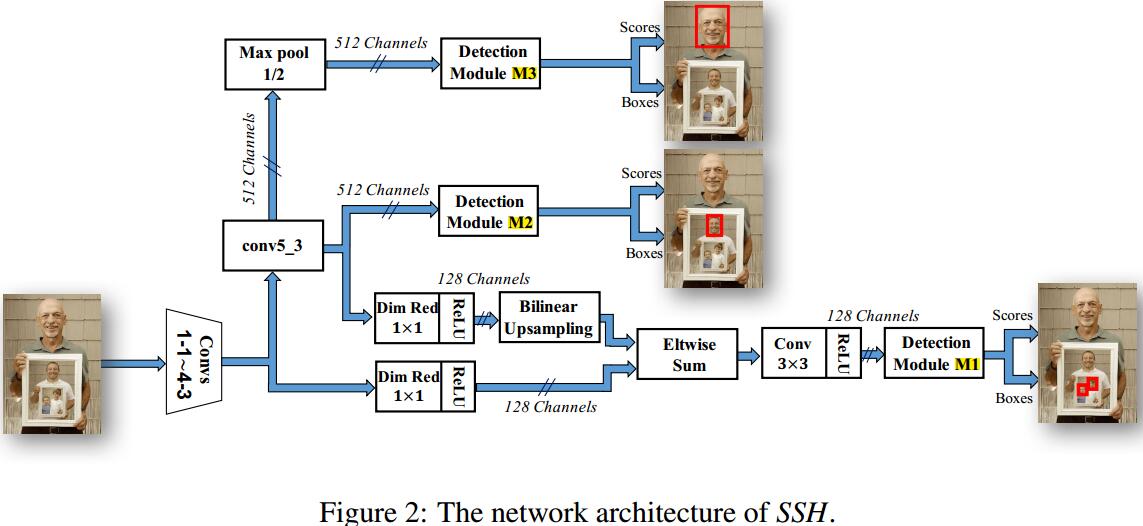
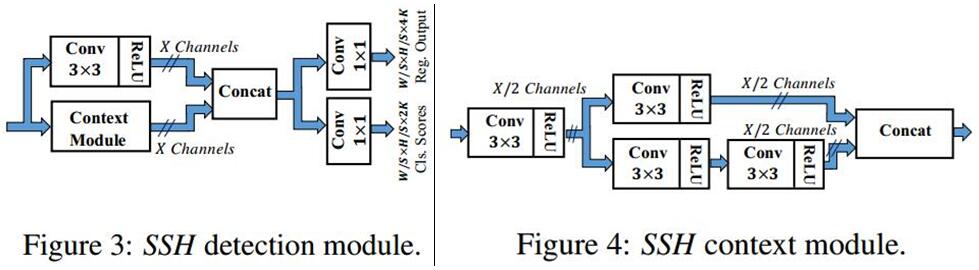
【一】《**SSH: Single Stage Headless Face Detector**》

SSH最大的特色就是尺度不相关性（scale-invariant），像MTCNN这样的网络在预测的时候，是通过一个for循环输入不同大小scale的图片进行预测再NMS，而SSH只需要一遍forward就可以搞定。实现的途径就是对VGG不同的卷积输出层做了3个分支（M1,M2,M3），每个分支都使用类似的流程预测检测和分类结果。这样的实现从理论来说性能也会比MTCNN有所提高，当然经过优化，MTCNN速度也会很快。



其中，M2和M3的流程基本一样，唯一的区别就是滑动步长的区别，其中，M1,M2,M3中的卷积核滑动步长分别为8,16,32。  
M1和M2,M3区别有点大，首先，M1的通道数为128，M2,M3的通道数为512，这里，作者使用了神奇的1\*1卷积核进行了降维操作。其次，将第四个卷积层和第五个卷积层的特征进行了融合（Elementwise sum），这里由于第五个卷积层的大小和第四个卷积层的大小不一样，作者队第五个卷积层的输出做了双线性插值的上采样。



其中，M模块的具体细节如上图所示，包含了分类和回归2个内容，分为2条卷积途径，第一个是普通的3\*3卷积，第二个，也就是context Module，作者为了获得更多的上下文信息，更大的感受野，对该模块使用了5\*5和7\*7的卷积分别进行操作，然后进行特征的concat，然后由于大的卷积核效率问题，根据INCEPTION的思想，使用2个3\*3的卷积核替代一个5\*5的卷积核，使用3个3\*3的卷积核替换1个7\*7的卷积核，最终形成了上图的context Module。

还有一个需要注意的就是最后作者没有使用全连接层，而是使用了1\*1的卷积达到同样的效果，亮点就是输入图片可以输入像fasterRCNN那样不同aspect ratio的图片了。从而可以使得输入图片再保证aspect ratio的同时，输入尽可能大的图片。

在上面的整个forward都走完后，再对输出的结果大概3000个（每个module输出1000个）做NMS（阈值0.3）操作，形成最终的结果。

另外一个需要注意的就是，作者anchor的设计，作者只考虑了scale的因素，aspect ratio的比例为固定的1/2，其中M1的scale为（1,2），M2的scale为（4,8），M3的scale为（16,32），其中base anchor为16 pix

其他方面，OHEM，smooth L1等都会对结果有提高。

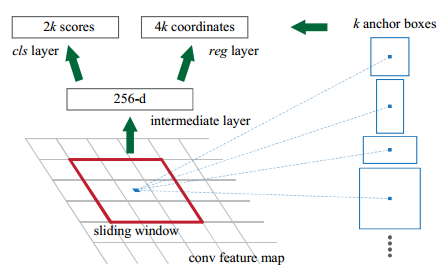
和MTCNN相比，略显遗憾的地方就是没有实现人脸检测和关键点检测的一体化。

# 【二】《**Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**》

由RCNN到FAST RCNN一个很重要的进步是实现了多任务的训练，但是仍然使用Selective Search算法来获得ROI，而FASTER RCNN就是把获得ROI的步骤使用一个深度网络RPN来实现。一个FASTER RCNN可以看作是一个RPN + FAST RCNN的组合，两者通过共享CONV LAYERS组合在一起。

**RPN网络**

一张图片先经过CONV LAYERS得到feature map，图片的大小是任意的。然后，使用一个小的滑动网络，它与feature map的一个n\*n的小窗口全连接。在论文中，n会取值3（但不知道是否指的是n\*n个像素的窗口），虽然小窗口不大，但实际上由于feature map经过pooling和convolution，映射回输入图像会有一个很大的感受野。通过这个滑动的小网络提取ROI并映射到一个低纬度的特征向量，用于回归和分类（两个1\*1的convolution layer）。



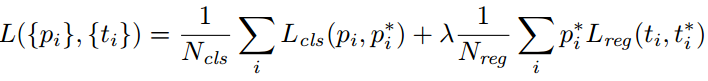
对于滑动的n\*n的窗口，在每个位置都会预测k个regional proposal，称为anchor。每个anchor都以窗口为中心，它们有不同的尺度以及宽高比，论文中去k = 9，即3种尺度与3种宽高比。映射得到的低维特征向量，输入到两个box回归层与softmax（二分类，即logistic，用于预测窗口中有或者没有物体的概率）。例如一个VGGnet，它映射到的低维特征为512维，由于k = 9，而且回归层和分类层都是1\*1的，因此输出层的参数有512 \* (4 + 2) \* 9个。如果feature map的大小为W \* H，则总共会有W \* H \* k个anchors。

**（n\*n应该是一个卷积层，而VGG最后会有512个通道，每个anchor都会映射到低维的特征，而这低维的特征应该与通道数相关。。。。猜测， 额，这里的映射到低维的特征与anchor的关系确实有点模糊……）**

论文中实现的不同尺度以及宽高比的anchor，而不需要多尺度的image或者filter。

**RPN的损失函数**

学习一个RPN的网络也是一个多任务的学习过程，是一个通过参数迁移后对网络进行fine tune的有监督过程，需要标示正类与负类。正例样本有两种（1）与一个ground truth的IOU最高的anchor（2）与任意一个ground truth的IOU高于0.7的anchor。于是，类似于FAST RCNN的定义，LOSS函数为：

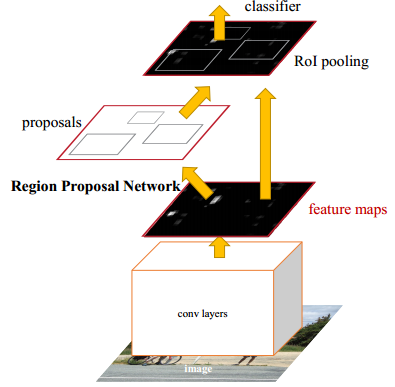


其中IMG_258为1如果anchor的正样例，否则为0。IMG_259表示anchor里是一个object的概率。而IMG_260即是FAST RCNN中smooth函数的定义。 IMG_261是一个平衡因子，具体设置可以参考论文，它与Ncls和Nreg有关。

**训练RPN**

RPN的训练使用的是BP算法，权值更新使用随机梯度下降法实现。Mini\_batch的抽样从单一的图片中选择anchor，尽量是的正例和负例的比是1:1。RPN权值的初始使用高斯分布初始化。

**RPN与FAST RCNN的共享**



由上图可以看出，RPN与FAST RCNN共享conv layer层以及feature map。论文中训练这个网络提供了几种方法：

（1）交替训练。首先训练RPN，得到proposal之后去训练FAST RCNN。得到的NETWORK又去训练RPN。迭代交替

（2）把两者近似的融合训练。如上图，中间层输出proposal去训练FAST RCNN。后向传播时，对于共享层像平常一样更新，把来自RPN的loss和FAST RCNN的loss结合。而对于proposal 的box坐标预测的梯度，直接忽略，因而得到一个近似的解。但实际上，proposal层的网络权值也是会更新的。

论文中采取一种新的不同于上述的方法：

（1）用imagenet模型初始化，独立训练一个RPN网络。

（2）仍然使用imagenet初始化，利用第一步得到的proposal作为输入训练一个FAST RCNN网络，此时参数完全不共享。

（3）用（2）得到的参数初始化RPN的网络，把RPN与FAST RCNN共享的卷积层的学习率设为0，仅更新RPN特有的网络层，重新训练，此时网络共享共有的卷积层。

（4）固定共享的层，加入FAST RCNN，fine tune它特有的网络。

**细节：**

对于每个anchor的三种scale为128\*128，256\*256，512\*512,，三种宽高比1:1,1:2,2:1。

对于一些RPN的proposal，采用非极大值抑制的办法选择过滤一部分，减少冗余。