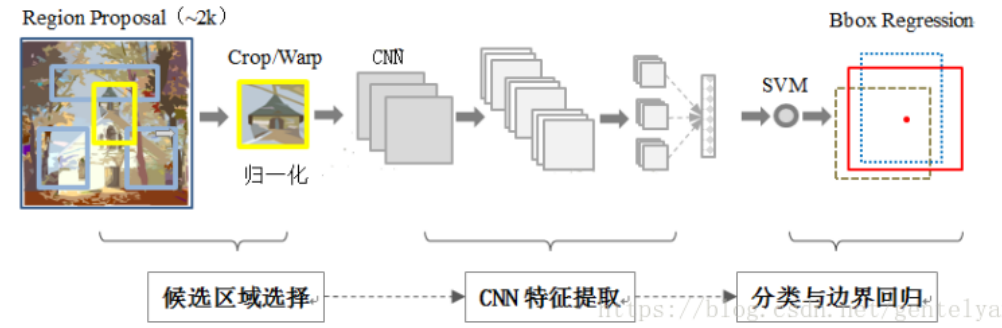
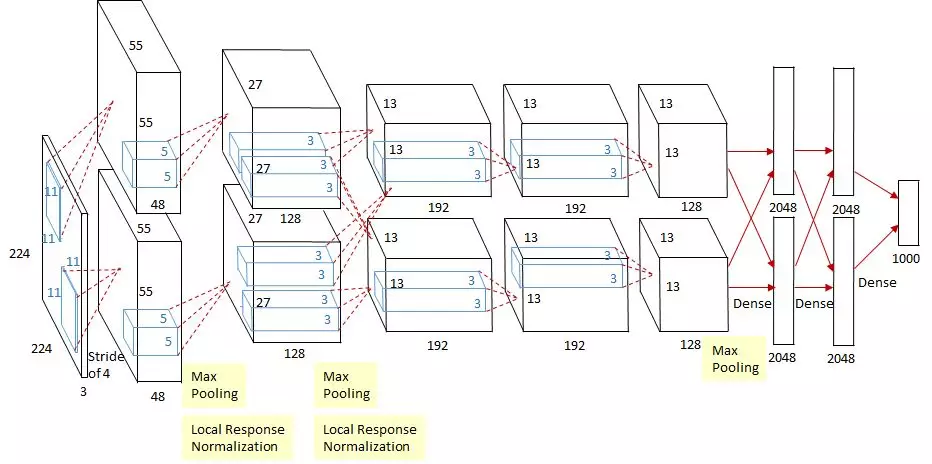
**RCNN**



1. 使用selective search算法生成大约2000个候选区域（region proposal）
2. 将所有候选区域缩放成相同尺寸（277\*277），送入CNN特征提取模块
3. CNN特征提取模块采用AlexNet模型





以C3卷积层为例：

C3卷积层输入的是大小为27\*27\*256的特征图，在C3卷积层中使用256个大小为3\*3\*256的卷积核进行步长为2的卷积，填充为valid（不填充，若为same则填充以保证输出特征图长宽与输入特征图一致），因此输出特征图大小为13\*13\*256（（27-3）/ 2 + 1 = 13），输出作为最大池化层的输入继续传播。

AlexNet采用双GPU协同训练。该模型一共分为8层，包括5个卷积层和3个全连接层，每个卷积层都包含[激活函数](https://so.csdn.net/so/search?q=%E6%BF%80%E6%B4%BB%E5%87%BD%E6%95%B0&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_52053775/article/details/_blank)、ReLU、池化和LRN（局部响应归一化）处理。AlexNet在第2,4,5层均是前一层自己GPU内连接，第3层是与前面两层全连接，全连接是2个GPU全连接。也就是说，第三层的输入的通道数是256而非128，其余卷积层都是总通道数的一半。

RCNN训练使用PASCAL VOC数据集，其中有20类物体。对Pre-trained AlexNet模型进行fine-tuning，将原来预训练模型最后的1000-way的全连接层（分类层）换成21-way的分类层（20类物体+背景）。训练前需要准备正负样本，如果某个region proposal和当前图像上的所有ground truth中重叠面积最大的那个的IOU大于等于0.5，则该region proposal作为这个ground truth类别的正样本，否则作为负样本（背景）。IOU=(A∩B)/(A∪B)

但是RCNN并不根据CNN最后21层做softmax分类，而是用F10层4096维的特征图作为输入重新训练21个SVM分类器。在SVM分类过程中，IOU<0.3被作为负例，ground-truth（即完全框住了物体，默认IOU＞0.7时）是正例，其余的全部丢弃。然后SVM分类器也会输出一个预测的labels，然后用labels和truth labels比较，计算出loss，然后训练SVM。

这是因为CNN需要大量的数据，所以设置的阈值较低（0.5），即使一个bounding box可能只包含物体的一部分，那么我也把它标注为正样本，用于训练CNN。如果数据过少，容易造成过拟合。而SVM适用于小样本训练，可以设置更严格的阈值。

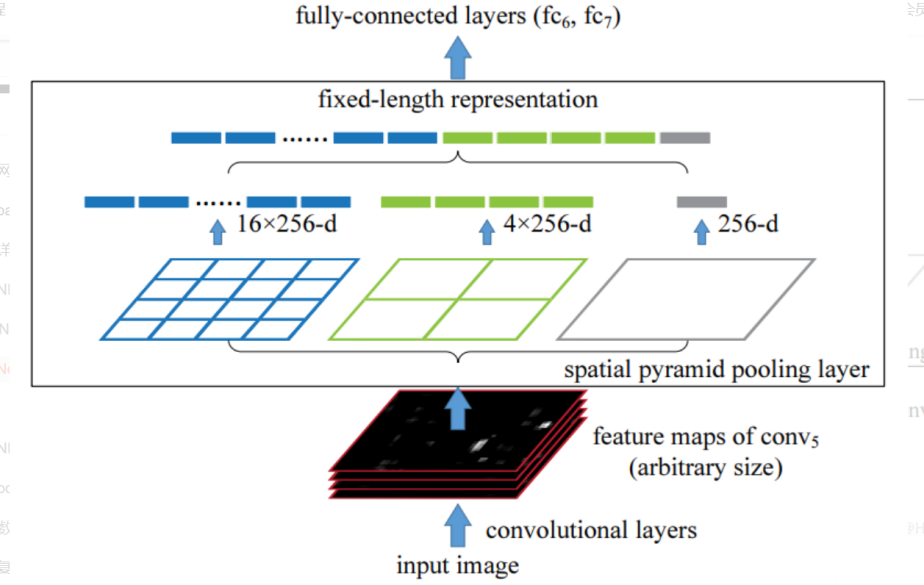
由于SVM是小样本训练，所以会存在负样本远多于正样本的情况。针对这种情况，作者使用了hard negative mining方法。初始时用所有样本训练，但是这样负样本可能远多于正样本，经过一轮训练后将score最高，即最容易被误判的负样本加入新的样本训练集，进行训练，重复以上步骤直至分类器性能不再提升。Data-minig的作用就是去掉那些对优化作用很小的Easy-examples，保留靠近分界面的Hard-examples。

SS算法：在cv2中有现成函数cv2.ximgproc.segmentation.createSelectiveSearchSegmentation()可以直接生成RP。其基本思路是使用一种过分割手段将图像分割成小区域，然后不断合并（根据颜色、梯度、面积、位置）可能性最高的小区域，直到整个图像合并成一个区域位置，曾经存在过的区域即为候选区域。

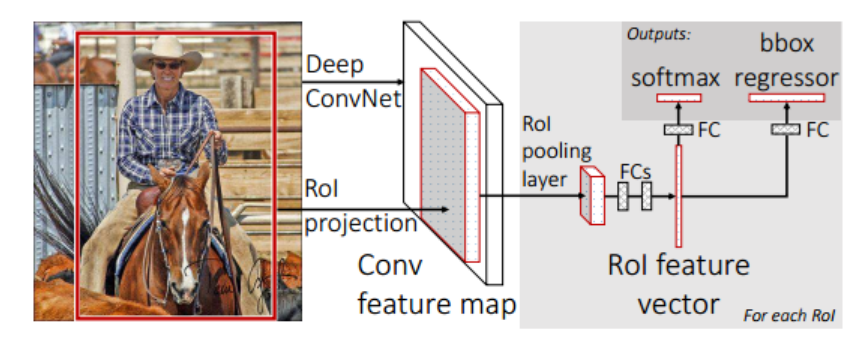
**Fast RCNN**

SPP-net：无需将输入的RP调整到规定大小，保留图像长宽比等信息。

RCNN需要固定大小的输入是因为全连接层需要固定尺寸的输入。SPP-net在全连接层前加入了空间金字塔池化层，将每一层特征图分别划分成 4×4、2×2、1×1 大小的子块，得到21个子块。对每一个子块做max pooling，得到大小为21的特征向量，将256个特征向量连接形成一个固定长度的特征向量，一共包含21\*256维特征。

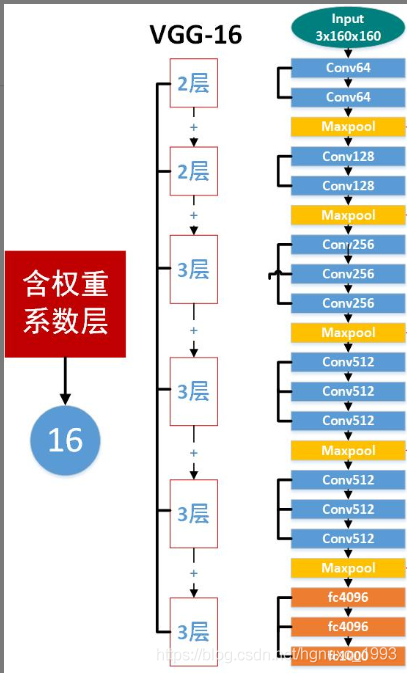


Fast RCNN

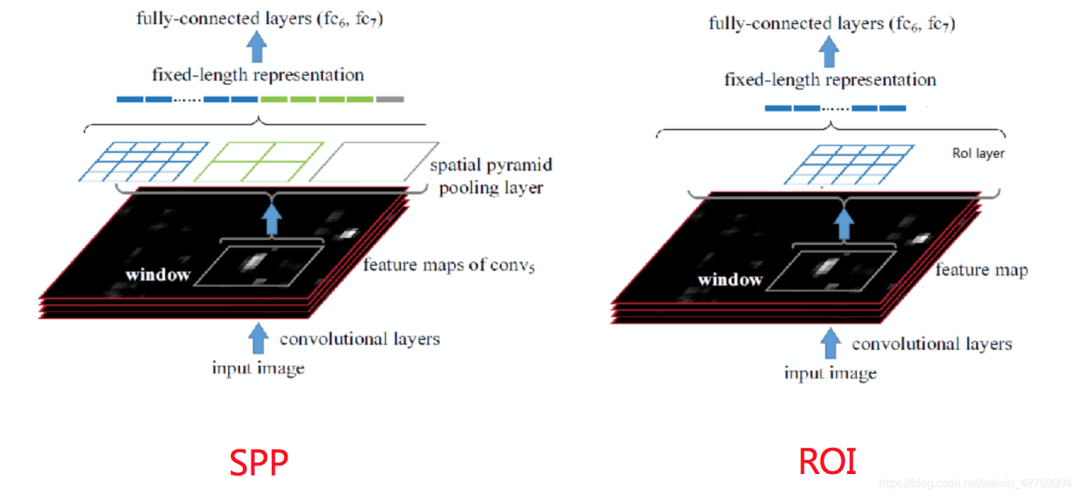


SPPnet依然存在和RCNN一样的一些缺点比如：训练步骤过多，需要训练SVM分类器，需要额外的回归器，特征也是保存在磁盘上。Fast RCNN相当于全面改进了原有的这两个算法，不仅训练步骤减少了，也不需要额外将特征保存在磁盘上。

Fast RCNN的输入是224\*224，Deep ConvNet基于VGG16模型（与AlexNet类似），由5个卷积层和2个降采样层构成（这两个降采样层分别跟在第一和第二个卷积层后面），通过Deep ConvNet得到整张图的feature map，再通过RoI projection将RP区域位置映射到feature map中。由于Fast RCNN是对整张图片而非每一个RP做卷积，因此可以节省大量的计算时间。



RoI pooling层的作用是对不同大小的region proposal，从最后卷积层输出的feature map提取大小固定的feature map。RoI是单尺度的，且大小可以动态调整，而SPP是多尺度的固定的4×4、2×2、1×1 大小的子块。单尺度会使精度略有降低，但能节省很多时间。

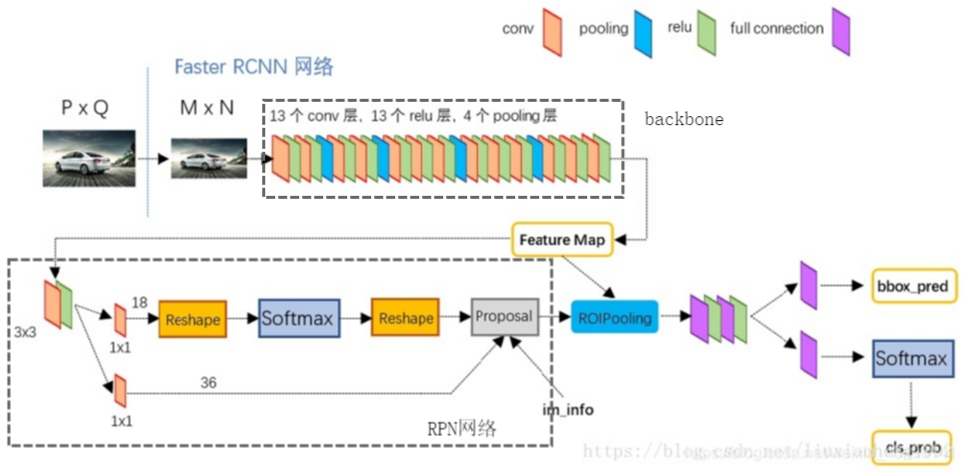


RoI pooling的输出经过两个output都是4096的全连接层，然后分别经过output个数是21和84的两个并列的全连接层，前者是分类的输出，代表每个region proposal属于每个类别（21类）的得分，后者是回归的输出，代表每个region proposal的四个坐标。最后是两个损失层，分类的是softmaxWithLoss，输入是label和分类层输出的得分；回归的是SmoothL1Loss，输入是回归层的输出和target坐标及weight。

Fast RCNN主要的改进在于其对整张图片进行卷积，ROI层采用单尺度，分类和回归在网络中一起训练，但是其RP的提取还是需要使用SS算法，这是其后续改进方向之一。

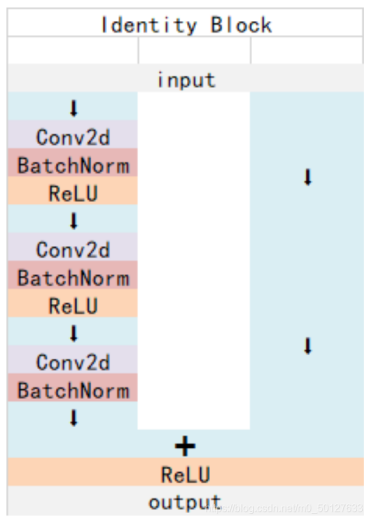
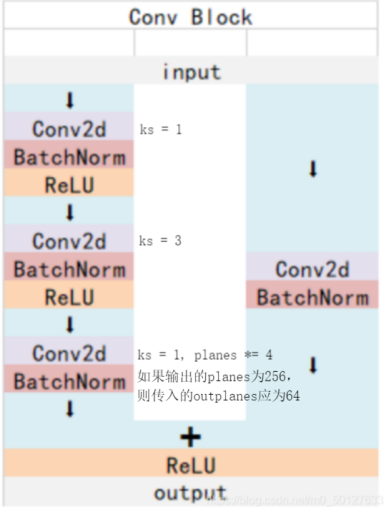
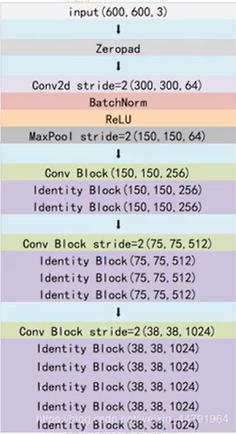
**Faster RCNN**

Faster RCNN使用RPN网络（Region Proposal Network)取代了Selective Search，不仅速度更快，结果也更加精确。Faster RCNN对输入图片的大小没有要求，但通常把输入不失真地resize到短边为600。



**特征提取：**

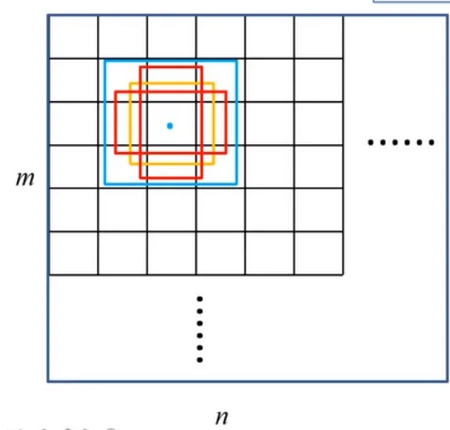
Backbone使用的是resnet50模型。Resnet50有两种基本块：Conv Block和Identity Block。两者的区别在于残差边是否有卷积。Conv block可以通过改变残差边卷积的步长和通道数以改变输出特征层维度，而identity block输入输出维度一样，用于串联加深网络。 Resnet50整体框架：



**建议框网络构建：**

得到feature map后，将feature map送入一个3\*3的卷积，输入通道数为1024，输出通道数为512，步长为1，有填充，输出特征图大小为38\*38\*512，然后送入通道数分别为36和18的1\*1卷积中，步长为1，无填充。其中36=9\*4表示9个先验框的长宽变化情况（中心点位置偏移、长宽缩放）（结合先验框获得候选框），18=9\*2表示这9个候选框中是否有物体。进行通道为18的卷积后还需要对结果进行softmax，以得到框中存在物体的概率（置信度）。这一步操作的含义是将原图像划分成了38\*38个区域，然后以每个区域中心为中点构建9个候选框。

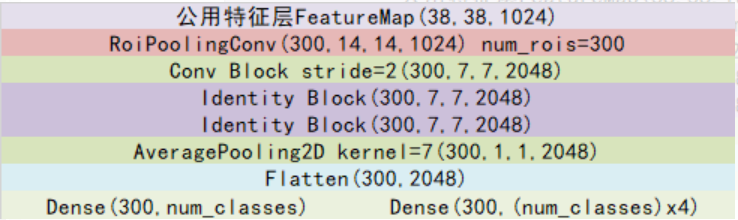
先验框（图中的im\_info）是预先设定好的9个框。在faster rcnn中，feature map大小为38\*38，在卷积过程中经过了四次长宽减半，因此可以认为38\*38中的每个点都是16\*16网格的中心点。（周围可能有一些像素也被卷积）先验框是以38\*38网格中每个点为中心画的9个框，共有38\*38\*9个。



**ROI pooling：**

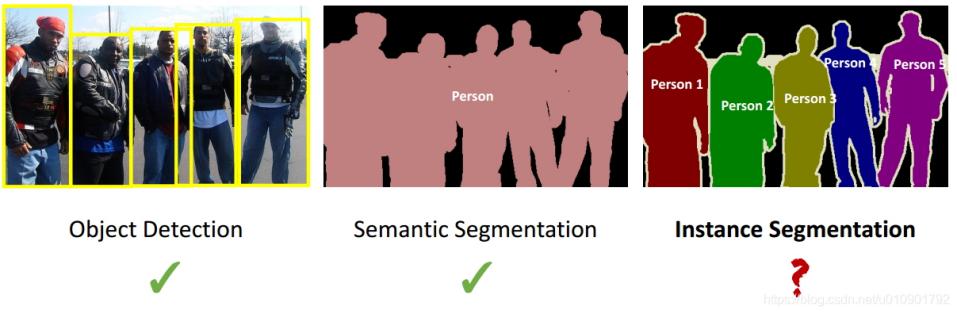
通过建议框网络我们得到了300个建议框（RP or ROI，用得分筛出300个）。在Faster RCNN的ROI pooling中，每个ROI被分割成14\*14的区域，然后在每个区域中做最大池化，得到大小为（300，14，14，1024）的特征图。这一步骤在pytorch中用自带的RoIPool函数可以完成。

上述操作完成后，Faster RCNN对每一个建议框进行resnet原有的第五次压缩（一次conv block和两次identity block），之后进行全局平均池化，通过两个全连接层分别进行分类和回归。特征图的大小变化如下，其中num\_class表示物体的种类数（21），num\_class \* 4对应每个种类给出的边框的四个参数。

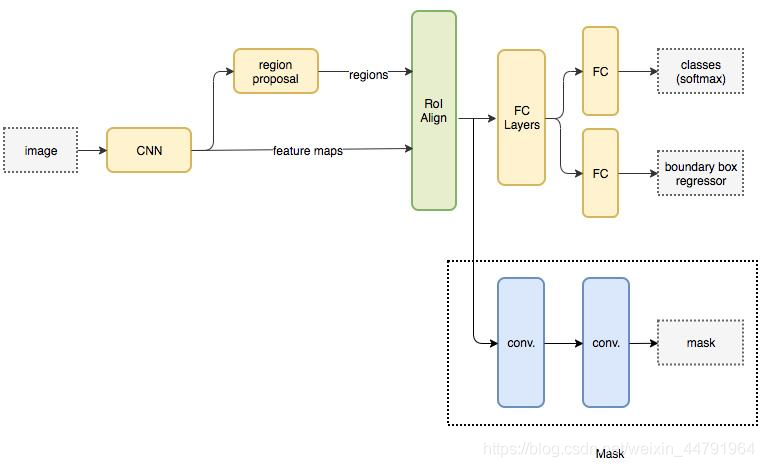


**Mask RCNN**

在下图中，我们有三种方法表示图中人的位置。第一种是目标检测算法（如Faster RCNN），把图中的目标框出来；第二种是语义分割算法（如FCNN），用像素点将不同类物体标出来。第一种算法能区分个体，但在标注物体时会把背景也框起来；第二种算法能准确分割不同类物体，但同类物体无法区分个体。而以Mask RCNN为例的实例分割算法能集结两者优点，能同时准确区分类别和个体。



Mask RCNN网络结构

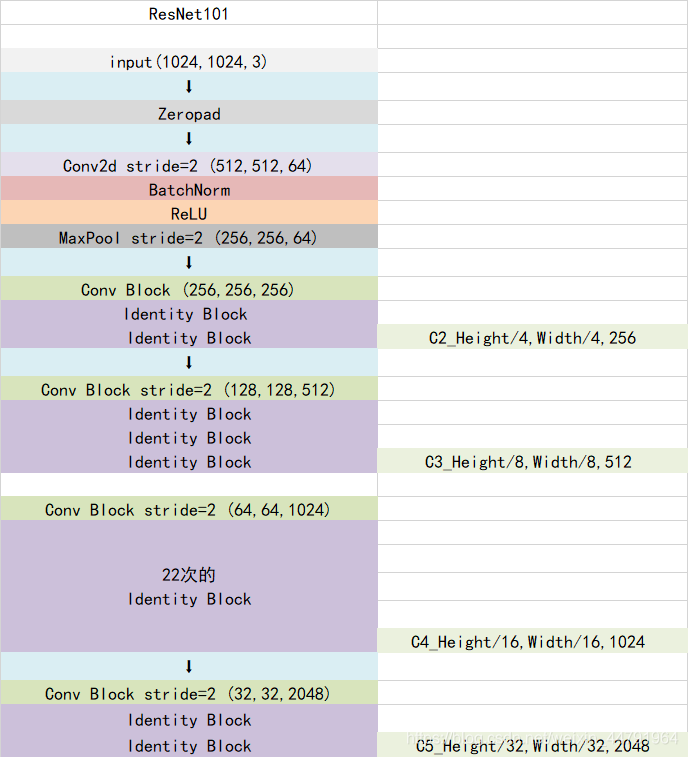


输入要求：用灰条补充图片使图片为正方形，并缩放图片使边长整除2^6。

**CNN模块：**

CNN由两部分组成：主干特征提取网络Resnet101和特征金字塔FPN。

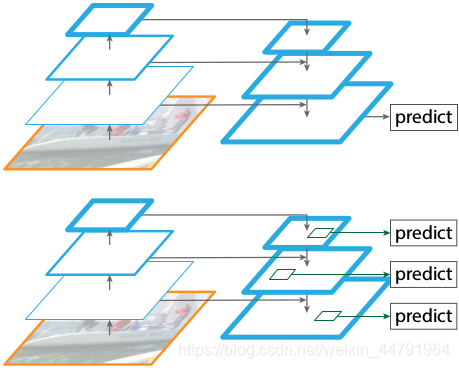
**Resnet101**（Identity Block和Conv Block定义与Resnet50相同）：



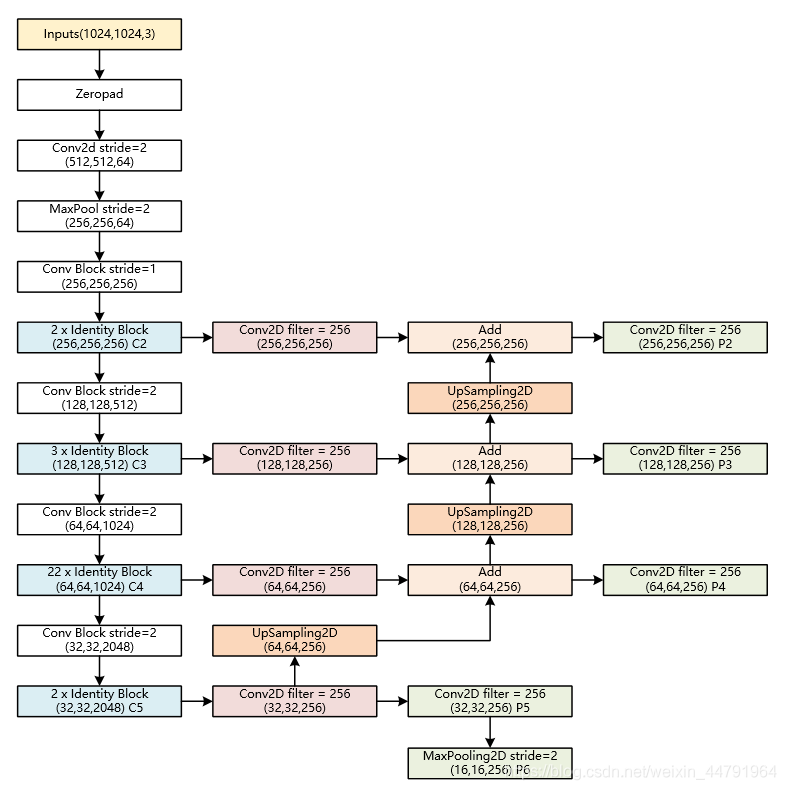
图片进行两次、三次、四次、五次压缩后的特征图用于构建特征金字塔。

**特征金字塔**的构建是为了实现特征多尺度的融合。在检测小目标时，由于大目标所占像素点远大于小目标像素，所以大的目标被经过卷积核的次数远比小的目标多，因此大目标的特点更容易得到保留，小目标的特征点容易被跳过，经过很多层的卷积之后，小目标的特点会越来越少，越小越小。特征金字塔重新进行上采样，使得特征层的长宽重新变大，用大size的feature map去检测小目标。这会使小目标的特征信息不明确，因此可以将下采样中与上采样中长宽相同的特征层进行堆叠，这样可以保证小目标的特征与信息。

注：这里的堆叠有两种解释，即add和concat。Add相当于concat之后两层特征图用相同的卷积核卷积，当两路输入具有“对应通道的特征图语义类似”的性质的时候，可以用add来替代concat，这样更节省参数和计算量。在FPN中可以使用add。

如图，我们不检测猫而检测落叶，经过多层卷积后目标的特征会越来越少。

将特征提取与FPN结合得到下图：



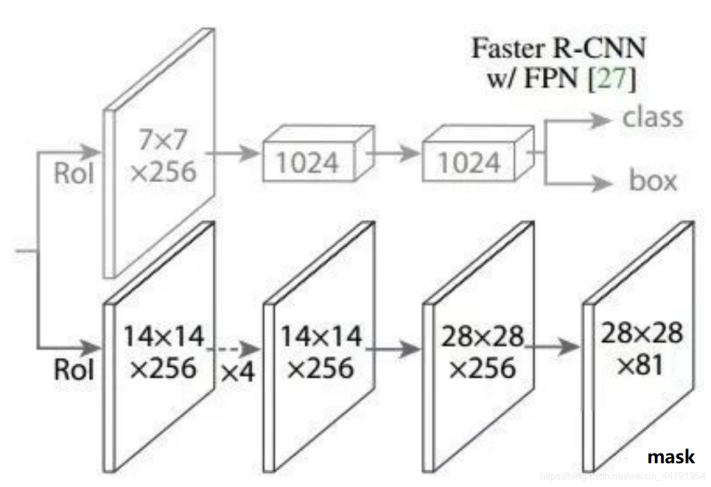
提取到的P2、P3、P4、P5、P6可以作为RPN网络的有效特征层，利用RPN建议框网络对有效特征层进行下一步的操作，对先验框进行解码获得**建议框**。P2、P3、P4、P5可以作为Classifier和Mask网络的有效特征层，利用Classifier预测框网络对有效特征层进行下一步的操作，对建议框解码获得最终**预测框**；利用Mask语义分割网络对有效特征层进行下一步的操作，获得每一个预测框内部的**语义分割结果**。

Region Proposal：使用有效特征层（P2、P3、P4、P5、P6）获得**建议框**

与Faster RCNN类似，建议框网络首先通过一个通道数为512的3\*3的卷积层，然后再分别进行一次通道数为anchors\_per\_location\*4的卷积和一次anchors\_per\_location\*2的大小为1\*1的卷积，分别用于预测先验框的变化情况和预测框内是否包含了物体。anchors\_per\_location默认为3，表示一个网格中默认存在三个建议框。对五个有效特征层分别进行上述卷积，结合先验框并对偏置解码得到建议框。（输入为（1024,1024,3）时，一共有256\*256\*3+128\*128\*3+64\*64\*3+32\*32\*3+16\*16\*3=261888个建议框）

RoI Align：对建议框加以利用

公用特征层里的每一个点相当于原图片上某个区域内部所有特征的浓缩。建议框会对其对应的公用特征层进行截取，然后将截取的结果进行resize，在classifier模型里，截取后的内容会resize到7x7x256的大小。在mask模型里，截取后的内容会resize到14x14x256的大小。



在Classifier模型中，使用一次7\*7\*1024的卷积和一次1\*1\*1024的卷积，用于模拟两次通道数为1024的全连接层，再分别全连接到num\_classes和num\_classes\*4上，代表这个建议框内的物体，以及这个建议框的调整参数。

在mask模型里，其首先会对resize后的局部特征层进行四次3x3的256通道的卷积，再进行一次反卷积，再进行一次通道数为num\_classes的卷积，最终结果代表每一个像素点分的类。最终的shape为28\*28\*num\_classes，代表每个像素点的类别。

预测框的解码：

对那些非背景且得分较高的建议框（>config.DETECTION\_MIN\_CONFIDENCE）解码得到最终预测位置，并利用预测框位置进行非极大抑制防止重复检测。

得到最终的预测框后，用这个预测框对mask模型中使用的公用特征层进行截取，再送入mask模型中进行像素点的分类，得到语义分割结果。