论文笔记：泛读、精读

每篇论文笔记包含以下几个部分

1. 文献名字和作者
2. 阅读时间
3. 文献贡献点（架构、创新点）
4. 实验结果
5. 感悟

背景知识：

Cnn卷积神经网络：

卷积神经网络较浅的卷积层感知域较小，学习到一些局部区域的特征；较深的卷积层具有较大的感知域，能够学习到更加抽象的特征。这些特征对物体的大小、位置和方向等敏感性更低，从而有助于识别性能的提高。这些抽象特征对于分类很有帮助，但是因为丢失了一些物体的细节，不能很好地给出物体具体的轮廓，不能对物体做精确的分割。浅层的特征图分辨率较高，包含了物体的一些细节，因此，很多论文文献中都是结合浅层和深层的特征图来获取最终的特征。

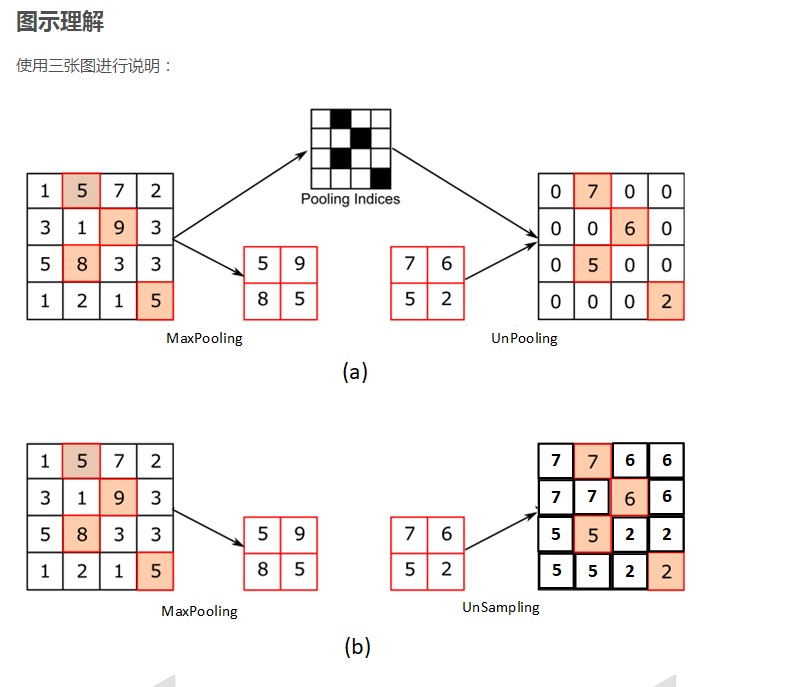
传统的基于CNN的分割方法：为了对一个像素进行分类，传统的方法是将像素与周围的像素组成像素块作为CNN的输入用于训练和预测。这种方法存储开销大、计算效率低下，而且像素块的大小相对于整幅图像而言小很多，只能提取一些局部的特征，从而导致分类的性能受到限制

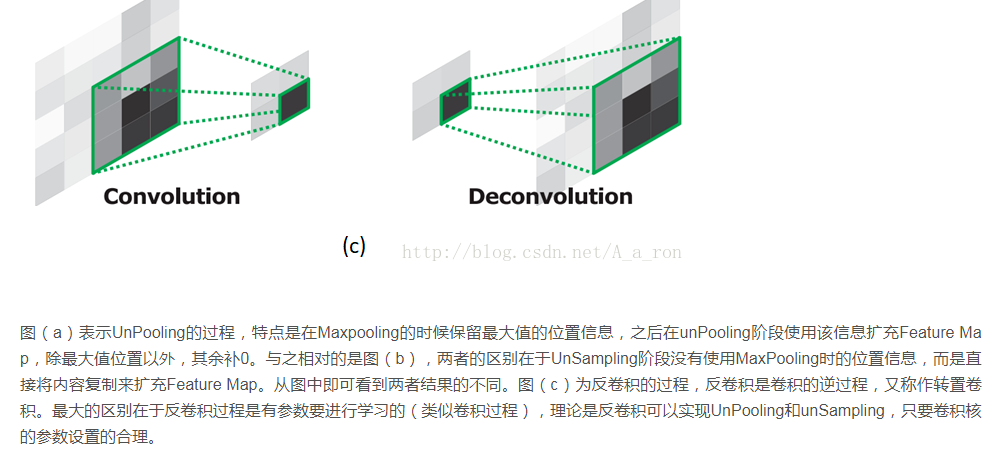
Fcn:

FCN对图像进行像素级的分类，从而解决了语义级别的图像分割问题。与经典的CNN在卷积层使用全连接层得到固定长度的特征向量进行分类不同，FCN可以接受任意尺寸的输入图像，采用反卷积层对最后一个卷基层的特征图（feature map）进行上采样，使它恢复到输入图像相同的尺寸，从而可以对每一个像素都产生一个预测，同时保留了原始输入图像中的空间信息，最后经过在反卷积（转置卷积）的特征图进行像素的分类。

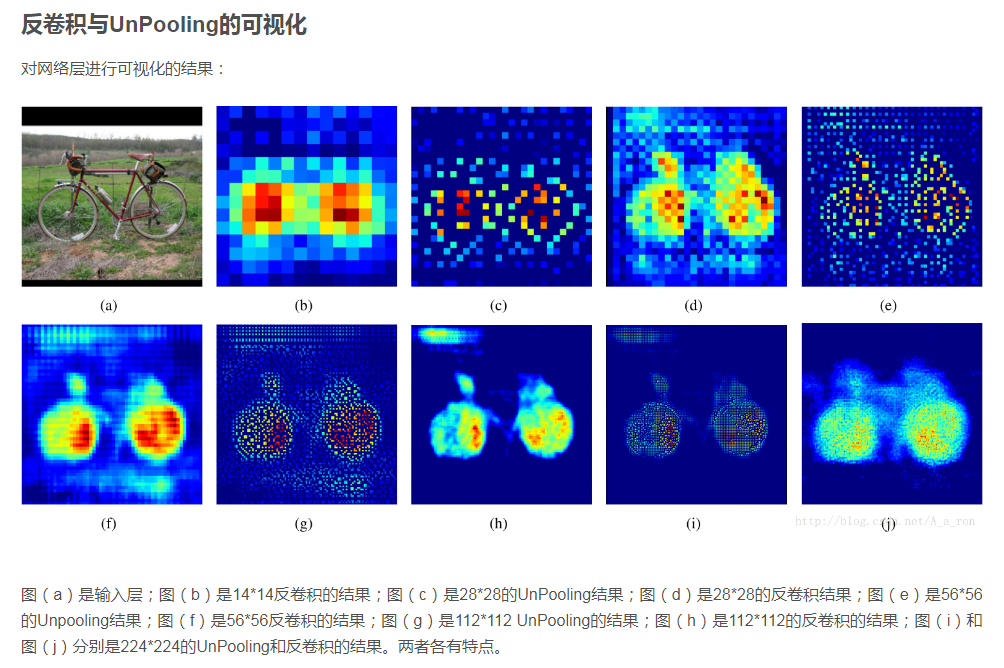
GUANYU :

反卷积(Deconvolution)、上采样(UNSampling)与上池化(UnPooling)的理解：

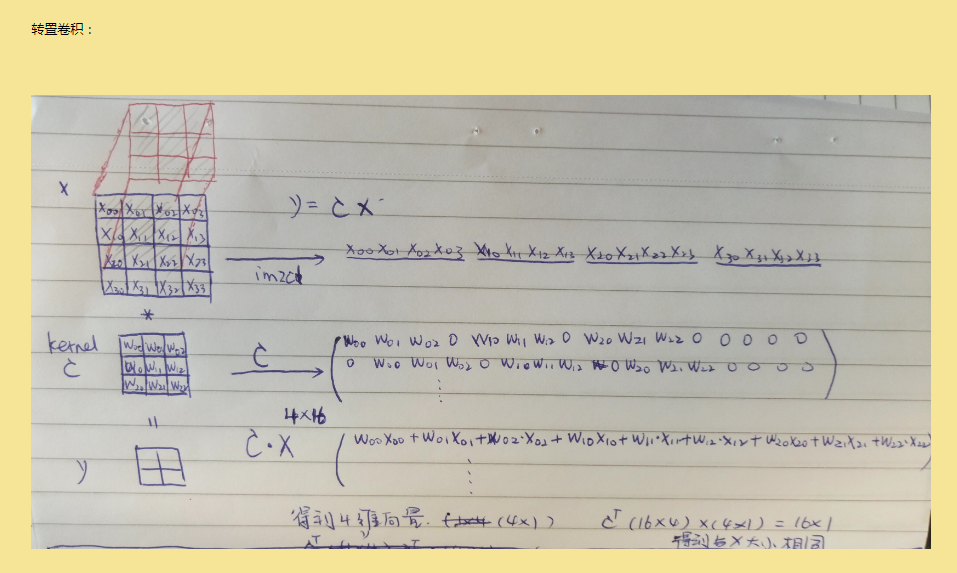




反卷积类似于一个卷积的过程，只不过在卷积之前将输入特征插值到一个更大的特征图然后进行卷积



从图中可以看出，越深的网络反卷积出的结果越模糊，提取的特征越抽象，缺少细节，不利于分割，但是适合分类和回归。



1. 文献名字和作者

FCN是Jonathan Long和Evan Shelhamer于2015年提出的网络结构

论文：《Fully Convolutional Networks for Semantic Segmentation》

代码：<https://github.com/shelhamer/fcn.berkeleyvision.org>

1. 阅读时间

10.8

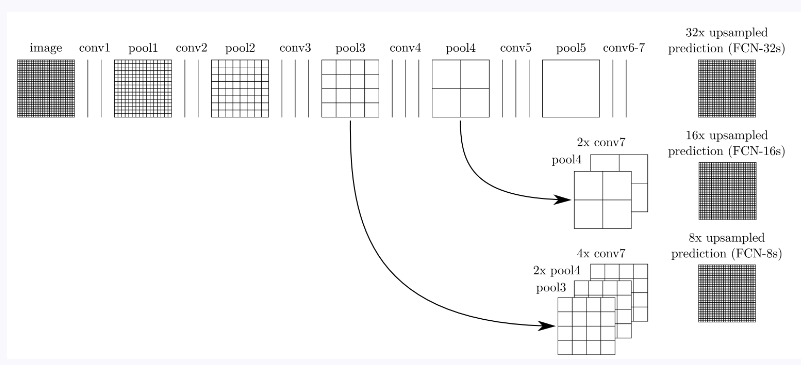
1. 文献贡献点

* 使用了全卷积结构
* 采用反卷积解码成和原图一样的大小

在一般的CNN结构中均是使用pooling层来缩小输出图片的size，如果在VGG16中，五次pooling之后特征图的大小比输入图缩小了32倍。而在fcn网络中，要求网络输出与原图size相同的分割图，因此我们需要对最后一层进行上采样。在caffe中也被称为反卷积（Deconvolution）。

虽然转置卷基层和卷积层一样，也是可以训练参数的，但是在实验中发现，让转置卷基层可学习，并没有带来性能的提高，所以在实验中转置卷基层的lr全部设为0

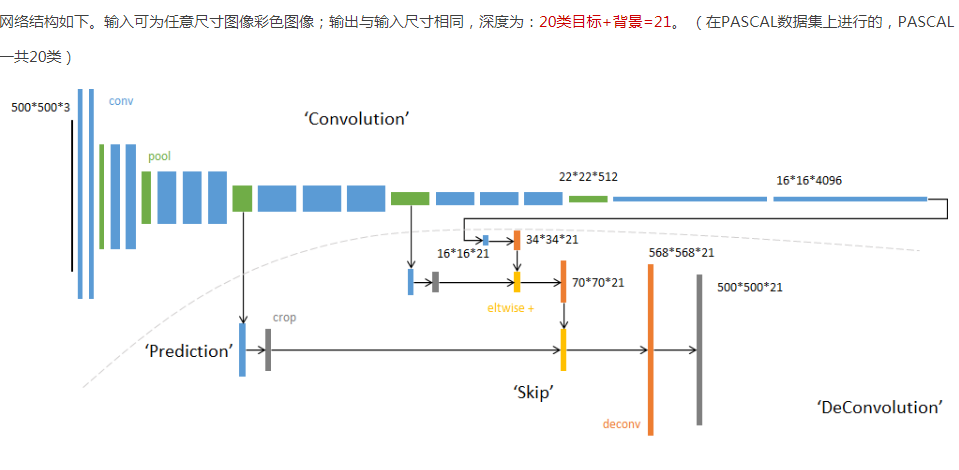
* 采用了skip结构，充分利用了浅层和深层的信息使得分类更精细



如上图所示：对原图进行卷积conv1、pool1后图像缩小为1/2；对图像进行第二次卷积conv2、pool2后图像缩小为1/4；对图像进行第三次卷积conv3、pool3后图像缩小为1/8，此时保留pool3的featuremap；对图像进行第四次卷积conv4、pool4后图像缩小为1/16，此时保留pool4的featuremap；对图像进行第五次卷积conv5、pool5后图像缩小为1/32，然后把原来CNN操作过程中的全连接编程卷积操作的conv6、conv7，图像的featuremap的大小依然为原图的1/32,此时图像不再叫featuremap而是叫heatmap。

其实直接使用前两种结构就已经可以得到结果了，这个上采样是通过反卷积（deconvolution）实现的，对第五层的输出（32倍放大）反卷积到原图大小。但是得到的结果还上不不够精确，一些细节无法恢复。于是将第四层的输出和第三层的输出也依次反卷积，分别需要16倍和8倍上采样，结果过也更精细一些了。这种做法的好处是兼顾了local和global信息。

网络结构：



注意点：1、反卷积得到的特征图和浅层图融合的采用的是eltwise层（caffe层：对应特征图像素点相加，与Unet不一样， unet是concat，拼接）；

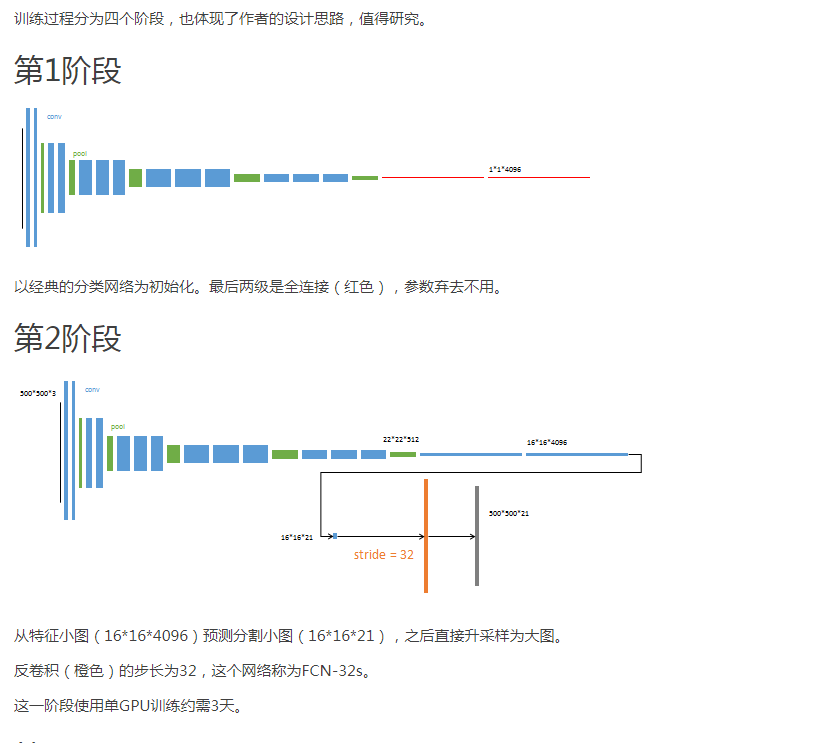
2、输入图像会进行padding100大小，网络分析：<http://melonteam.com/posts/quan_juan_ji_shen_jing_wang_luo_fcn_xue_xi_bi_ji/>

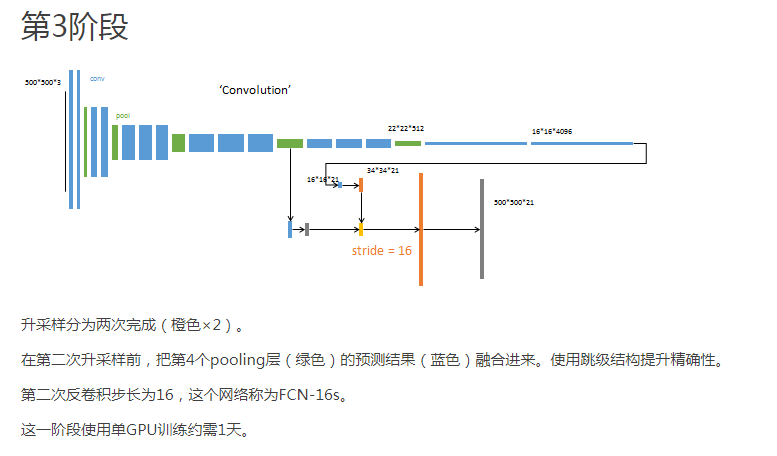
损失函数：

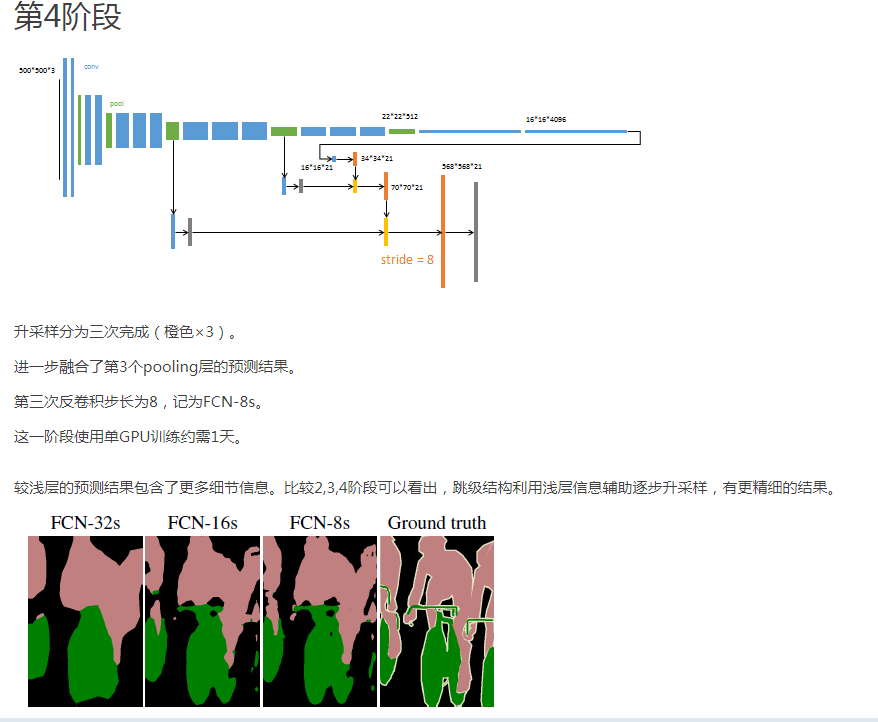
pixel-wise classification loss

最后的输出为500\*500\*21，然后接损失函数。损失函数是在最后一层的spatial map上的每一个pixel的loss和，在每一个pixel使用softmax los。

训练过程：







1. 实验结果

取得了当年最新的结果

1. 总结

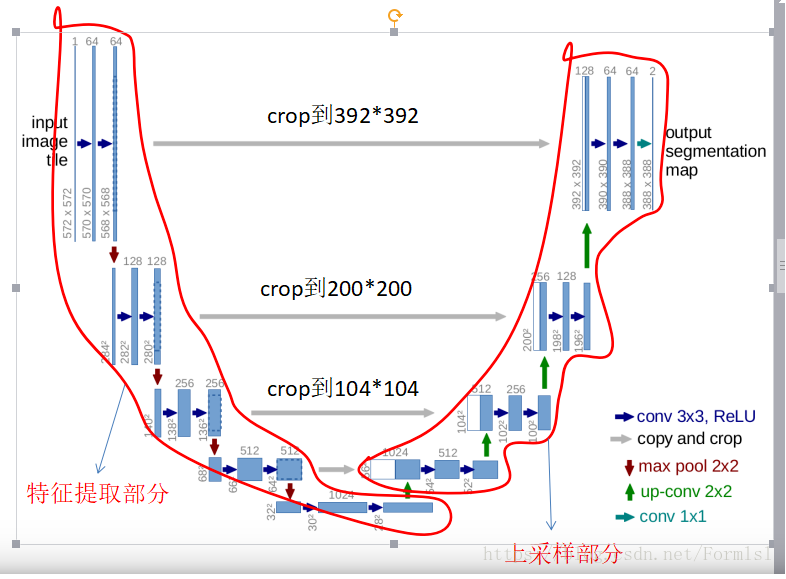
与传统用CNN进行图像分割的方法相比，FCN有两大明显的优点：一是可以接受任意大小的输入图像，而不用要求所有的训练图像和测试图像具有同样的尺寸。二是更加高效，因为避免了由于使用像素块而带来的重复存储和计算卷积的问题。

同时FCN的缺点也比较明显：一是得到的结果还是不够精细。进行8倍上采样虽然比32倍的效果好了很多，但是上采样的结果还是比较模糊和平滑，对图像中的细节不敏感。二是对各个像素进行分类，没有充分考虑像素与像素之间的关系，忽略了在通常的基于像素分类的分割方法中使用的空间规整（spatial regularization）步骤，缺乏空间一致性

医学图像分割：《U-Net: Convolutional Networks for Biomedical Image Segmentation》

1、文献作者：

2、网络结构



这里的up-conv 是先上采样，然后在卷积 和FCN里的反卷积有些区别。

1. 特征提取部分，每经过一个池化层就一个尺度，包括原图尺度一共有5个尺度。
2. 上采样部分，每上采样一次，就和特征提取部分对应的通道数相同尺度融合，但是融合之前要将其crop。这里的融合也是拼接。   
   **个人认为改进FCN之处有：**
   1. 多尺度
   2. 适合超大图像分割，适合医学图像分割

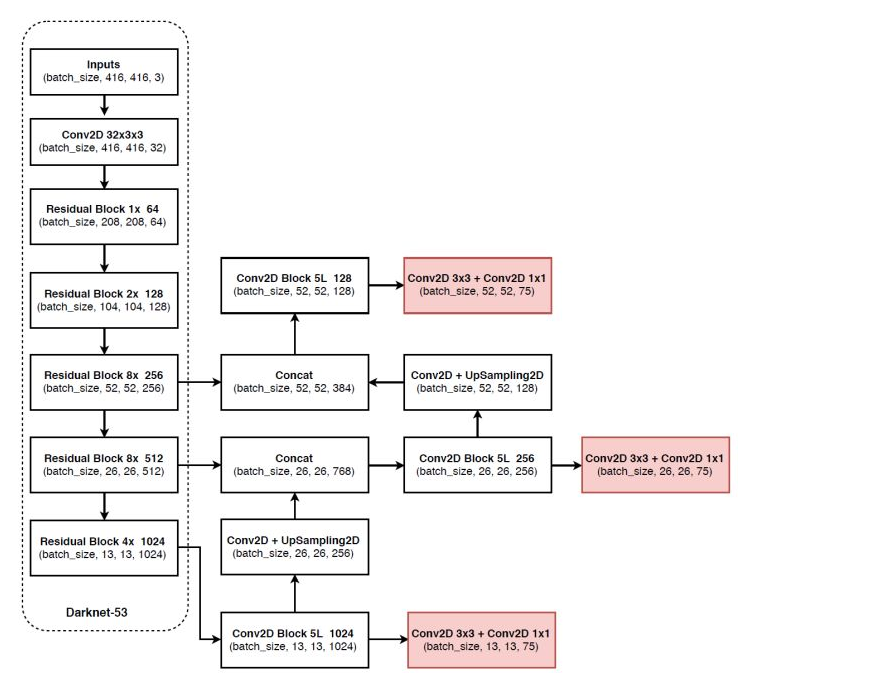
Yolo:

1. 文献作者

YOLOv3: An Incremental Improvement

1. 网络结构

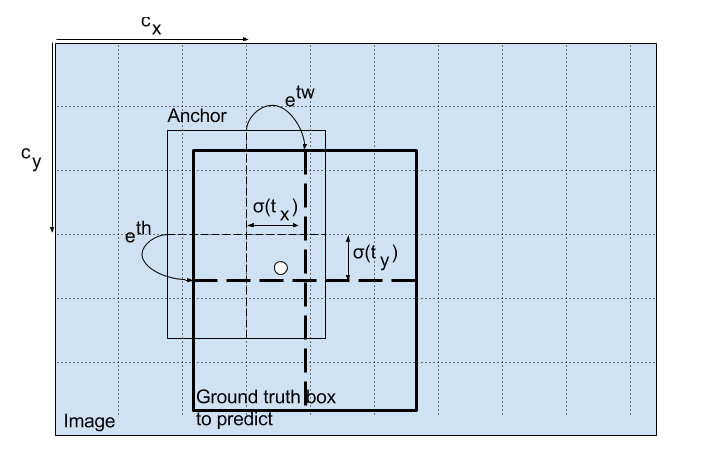
网络结构特点：采用全卷积网络（FCN），没有pool层，包含上采样层，concat连接方式，卷积核大小为3\*3或者1\*1，只有卷积步数（stride）为2时才减小卷积特征图，这样的原因是防止下采样丢失低级特征（This helps in preventing loss of low-level features often attributed to pooling）。416 x 416 经过stride = 32后，得到最终13\*13的特征图。使用激活函数使用 leak RELU。

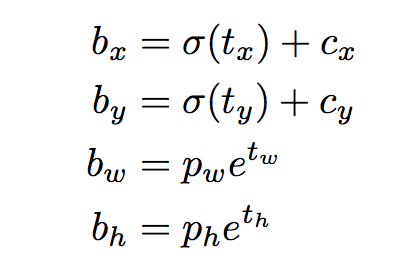


1. 核心思想

(B x (5 + C))的理解：B 代表每个单元可以预测的边界框数量， 每个边界框都有 5+C 个属性，分别描述每个边界框的中心坐标、维度、objectness 分数和 C 类置信度。YOLO v3 在每个单元中预测 3 个边界框。

网络输出的转换过程：





其中，tx,ty,tw,th为预测输出结果，cx,cy为左上角网格子的坐标，pw,ph为anchor的宽和高。注意网络最后的输出都是小于1的，需要经过转换。

中心坐标：假设第6行第5列的网格，预测的结果为0.6,0.6，那么他的中心结果为6+sigmoid(0.6) 和5 +sigmoid(0.5),这个中心结果是相对最终特征图13\*13的结果，为什么要经过sigmoid，如果预测结果为（1.6,1.5），和左上角相加得到的结果就是（7.6,6.5），但是他实际上是最大为（7， 6），和作者的核心思想对应的网格只预测中心在对应网格的目标 矛盾了，所以要经过sigmoid。

边界框发维度：假设预测的边界宽和高结果是（0.3， 0.7），则13\*13对应的特征图转换后的结果是（13\*0.3， 13\*0.7）。

结果过滤：

作者使用了多尺度进行预测，416\*416的图片，假设特征图步幅分别是 32、16、8，最终会在尺度 13 x 13、26 x 26 和 52 x 52 上执行检测，YOLO v3 在每个单元中预测 3 个边界框，那么最终的预测框达到((52 x 52) + (26 x 26) + 13 x 13)) x 3 = 10647 个边界框。处理办法：分别使用object\_score(表示是否包含目标)和NMS（非极大抑制）过滤。

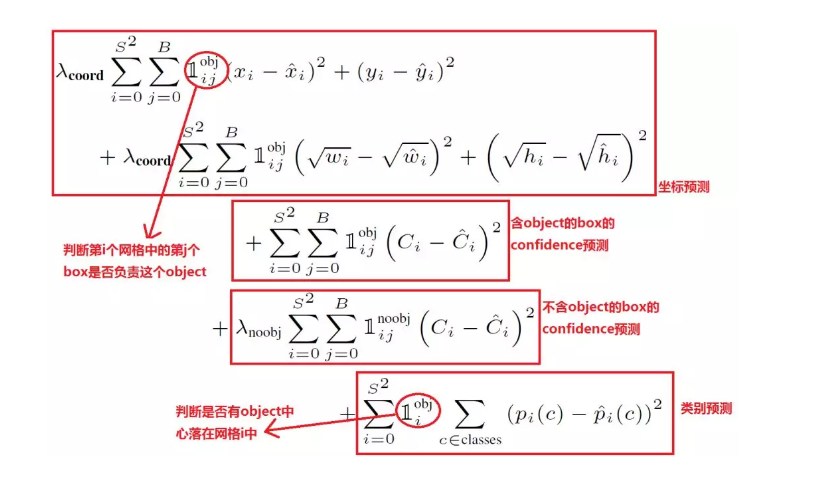
Object\_core: Object 分数表示目标在边界框内的概率。红色网格和相邻网格的 Object 分数应该接近 1，而角落处的网格的 Object 分数可能接近 0, objectness 分数的计算也使用 sigmoid 函数，因此它可以被理解为概率。

Class\_confidence(类别置信度):

类别置信度表示检测到的对象属于某个类别的概率（如狗、猫、香蕉、汽车等）。在 v3 之前，YOLO 需要对类别分数执行 softmax 函数操作。

但是，YOLO v3 舍弃了这种设计，作者选择使用 sigmoid 函数。因为对类别分数执行 softmax 操作的前提是类别是互斥的。简言之，如果对象属于一个类别，那么必须确保其不属于另一个类别。这在我们设置检测器的 COCO 数据集上是正确的。但是，当出现类别「女性」（Women）和「人」（Person）时，该假设不可行。这就是作者选择不使用 Softmax 激活函数的原因。

损失函数：



这个损失函数中：

只有当某个网格中有 object 的时候才对 classification error 进行惩罚。只有当某个 box predictor 对某个 ground truth box 负责的时候，才会对 box 的 coordinate error 进行惩罚，而对哪个 ground truth box 负责就看其预测值和 ground truth box 的 IoU 是不是在那个 cell 的所有 box 中最大。

其他细节：

每个 grid 有 30 维，这 30 维中，8 维是回归 box 的坐标，2 维是 box的 confidence，还有 20 维是类别。

其中坐标的 x, y 用对应网格的 offset 归一化到 0-1 之间，w, h 用图像的 width 和 height 归一化到 0-1 之间。

在实现中，最主要的就是怎么设计损失函数，让这个三个方面得到很好的平衡。作者简单粗暴的全部采用了 sum-squared error loss 来做这件事。

这种做法存在以下几个问题：   
第一，8维的 localization error 和20维的 classification error 同等重要显然是不合理的；   
第二，如果一个网格中没有 object（一幅图中这种网格很多），那么就会将这些网格中的 box 的 confidence push 到 0，相比于较少的有 object 的网格，这种做法是 overpowering 的，这会导致网络不稳定甚至发散。

解决办法：

第一：更重视8维的坐标预测，给这些损失前面赋予更大的 loss weight, 记为在 pascal VOC 训练中取 5。

第二：对没有 object 的 box 的 confidence loss，赋予小的 loss weight，记为在 pascal VOC 训练中取 0.5。

第三：有 object 的 box 的 confidence loss 和类别的 loss 的 loss weight 正常取 1。

第四：对不同大小的 box 预测中，相比于大 box 预测偏一点，小 box 预测偏一点肯定更不能被忍受的。而 sum-square error loss 中对同样的偏移 loss 是一样。

为了缓和这个问题，作者用了一个比较取巧的办法，就是将 box 的 width 和 height 取平方根代替原本的 height 和 width。这个参考下面的图很容易理解，小box 的横轴值较小，发生偏移时，反应到y轴上相比大 box 要大。（也是个近似逼近方式）

1. 实验结果
2. 总结