一. 卷积基础知识

卷积的特点: **局部连接,权值共享**

1.1 感受野的计算

- 对于第i层为卷积或者池化层:

$$R_e^i = min(R_e^{i-1} + (K_e^i - 1) \times \prod_{j=1}^{i-1} s_e^j), L_e)$$

其中:

$$R_e^0 = 1, s_e^0 = 1$$

R表示感受野. s表示步长, k表示kernel. L表示原始特征图的尺寸.

- 第i层为激活层:

$$R_e^i = R_e^{i-1}$$

- 第i层为全连接层:

$$R_e^i = L_e$$

1.2 卷积层的输出尺寸

$$l_e^o = \lfloor \frac{l_e^i + 2P_e - K_e}{s_e} \rfloor + 1$$

二. 卷积变种

2.1 分组卷积与深度可分离卷积

这两种卷积都可以用来降低模型的容量.

2.1.1 分组卷积

分组卷积最早出现在alexnet中,主要用来解决在单个gpu显存较小无法处理含有较大计算量与存储需求的卷积层, 因此采用分组卷积来将计算与存储分离在不同的gpu上.

后来在ResNeXt中,作者广泛采用分组卷积来降低模型的计算量与容量.

分组卷积就是自己处理自己的那一部分.

例如针对一个`C x H x W`的一组特征图, 如果采用普通的卷积,需要的卷积核的大小为`C x K x K x O`, 输出的特征层为`O x H x W`(假设特征层的宽高不变).

如果采用分组卷积, 将其分为`0.5C x H x W`与`0.5C x H x W`的两部分. 采用卷积核的大小为`0.5C x K x Kx 0.5O` 与`0.5C x K x K x 0.5O`, 输出的特征层分别为`0.5O x H x W` 与`0.5O x H x W`, 然后二者合成原来的`O x H x W`.

2.1.2 深度可分离卷积

深度可分离卷积就是采用depth wise卷积加上point wise卷积得到的层，相比较普通卷积而言，可以很大程度上减小参数量

对一个`C x H x W`的一组特征图，如果采用普通的卷积,需要的卷积核的大小为`C x  K x K x O`, 输出的特征层为`O x H x W`(假设特征层的宽高不变).

如果采用深度可分离卷积, 需要由两部分组成，第一部分是`C x K x K x 1`的depth wise卷积， 第二部分是`C x 1 x 1 x O`的point wise卷积

2.2 转置卷积

转置卷积(又叫做反卷积).其实也不能称作反卷积,因为特征层经过普通卷积操作后然后经过转置卷积仅仅能恢复原来的形状大小,不能恢复原来的数值.

具体的介绍可以参考大佬的博客: https://blog.csdn.net/lanadeus/article/details/82534425

2.3 空洞卷积

通常可以采用pooling的方法来扩大感受野, 但是pooling的过程会降低特征图的分辨率,损失信息,导致上采样过程中很难恢复原来的信息.

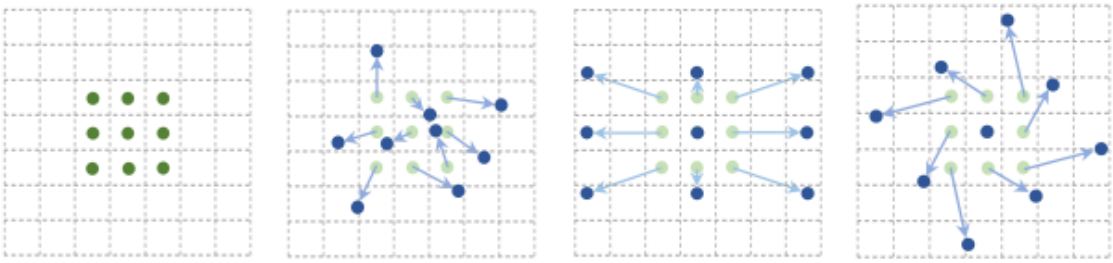
空洞卷积在标准的卷积中插入空洞来增加卷积核的感受野，空洞卷积引入扩张率(dilation rate)这个超参数, 就是在卷积核中插入`r-1`个空洞(0), r为扩张率.

对于标准的卷积核`K x K`, 使用r的扩张率后,卷积核的尺寸变为`K + (r-1)* (k-1)`

2.4 可变形卷积

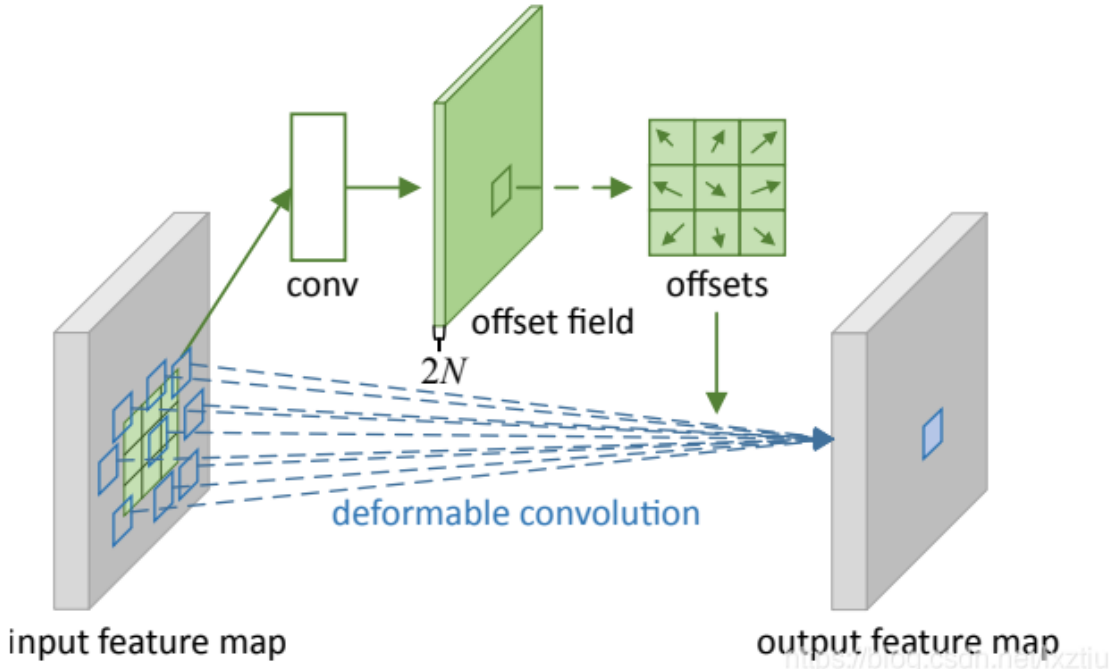
普通卷积是在固定的, 规则的网格点上进行采样,这就束缚了网格感受野的相撞,限制了其几何形变的适应能力. 为了克服这个限制,可变形卷积(DCN)在每个采样点上添加一个可学习的偏移量(offset), 让采样点不再局限于规则的网格点.

如图所示:



如上图,最左端为普通的卷积,后边的为可变形卷积,通过不同的offset,得到可变形的效果.

offset的取得由卷积层来实现, 引入一个平行分支,根据输入的特征图计算出采样点的偏移量, 然后在输入特征图上对应的点进行卷积运算.



三. 卷积整体结构

主要介绍如下几种网络结构:

> AlexNet-> VGGNet->GoogleNet/Inception-v1 -> inception v2/inception v3 -> ResNet -> Inception-v4/ Inception-resnet-> ResNeXt.

分别介绍这几种网络的特点:

3.1 AlexNet

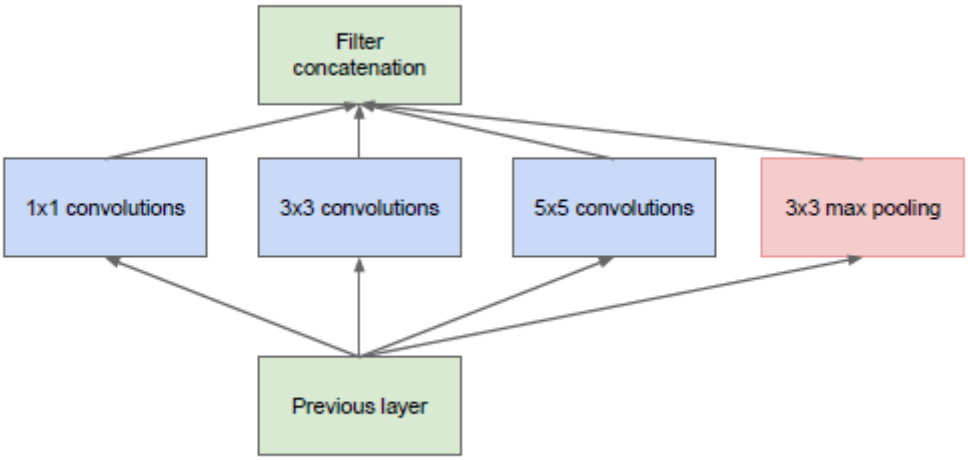
- 采用ReLU作为激活函数
- 采用局部响应归一化LRN
- dropout与数据增广
- 利用分组卷积解决gpu显存不足的问题

3.2 VGGNet

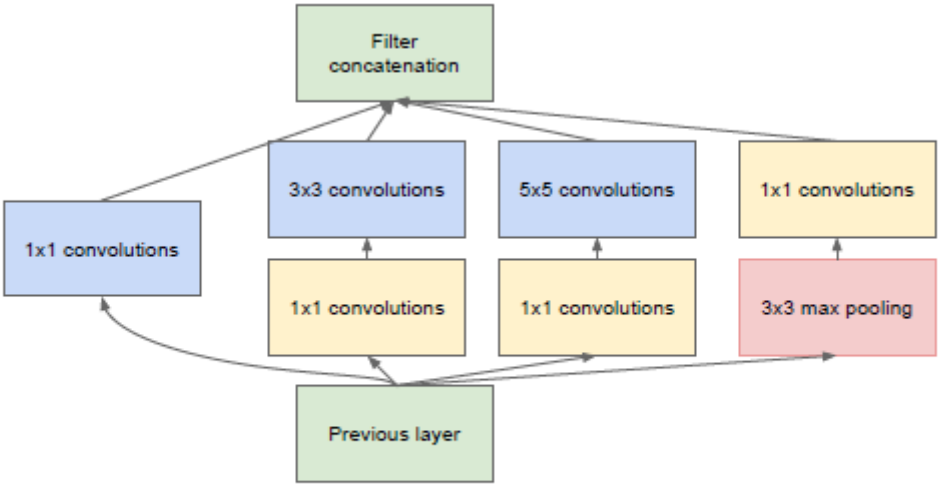
- 3x3的小卷积核代替之前较大的5x5与7x7卷积核.
- 2x2的池化代替3x3的池化
- 去掉了局部响应归一化

3.3 GoogleNet/ Inception-v1

- inception模块
- bottleneck模块
- 利用全局平均池化替代第一个卷积层



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

上边的两个结构分别为不带bottleneck的inception模块与带有bottleneck的版本.

3.4 Inception-v2/Inception-v3

提出了网络设计的准则:

- 避免表达瓶颈, 尽量让网络从前到后的各个层的信息表征能力逐渐降低不能剧烈下降,或者在中间某些节点出现瓶颈.
- 特征图的通道数越多,能表达的解耦信息就越多,更容易继续宁局部处理.
- 如果要在特征图上做空间域的聚合,可以先对特征图进行压缩,降低参数量,这样不会导致表达能力的损失
- 深度与宽度需要平衡

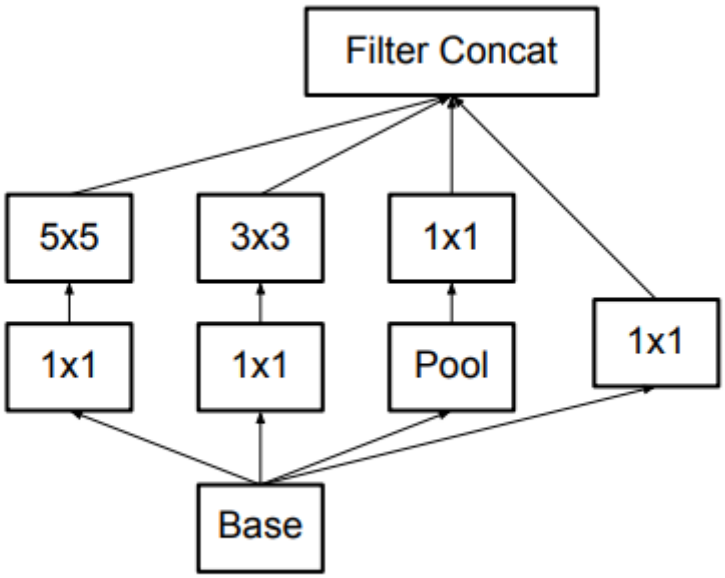


Figure 4. Original Inception module as described in [20].

使用3x3取代原始inception模块的5x5.

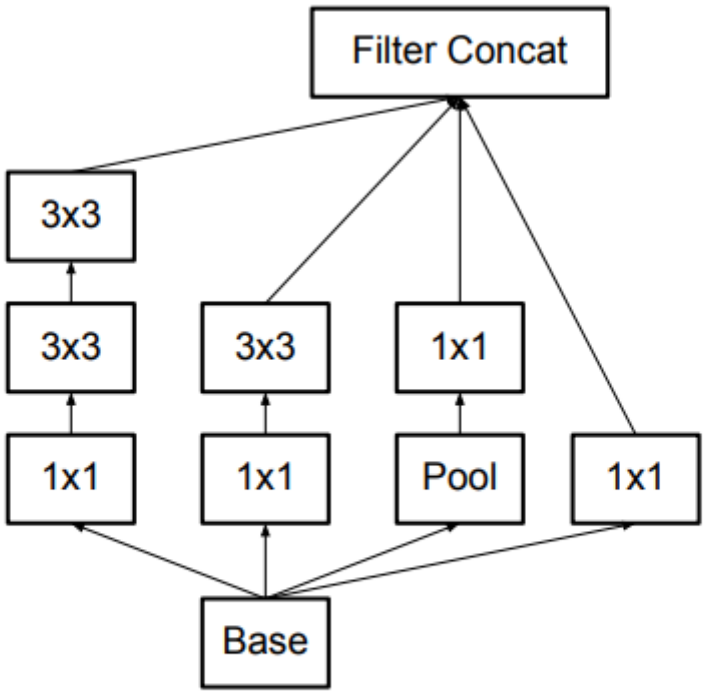
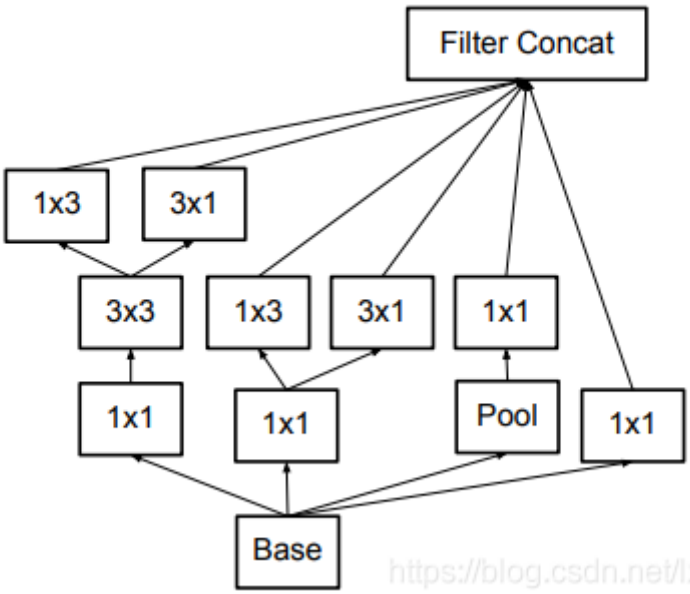


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

3x3分解为1x3与3x1的串联或者并联



3.5 ResNet

- 采用shortcut连接,可以一直梯度消失的现象.

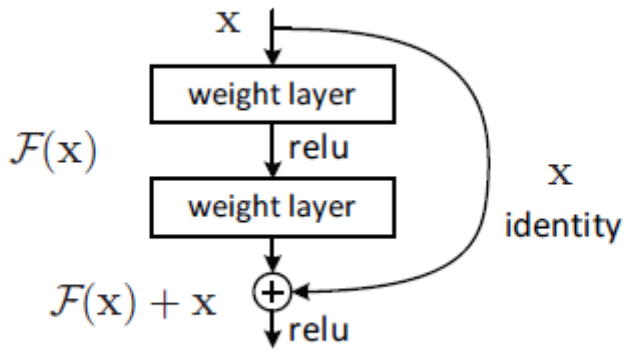


Figure 2. Residual learning: a building block.

3.6 Inception-v4/Inception-ResNet

进行了更加精细的设计,融入了resnet.

3.7 ResNeXt

- 采用分组卷积替代residual 模块中的普通卷积.

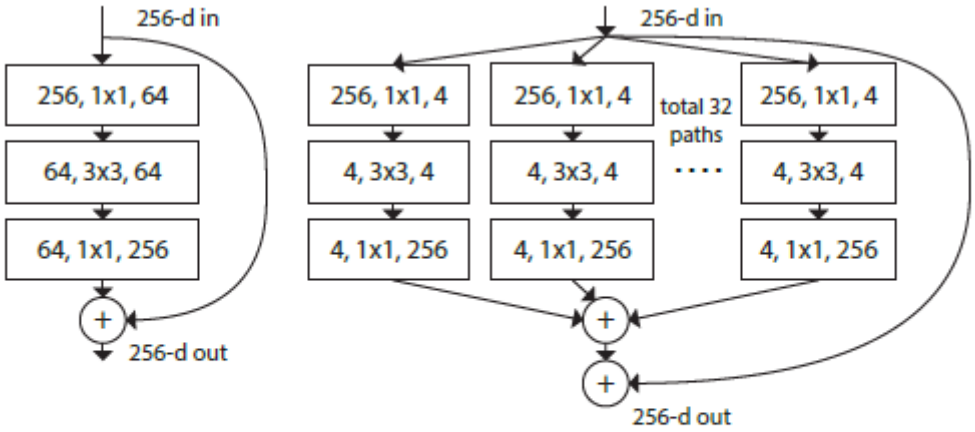


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

四. 卷积基础模块

4.1 BN

机器学习中,一般会假设模型的输入数据的分布是稳定的,如果**模型输入数据发生变化**,则称为协变量偏移,模型的训练与测试集的分布不一致,或者模型在训练过程中输入数据发生变化,这都属于**协变量偏移现象.**

对于深度神经完了过训练过程中,每一层的参数都会随之而更新,训练过程中如果之前层的参数被更新,那么该层的输入数据的分布必然也会发生变化,网络越深,这种内部协变量偏移的现象更加明显. 这种协变量偏移的会带来很多的问题:

- 网络的每一层都要不断适应其输入数据的变化,学习效率低,训练过程缓慢.
- 网络前几层的更新会使得后几层的输入变得过大或者过小,从而陷入激活函数的饱和区,学习过程停止.
- 为了降低协变量偏移的影响,学习率尽量使用小一些,降低收敛速度.

BN主要是为了确保即使网络的参数发生变化,各个层的输入输出数据的分布也不会发生较大的变化.

公式:

$$
\begin{aligned}
\mu_{\mathcal{B}} &\leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i && \text{// mini-batch mean} \\
\sigma_{\mathcal{B}}^2 &\leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_{\mathcal{B}})^2 && \text{// mini-batch variance} \\
\widehat{x}_i &\leftarrow \frac{x_i-\mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}} && \text{// normalize} \\
y_i &\leftarrow \gamma\widehat{x}_i+\beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) && \text{// scale and shift}
\end{aligned}
$$

x与y分别为原始输入数据与BN后的输出数据,有两个可学习的参数γ(缩放参数)与β(平移参数).在这两个参数的作用:

- 保留网络各层在训练中的成果,如果没有这两个参数,那么批归一化就会退化为普通的标准化,训练过程中参数虽然在更新,输出分布几乎不变(均值始终为0, 标准差为1), 不能有效的学习,有这两个参数

后,网络可以为每个神经元自适应的学习一个量身定制的分布,保留每个神经元的学习成果.
- 使BN有自我关闭的能力,若这两个参数分别取数据的均值与标准差,那么就可以复原初始的值,也就是BN被关闭.

BN放置在激活层之前之后均可

- 如果放置在激活层之前,那么就可以有效比秒BN破坏非线性特征的分布,另外BN也可以使得数据点尽量不落入激活函数的饱和区,防止梯度消失.
- 由于常用激活函数为ReLU,没有sigmoid的种种问题,因此,也可以放置在激活层之后,避免数据在激活层之前被转化为相似的模式从而使得非线性特征分布趋于同化.**工业界通常倾向于放置在激活层之后**

4.2 分类网络最后几层的变化

在AlexNet 与VGGNet中采用**两层的全连接层**,而在GoogleNet之后的网络中一般采用**全局平均池化层加上一层全连接层**实现.

全局平均池化的优点:

- 参数量与计算量大大降低.全局平均池化的参数量为0, 对于`cxwxh`的特征层, 次啊用全局平均池化的计算量为`cwh`, 采用连接k个节点的全连接层,计算量为`cwhk`.
- 具有较好的可解释性,我们可以知道特征图上哪些部分对于分类贡献最大(类别激活图CAM).

4.3 瓶颈与沙漏结构

- **瓶颈结构**
  采用1x1的卷积来压缩卷积层输入特征图的数目,减小计算量.完成之后根据需要会使用1x1的卷积将特征图的数目复原.
- **沙漏结构** : **一个编码解码器的结构**

hourglass网络,在目标检测的anchor-free方法中使用.

- 自底向上: 将特征图尺寸逐层压缩(编码器)
- 自顶而下: 利用反卷积或者插值等上采样操作将特征图的尺寸逐层扩大(解码器)
-

五. 目标检测

5.1 单步法与两步法性能差异的原因

- **单步法**: **没有独立的显式的提取候选区域**的过程, 直接由输入图像得到其中存在的物体的类别和位置信息.
- **两步法**: 有独立的显式的提取候选区域的过程,先在输入图像上筛选出可能存在物体的候选区域,然后针对每个候选区域,判断是否存在物体,如果存在就给出类别以及位置修正信息.

单步法在检测速度上有优势,两步法在检测精度上有优势.出现这种差异的原因:

- 单步法采用预设的anchor box来捕捉可能存在图像中各个位置的物体. 单步法模型会针对数量庞大的锚框进行是否含有物体以及物体类别的密集分类,**正负样本数目极不均衡**,Focal loss通过一直负样本对最终损失的贡献来提升网络的整体表现. 两步法有独**立的候选区域提取**的步骤,第一步就可以筛选掉大量不含有物体的区域,第二部进行分类以及候选框位置修正时,正负样本的比例已经比较均衡.
- 两步法在提取候选区域的过程中,会对候选区域的位置进行修正,候选区域的特征已经对其,第二部分类的效果更好, 另外在第二部中会对候选框进行**第二次的修正**,得到更高的定位精度. 但是这种操作

同时增加了模型的复杂度, 检测速度较慢.
- 以faster rcnn为代表的两步法模型在第二部对候选框进行分类和位置回归时,是针对每个候选区域单独进行,因此这一部分的算法复杂度正比于语塞的候选区域的数目, 这部分的数目往往十分大, 导致这部分的计算量非常大.

5.2 RCNN系列的发展过程

5.2.1 RCNN

- 无监督的选择性搜索将输入图像中具有相似颜色直方图的特征区域进行递归和步兵,产生候选区域
- 将候选区域的图像裁剪缩放,利用cnn提取特征,svm分类器进行分类
- NMS进行去重

5.2.2 SPPNet

- SPP为空间金字塔池化, 可以接受任意尺度的特征图作为输入,然后通过三个窗口大小可变的池化层输出具有固定大小的池化特征.这样可以输入任意尺度大小的图像
- 直接在特征图上针对对应的候选区域进行截取.

5.2.3 Fast RCNN

- 使用ROI Pooling代替SPP
- 全连接层代替SVM,实现端到端的训练

5.2.4 Faster RCNN

- 采用RPN代替选择性搜索.
- 一幅图像先使用RPN得到候选区域,然后再取出各个候选区域的特征图,送入Fast rcnn的后半部分进行分类与回归

5.3 yolo系列的发展过程

5.3.1 yolo

yolo整体使用一个端到端的cnn来直接预测目标的位置与类别, 实时性高,但是检测精度稍低.

yolo将输入图像划分成SxS的方格,每个方格需要检测出中心点位于方格内的物体,在具体实施时, 每个方格会预测B个边界框.

5.3.2 yolov2

yolov2针对yolo的低召回率与低的定位精度进行改进.

- 卷积层后边添加BN,加快收敛速度,防止过拟合.
- yolov2的backbone在进行检测任务之前,先在高精度图像上finetune 10个batch.使得检测精度能够提前适应高分辨率的图像.
- yolov2直接在预设的锚框上提取特征. yolo使用cnn作为特征提取器, 然后加上全连接层来预测中心位置大小与i及置信度.yolov2借鉴faster rcnn的思想,使用cnn直接在锚框上预测偏移量以及置信度.方法简单易于学习.
- yolov2将输入的图像尺寸从448变成了416, 图片通常是以某个物体为中心,采用416的输入图像经过下采样后图像的尺寸为13(416/32 = 13), 特征图的尺寸为奇数,方便检测出物体的中心.
- yolov2还将26x26x512的特征图经过直通层变成13x13x2048与13x13x1024的特征图相结合一起进行目标检测.来提高小目标的识别精度.
- yolov2使用多尺度的图像进行训练,增强鲁棒性
- 采用DarkNet-19代替VGG.

5.3.3 yolo9000

- 使用检测数据集与分类数据集进行联合训练,检测数据集相对分类数据集来说数据量比较小,类别少,获取困难, 采用分类与检测数据集进行联合寻来你, yolo9000采用字典树,合并了ImageNet的分类标签与COCO的检测标签.

5.3.4 yolov3

- 由于图像的标注并不保证一定互斥(例如: 女人与人), 因此采用sigmoid的激活函数代替softmax函数.
- 采用DarkNet-53
- 利用不同大小的特征图进行联合的寻来你,在小物体上的检测效果也更加的优良.

5.4 如何增强模型对小目标的检测效果

- **模型设计方面**, 采用特征金字塔(FPN), 沙漏结构等子结构来增强对小尺度特征的感知能力;尽可能提升网络的感受野,使得网络能够更多的利用上下文信息;减少总的下采样的比例,使得最后的检测特征分辨率更高.
- **训练时**, 提高小样本物体的比例,采用数据增广手段,将图像缩小来生成小物体的样本.
- 采用**更大的输入图像的尺寸**