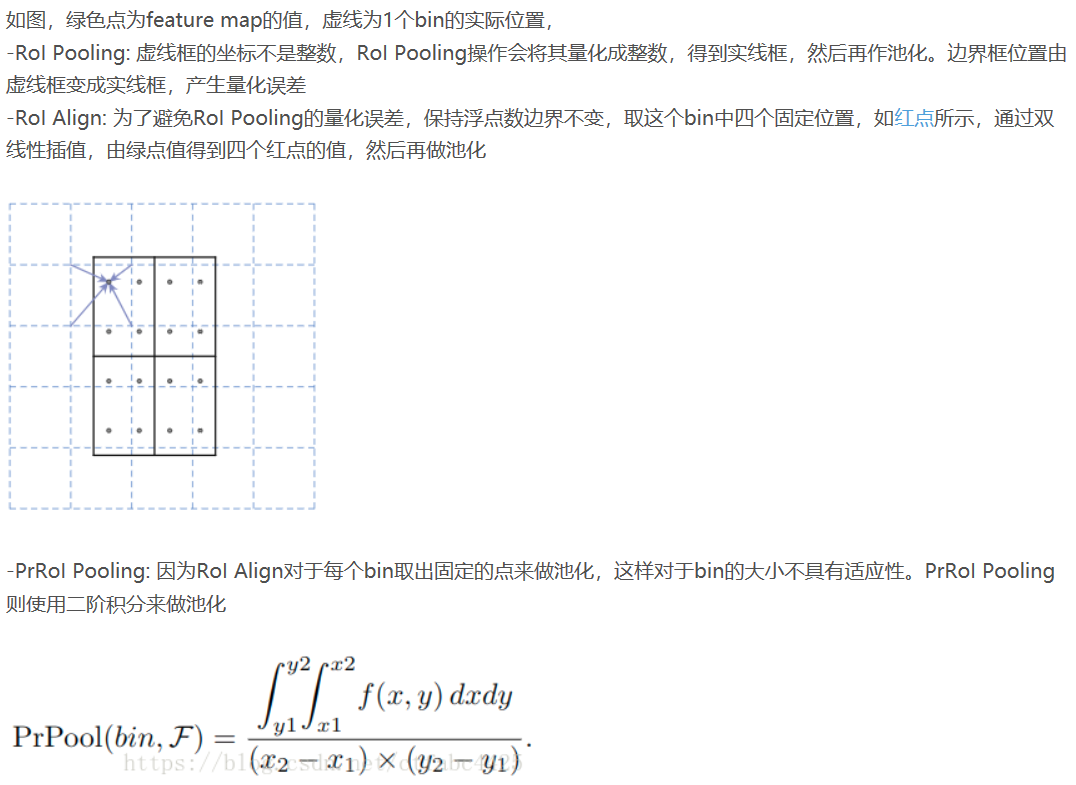


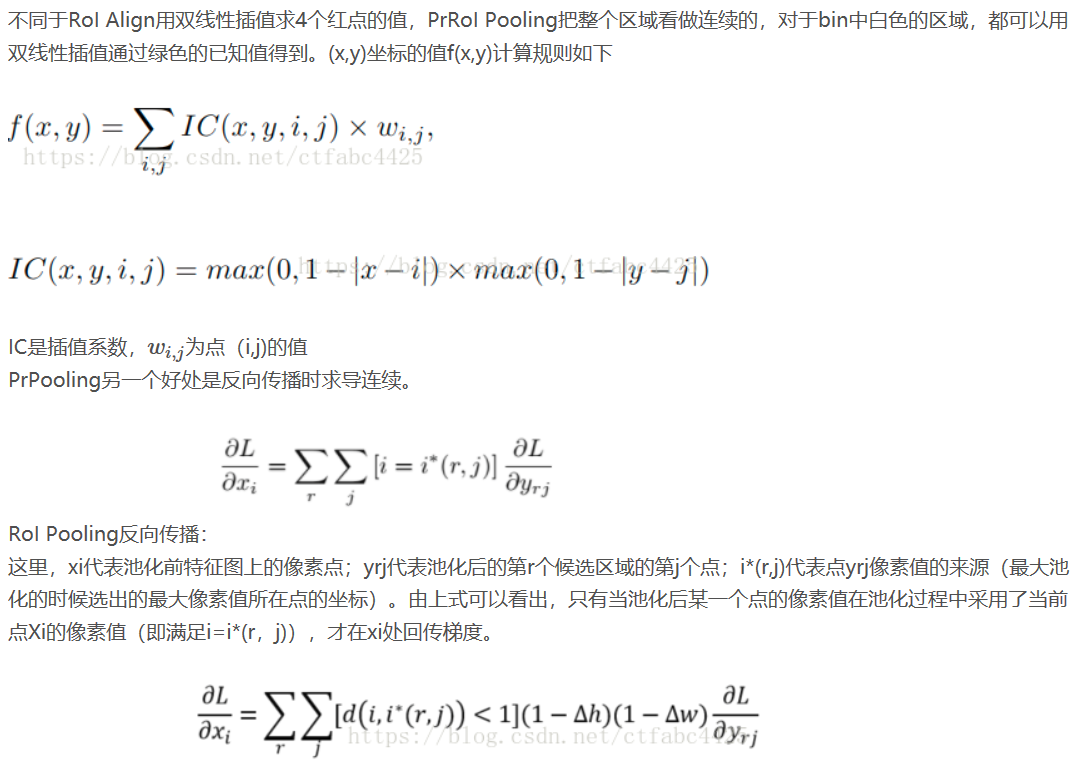


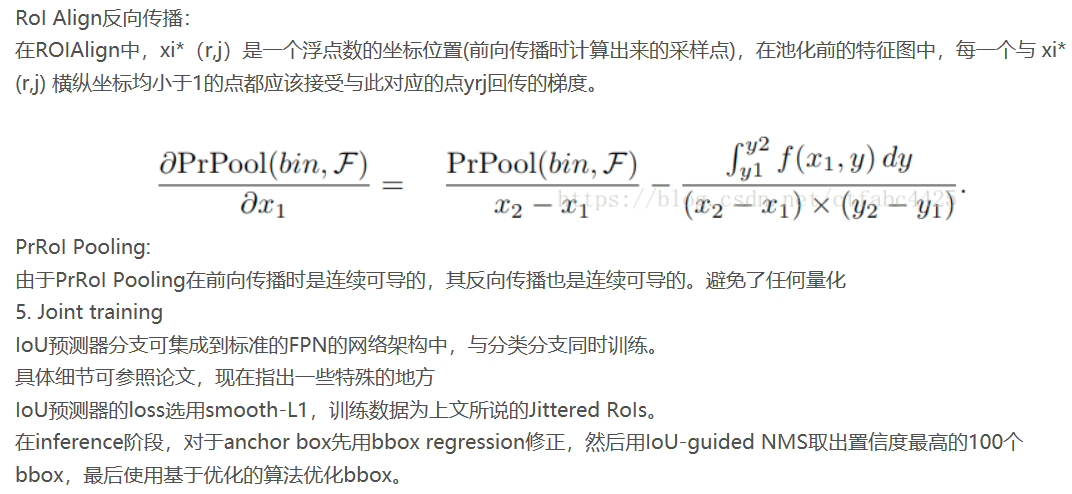












**Conclusion**

本文提出一篇新颖的修正目标定位的网络IoU-Net，指出了分类与定位置信度不匹配的问题，提出IoU-NMS来获取位置更精确的bbox，并将定位的回归问题重新定义为优化问题，提升了定位的精度。同时IoU预测器可方便集成到别的网络。