目标检测

2022年5月13日 8:59

目标检测

项目背景：

这项工作主要是对于高分辨率的遥感图像中包含的军、民飞机以及多种类型的舰船进行检测。

批注 [李1]: 军民飞机类型，舰船类型?

基线问题:

遥感图像与自然图像相比还是有挺大区别的，因此针对具体的检测任务，如果直接迁移在自然图像中表现比较好的模型到遥感图像上，是很难达到令人满意的检测效果的。就比如我们起初使用了 SSD 目标监测框架对遥感图像进行检测，但 mAP 只能达到 60%多，存在比较严重的小型飞机以及密集舰船漏检的问题。因此，针对于以上两个大的问题，我们对模型进行了几点优化。

批注 [李2]: 遥感与自然图像的区别?

批注 [李3]: 迁移学习?

批注 [李4]: SSD 原理?

批注 [李5]: mAp 计算方式?

痛点问题 1（小目标）:

针对小目标问题，我们在 SSD 的多层检测特征图之间添加了 FPN 策略，为负责检测小目标的浅层特征图添加更多的语义信息，但最后的检测效果依旧不太理想，小飞机的检测性能确实有所提升，但又出现了大目标的重复检测问题。

批注 [李6]: FPN 原理?

痛点问题 2（尺度差异大）:

通过分析，我们发现不能像自然图像一样只关注小目标的检测性能提升，对于高分辨遥感图像来说，它的主要问题在于同一复杂场景下目标尺寸差异较大的问题。（举个例子，就比如操场旁边的一辆小汽车，或者我们所用的数据集中大舰船旁边的小舰船）同一场景下，大小尺寸的目标同时存在，网络通常会对提取的大目标特征产生比较强的响应，而小目标的响应相对来说弱一些，那么在网络学习过程中，就会更倾向于关注大尺寸目标，而忽略小目标的捕获。

解决方案 2:

针对目标尺度差异过大的问题，我们在网络中添加了一种特征分离再合并策略。这个策略的作用在于解决上述大小目标特征混淆问题。它将负责检测小目标的浅层特征图中包含的大目标特征擦除掉，只保留小目标信息，使得网络能更加的关注浅层中小目标特征的学习，而将擦除的这部分大目标信息，我们会向后传递到用于检测大目标的深层特征图中，弥补深层特征图细节信息不足的缺陷。

实施细节 2:

接下来我讲一下具体的擦除操作是怎么实现的。我们首先通过最大和平均池化求得一张掩膜来反应深层特征图的特征分布情况，再将这张掩膜取反后（1-mask）作用于浅层特征图，利用这种方式消除掉浅层特征图中比较显著的大尺寸目标信息（因为随着网络的加深，小目标的特征以及位置信息逐渐减少甚至消失）。

批注 [李7]: 最大池化，平均池化?

在特征传递阶段，我们依旧利用深层特征图的掩膜作用于较浅层的特征图，这次通过点乘的

批注 [李8]: PyTorch 元素级相乘等操作实现

方式去强调浅层特征图中的大目标的强响应信息，再将这些信息融合到深层特征图上，增强深层特征图的细节信息表达能力。

通过这种方式之后，浅层特征图就主要聚焦于小目标特征，而深层聚焦于大目标特征。这样就让浅层和深层特征图各司其职，避免不必要的特征混淆问题。

痛点问题 3（位置偏差）：

在上述的特征分离在合并策略中，我们通过深层特征图上采样后再池化求掩膜的方式来传递或者消除信息。这就需要相邻特征图之间的空间位置信息分布尽可能的对应，但只采用简单的上采样是难以实现特征对齐的。

批注 [李9]: 上采样方式?

解决方案 3：

因此，针对上一个策略可能存在的位置偏差问题，我们引入了一些监督信息在网络的损失函数部分做了一些优化。

实施细节 3：

具体来说，我们获取了 GT 中所有标定框的中心坐标集合(x,y)，接着获取了检测特征图中可能存在于目标区域的关键点坐标集合（a,b），关键点的获取方式是：首先对特征图进行激活获得一张热力图，对于热力图中的像素点 A,如果它的值大于它的八领域的值，我们就将它认定为关键点。对应于 GT 中的中心点，我们将所有特征图求得的关键点聚合在一起，目的在于计算两者的偏差。

批注 [李10]: GT 标定格式

批注 [李11]: 激活函数选择?

除此之外我们利用卷积在每个待检测特征图上预测一组位置偏差（deltax,deltay），最后利用预测偏差，关键点集合(a,b)以及 GT 中心点集合（x,y）构造损失函数，利用 GT 与关键点之间的真实偏差作为监督信号对预测偏差进行牵引，从而实现对相邻特征图之间的位置偏差的校正。实现有监督的信息消除与传递。

痛点问题 4（密集目标）：

最后，我们还面临目标密集排列的问题（港口处的船）。由于目标密集排列，网络提取的特征存在边界模糊的问题，这就导致了目标的漏检。

解决方案 4：

针对这个问题，我们在网络中添加了一个目标增强策略，目的在于实现前景和背景的有效分离。

实施细节 4：

具体来说，对于每一幅特征图我们会先计算一个它的背景值，初始的时候，我们是利用特征图边缘像素点的值求和再平均来求得这个背景值（因为图像的边界像素点一般是不包含目标），之后，求特征图中每个像素点的特征值与背景值的差值，这个差值越大，证明该像素点表示目标区域的可能性更大。我们定义了一个隶属度函数来表示这种正相关的映射关系，利用这个函数我们可以获得一张掩膜，将这张掩膜作用于原始特征图，从何实现前景与背景的区分。

批注 [李12]: PyTorch 自定义卷积核

最终结果：

最后，对与增强的多尺度特征图，我们采用了与 SSD 相同的检测头（也就是利用卷积核实现类别预测和边界回归）进行最终的目标检测。
mAP 最终达到 95%。目标检测器对同一场景下尺度差异较大的目标的检测性能达到了较好的平衡，对于密集目标的检测能力也进一步提升。

批注 [李13]: SSD 检测头

优化方向：

我们这套框架其实采用的依旧是比较传统的水平框，这也是密集目标检测率低的一个原因，我们这里是利用很传统的方式对特征进行精细的处理来达到较好的检测效果。如果我们用带有角度回归的斜框又或者任意四边形的那种框来标定的话，效果可能会更加好一些。

批注 [李14]: 了解斜框

- **飞机类型、船舶类型**

  形状差异小，更小 战斗机（F-18、F-22、F-16）；灰色 更大 运输机（C-5）；灰色 运输机（C-17）

  军用飞机：F-18，F-22，F-16，C-5，C-17

  民用：Boeing-737，Boeing-707，Boeing-767，Boeing-787

  船舶：大型船舶，密集船舶.

- **遥感图像与自然图像的区别**

  - 尺寸多样性：取决于遥感成像设备的高度及成像时的比例选取.
  - 成像范围更广泛：遥感图像内目标密度分布不均匀，图像内目标密集排列 or 不包含目标，图像背景更复杂　（对于自然图像来说，目标数量通常不会太多）
  - 成像视角更特殊：遥感图像成像视角为俯视或斜下视，角度不同目标的形状也会有较大差异.　（自然图像为正视 or 侧视，目标形状不会有太大差别）
  - 方向多样性：遥感图像从高空中对目标成像，目标姿态可以任意旋转　（自然图像在地面拍摄图片，目标的方位差异不会很大）
  - 图像通道数：自然图像通常为RGB三通道，遥感图像通道数由成像设备决定，如多光谱、高光谱和SAR图像等.

- **迁移学习**

  - 迁移学习：迁移学习就是从相似性出发，将旧领域学习过的模型应用到新领域上.

- 为什么需要迁移学习：

  ①. 大数据与少标注的矛盾：数据大量存在，但大部分都没有标注，人工标注太耗时.

  ② 大数据与弱计算的矛盾：普通人无法拥有充足的计算资源，因此需要借助模型迁移

  ③. 普适化模型与特定需求的矛盾：即使在同一个任务上，一个模型也难以满足不同人的需求，因此需要在不同的人之间做模型的适配

- 迁移学习与多任务学习：

  多任务学习：多个相关任务一起协同学习

  迁移学习：强调信息复用，从一个领域迁移到另一个领域.

- 负迁移

  - 负迁移含义：从源域上学习到的知识，对目标域上的学习产生负面作用

  - 产生负迁移的原因：

    ①数据问题：目标域与源域的数据分布不相似

    ②方法问题：目标域与源域相似，但迁移方法选择不恰当，没有找到可迁移的成分.

- 迁移学习基本思路:
  - 基于样本的迁移学习方法: 根据一定的权重生成规则对源域中的数据进行重用
    - 对源域与目标域中的数据分布比值进行估计，所得到的比值即为样本权重。
  - 基于特征的迁移学习方法: 通过特征变换的方式互相迁移来减少目标域与源域之间的差距 or 将目标域与源域的数据特征统一到同一特征空间中，再利用传统的机器学习方法进行识别。
  - 基于模型的迁移学习方法: 从源域与目标域中找到它们之间共享的参数信息，以实现迁移。
  - 基于关系的迁移学习方法: 更加关注目标域与源域的样本之间的关系，可以通过马尔科夫逻辑网络来挖掘不同领域之间的关系相似性
- 深度神经网络中的微调 (Fine tune)
  - Fine tune: 利用别人已经训练好的网络，针对自己的任务再进行调整。
  - 为什么需要已经训练好的模型:
    - ① 耗时: 在实际任务中，我们通常不会选择去从头开始训练一个神经网络，很耗时
    - ② 数据少: 我们的数据可能不会像ImageNet那样大，可以训练出泛化能力很强的深度神经网络。
  - 为什么要 fine tune: 别人训练好的模型不一定完全适用于我们的任务（数据分布不一致，参数不同等）

- finetune的优势:

    ① 不需要针对新任务,从头开始训练网络,节省了时间成本

    ② 预训练模型通常都是在大型数据集上训练上,使得模型更鲁棒,泛化性能更好

- mAP

mAP即 mean AP,不同类别样本AP的均值,p-R曲线下的面积即为AP.(precision为纵轴,recall为横轴)

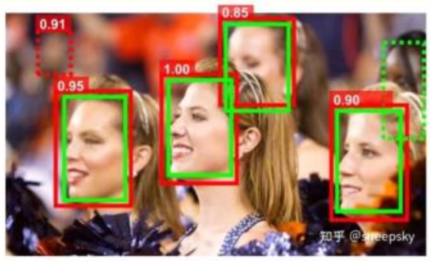
- 目标检测中的Tp, TN, Fp, FN.

    - Tp: 与GT的IoU大于阈值, confidence>阈值, 与GT的IoU最大(对于某个目标来说,没有比本框更好的框了)

    - Fp: 检测框里没有目标　与GT的IoU<阈值　有一个bbox但没有目标.

    - TN: 没有目标的位置没有框.

    - FN: 有目标的位置没有框, 此时的GT相当于一个FN



绿色实线或虚线框:人脸的真实标注

红色实线或虚线框:输出的检测结果

红色数字:检测结果为人脸的概率

参考上图与我们定义的概念,我们可以得知四个红色实线框将贡献TP值,红色虚线框贡献FP值,绿色虚线框则贡献FN值。当然你会发现TN值难以度量,但其实我们并不需要它。

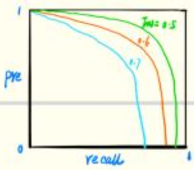
- precision 和 recall

$$precision = \frac{TP}{TP+FP} = \frac{TP}{all\ detections}$$

$$recall = \frac{TP}{TP+FN} = \frac{TP}{all\ ground\ truths}$$

通过设定不同的 confidence 阈值，一些检测出界框会由 TP 变为 FN，因此 confidence 阈值设定不同，得到的 precision 和 recall 也不同。

- P-R曲线



precision 和 recall 是鱼与熊掌不可兼得的关系。

- mAP 计算

- AP：每个类别的 AP 即为每个类别 P-R 曲线下面的面积。

- mAP：$mAP = \frac{\sum AP}{n}$ 不同类别的 AP 的平均值。

VOC 格式数据集计算 mAP 的编码中：将关于某一类检测出的所有边界框的 confidence 分别作为阈值来计算 precision 和 recall。e.g：在 1000 张图片里检测了 30000 个 bbox，则将这 30000 个 bbox 的 confidence 分别作为阈值来计算 precision 和 recall，从而使得 P-R 曲线更精细。

将 recall 的 [0,1] 区间分为 11 个区间段 [0,0.1]，[0.1,0.2]…[0.9,1.0] 等，从每个区间算出最大的 precision，再求和求平均得到 AP。（面积的粗略估计）

- **最大池化和平均池化.**

  - max-pooling: 更多的保留纹理信息　　由于图像包含较多的噪音和与目标处理无关的信息，因此在网络前几层用Max-pooling去除无用信息.

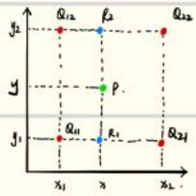
  - avg-pooling: 更多的保留背景信息.　　需要综合图像信息时，常用avg-pooling，网络的最后几层用avg-pooling

- **上采样方法.**

  - 双线性插值

双线性插值的本质是在两个方向上进行线性插值.

若想求得p点处的插值，则 ① Q_{11}, Q_{21} 在x方向插值求得 R_1，Q_{12}, Q_{22} 在x方向差值求得 R_2



② R_1, R_2 在y轴方向插值，求得p处的值.

①. 在x方向上: $f(R_1) = \frac{x - x_1}{x_2 - x_1} f(Q_{21}) + \frac{x_2 - x}{x_2 - x_1} f(Q_{11})$　　$f(R_2) = \frac{x - x_1}{x_2 - x_1} f(Q_{22}) + \frac{x_2 - x}{x_2 - x_1} f(Q_{12})$

②. 在y方向上: $f(p) = \frac{y - y_1}{y_2 - y_1} f(R_2) + \frac{y_2 - y}{y_2 - y_1} f(R_1)$

- **Voc数据集标注格式.**

文件夹: Annotations , Imagesets , JPEGImages

- Annotations → 图片.xml.
- Imagesets → Main → test.txt, val.txt. → 图像名称
- JPEGImages → 原始的图片.

标签: <bndbox> xmin, ymin, xmax, ymax <bndbox>   <name> airplane <name>.

- **Pytorch 自定义卷积核.**

函数: torch.nn.functional.conv2d( )   (原本为 torch.nn.Conv2d( ))

F.conv2d( ) (是一个函数.) 的 weight 可以自己设定，它的参数既可以写死不让求导，也可以写成可学习参数，在反向传播过程中进行优化.

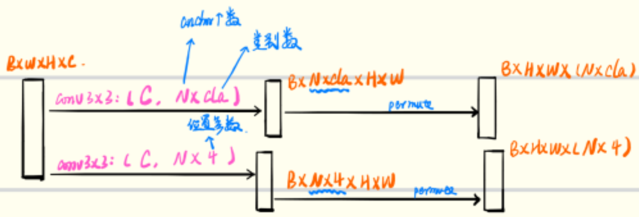
数据采用 torch.tensor格式.

与 nn.Conv2d( ) 相同，也有 input_channel, out_channel, 卷积核大小, 卷积核步长, 可以很方便的写成一个卷积层.

F.conv2d(input, filter, bias, stride, padding, dilation, groups)

- input: [B, C, H, W]
- filter: [out_c, in_c, H, W]
- dilation: 空洞率: 卷积核内部每个元素之间间隔的元素数目.

- **SSD 检测头.**



两个卷积分支，一个用于分类，一个用于边界框位置回归.

- 斜框

旋转同标检测：从图像中定位出具有旋转角度的用标。

①. 一种是带角度的边界框 (x, y, w, h, θ)

②. 四边边界框的4个顶点坐标 $(x_1, y_1, x_2, y_2, \cdots x_4, y_4)$

- anchor-free

论文笔记(object as point).

目标检测项目问题

2022年7月4日 20:53

1. 基线网络是怎么选择的（Faster R-CNN，SSD，YOLO）
    - Anchor-Free
        - Center-Net：预测中心点+宽+高
        - 原始图像输入全卷积网络得到一个关键点估计热力图
        - 训练阶段，GT通过按比例缩放以及高斯核处理也得到了一个GT中心点热力图（但热力图在生成过程中，如果遇到两个高斯生成点重合的情况，只选择保留一个）
        - 这种策略对于密集目标来说是不友好的，如果两个目标的关键点重合只保留一个
    - YOLO
        - 每个cell（格子）只能拥有一个CLASS和两个BBOX，这个空间局限性，使得对小物体检测效果不好（尤其是密集的小物体）
        - 体量大
    - Faster R-CNN
        - 双阶段网络，检测精度相比单阶段高，但检测时只利用了最高层Feature maps，利用FC层进行检测，对图像输入有要求
        - 速度慢
    - SSD
        - 对一张图片，SSD采用金字塔结构，结合多个大小不同的feature map的预测结果，能够处理大小不同的物体
        - 运行速度可以和YOLO媲美，检测精度可以和Faster R-CNN媲美
        - 对小目标的recall依然一般，并没有达到碾压Faster R-CNN的级别
            - SSD使用conv4_3低级feature去检测小目标，而低级特征卷积层数少，存在特征提取不充分的问题
2. 输入图像尺寸
    - 原始图形的尺寸800-1000不等
    - 输入网络是压缩到400（设备原因，太大就会爆显存）
3. 目标尺寸
    - 小目标压缩后：15pixel 以下（<0.12%）（0.09%）（144/160000=0.09%）
    - 大目标压缩后：100pixel 以内（大舰船）
4. 检测阶段用了几个尺寸特征图
    - 8倍下采样：50（VGG16最后一层输出是8倍下采样）
    - 16倍下采样：25
    - 32倍下采样：13
5. 先验框设置
    - 大小：15，55，130
    - 纵横比：0.65，1，2，1/2
6. 超参设置
    - 训练集：1000
    - Batch-size：16
    - Epoch：20-30
7. 每个尺寸上的检测目标的大小是怎么定义的
    - 没有固定定义每个待检测特征图，用于检测多大的目标
    - 而是利用浅层与深层目标特征的分布状况来确定
8. 分离再合并是用于每一层特征图
    - 在待检测的三个尺寸特征图上进行
    - S1的消除利用了S2和S3
    - S3是最后一层特征，因此不需要消除操作

    - S1是第一层特征不需要消除操作
    - S2既包含消除又包含传递操作
9. 项目里的方法是其他论文里的，还是自己的
    - 首先就是基线在我们自己的数据集上训练去发现问题
    - 结合问题也看了相关文献，但大部分关于自然图像，遥感数据集较少
    - 会借鉴一些比较有效的思想和方法，但具体的改进措施还是我们根据自己的数据集的特性，想出来的
10. 数据标注
    - 标注工具：LabelImg（VOC数据格式）
    - 标注格式：水平框
11. 推理速度
    - SSD：1秒 100张左右
    - OURS：一秒 50张左右
12. 数据集多大
    - 总共：1700多张图片
    - 训练集：700多张
    - 验证集：300多张
    - 测试集：300多张