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。

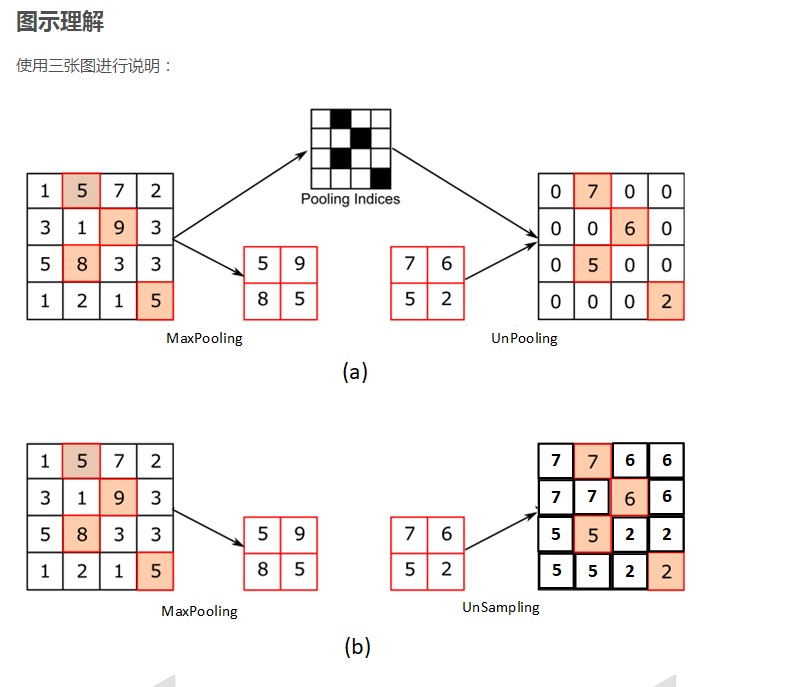
传统的基于CNN的分割方法：为了对一个像素进行分类，传统的方法是将像素与周围的像素组成像素块作为CNN的输入用于训练和预测。这种方法存储开销大、计算效率低下，而且像素块的大小相对于整幅图像而言小很多，只能提取一些局部的特征，从而导致分类的性能受到限制

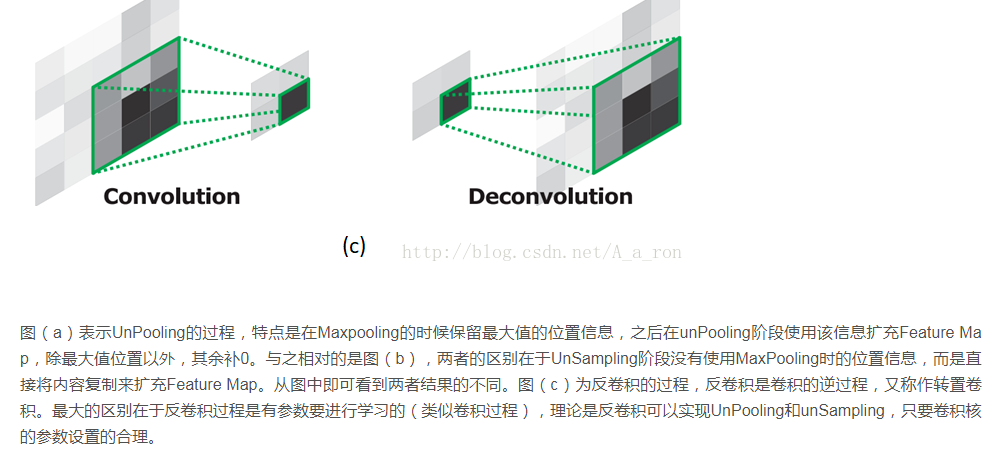
Fcn:

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。

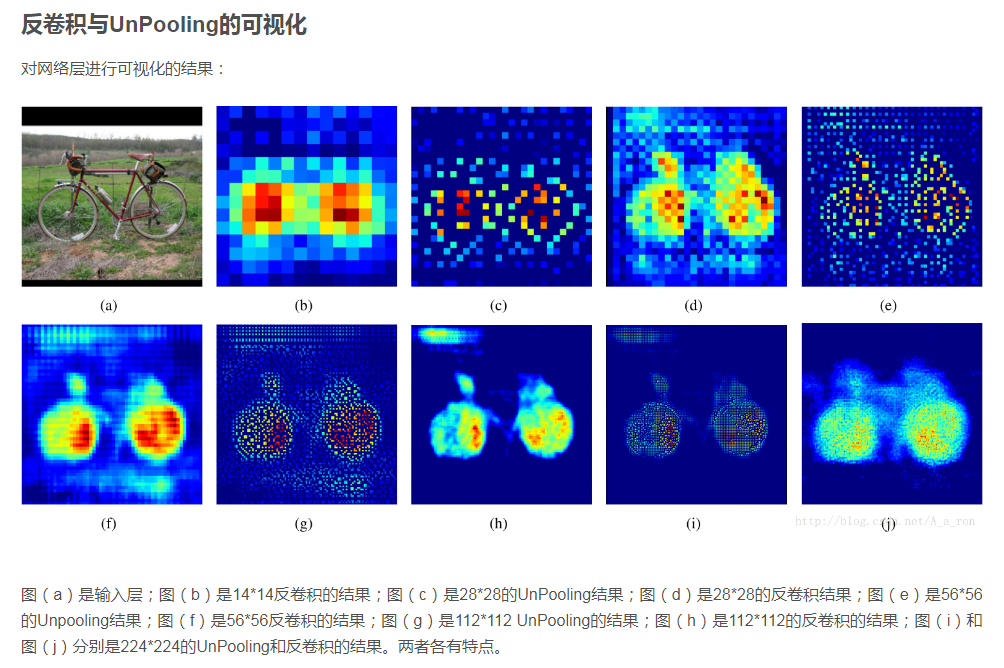
GUANYU :

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

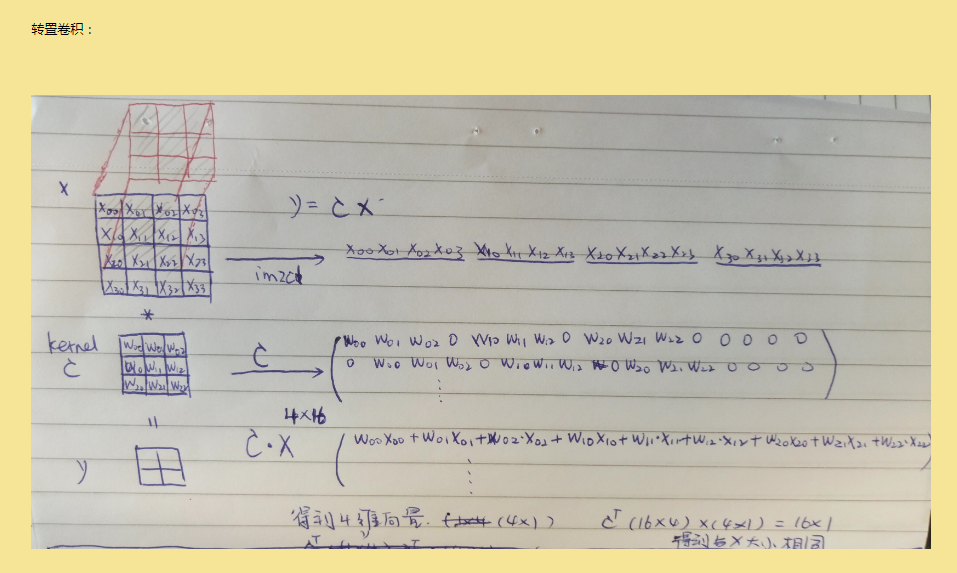




反卷积类似于一个卷积的过程，只不过在卷积之前将输入特征插值到一个更大的特征图然后进行卷积



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

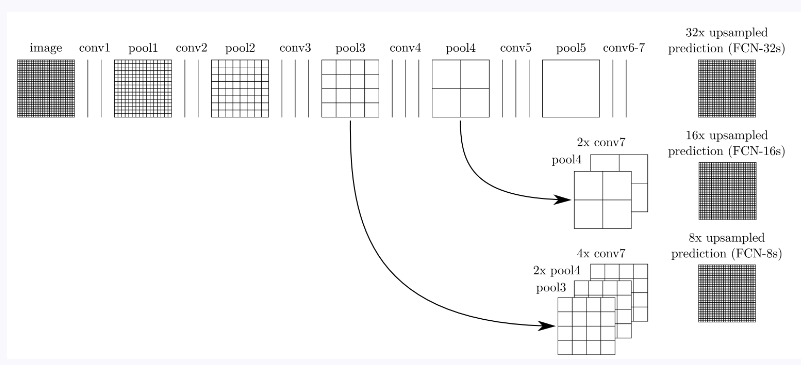
1. 文献贡献点

* 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

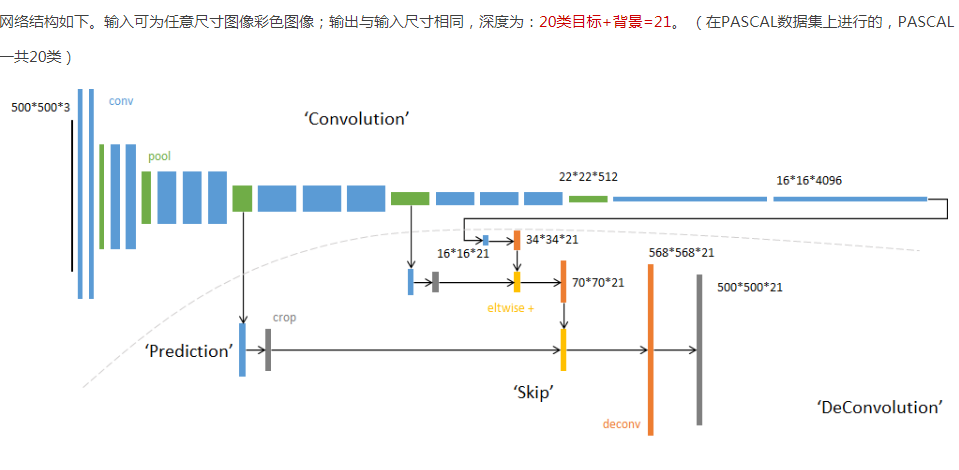
* 采用了skip结构，充分利用了浅层和深层的信息使得分类更精细



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/>

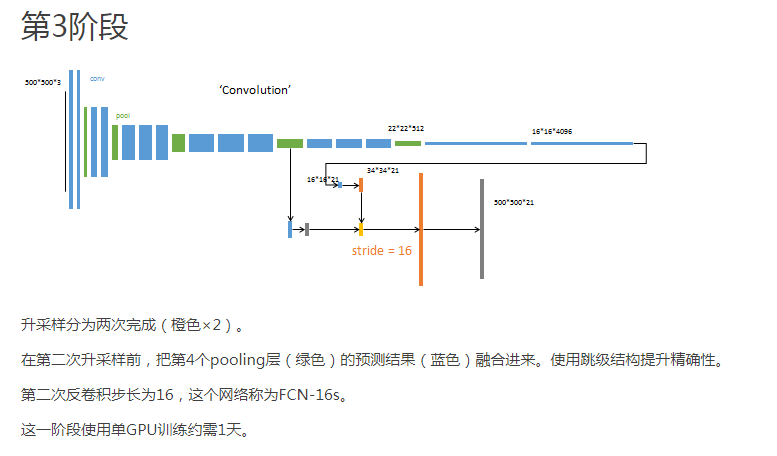
损失函数：

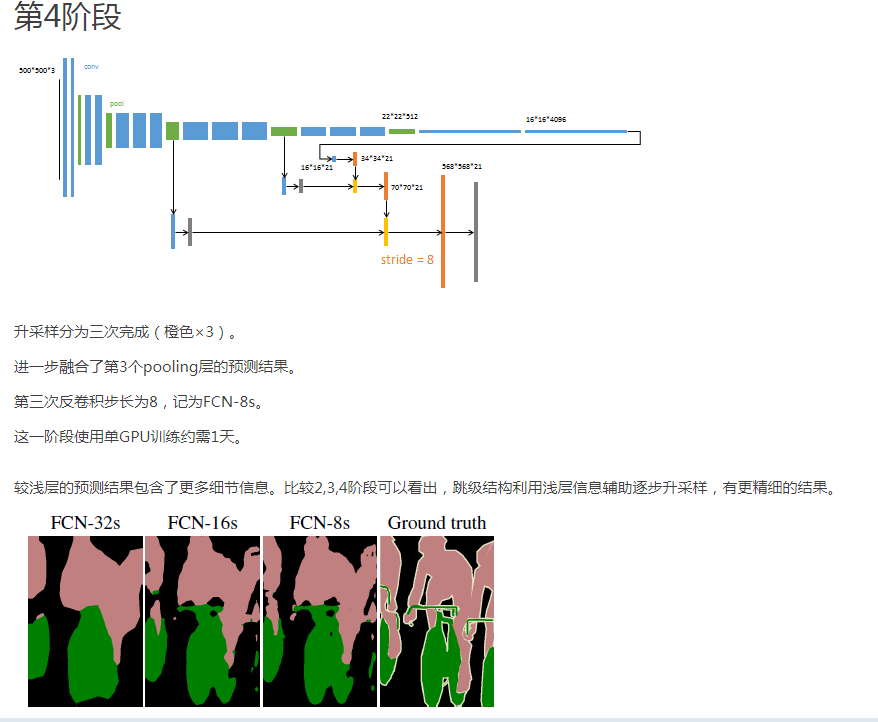
pixel-wise classification loss

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

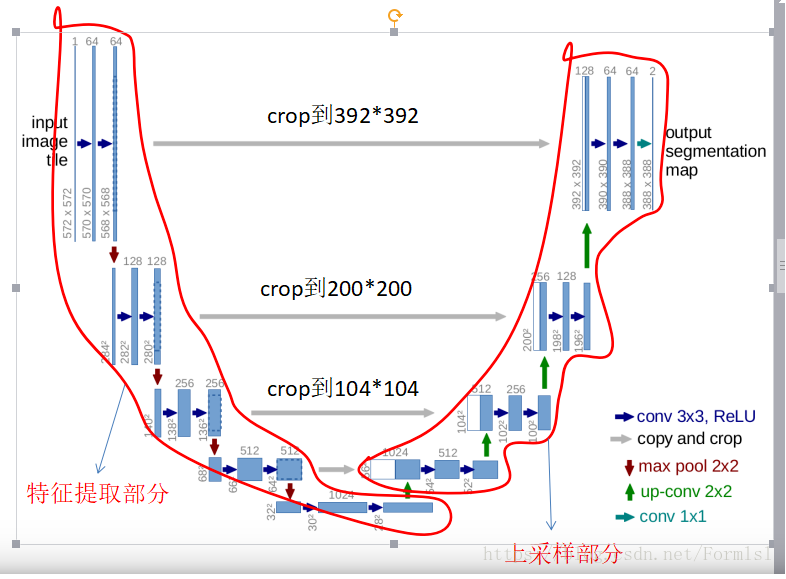
与传统用CNN进行图像分割的方法相比，FCN有两大明显的优点：一是可以接受任意大小的输入图像，而不用要求所有的训练图像和测试图像具有同样的尺寸。二是更加高效，因为避免了由于使用像素块而带来的重复存储和计算卷积的问题。

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



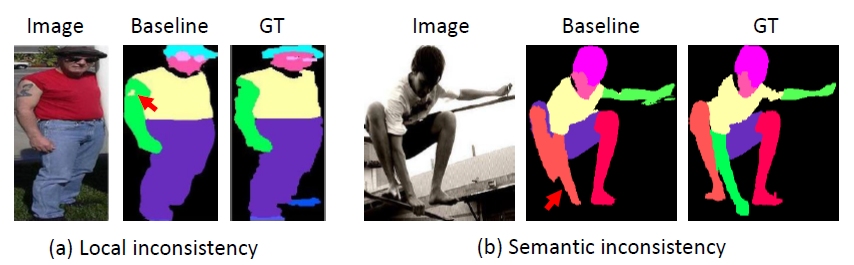
这里的up-conv 是先上采样，然后在卷积 和FCN里的反卷积有些区别。

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

MMAN

问题的提出：

人体部位分割（human parsing）研究中，大家主要使用基于像素级别分类的损失函数（pixel-wise classi\_cation loss），存在两个明显的缺点：低级别局部不一致性（low-level local inconsistency），比如分割结果局部模糊和有黑洞等， 和高级别语义不一致性（semantic inconsistency），比如错位的身体部位。如下图所示：

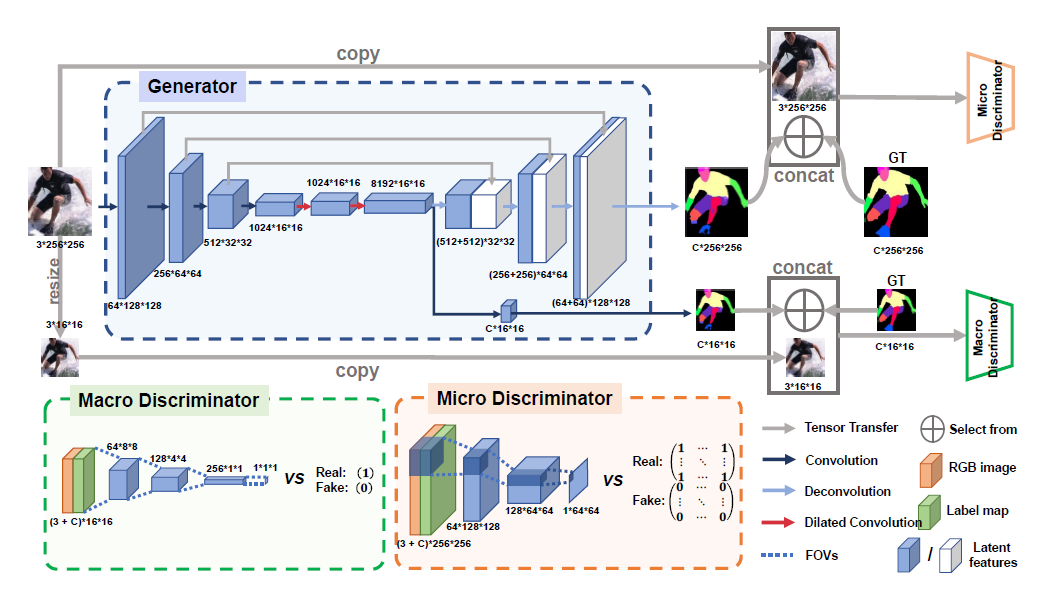


作者提出了宏微观对抗生成网络（MMAN）解决这个问题：

网络包含两个判别器和一个生成器，其中一个是宏观判别器器，主要充当低像素标签的映射，用于解决错乱的身体部位等语义不一致问题；另一个是微观判别器，充当高像素标签的映射，用于解决局部模糊、局部存在洞等局部不一致问题。

（It has two discriminators. One discriminator, Macro D, acts on the low-resolution label map and penalizes semantic inconsistency, e.g., misplaced body parts. The other discriminator, Micro D, focuseson multiple patches of the high-resolution label map to address the local inconsistency, e.g., blur and hole.）

网络架构：



## FPN

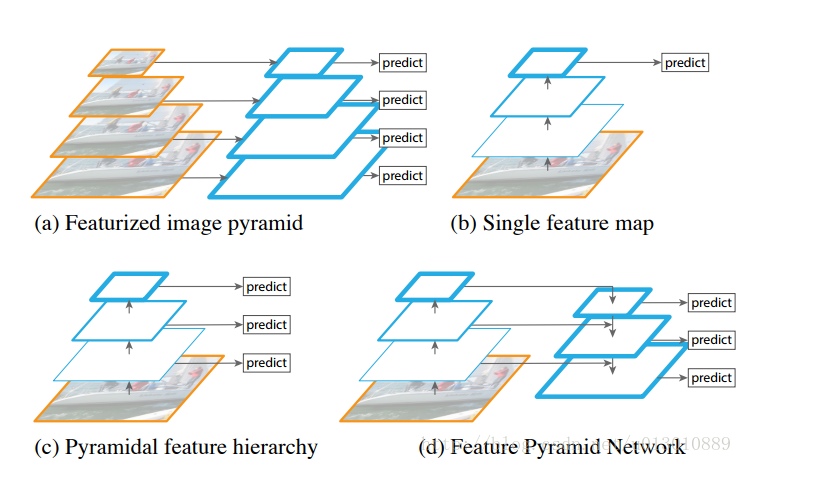
1、论文作者

论文链接：<https://arxiv.org/abs/1612.03144>

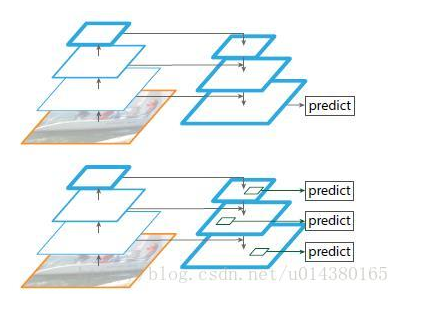
2、核心思想

**低层的特征语义信息比较少，但是目标位置准确；高层的特征语义信息比较丰富，但是目标位置比较粗略；**

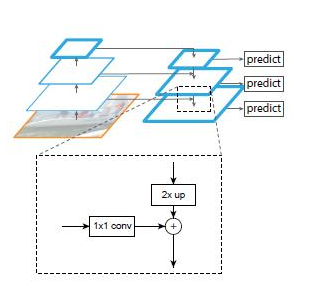
**另外虽然也有些算法采用多尺度特征融合的方式，但是一般是采用融合后的特征做预测，而本文不一样的地方在于预测是在不同特征层独立进行的。**



1. 图像金字塔，即将图像做成不同的scale，然后不同scale的图像生成对应的不同scale的特征。这种方法的缺点在于增加了时间成本；
2. 像SPP net，Fast RCNN，Faster RCNN是采用这种方式，仅采用网络最后一层的特征
3. 像SSD（Single Shot Detector）采用这种多尺度特征融合的方式，没有上采样过程，即从网络不同层抽取不同尺度的特征做预测，这种方式不会增加额外的计算量。作者认为SSD算法中没有用到足够低层的特征（在SSD中，最低层的特征是VGG网络的conv4\_3），而在作者看来足够低层的特征对于检测小物体是很有帮助的
4. 本文作者是采用这种方式，顶层特征通过上采样和低层特征做融合，而且每层都是独立预测的。



**在FCN中，如上面一个带有skip connection的网络结构在预测的时候是在finest level（自顶向下的最后一层）进行的，简单讲就是经过多次上采样并融合特征到最后一步，拿最后一步生成的特征做预测**。而下面一个网络结构和上面的类似，**区别在于预测是在每一层中独立进行的。**



作者的算法大致结构如下Fig3：一个自底向上的线路，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。

## 目标检测中的评价指标：

1.对于某个类别C，在某一张图片上

首先计算C在一张图片上的Precision=在一张图片上类别C识别正确的个数（也就是IoU>0.5）/一张图片上类别C的总个数

2.依然对于某个类别C，可能在多张图片上有该类别，下面计算类别C的AP指数：

AP=每张图片上的Precision求和/含有类别C的图片数目

3.对于整个数据集，存在多个类别C1、C2、C3：

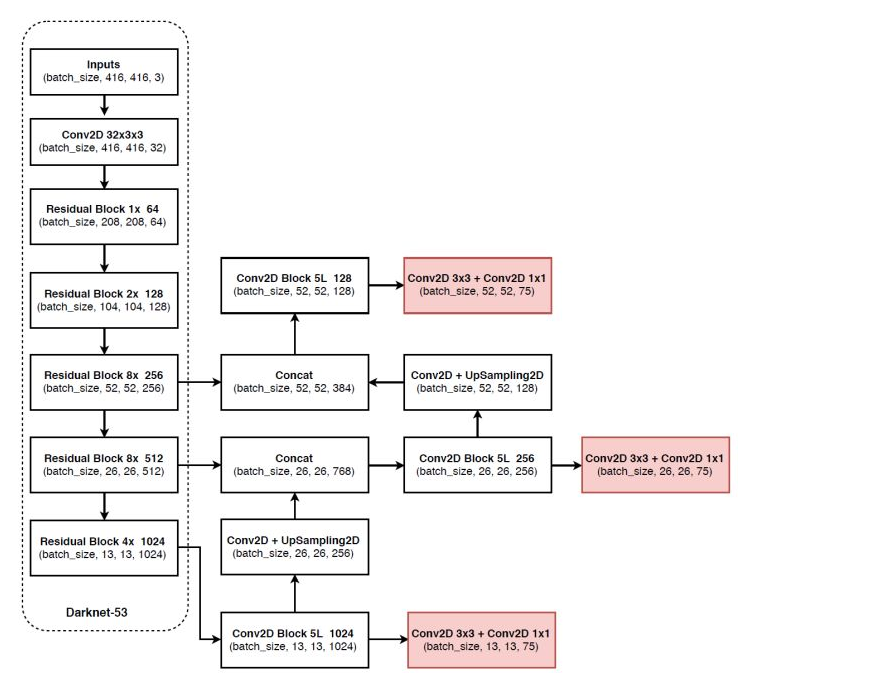
## Yolo:

1. 文献作者

YOLOv3: An Incremental Improvement

1. 网络结构

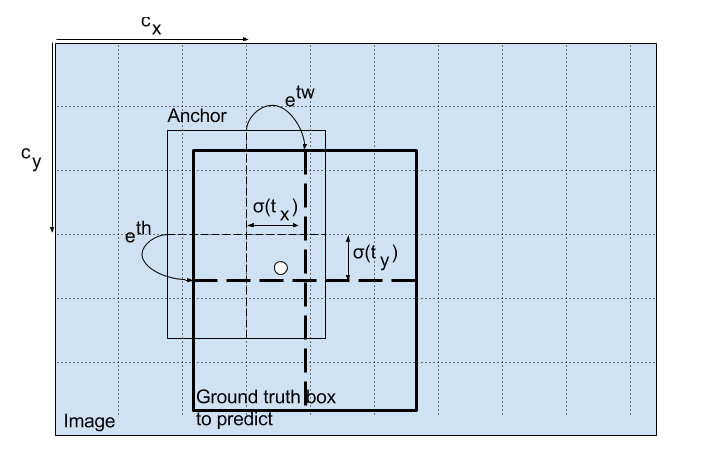
网络结构特点：采用全卷积网络（FCN），没有pool层，包含上采样层，concat连接方式，卷积核大小为3\*3或者1\*1，只有卷积步数（stride）为2时才减小卷积特征图，这样的原因是防止下采样丢失低级特征（This helps in preventing loss of low-level features often attributed to pooling）。416 x 416 经过stride = 32后，得到最终13\*13的特征图。使用激活函数使用 leak RELU。

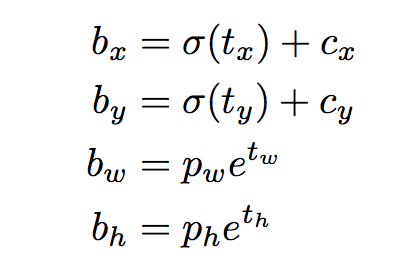


1. 核心思想

(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。

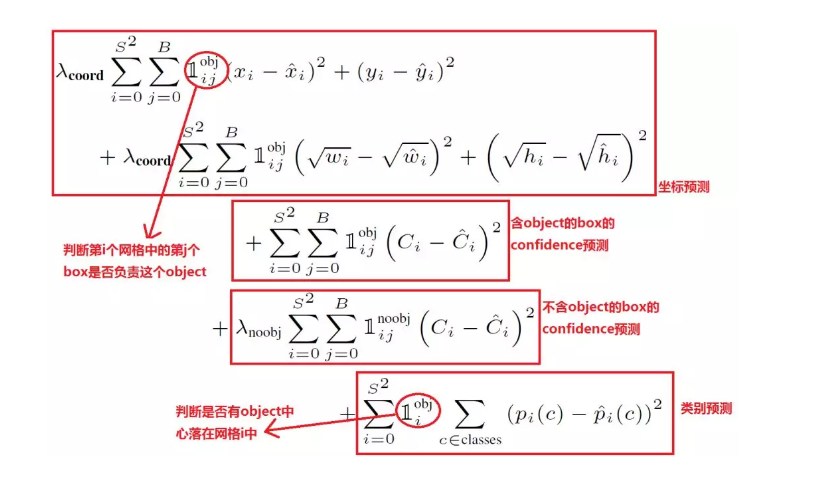
Object\_core: Object 分数表示目标在边界框内的概率。红色网格和相邻网格的 Object 分数应该接近 1，而角落处的网格的 Object 分数可能接近 0, objectness 分数的计算也使用 sigmoid 函数，因此它可以被理解为概率。

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

多标签分类：

论文: Multi-Label Fashion Image Classification with Minimal Human Supervision

作者引用的@21文献提出：(they showed that exploiting a largenumber of fashion images with only weak labels and de-signing a CNN to learn from them can provide a better fea-ture representation than ImageNet or Places for one of thefashion-related tasks, i.e., fashion style classification)

当搜集一个大的fashion数据集，然后只需要少量的数据进行finetune就能获得比在Imagenet上finetune的效果好.

摘要：we buildupon a recent approach that both cleans the noisy label sand learns to classify, and introduce simple changes thatcan significantly improve the performance.

作者说：影响分类效果有两个因素，一是样本大小；二是标签标注的质量

核心思想：

(DeepSAR and DeepMAR): Multi-attribute Learning for Pedestrian Attribute Recognitionin Surveillance Scenarios 2015

1、作者贡献：分别提出两种网络，DeepSAR采用sigmoid对每个标签二分类；DeepMAR利用了标签之间的关系；

To handle the complicated surveillance scenarios, theautomatically learned features are introduced intopedestrian attributes recognition instead of handcraft-ed features. Treating each attribute as an independentcomponent, the DeepSAR model is proposed to recog-nize each attribute one by one；

To exploit the relationship among attributes effectively,the unified multi-attribute jointly learning frameworkDeepMAR is proposed to recognize multi-attribute si-multaneously. In addition, the weighted sigmoid crossentropy loss is proposed to handle the unbalance a-mong attributes and obtain the state-of-the-art results

代码实现：

1. 多个标签单独分类，loss直接相加

loss\_fun = torch.nn.CrossEntropyLoss(weight=weight.cuda())

loss0 = loss\_fun(predict[0], target[:, 0])

loss = 0.3 \* loss0 + 0.1 \* loss1 + 0.1 \* loss2 + …

1. 采用sigmoid多变迁二分类

loss\_func = torch.nn.MultiLabelSoftMarginLoss()

loss = self.loss\_func(outputs, target)

其中target为one-short格式，总共19个，表示19个标签

y\_pred = utils.sigmoid(y\_pred)

accuracy = utils.accuracy(y\_true, y\_pred)

这里的实现需要给sigmoid设定阈值，大于阈值判为正例，即为1，小于为0

一些提升的trick:

1. 采用finetune，可以提升5到10个点；
2. 采用辅助分类网络，可以提升2个点左右；
3. 采用更好的模型，如resnet
4. 融合局部特征和全局特征进行分类，采用MGN网络，可提升2个点左右；
5. 图像增强，如左右翻转、随机遮挡、增加颜色噪点、亮度等，估计可以提升2个点；
6. 数据不平衡，可以设置loss中的weight比例，能增加recall；
7. 对于当个标签，采用粗细粒度分类，预计会提升2个点；

## FasterRcnn的细节理解

1. RPN和提取Feature层共享网络层，主要由卷积层和pool层组成；
2. RPN层最终得到相对于原图大小的roi，然后通过卷积下采样关系得到其主网络的Feature map的位置，最后映射到7\*7大小的Roipool层，最后输出分类回归；
3. Roipool层输入为proposals和Feature map。

依作者看来，如图1，Faster RCNN其实可以分为4个主要内容：

1. Conv layers。作为一种CNN网络目标检测方法，Faster RCNN首先使用一组基础的conv+relu+pooling层提取image的feature maps。该feature maps被共享用于后续RPN层和全连接层。
2. Region Proposal Networks。RPN网络用于生成region proposals。该层通过softmax判断anchors属于foreground或者background，再利用bounding box regression修正anchors获得精确的proposals。
3. Roi Pooling。该层收集输入的feature maps和proposals，综合这些信息后提取proposal feature maps，送入后续全连接层判定目标类别。
4. Classification。利用proposal feature maps计算proposal的类别，同时再次bounding box regression获得检测框最终的精确位置。

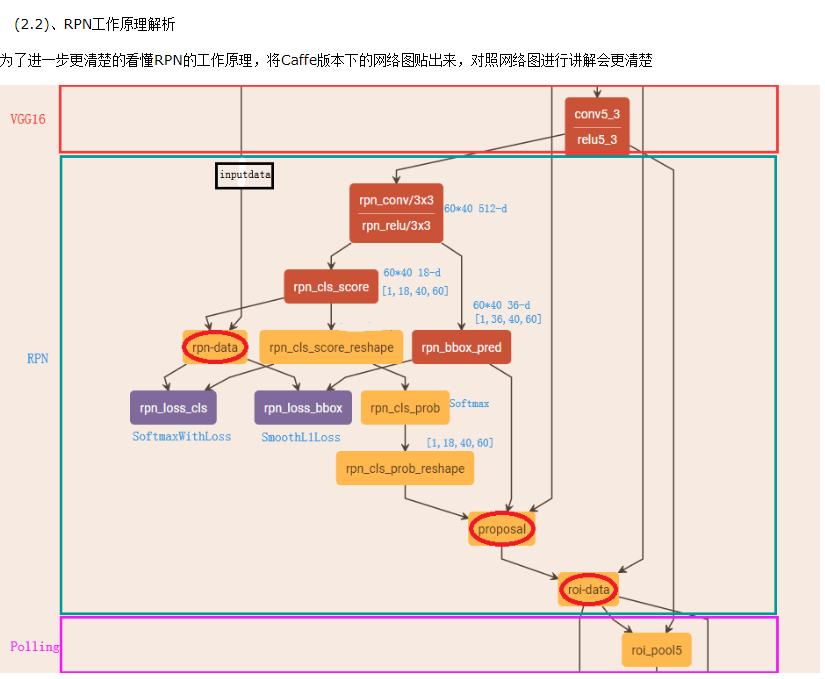
Faster-rcnn源码阅读笔记：

<https://blog.csdn.net/williamyi96/article/details/77648047>

<http://www.cnblogs.com/wangyong/p/8513563.html>

网络架构：





输入：vgg\_feature 层的输出con5\_3 [1, 512, 37, 50]， batch\_size为1

Rpn\_cls\_score 经过Rpn\_conv/3\*3 输出 [1, 512, 37, 50]

Rpn\_cls\_score\_reshape 为了适应softmax函数 [1, 2, 333, 50]

Rpn\_cls\_prob 直接预测是否前背景 [1, 2, 333, 50]

Rpn\_cls\_prob\_reshape 包括9个anchors \* 前背景 [1, 18, 37, 50]

Rpn\_bbox\_pred 包括9个anchors \* 四个坐标 [1, 18, 37, 50]

Rpn\_data : 训练的时候用于求loss,主要生成分类和回归坐标的labels

这一层主要是为特征图37\*50上的每个像素生成9个Anchor box，并且对生成的Anchor box进行过滤和标记，得到gt\_bbox的labels，生成规则：

①    去除掉超过1000\*600这原图的边界的anchor box

②    如果anchor box与ground truth的IoU值最大，标记为正样本，label=1

③    如果anchor box与ground truth的IoU>0.7，标记为正样本，label=1

④    如果anchor box与ground truth的IoU<0.3，标记为负样本，label=0

     剩下的既不是正样本也不是负样本，不用于最终训练，label=-1

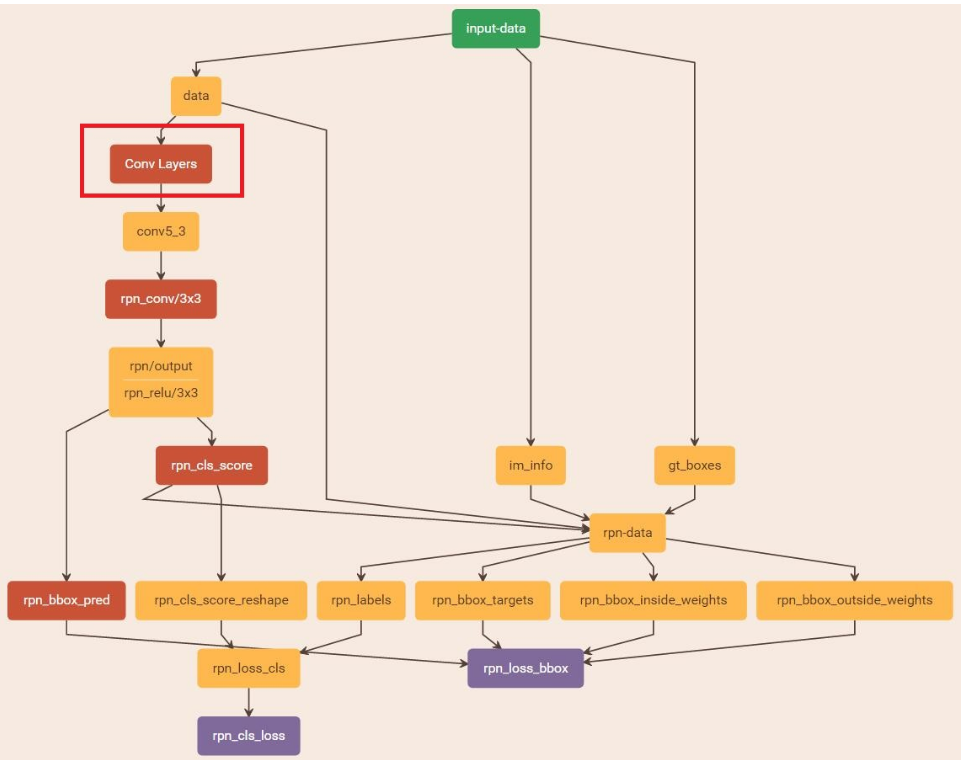
Proposal：用于生成粗略的region\_proposal，为特征图37\*50上的每个像素生成9个Anchor box，并且对生成的Anchor box进行过滤，得到2000个左右 [1,2000,5]

Roi\_data: 进一步过滤region\_proposal，得到300个左右的region\_proposal以及总类别的label；

①       RPN层只是来确定region proposal是否是物体(是/否),这里根据region proposal和ground truth box的最大重叠指定具体的标签(就不再是二分类问题了，参数中指定的是81类)

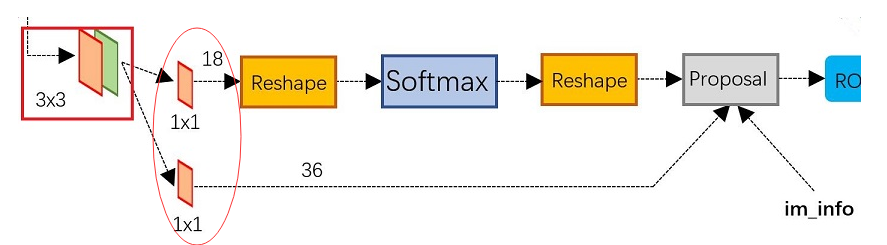
②       计算region proposal与ground truth boxes的偏移量，计算方法和之前的偏移量计算公式相同

Rpn网络的训练：



Faster-rcnn\yolov1\yolov3\mtcnn分类回归阶段的分析？

Fasterrcnn关于rpn处的分类和回归：



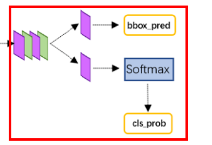
如上图所示，假设主网络输出为512\*47\*37，经过3\*3\*512的卷积运算后仍然是512\*47\*37，47\*37的每个像素固定对应9个anchors。

分类：512\*47\*37 和18个 1\*1\*512 卷积 =》18\*47\*37 =》9\*47\*37\*2 =》softmax

这里可以这样理解，这里只是softmax二分类，只不过这种图片对应的正负样本（roi）有9\*47\*37个，当然不是所有的样本都会拿去训练，最终会根据Roi是否越界、iou等来确定最终的训练正负样本；

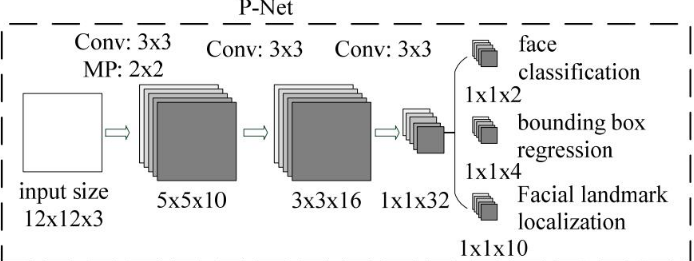
回归：对上面帅选出来的roi采用smooth1 loss回归。

Fasterrcnn关于最终结果的分类和回归：



如上图所示，假设总共有20类，在vgg中，最后全连接层是4096，然后输出对应两个分支。一个用于分类，总共20个输出，分别对于20类，一个对于回归，总共80个输出分别对于20类的四个坐标。

Mtcnn的回归和分类：



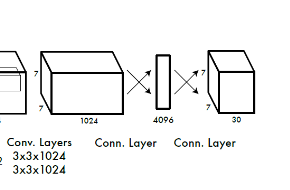
Mtcnn从输入到输出刚好对于12个步长，假设提取特征的最后特征图是32\*5\*5，那么5\*5大小特征图对于25个输出，每个像素对应一个。

分类：32\*5\*5 和2个1\*1\*32卷积核卷积运算 =》5\*5\*2 =》softmax二分类

这里同样不会拿全部25个样本去训练，正负样本由随机取的框与groundtruth的overlap区域比例的大小决定，大于0.65为positives，小于0.3为negatives，0.4到0.65之间为partfaces。positives和negatives参与到分类任务

回归：32\*5\*5 和4个1\*1\*32卷积核卷积运算 =》MSELoss 均方差回归。

Yolov1的分类和回归：



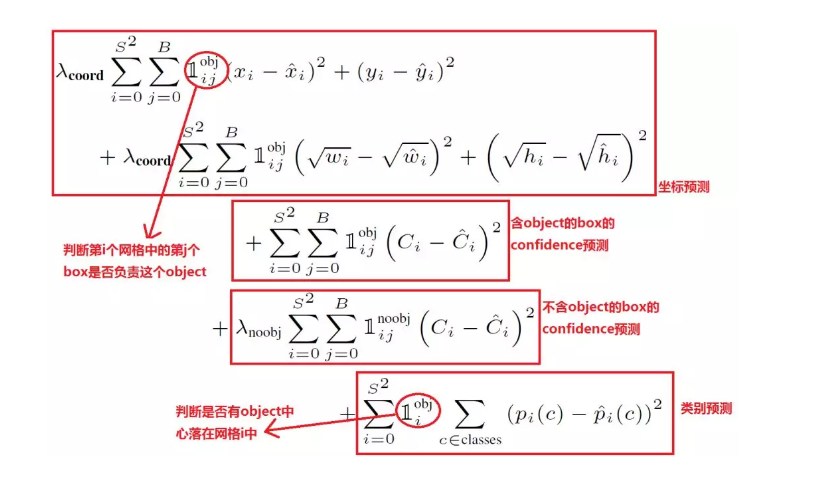
Yolov1的loss全部时候均方差Mseloss。

如图所示，最后一层特征提取层大小是4094，然后连接到1470的连接层，再sigmoid归一化reshape成7\*7\*30的大小。其中7\*7的大小对应上7\*7的网格，30代表的意义是两个bbox的x,y,w,h,confidence以及20类。

坐标x,y代表了预测的bounding box的中心与栅格边界的相对值。

坐标w,h代表了预测的bounding box的width、height相对于整幅图像width,height的比例。 confidence就是预测的bounding box和ground truth box的IOU值。

现在的问题是怎么区分正负样本？



上图是loss的组成。虽然每个格子对应两个bbox，但是一个格子最终只有一个用来预测目标，loss组成分析如下：

不包含目标的confidence loss，对应第三项

包含目标但是不负责预测的confidence loss 对应第二项

包含目标且负责预测的confidence loss 对应第二项

包含目标且负责预测的 坐标预测 loss 对应第一项

包含目标的分类loss，对应第四项

首先说明下yolov1的标签格式，target的size也是7\*7\*30，其中第五个和第十个代表某个网格是否包含目标。初始化如下：

target = torch.zeros((grid\_num,grid\_num,30))

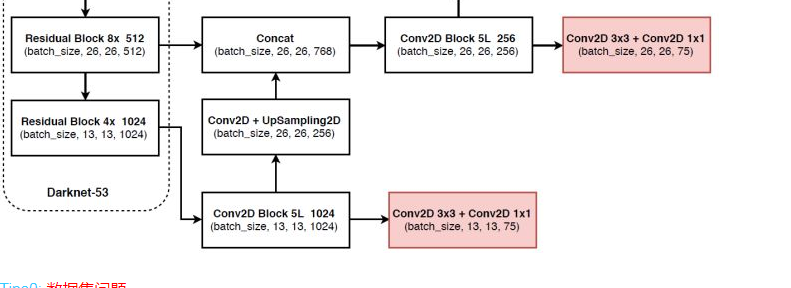
注意：包含目标时，通过计算iou来判断是否负责预测目标）

所以，

是否包含目标的正负样本划分：根据标签来确定，由于样本不均衡，所以在loss前面要乘以，分类和坐标预测的样本数由target确定，一个格子对应一类。

Yolov3的回归和分类：

yolov3由多尺度输出，浅层的输出融合深层的特征，主要负责小目标的检测，深层的负责大一点的目标检测，这样有利于recall的提升。



上图是某个尺寸的输出，最终得到的是1\*13\*13\*75，其中75是根据类别个数计算的，比如有20类，则（20+5）\* 3,3表示3个anchor，5分别表示x,y,w,h,confidence.

坐标x,y代表了预测的bounding box的中心与栅格边界的相对值，经过sigmoid归一化

坐标w,h代表了预测的bounding box的width、height相对于anchor的width,height的比例。 Confidence表示包含目标的可能性，sigmoid归一化。

注意与yolov1的区别，多尺度输出，每个格子预测三个目标。

怎么区分正负样本？

对于某个尺度的输出，最终包含13\*13\*3个anchor，即要预测目标个数。

计算每个gtbox与anchors的iou，其中iou最大的为正例，iou小于阈值的为负例

loss\_x = self.bce\_loss(x \* mask, tx \* mask)

loss\_y = self.bce\_loss(y \* mask, ty \* mask)

loss\_w = self.mse\_loss(w \* mask, tw \* mask)

loss\_h = self.mse\_loss(h \* mask, th \* mask)

loss\_conf = self.bce\_loss(conf \* mask, mask) + \

0.5 \* self.bce\_loss(conf \* noobj\_mask, noobj\_mask \* 0.0)

loss\_cls = self.bce\_loss(pred\_cls[mask == 1], tcls[mask == 1])

# total loss = losses \* weight

loss = loss\_x \* self.lambda\_xy + loss\_y \* self.lambda\_xy + \

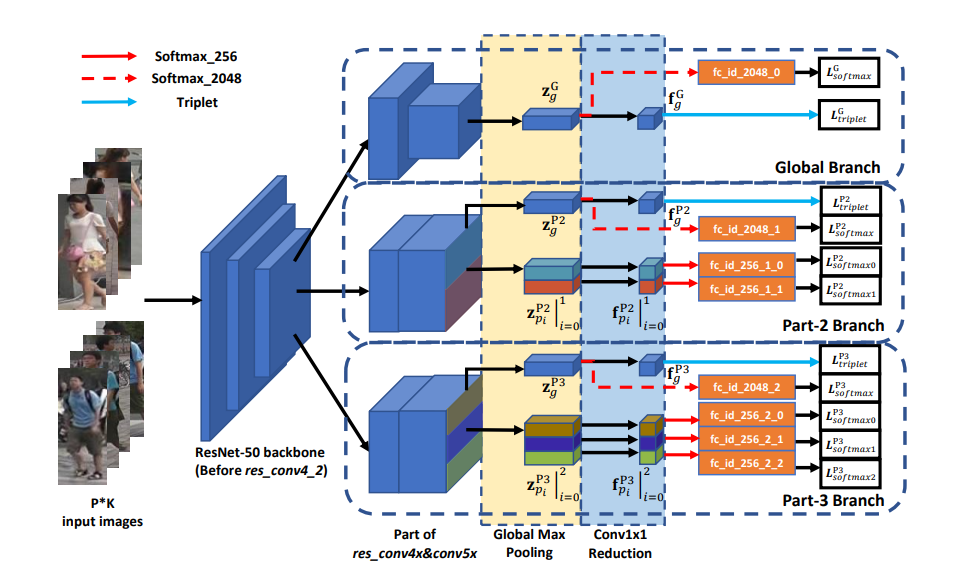
loss\_w \* self.lambda\_wh + loss\_h \* self.lambda\_wh + \

loss\_conf \* self.lambda\_conf + loss\_cls \* self.lambda\_cls

上式中，x,y,conf,class采用了sigmoid的交叉熵损失函数，w,h使用了均方差损失函数。

行人重识别（MGN）总结：

1. 网络结构



网络组成：Backone + 3个分支

Backone：由resnet—resnet.layer3[0]

Branch1：resnet.layer3[1:]—resnet.layer4-- resnet.avg

Branch2：resnet.layer3[1:]—resnet.custom\_layer4()(无下采样)--resnet. custom\_avg

Branch3：resnet.layer3[1:]-- resnet. custom\_layer4-- resnet. custom\_avg

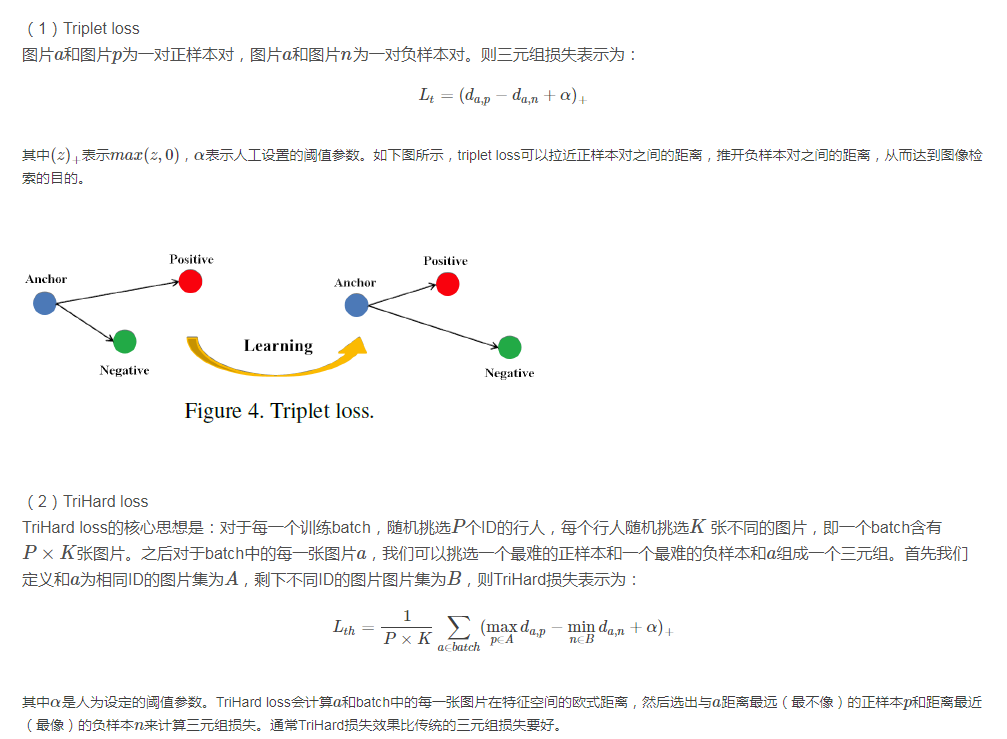
第二、三个分支和第一个分支的区别在于layer4没有下采样操作，这样有利于网络学习更细节的特征。

1. Loss设计：

Triplet\_loss + softmax

对于全局特征采用Triplet\_loss，每个分支再经过softmax分类，分类总数是行人库的不重复的人数。

1. 训练技巧

在 ReID 方案里面我给大家介绍一个 Batchhard的策略，因为 TripletLoss 在设计时怎么选这三张图是有很多文章在实现不同算法，我们的文章里用的是 Batchhard算法，就是我们从数据集随机抽取 P 个人，每个人 K 张图片形成一个 Batch，每个人的 K 张图片之间形成一个 K×（K-1）个 ap 对，再在剩下其他人里取一个与该 ap 距离最近的 negtive，组成 apn 组，然后我们这个模型使得 apn 组成的 Loss 尽量小。

1. 关于rerank

假设查询图片是q，图片库是p

第一步q查询1000张图片p1…p1000;

第二步，p1从1000张图片中查询6张图片c1,c2,c3,c4,c5,c6;c1再从1000张查询3张图e1,e2,e3,然后取c\*和e\*交集，如果超过2/3，则将e\*合并到c\*，最终结果是r1\*

P2..p1000重复这样操作，得到各自的r\*;

第三步，得到q的rq\*;

第四步，求rq\*和rj\*的杰卡德距离。

第五步，根据最初的欧氏距离和杰卡德距离得到最终的距离,即\*d + (1-@)j,然后重排序。

上述中1000,6,3都可以自定义设置

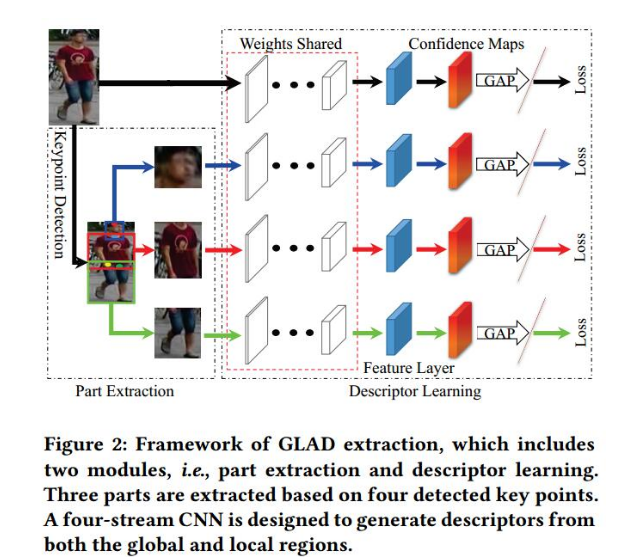
其中欧式距离可以进一步换成平均欧氏距离

关于多图片查询：多张图片输入搜索，然后取对应位置最小的距离（最相似）

### GLAD: Global-Local-Alignment Descriptor for Pedestrian Retrieval

地址: <https://arxiv.org/abs/1709.04329>

框架：



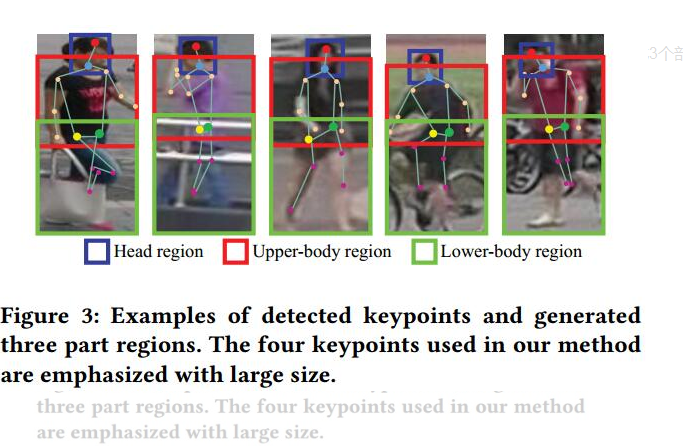
17年的一篇论文，基于局部特征的reid方法，网络结构包含两个部分：

Part Extraction:提取头部、上半身、下半身三个部位。

Descriptor Learning(描述子学习):将全局以及提取出的三个部位输入到网络中进行分类。

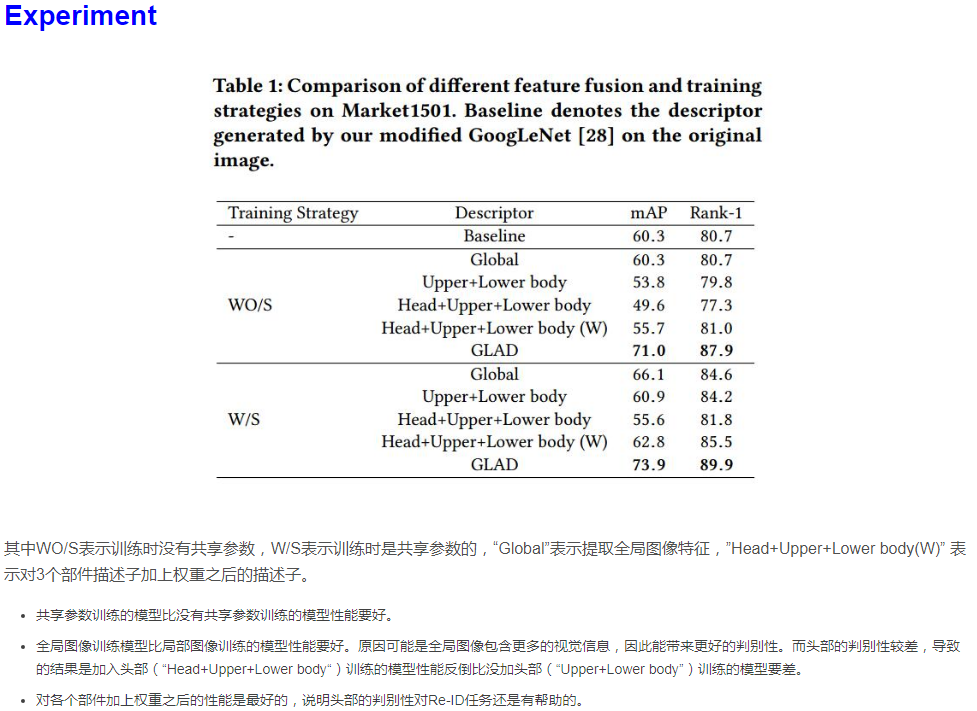
实现细节：

* 采用Deeper Cut算法估计行人的4个关键点（upper-head, neck, right-hip, left-hip），依据这4个关键点将输入图像分成3个部件（head, upper-body, and lower-body）



* 使用GoogleNet作为主干网络，去除最后两个全连接层，改成卷积层，第一个卷计层作为feature map用来提特征，第二个卷积层作为confidence map，该层特征图个数和训练集的类别个数一致，在这层之后加上global pooling层用作softmax分类。替换掉全连接层的好处：减少参数、适应任意大小的输入图片。
* 提取特征的时候，使用第一个卷积层提取的特征，然后经过GAP得到最后的特征，假设提取到某个部位的最后的特征维度是M，则最后的特征是4\*M维。

实验结果：



这篇论文和MGN的区别和联系：

首先都是使用了全局特征融合局部特征的方法；不同点：第一，GLAD是通过关键点检测来剪切图片，这里由于关键点检测的错误会引入一点误差；第二个是GLAD需要每个部位都经过一次提取特征，计算量增大；第三就是GLAD多个部位提取特征时，主干卷积层共享，只是在最后的分类阶段不同，而MGN则是在通用的卷计层共享特征（layer3[1:0]之前）,后面都是各自提取特征；第四MGN训练的时候才用了tribloss和softmax，GLAD只是每个部位分类，分别训练网络参数。