综述：目标检测中的多尺度检测方法

极市平台 2022-02-27 22:00:00 手机阅读

以下文章来源于AI算法修炼营，作者SFXiang

作者丨SFXiang

来源丨AI算法修炼营

编辑丨极市平台

极市导读

本文从**降低下采样率与空洞卷积、多尺度训练、优化Anchor尺寸设计、深层和浅层特征融合**等多个方面入手，对目标检测中的多尺度检测方法进行了全面概述，并介绍了多尺度检测相关方法。

前面的话

传统卷积网络通常采用从上到下的单行结构。**对于大物体而言，其语义信息将出现在较深的特征图中；而对于小物体，其语义信息出现在较浅的特征图中，随着网络的加深，其细节信息可能会完全消失。**

多尺度检测也是当今物体检测领域最为活跃的研究主题之一，本文主要介绍：**多尺度检测**。

多尺度是目标检测与图像分类两个任务的一大区别。分类问题通常针对同一种尺度，如ImageNet中的224大小；而目标检测中，模型需要**对不同尺度的物体都能检测出来**，这要求模型对于尺度要具有鲁棒性。

在多尺度的物体中，大尺度的物体由于面积大、特征丰富，通常来讲较为容易检测。难度较大的主要是**小尺度的物体**，而这部分小物体在实际工程中却占据了较大的比例。通常认为**绝对尺寸小于32×32的物体**，可以视为小物体或者**物体宽高是原图宽高的1/10以下**，可以视为小物体。

小物体由于其尺寸较小，可利用的特征有限，这使得其检测较为困难。当前的检测算法对于小物体并不友好，体现在以下4个方面：

1、过大的下采样率：假设当前小物体尺寸为15×15，一般的物体检测中卷积下采样率为16，这样在特征图上，过大的下采样率使得**小物体连一个像素点都占据不到**。

2、过大的感受野：在卷积网络中，特征图上特征点的感受野比下采样率大很多，导致在特征图上的一个点中，小物体占据的特征更少，会包含大量周围区域的特征，从而影响其检测结果。

3、语义与空间的矛盾：当前检测算法，如Faster RCNN，其Backbone大都是自上到下的方式，深层与浅层特征图在语义性与空间性上没有做到更好的均衡。

4、SSD一阶算法缺乏特征融合：SSD虽然使用了多层特征图，但浅层的特征图语义信息不足，没有进行特征的融合，致使小物体检测的结果较差。

多尺度的检测能力实际上体现了**尺度的不变性**，当前的卷积网络能够检测多种尺度的物体，很大程度上是由于其本身具有超强的拟合能力。

较为通用的提升多尺度检测的经典方法有：

降低下采样率与空洞卷积可以显著提升小物体的检测性能；**设计更好的Anchor**可以有效提升Proposal的质量；**多尺度的训练**可以**近似构建出图像金字塔**，增加样本的多样性；**特征融合**可以构建出特征金字塔，将浅层与深层特征的优势互补。下面将详细介绍：

1 降低下采样率与空洞卷积

对于小物体检测而言，降低网络的下采样率通常的做法是**直接去除掉Pooling层。**

例如，将原始的VGGNet-16作为物体检测的Backbone时，通常是将第5个Pooling层之前的特征图作为输出的特征图，一共拥有4个Pooling层，这时下采样率为16。为了降低下采样率，我们可

以将第4个Pooling层去掉，使得下采样率变为8，减少了小物体在特征图上的信息损失。

但是，**如果仅仅去除掉Pooling层，则会减小后续层的感受野**。如果使用预训练模型进行微调（Fine-tune），则仅去除掉Pooling层会使得后续层感受野与预训练模型对应层的感受野不同，从而导致不能很好地收敛。

因此，需要在去除Pooling的前提下增加后续层的感受野，**使用空洞卷积可以在保证不改变网络分辨率的前提下增加网络的感受野。**

需要注意的是，采用空洞卷积也不能保证修改后与修改前的感受野完全相同，但能够最大限度地使感受野在可接受的误差内。

2 多尺度训练

多尺度类似于数字图像处理中的图像金字塔，即将输入图片缩放到多个尺度下，每一个尺度单独地计算特征图，并进行后续的检测。这种方式虽然一定程度上可以提升检测精度，但由于多个尺度完全并行，耗时巨大。

多尺度训练（Multi Scale Training, MST）通常是指设置几种不同的图片输入尺度，**训练时从多个尺度中随机选取一种尺度，将输入图片缩放到该尺度并送入网络中**，是一种简单又有效的提升多尺度物体检测的方法。

虽然一次迭代时都是单一尺度的，但每次都各不相同，增加了网络的鲁棒性，又不至于增加过多的计算量。而在测试时，为了得到更为精准的检测结果，也**可以将测试图片的尺度放大，例如放大4倍，这样可以避免过多的小物体。**

多尺度训练是一种十分有效的trick方法，放大了小物体的尺度，同时增加了多尺度物体的多样性，在多个检测算法中都可以直接嵌入，在不要求速度的场合或者各大物体检测竞赛中尤为常见。

3 优化Anchor尺寸设计

现今较为成熟的检测算法大都采用Anchor作为先验框，如Faster RCNN和SSD等。模型在Anchor的基础上**只需要去预测其与真实物体边框的偏移即可**，可以说是物体检测算法发展中的一个相当经典的设计。

Anchor通常是多个不同大小与宽高的边框，这个大小与宽高是一组超参数，需要我们**手动配置**。在不同的数据集与任务中，由于物体的尺度、大小会有差距，例如行人检测的数据集中，行人标签宽高比通常为0.41，与通用物体的标签会有所区别，这时就需要相应地调整Anchor的大小与宽高。

如果Anchor设计的不合理，与数据集中的物体分布存在差距，则**会给模型收敛带来较大的困难，影响模型的精度，甚至不会收敛。**

另外，Anchor的设计对于小物体的检测也尤为重要，**如果Anchor过大，即使小物体全部在Anchor内，也会因为其自身面积小导致IoU低，从而造成漏检。**

通常来讲，可以从以下两个角度考虑如何设计一组好的Anchor。

1. 统计实验，手工设计

在Faster RCNN的RPN阶段，所有Anchor会与真实标签进行匹配，根据匹配的IoU值得到正样本与负样本，正样本的IoU阈值为0.7。在这个过程中，Anchor与真实标签越接近，正样本的IoU会更高，RPN阶段对于真实标签的召回率会越高，正样本也会更丰富，模型效果会更好。

因此，可以**仅仅利用训练集的标签与设计的Anchor进行匹配试验，试验的指标是所有训练标签的召回率，以及正样本的平均IoU值**。当然，也可以增加每个标签的正样本数、标签的最大IoU等作为辅助指标。

为了方便地匹配，在此不考虑Anchor与标签的位置偏移，而是把两者的中心点放在一起，仅仅利用其宽高信息进行匹配。这种统计实验实际是通过手工设计的方式，寻找与标签宽高分布最为一致的一组Anchor。

2．边框聚类

相比起手工寻找标签的宽高分布，也可以利用聚类的思想，在训练集的标签上直接聚类出一组合适的Anchor。由于一组Anchor会出现在特征图的每一个位置上，因此没有位置区别，可以只关注标签里的物体宽高，而没必要关心物体出现的位置。

边框聚类时通常使用K-Means算法，这也是YOLO采用的Anchor聚类方法。K-Means算法输入超参数K，即最终想要获得的边框数量，首先随机选取K个中心点，然后遍历所有的数据，并将所有的边框划分到最近的中心点中。在每个边框都落到不同的聚类后，计算每一个聚类的平均值，并将此平均值作为新的中心点。重复上述过程，直到算法收敛。

在聚类过程中，**Anchor的数量K是一个较为重要的超参**，数量越多，精度越高，但与此同时会带来计算量的增加。对于使用Anchor的物体检测算法而言，设计一组好的Anchor是基础，这对于多尺度、拥挤等问题都有较大的帮助。

4 深层和浅层特征融合

传统的卷积网络通常是自上而下的模式，随着网络层数的增加，感受野会增大，语义信息也更为丰富。这种自上而下的结构本身对于多尺度的物体检测就存在弊端，尤其是小物体，其**特征可能会随着深度的增加而渐渐丢失，从而导致检测性能的降低。**

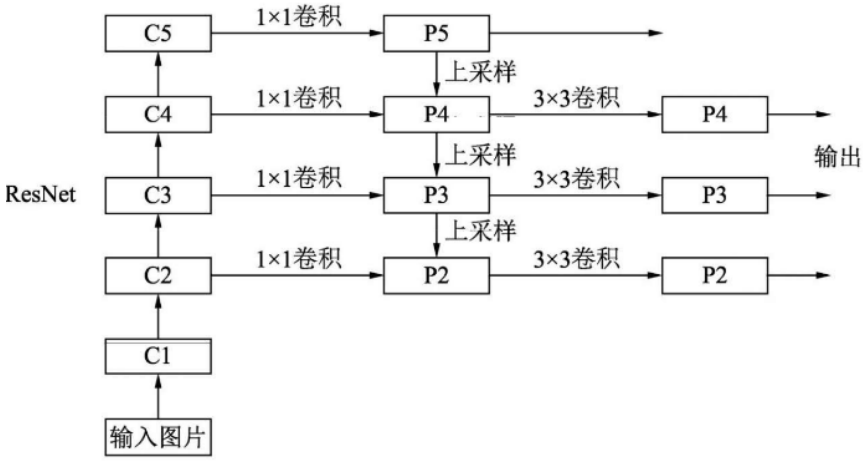
可以**将深层的语义信息添加到浅层的特征图中**，融合两者的特征，优势互补，从而提升对于小物体的检测性能。

特征融合有多种方式，**增大特征图尺寸**可以使用上采样、反卷积等，**融合方法**有逐元素相加、相乘和通道拼接等，具体哪种效果更好，还要看实际的检测任务及使用的检测算法。特征融合的普遍缺点是通常**会带来一定计算量的增加**。

特征融合方法示例：

1. FPN（Feature Pyramid Network）

将深层信息上采样，与浅层信息逐元素地相加，从而构建了尺寸不同的特征金字塔结构，性能优越，现已成为目标检测算法的一个标准组件。FPN的结构如下所示。



自下而上：最左侧为普通的卷积网络，默认使用ResNet结构，用作提取语义信息。C1代表了ResNet的前几个卷积与池化层，而C2至C5分别为不同的ResNet卷积组，这些卷积组包含了多个Bottleneck结构，组内的特征图大小相同，组间大小递减。

自上而下：首先对C5进行1×1卷积降低通道数得到P5，然后依次进行上采样得到P4、P3和P2，目的是得到与C4、C3与C2长宽相同的特征，以方便下一步进行逐元素相加。这里**采用2倍最邻近上采样，即直接对临近元素进行复制，而非线性插值。**

横向连接（Lateral Connection）：目的是为了**将上采样后的高语义特征与浅层的定位细节特征进行融合。**高语义特征经过上采样后，其长宽与对应的浅层特征相同，而通道数固定为256，因此需要对底层特征C2至C4进行11卷积使得其通道数变为256，然后两者进行逐元素相加得到P4、P3与P2。**由于C1的特征图尺寸较大且语义信息不足，因此没有把C1放到横向连接中。**

卷积融合：在得到相加后的特征后，利用3×3卷积对生成的P2至P4再进行融合，目的是**消除上采样过程带来的重叠效应，以生成最终的特征图。**

FPN对于不同大小的RoI，使用不同的特征图，大尺度的RoI在深层的特征图上进行提取，如P5，小尺度的RoI在浅层的特征图上进行提取，如P2。**FPN的代码实现如下：**

```
import torch.nn as nn
import torch.nn.functional as F
import math

class Bottleneck(nn.Module):
    expansion = 4
    def __init__(self, in_planes, planes, stride=1,
downsample=None):
        super(Bottleneck, self).__init__()
        self.bottleneck = nn.Sequential(
                nn.Conv2d(in_planes, planes, 1, bias=False),
                nn.BatchNorm2d(planes),
                nn.ReLU(inplace=True),
                nn.Conv2d(planes, planes, 3, stride, 1, bias=False),
                nn.BatchNorm2d(planes),
                nn.ReLU(inplace=True),
                nn.Conv2d(planes, self.expansion * planes, 1,
bias=False),
                nn.BatchNorm2d(self.expansion * planes),
            )
        self.relu = nn.ReLU(inplace=True)
        self.downsample = downsample
    def forward(self, x):
        identity = x
```

```
        out = self.bottleneck(x)
        if self.downsample is not None:
            identity = self.downsample(x)
        out += identity
        out = self.relu(out)
        return out

class FPN(nn.Module):
    def __init__(self, layers):
        super(FPN, self).__init__()
        self.inplanes = 64
        self.conv1 = nn.Conv2d(3, 64, 7, 2, 3, bias=False)
        self.bn1 = nn.BatchNorm2d(64)
        self.relu = nn.ReLU(inplace=True)
        self.maxpool = nn.MaxPool2d(3, 2, 1)

        self.layer1 = self._make_layer(64, layers[0])
        self.layer2 = self._make_layer(128, layers[1], 2)
        self.layer3 = self._make_layer(256, layers[2], 2)
        self.layer4 = self._make_layer(512, layers[3], 2)
        self.toplayer = nn.Conv2d(2048, 256, 1, 1, 0)

        self.smooth1 = nn.Conv2d(256, 256, 3, 1, 1)
        self.smooth2 = nn.Conv2d(256, 256, 3, 1, 1)
        self.smooth3 = nn.Conv2d(256, 256, 3, 1, 1)

        self.latlayer1 = nn.Conv2d(1024, 256, 1, 1, 0)
        self.latlayer2 = nn.Conv2d( 512, 256, 1, 1, 0)
        self.latlayer3 = nn.Conv2d( 256, 256, 1, 1, 0)

    def _make_layer(self, planes, blocks, stride=1):
        downsample  = None
        if stride != 1 or self.inplanes != Bottleneck.expansion *
planes:
            downsample  = nn.Sequential(
                nn.Conv2d(self.inplanes, Bottleneck.expansion *
planes, 1, stride, bias=False),
                nn.BatchNorm2d(Bottleneck.expansion * planes)
            )
        layers = []
```

```
        layers.append(Bottleneck(self.inplanes, planes, stride,
downsample))
        self.inplanes = planes * Bottleneck.expansion
        for i in range(1, blocks):
            layers.append(Bottleneck(self.inplanes, planes))
        return nn.Sequential(*layers)

    def _upsample_add(self, x, y):
        _,_,H,W = y.shape
        return F.upsample(x, size=(H,W), mode='bilinear') + y

    def forward(self, x):

        c1 = self.maxpool(self.relu(self.bn1(self.conv1(x))))
        c2 = self.layer1(c1)
        c3 = self.layer2(c2)
        c4 = self.layer3(c3)
        c5 = self.layer4(c4)

        p5 = self.toplayer(c5)
        p4 = self._upsample_add(p5, self.latlayer1(c4))
        p3 = self._upsample_add(p4, self.latlayer2(c3))
        p2 = self._upsample_add(p3, self.latlayer3(c2))

        p4 = self.smooth1(p4)
        p3 = self.smooth2(p3)
        p2 = self.smooth3(p2)
        return p2, p3, p4, p5
```

2. DetNet：专为目标检测而生的Backbone，**利用空洞卷积与残差结构，使得多个融合后的特征图尺寸相同，从而也避免了上采样操作。**

3. Faster RCNN系列中，HyperNet将第1、3、5个卷积组后得到的特征图进行融合，**浅层的特征进行池化、深层的特征进行反卷积，**最终采用**通道拼接**的方式进行融合，优势互补。

4. SSD系列中，DSSD在SSD的基础上，**对深层特征图进行反卷积，与浅层的特征相乘**，得到了更优的多层特征图，这对于小物体的检测十分有利。

5. **RefineDet将SSD的多层特征图结构作为了Faster RCNN的RPN网络**，结合了两者的优点。特征图处理上与FPN类似，**利用反卷积与逐元素相加，将深层特征图与浅层的特征图进行结合**，实现了一个十分精巧的检测网络。

6. YOLO系列中，YOLO v3也使用了特征融合的思想，通过上采样与通道拼接的方式，最终输出了3种尺寸的特征图。

5 SNIP：尺度归一化

论文地址：
https://arxiv.org/abs/1711.08189

代码地址：
https://github.com/mahyarnajibi/SNIPER

当前的物体检测算法通常使用微调的方法，即先在ImageNet数据集上训练分类任务，然后再迁移到物体检测的数据集上，如COCO来训练检测任务。我们可以将ImageNet的分类任务看做224×224的尺度，而COCO中的物体尺度大部分在几十像素的范围内，并且包含大量小物体，物体尺度差距更大，因此两者的样本差距太大，会导致**映射迁移（Domain Shift）**的误差。

SNIP是多尺度训练（Multi-Scale Training）的改进版本。MST的思想是使用随机采样的多分辨率图像使检测器具有尺度不变特性。然而作者通过实验发现，在MST中，**对于极大目标和过小目标的检测效果并不好**，但是MST也有一些优点，比如对一张图片会有几种不同分辨率，每个目标在训练时都会有几个不同的尺寸，那么总有一个尺寸在指定的尺寸范围内。
SNIP的做法是**只对size在指定范围内的目标回传损失**，即训练过程实际上**只是针对某些特定目标进行**，这样就能减少domain-shift带来的影响。

SNIP的网络结构如下图所示：

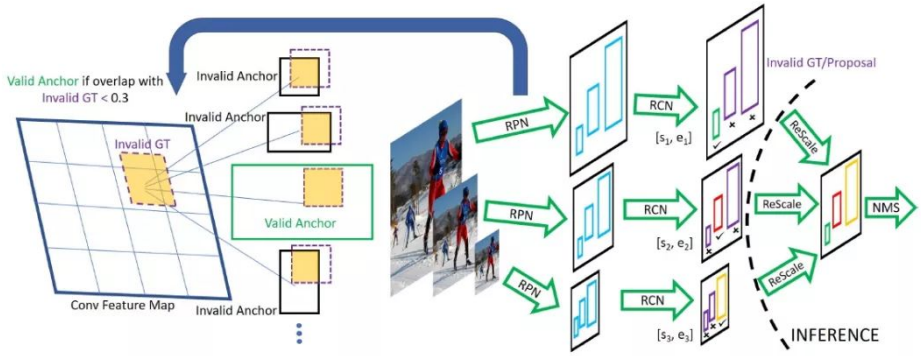


Figure 6. SNIP training and inference for IPN is shown. Invalid RoIs which fall outside the specified range at each scale are shown in purple. These are discarded during training and inference. Each batch during training consists of images sampled from a particular scale. Invalid GT boxes are used to invalidate anchors in RPN. Detections from each scale are rescaled and combined using NMS.

具体的实现细节

（1）**3个尺度分别拥有各自的RPN模块**，并且各自预测指定范围内的物体。

（2）对于**大尺度**的特征图，其RPN**只负责预测被放大的小物体**，对于**小尺度**的特征图，其RPN**只负责预测被缩小的大物体**，这样真实的物体尺度分布在较小的区间内，避免了极大或者极小的物体。

（3）在RPN阶段，如果真实物体不在该RPN预测范围内，会被判定为无效，并且与该无效物体的IoU大于0.3的Anchor也被判定为无效的Anchor。

（4）在训练时，**只对有效的Proposal进行反向传播**。在测试阶段，**对有效的预测Boxes先缩放到原图尺度，利用Soft NMS将不同分辨率的预测结果合并。**

（5）实现时SNIP采用了**可变形卷积**的卷积方式，并且为了降低对于GPU的占用，将原图随机裁剪为1000×1000大小的图像。

总体来说，SNIP**让模型更专注于物体本身的检测**，剥离了多尺度的学习难题。在网络搭建时，SNIP也使用了类似于MST的多尺度训练方法，构建了3个尺度的图像金字塔，但在训练时，只对指定范围内的Proposal进行反向传播，而忽略掉过大或者过小的Proposal。

SNIP方法虽然实现简单，但其背后却蕴藏深意，更深入地分析了当前检测算法在多尺度检测上的问题所在，在训练时只选择在一定尺度范围内的物体进行学习，在COCO数据集上有3%的检测精度提升，可谓是大道至简。

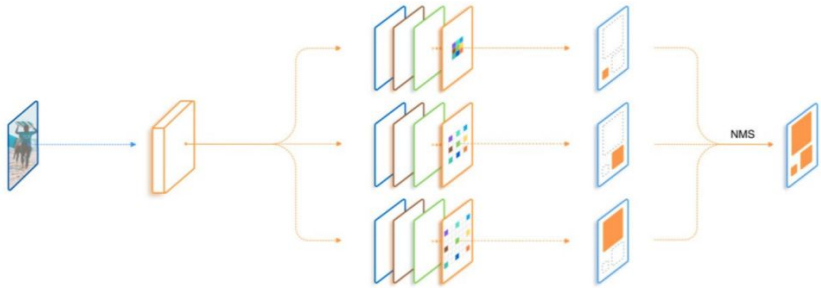
6 TridentNet：三叉戟网络

论文地址：
https://arxiv.org/abs/1901.01892

代码地址：
https://github.com/TuSimple/simpledet/tree/master/models/tridentnet

传统的解决多尺度检测的算法，大都依赖于图像金字塔与特征金字塔。与上述算法不同，图森组对**感受野**这一因素进行了深入的分析，并利用了**空洞卷积**这一利器，构建了简单的三分支网络TridentNet，对于多尺度物体的检测有了明显的精度提升。

TridentNet网络的作者**将3种不同的感受野网络并行化**，提出了如下图所示的检测框架。采用ResNet作为基础Backbone，前三个stage沿用原始的结构，在第四个stage，使用了三个感受野不同的并行网络。



具体实现细节

（1）3个不同的分支**使用了空洞数不同的空洞卷积，感受野由小到大**，可以更好地覆盖多尺度的物体分布。

（2）由于3个分支要检测的内容是相同的、要学习的特征也是相同的，只不过是形成了不同的感受野来检测不同尺度的物体，因此，**3个分支共享权重**，这样既充分利用了样本信息，学习到更本质的目标检测信息，也减少了参数量与过拟合的风险。

（3）借鉴了SNIP的思想，**在每一个分支内只训练一定范围内的样本**，避免了过大与过小的样本对于网络参数的影响。

在训练时，TridentNet网络的三个分支会接入三个不同的head网络进行后续损失计算。在测试时，由于没有先验的标签来选择不同的分支，因此只保留了一个分支进行前向计算，这种前向方法只有少量的精度损失。

具体细节可以参考论文。

参考资料

1.深度学习之PyTorch物体检测实战 董洪义

2.https://link.zhihu.com/?target=https%3A//arxiv.org/abs/1901.01892

3.https://zhuanlan.zhihu.com/p/74415602

4.https://zhuanlan.zhihu.com/p/61536443

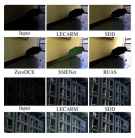
喜欢此内容的人还喜欢

YOLOv5帮助母猪产仔？南京农业大学研发母猪产仔检测模型并部署到Jetson Nano开发板

极市平台

ICCV23｜将隐式神经表征用于低光增强，北大张健团队提出NeRCo

极市平台



ICCV 2023｜南开程明明团队提出适用于SR任务的新颖注意力机制（已开源）

极市平台