**目录**

第一章 绪论

1.1 研究背景及意义

1.2 本文主要工作

1.3 论文结构

第二章 图像语义分割技术研究现状

2.1 基于低级特征提取的图像分割方法

2.2 基于卷积网络的语义分割方法

2.3 基于区域选择的分割算法改进

2.4 本章小结

第三章 图像语义分割的基本算法实现

3.1 方法步骤流程

3.2 基于labelme的图像数据预处理

3.3 基于YOLACT的实时实例分割

3.3.1 核心算法与工作原理

3.3.2 数据增强代码与分析

3.3.3 训练过程参数的影响

第四章 图像语义分割的实际应用与细节优化

4.1 针对嵌套圆管的数据集制作

4.2 实验环境和参数设置

4.3 应用过程中的细节优化

4.4 实验结果展示与分析

第五章 总结与展望

5.1 工作总结

5.2 未来展望

1. **绪论**

本章探讨本文的研究背景及意义，对本文工作内容和论文结构进行简要的介绍。

* 1. **研究背景及意义**

二十一世纪是人工智能的时代，人工智能也被认为是第四次工业革命，许多全球顶尖的技术公司都把目光转向人工智能。人工智能，即模仿生物神经网络的原理建造实用的人工神经网络模型，设计相应的学习算法，模拟人脑的某些智能活动，比如视觉、听觉，以及对所视、所听的理解，甚至于思考并作出反应。借此，计算机视觉这一研究方向应运而生。

那么，人类是如何描述场景的呢？我们可能会说“窗户下有一张桌子”，或者“沙发右边有一盏灯”。图像理解的关键在于将一个整体场景分解成几个单独的实体，这也有助于我们推理目标的不同行为。

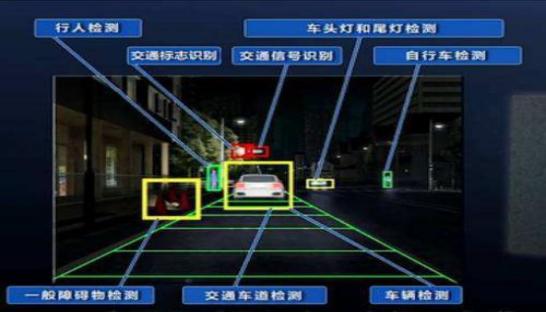
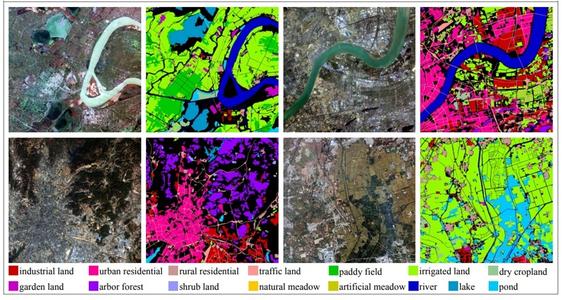
在计算机视觉领域，目前神经网络的应用主要包括图像识别、目标定位与检测以及语义分割。图像识别告诉你图像是什么，目标定位与检测告诉你图像中的目标在哪里，而语义分割则是让计算机从像素级别回答上述两个问题。虽然目标定位与检测可以帮助我们绘制某些确定实体的边框，但人类对场景的理解远远不止于此，我们能以像素级的精细程度对每一个实体进行检测并标记精确的边界，因此计算机仅仅做到识别、定位与检测是不够的，根据图像语义精确分割出实例变得越来越重要。

语义分割是计算机视觉中的基本任务，在语义分割中我们需要将视觉输入分为不同的语义可解释类别，“语义的可解释性”即分类类别在真实世界中是有意义的。例如，我们可能需要区分图像中属于汽车的所有像素，并把这些像素涂成同一种颜色，如图1-1所示。

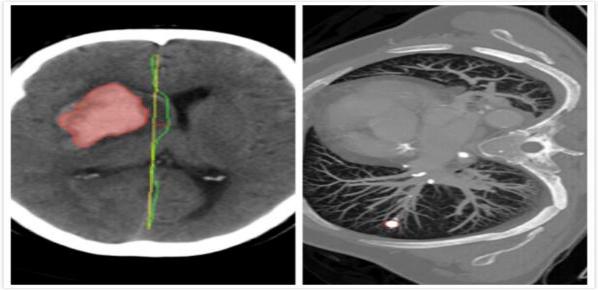


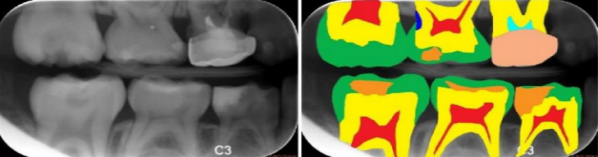
图1-1 语义分割演示视频截图

由于图像、视觉与人们生活息息相关，所以语义分割在现代生活中应用十分广泛。如图1-2（a）所示，在地理信息系统中，通过训练神经网络让机器输入卫星遥感影像，自动识别道路，河流，庄稼，建筑物等，并且对图像中每个像素进行标注；如图1-2（b）所示，在无人驾驶交通中，车载摄像头，或者激光雷达探查到图像后输入到神经网络中，后台计算机可以自动将图像分割归类，以避让行人和车辆等障碍；如图1-2（c）所示，在医疗影像分析中，通过对图像的语义分割诊断肿瘤、龋齿等不良生理现象。



（a）地理信息系统 （b）无人驾驶





（c）医疗影像分析

图1-2 语义分割在现代生活中的应用

由此可见，随着越来越多的领域以及应用程序通过从图像中推断知识来获取“营养”，“对场景的理解”逐渐成为一个核心的计算机视觉问题。而从宏观上看，在计算机视觉的应用中，语义分割相当于解决整个问题的预处理阶段，是实现场景完整理解的基础。可谓是，没有正确的分割就不可能有正确的识别。因此，语义分割是当今计算机视觉领域的关键问题之一。

语义分割是图像处理中的一项关键技术，也是一个经典难题，发展至今仍没有找到一个比较通用的方法，也没有制定出判断分割算法好坏的标准。第二章中将对近年来出现的多种语义分割方法进行较为全面的介绍，探讨语义分割技术的发展方向。

近十年以来，人工神经网络识别技术已经引起了广泛的关注，并且大量应用于语义分割领域。现有的语义分割体系结构通常是一个编码器网络和一个解码器网络。编码器通常是一个预先训练好的分类网络，如VGG或者ResNet等，其作用是学习图像的特征。基于神经网络的语义分割方法的基本思想，是通过训练多层感知机来得到线性决策函数，然后用决策函数对像素进行分类来达到语义分割的目的，这种方法需要投入大量的训练数据。神经网络存在巨量的连接，容易引入空间信息，能较好地解决图像中的噪声和不均匀问题。选择何种网络结构是此类基于神经网络的技术要解决的主要问题。

以下介绍一些对计算机视觉领域做出重大贡献的标准深层网络，因为它们经常被用作语义分割模型的基础。

Alexnet：Toronto首创的Deep CNN，以84.6%的测试准确率赢得了2012年Imagenet竞赛。它由5个卷积层、最大池层、作为非线性的ReLUs、3个完全卷积层和dropout组成。

VGG-16：这款牛津的模型以92.7%的准确率赢得了2013年的Imagenet竞赛。它使用第一层中具有小接收场的卷积层堆栈，而不是具有大接收场的少数层。

GoogLeNet：GoogLeNet赢得了2014年Imagenet的竞争，准确率为93.3%。它由22层和一个新引入的称为初始模块的构建块组成。该模块由网络层网络、池操作、大卷积层和小卷积层组成。

ResNet：这款微软的模型以96.4%的准确率赢得了2016年的Imagenet竞争，这得益于它的152层的深度和残余块的引进。残余块通过引入标识跳过连接来解决训练深层架构的问题，以便当前层可以将其输入复制到下一层。这个网络的简化版也是本文实验中所使用的YOLACT模型的底层网络。

在人工智能发展的三大驱动力“数据，算法，计算力”中，眼下最重要的就是数据。数据集在人工智能中有着举足轻重的地位，具体根据不同的应用领域，需要选择不同的数据集。目前常用的公共数据集主要有：

（1）Pascal VOC系列：通常采用PASCAL VOC 2012，最开始有1464 张具有标注信息的训练图片，2014 年增加到10582张训练图片。主要涉及了日常生活中常见的物体，包括汽车，狗，船等20个分类。

（2）Microsoft COCO：微软开发维护的大型图像数据集，一共有80个类别。数据集标注类型对应任务包括物体检测、关键点检测、实例分割、 stuff分割 （没有特定形状的物体） ，全景分割人体关键点，人体密度检测等等，也是本文实验中所使用的数据集格式。

（3）Cityscapes：适用于汽车自动驾驶的训练数据集，包括19种都市街道场景：road、side-walk、building、wal、fence、pole、traficlight、trafic　sign、vegetation、terain、sky、person、rider、car、truck、bus、train、motorcycle 和 bicycle。该数据库中用于训练和校验的精细标注的图片数量为3475张，同时也包含了 2 万张粗糙的标记图片。

* 1. **本文主要工作**

本文研究的课题是基于YOLACT的图像语义分割模型的复现以及参数的训练优化，并将模型应用于物品复杂摆放的图像上，正确地识别并标注出每一物品实例。主要工作如下：

（1）实现YOLACT的经典算法，掌握调参、训练、评估的方法。

（2）根据分割目标制作自己的数据集，用于语义分割的训练集，并将模型测试的输出数据进行标签细化。

（3）将训练得到的模型进行一定量的测试，根据测试结果的不足，对模型参数进行调整，或对算法进行改进。

* 1. **论文结构**

本文的各章节组织结构如下：

第一章：绪论。简要介绍了图像语义分割的研究背景和意义，并概述了本文的主要工作和行文结构。

第二章：图像语义分割技术研究现状。介绍了近年来图像识别与语义分割的发展以及国内外的研究现状，包括了模型与算法的演变过程。

第三章：图像语义分割的基本算法实现。详述了本文所重点采用的YOLACT模型的算法原理和实现思路。

第四章：图像语义分割的实际应用与细节优化。针对本文对复杂图片进行分割的任务特点，对上述模型与算法进行细节上的优化，最后展示分割效果，并分析改进后精度和效率的提升。

第五章：对本文的工作进行总结，讨论存在的问题以及对未来工作的展望。

1. **图像语义分割技术研究现状**

图像语义分割，即让计算机根据图像的语义对图像中的不同对象进行分割与标注。相关技术经过多年的研究与发展，其算法也不断迭代更新，以下将以时间顺序介绍各个时期涌现的具有代表性的算法。

**2.1 基于低级特征提取的图像分割方法**

普通的图像分割，通常意味着传统语义分割，这个时期的图像分割（大概2010年之前），由于计算机计算能力有限，早期只能处理一些灰度图，后来才能处理RGB图，这个时期的分割主要是通过提取图片的低级特征，然后进行分割，分割出来的结果并没有语义的标注，换句话说，并不知道分割出来的东西是什么。这个时期涌现出一些方法，如Ostu、FCM、分水岭、N-Cut等。

N-cut，全称为Normalized cut，是一种图划分方法，它的思想主要是综合考虑像素和像素之间的关系权重，根据给出的阈值，将图像一分为二。图2-1是将像素间的关系信息简单描述成距离，根据距离差距来划分图像的例子。

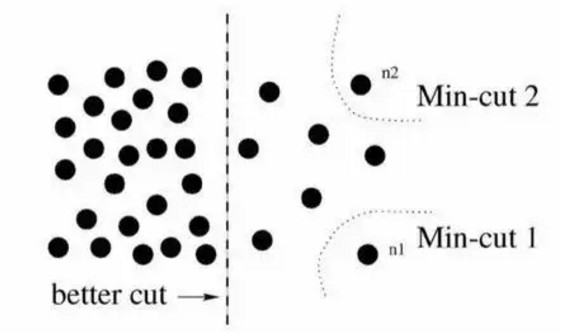


图2-1 根据距离关系划分图像的N-cut示意图

这个方法给出的结果在当时可以算是还不错，但缺点也很明显，首先，它只能做二类语义分割，在实际应用中，每运行一次 N-cut，只能切割一次图片，为了分割出图像上的多个物体，需要多次运行。其次，在遇到稍微复杂一点的图像时需要进行人工干预，这个缺点在将来批量化处理的智能时代简直就是死穴。

**2.2 基于卷积网络的语义分割方法**

之后，随着计算能力的提高，人们开始考虑分割并获得图像的语义，这里的语义目前还是低级语义，主要指分割出来的物体的类别，这个阶段（大概是2010年到2015年）人们开始引入卷积神经网络CNN，使用机器学习的方法进行语义分割。

2015年，国际计算机视觉与模式识别会议（CVPR）上发表了一篇论文，论文第一次提出了全卷积网络FCN，获得了当年的最佳荣誉提名论文。随着FCN的出现，深度学习正式进入语义分割领域，这里的语义仍主要指分割出来的物体的类别，但从分割结果已经可以清楚地知道分割出来的是什么物体，比如猫、狗等等。Faster RCNN是两阶段的目标检测算法，阶段一使用RPN（Region Proposal Network）去提取出ROI，然后使用ROI pooling将这些ROI全部变成固定尺寸后，提供给阶段二的全连接层进行Bounding box回归和分类预测。Faster RCNN的整个流程如图2-2所示。

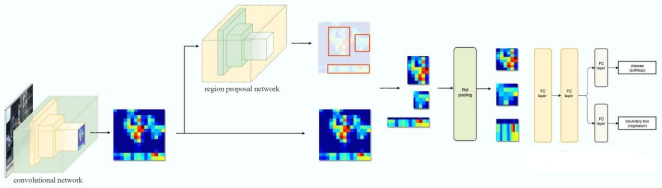


图2-2 Faster RCNN流程示意图

对于一般的CNN网络，如VGG和ResNet，都会在网络的最后加入一些全连接层，经过softmax函数的预测后就可以获得类别概率信息。但是这个概率信息是1维的，即只能标识整个图片的类别，不能标识每个像素点的类别。而FCN则是把后面几个全连接都换成卷积，这样就可以获得一张2维的特征图，后接softmax获得每个像素点的分类信息，如图2-3所示。

（1）对于FCN-32s，直接对pool5特征图进行32倍上采样获得32倍特征图，再对32倍特征图每个点做softmax预测获得FCN-32s预测图（即分割图）。

（2）对于FCN-16s，首先对pool5特征图进行2倍上采样获得2倍特征图，再把pool4特征图和2倍特征图逐点相加，然后对相加的特征图进行16倍上采样，并做softmax预测，获得FCN-16s预测图。

对于FCN-8s，首先进行pool4 + 2x逐点相加后两倍上采样，然后又进行pool3 + 2x逐点相加，即进行更多次特征融合……

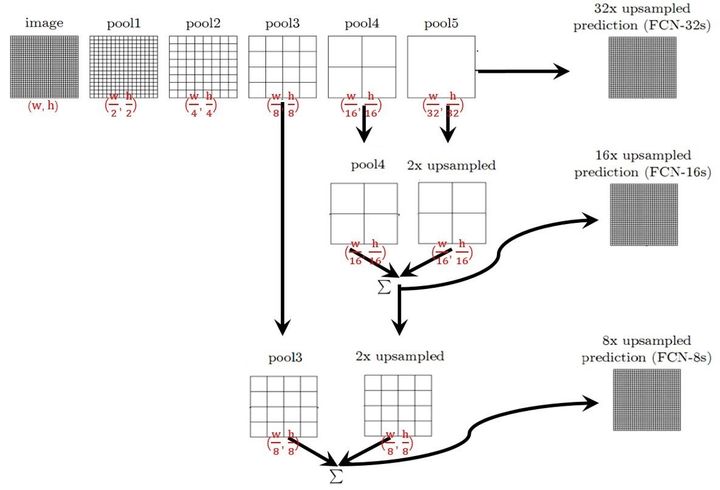


图2-3 不同倍数的FCN预测图生成过程示意图

如图2-4所示，从精度上看，FCN-32s＜FCN-16s＜FCN-8s，即使用多层特征融合有利于提高分割准确性。

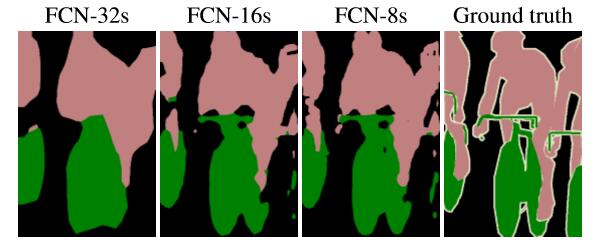


图2-4 不同倍数上采样结果对比图

因此可以看出，CNN的识别是图像级的识别，也就是从图像到结果；而FCN的识别是像素级的识别，对输入图像的每一个像素在输出上都有对应的判断标注，标明这个像素最可能是属于一个什么物体或类别。但是FCN并不是完美无缺，主要有以下两个缺点：

（1）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。

（2）FCN是对各个像素进行分类，因此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步骤，缺乏空间一致性。

**2.3 基于区域选择的分割算法改进**

在语义分割领域，基于区域选择的几个算法主要是由前人的有关于目标检测的工作渐渐延伸到语义分割的领域的。目标检测的主要研究进程由R-CNN到Fast R-CNN再到Faster R-CNN。

Faster R-CNN是两阶段的目标检测算法，包括阶段一的Region proposal以及阶段二的Bounding box回归和分类。Faster R-CNN的整个流程如图2-5所示。

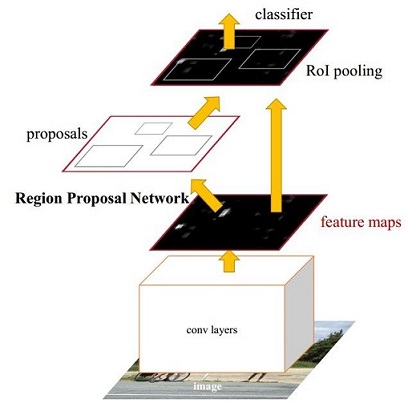


图2-5 Faster R-CNN算法流程示意图

Faster RCNN使用CNN提取图像特征，然后使用Region Proposal Network（RPN）去提取出ROI（感兴趣区域），然后使用ROI池化将这些ROI全部变成固定尺寸，再喂给全连接层进行Bounding box回归和分类预测。

随着技术的发展，多尺度检测在目标检测中变得越来越重要，对小目标的检测尤其如此。现在主流的目标检测方法很多都用到了多尺度的方法，FPN（Feature Pyramid Networks，即特征金字塔网络）则是一种精心设计的多尺度检测方法。

FPN结构中包括自下而上，自上而下和横向连接三个部分，如图2-6所示。这种结构可以将各个层级的特征进行融合，使其同时具有强语义信息和强空间信息。

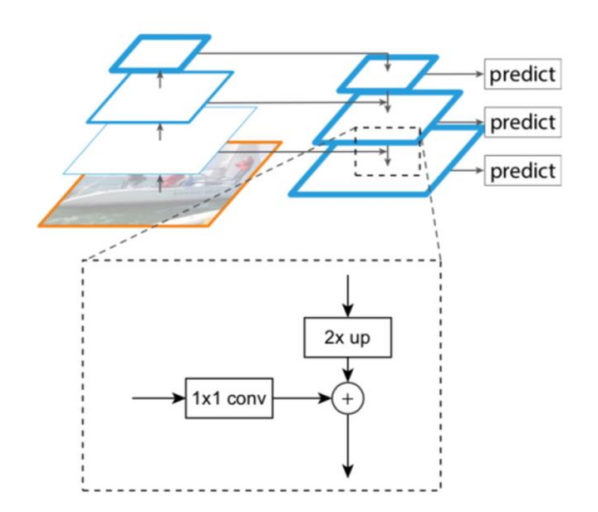


图2-6 FPN结构示意图

FPN实际上是一种通用架构，可以结合各种骨架网络使用，比如与VGG或ResNet等底层网络架构结合而成的VGG-FPN或ResNet-FPN。于是可以通过添加FPN结构，对Faster R-CNN中提取特征的过程进行改进，再加上mask预测，从而最终得到下文介绍的Mask R-CNN模型。

在此基础上，我国何恺明团队提出一个基于Faster R-CNN模型的一种新型的分割模型Mask R-CNN，可以在目标检测和分类的同时进行语义分割，此论文斩获ICCV 2017的最佳论文。Mask R-CNN的构建是在ROI pooling（实际上用到的是加以改进的ROI Align）之后添加卷积层，进行mask预测的任务，主要由以下两个网络构成：

（1）骨干网络ResNet-FPN，用于特征提取。另外，ResNet还可以是：ResNet-50，ResNet-101， ResNeXt-50，ResNeXt-101。这里ResNet-101与本文实验将使用的YOLACT模型的默认骨干网络相同。

（2）头部网络，包括边界框识别（分类和回归）+mask预测。头部结构如图2-7所示。

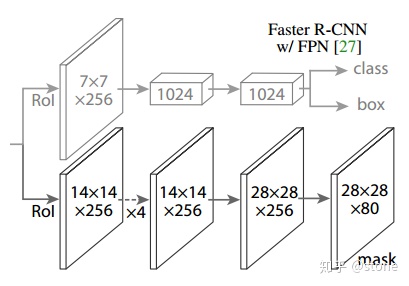


图2-7 Mask R-CNN的头部网络结构示意图

实际上，Mask R-CNN中还有一个很重要的改进，就是ROIAlign。Faster R-CNN存在的问题是：特征图与原始图像是不对准的（mis-alignment），所以会影响检测精度。而Mask R-CNN提出了RoIAlign的方法来取代ROI pooling，因为RoIAlign可以保留大致的空间位置。

**2.4 本章小结**

由本章内容可知，图像语义分割技术随着精度要求的提高而不断地改进和发展，而算法的更新迭代总是在解决前一阶段遇到的问题，这个问题可以概括为对“语义”的理解过程。从分割而不识别的基于低级特征提取的图像分割时期，到分割同时进行分类的基于卷积网络的语义分割时期，再到可以将同一类别的不同实例分割出来的实例分割时期，对“语义”的理解也逐渐深入透彻，逐渐考察更多的细节，逐渐向人类对图像的认知靠近。近几年不断有更新颖、更前沿的模型被提出，如Mask Scoring R-CNN、YOLACT、Blend Mask等。此次毕业设计具体便选择了其中的YOLACT模型进行深入研究。

1. **语义分割的基本算法实现**

**3.1 方法步骤流程**

本实验所使用的方法主要包括图像数据预处理和图像语义分割训练两个阶段。一阶段是选取合适的原始数据集，使用合适的标注方法对原始数据集进行手动标签标注，并将标注后的数据文件转换为合适的格式。二阶段是将数据集投入训练，根据训练结果改进数据集的选取和标注方式。具体方法步骤流程如图3-1所示。

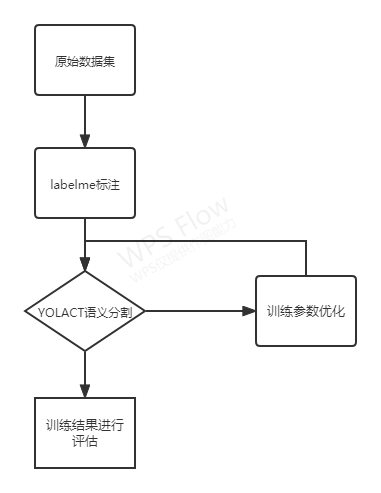


图3-1 本文实验具体方法步骤流程示意图

**3.2 基于labelme的图像数据预处理**

图像数据的预处理主要分为两个部分：

（1）使用labelme软件对数据集（包括训练集和验证集）中每张图片里的每个实例进行标记，获取对应的json文件。

（2）将第一步得到的json文件转为COCO数据集的格式。

**3.3 基于YOLACT的实时实例分割**

**3.3.1 核心算法与工作原理**

在目标检测模型设计上，主要有两大路线：Two-stage型网络，即“先检测后分割”的方法，首先定位到目标物体的边框，然后在边框内分割目标物体，典型的代表是Faster R-CNN，这类模型更加重视精度；One-stage型网络则更加重视提升速度，代表作YOLO、SSD等。

由于实例分割建立在目标检测任务的基础上，因此同样可以有两条路线。Mask R-CNN模型是实例分割模型的典范，它基于Faster R-CNN模型改进而来，奠定了Two-stage型实例分割的基调，因此同样地Mask R-CNN的重点也是在于精度的提升。然而另一条路线却出现了困难：SSD/YOLO通过移除第二个stage，并以其他方式弥补性能缺失来加速Faster R-CNN。然而这种思想不易直接应用在实例分割任务上，因为实例分割为了产生mask，严重依赖于特征定位。在Two-stage模型中，通过repooling型操作（如ROI align、ROI pooling）来将特征映射到包围框中，这种做法从逻辑上就是串行的，所以很难加速。FCIS虽然将这些操作进行并行化，但由于后处理步骤太多，同样很难达到加速的目的。

基于以上考虑，Daniel Bolya团队另辟蹊径提出了YOLACT模型。YOLACT是2019年发表在ICCV上面的一个全卷积的实时实例分割模型，根据评估，该模型在处理MSCOCO数据集的图片时，可以在29.8mAP的精度下达到33.5FPS的速度。这是只在一个GPU上进行训练就得到了这个结果，比以往任何方法都要快得多，而互联网上多数视频一般是30FPS，这也就是“实时”的含义了。YOLACT的是You Only Look At CoefficienTs的简写，其中coefficients也就是这个模型的输出之一，这个命名风格明显致敬了目标检测模型YOLO。

YOLACT模型并非只是在Mask R-CNN的基础上进行小修小补，而是基于One-stage全卷积算法做了重新设计，虽然在精度上稍低于Mask R-CNN，但是在精度满足大部分需求的同时，速度大大提升，达到了实时的效果，并且容易部署，可广泛应用于多种不同的场景。

YOLACT模型的框架如图3-2所示，该模型非常具有创新性地摒弃了隐含的特征定位步骤，将实例分割分解为两个并行的子任务：

1. Prediction Head分支生成各个anchor的类别置信度、位置回归参数以及mask系数；
2. Protonet分支生成一组独立于实例的原型mask，然后将mask与mask系数线性结合，从而得到图片中每一个目标物体的mask。

由于这个过程不依赖于重新池化，这种方法可以产生非常高质量的mask，并以极低地代价显示出时间稳定性。

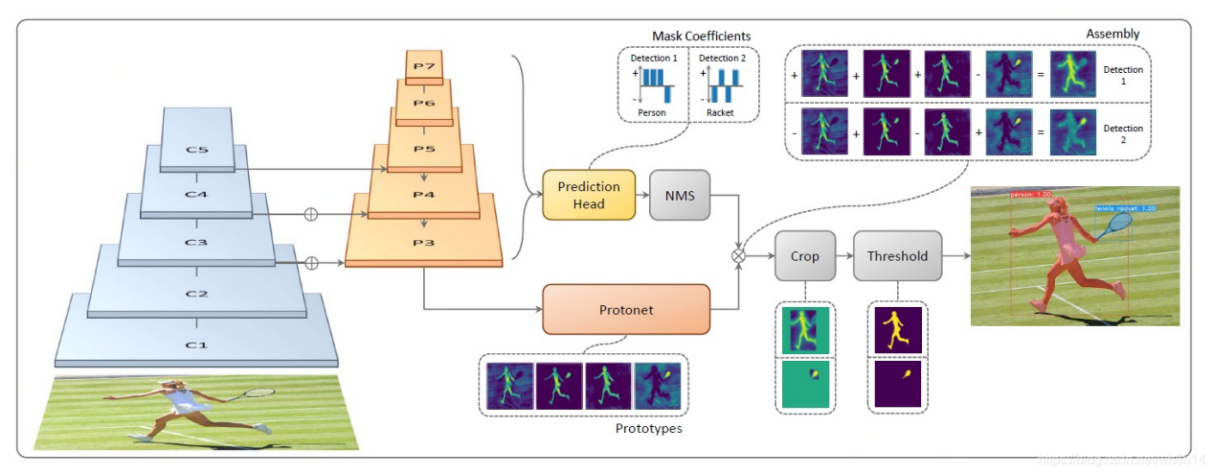


图3-2 YOLACT模型的框架示意图

1. 骨干网络ResNet101-FPN：五层卷积C1~C5，然后由C5经过1个卷积层得到P5，然后对P5采用双线性插值使特征图扩大一倍，与经过卷积的C4相加得到P4，再采用同样的方法即可得到P3。再然后，对P5进行卷积和下采样得到P6，对P6进行同样的卷积和下采样得到P7，从而建立FPN网络。P3被送入Protonet，同时P3-P7被送到Prediction Head中。
2. Protonet分支：其网络结构如图3-3所示，由若干卷积层组成，输出k个通道的prototype mask。

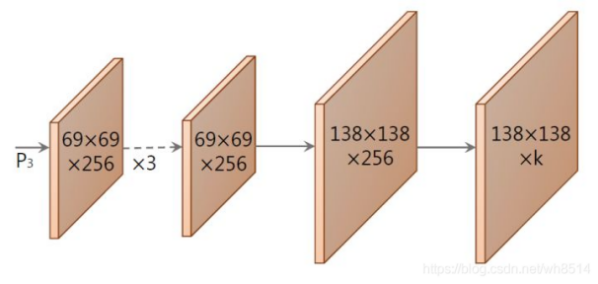


图3-3 Protonet分支的网络结构示意图

1. Prediction Head分支：其网络结构如图3-4所示，在RetinaNet的基础上改进得到的，采用共享卷积网络，从而可以提高速度，达到实时分割的目的。它的输入是 P3 ~ P7 共五个特征图，每个特征图先生成anchor，然后生成每个anchor的类别置信度、位置回归参数以及mask系数。

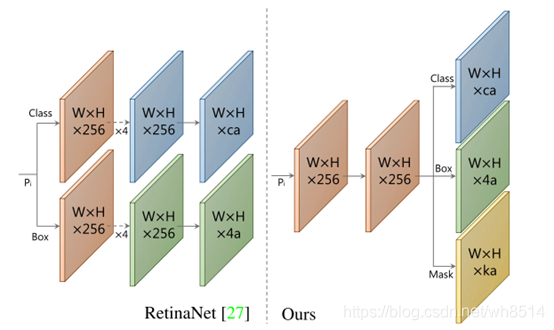


图3-4 Prediction Head分支的网络结构示意图

1. NMS：通过Prediction Head分支网络后会得到很多anchor，可以在anchor的位置加上位置偏移得到RoI位置。由于RoI存在重叠，NMS是常用的筛选算法。
2. Crop & Threshold：将通过Prediction Head分支得到的mask系数和Protonet分支得到的prototype mask做矩阵乘法，就可以得到图像中每一个目标物体的mask。Crop指的是将边界外的mask清零，Threshold指的是以0.5为阈值对生成的 mask进行图像二值化处理。

经过实验分析，Daniel Bolya认为YOLACT通过原型mask去自适应的学习如何定位实例目标。原型mask的数量与类别数目无关，也就是说每张原型mask都包含了跨类别的信息，YOLACT旨在学习一种分布的表示方法，这样每个实例可以通过该表示方法对多张原型mask进行线性组合，来得到自己的mask。通过训练，每张原型mask会学习到去抽象输入图像的一些细节信息，比如边缘信息、位置信息，或者对特定区域响应的信息。

YOLACT有三个显著的优势：

（1）速度快，因为One-stage；

（2）mask质量高，因为不包含repooling类操作；

（3）普适性强，这种生成原型mask和mask系数的思路可以应用在目前很多流行的检测器上。

与人类视觉系统相比，YOLACT也非常直观：线性系数组合和检测分支像是在解决“是什么”的问题，原型mask的生成像是在解决“在哪里”的问题。

YOLACT与其他一些现有的实例分割模型在COCO数据集上的表现对比如图3-5所示。

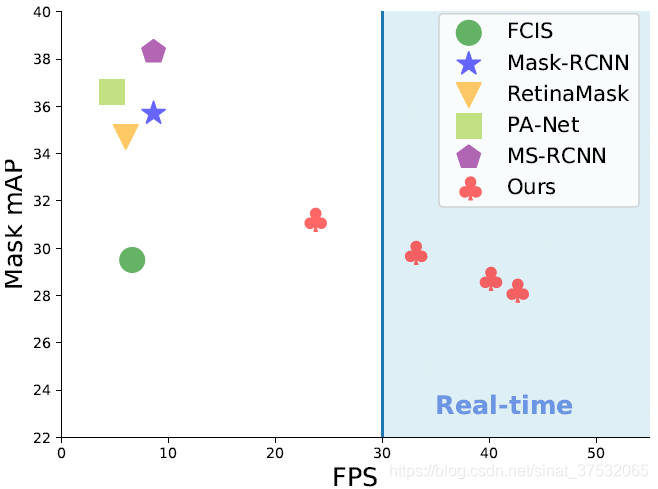


图3-5 多种实例分割模型在COCO数据集上的效果对比图

综上所述，可见YOLACT模型的算法是非常具有创新性的，大胆抛弃了Mask R-CNN那一套较为繁琐冗杂的设计，整个流程显得非常的简洁且高效，并行预测当前图片的原型掩码（prototype mask) 和每个bbox实例的掩码系数（mask coefficients)，然后通过将原型与掩模系数线性组合来生成实例掩码（instance masks)，思路很有独到之处，是近几年被提出的前沿算法中非常值得研究与学习的对象。

**3.3.2 数据增强代码与分析**

与数据增强相关的代码位于train.py文件中。首先是数据的加载，这里传入的参数包括训练集图像的路径image\_path、训练集标记文件info\_file，以及使用的数据增强方式transform。

1. dataset = COCODetection(image\_path=cfg.dataset.train\_images,
2. info\_file=cfg.dataset.train\_info,
3. transform=SSDAugmentation(MEANS))

在这里重点关注一下SSDAugmentation类中包含了哪些数据增强方法。

1. **class** SSDAugmentation(object):
2. """ Transform to be used when training. """
4. **def** \_\_init\_\_(self, mean=MEANS, std=STD):
5. #构造Compose的实例对象，传入的是一个列表
6. self.augment = Compose([
7. ConvertFromInts(), #将图片数据转为np.float32的数据类型
8. ToAbsoluteCoords(), #计算bbox的绝对坐标
9. enable\_if(cfg.augment\_photometric\_distort, PhotometricDistort()), #光度扭曲
10. enable\_if(cfg.augment\_expand, Expand(mean)), #扩张
11. enable\_if(cfg.augment\_random\_sample\_crop, RandomSampleCrop()), #随机裁剪
12. enable\_if(cfg.augment\_random\_mirror, RandomMirror()), #随机镜像
13. enable\_if(cfg.augment\_random\_flip, RandomFlip()), #随机翻转
14. enable\_if(cfg.augment\_random\_flip, RandomRot90()), #随机旋转
15. Resize(),
16. enable\_if(**not** cfg.preserve\_aspect\_ratio, Pad(cfg.max\_size, cfg.max\_size, mean)),
17. ToPercentCoords(),
18. PrepareMasks(cfg.mask\_size, cfg.use\_gt\_bboxes),
19. BackboneTransform(cfg.backbone.transform, mean, std, 'BGR')
20. ])

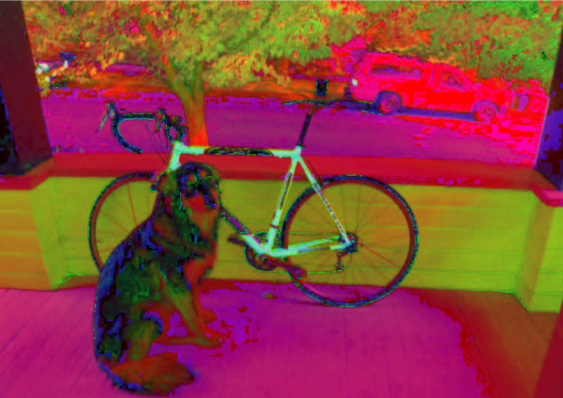
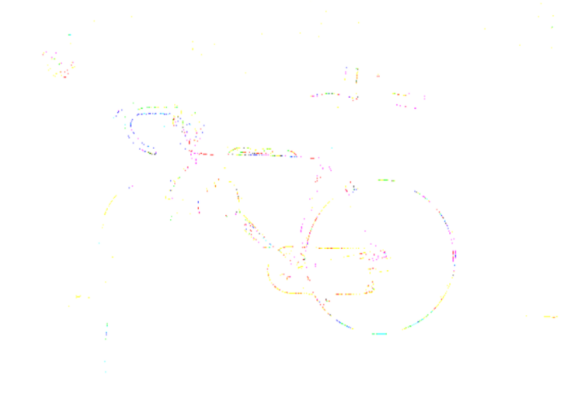
这里用到了光度扭曲、图像扩张、随机裁剪、随机镜像、随机翻转、随机旋转等数据增强方法。对图像进行几何变换的数据增强方法比较容易理解，但类似于光度扭曲的方法可能就难以想象出其效果。于是通过调用与光度扭曲相关的代码，可对这种数据增强方法进行直观地展示。光度扭曲效果如图3-6所示。



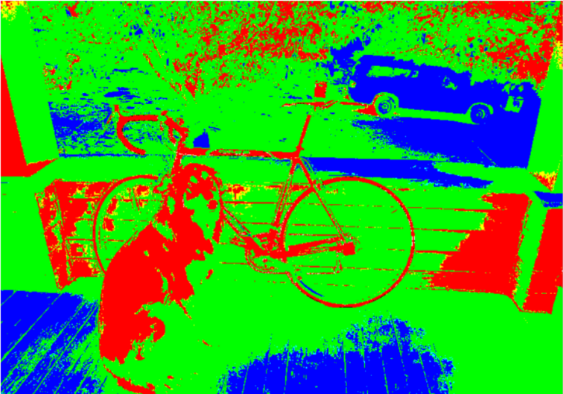
（a）处理前 （b）处理后

图3-6 光度扭曲效果图

光度扭曲实际上是由多种像素操作组合而得到的，包括随机对比度、颜色通道转换、随机亮度等操作，其效果在图3-7中分别进行展示。



（a）随机对比度处理 （b）颜色通道转换处理



（c）随机亮度处理

图3-7 三种不同的像素操作效果

数据增强在深度学习中是不可缺少的步骤，其作用有：

（1）避免过拟合。当数据集具有某种明显的特征，例如数据集中的图片基本在同一个场景中拍摄，数据增强可以避免模型学习到与目标无关的信息。

（2）提升模型的鲁棒性，降低模型对图像的敏感度。当训练数据都处于比较理想的状态时，如果碰到一些特殊情况，类似于遮挡、亮度不均、模糊等特殊情况很容易出现识别错误，对训练数据进行数据增强，如加上噪声、掩码等方法，可以有效提升模型的鲁棒性。

（3）增加训练数据量。提高模型泛化能力。

（4）避免样本不均衡。在工业缺陷检测、医疗疾病识别等方面，很容易出现正负样本极度不平衡的情况，通过对少样本进行一些数据增强，可以有效缓解样本不均衡比例。

继续看看定义在augmentions.py文件中的COCODetection的构造函数：

1. **def** \_\_init\_\_(self, image\_path, info\_file, transform=None,
2. target\_transform=None,
3. dataset\_name='MS COCO', has\_gt=True):
4. # Do this here because we have too many things named COCO
5. **from** pycocotools.coco **import** COCO
7. **if** target\_transform **is** None:
8. target\_transform = COCOAnnotationTransform()
10. self.root = image\_path
11. self.coco = COCO(info\_file) #将标签文件导入COCO API
13. #self.coco.imgToAnns 里面包含了标签文件中所有的bbox、category\_id、image\_id、segmentation信息
14. self.ids = list(self.coco.imgToAnns.keys())
15. **if** len(self.ids) == 0 **or** **not** has\_gt:
16. self.ids = list(self.coco.imgs.keys())
18. self.transform = transform
19. #transform是SSDAugmentation的实例对象
20. #COCOAnnotationTransform这个类作用：将COCO的标签转换成bbox coords and label index的张量
21. self.target\_transform = COCOAnnotationTransform()
23. self.name = dataset\_name
24. self.has\_gt = has\_gt

关注这里的COCOAnnotationTransform()，这个类的\_\_call\_\_方法将COCO标签中的bbox和category\_id信息存到一个列表中：[xmin, ymin, xmax, ymax, category\_id]。根据COCO数据集官方文档，预测的结果格式以列表的形式保存，列表中每个元素对应一个检测目标，每个目标中记录了4个信息：

（1）image\_id：记录该目标所属图像的id。

（2）category\_id：记录预测该目标的类别索引。

（3）bbox：记录预测该目标的边界框信息，对应目标的[xmin，ymin，width，height]。其中，(xmin, ymin)为目标边界框左上角的(x,y)坐标；width为目标的宽度，height为目标的高度。

（4）score：记录预测该目标的概率。

回到train.py文件中创建yolact算法模型对象的地方。

1. yolact\_net = Yolact()
2. net = yolact\_net
3. net.train()

Yolact()类定义在yolact.py文件中，其定义中默认以ResNet101网络作为backbone。

**3.3.3 训练过程参数的影响**

在深度学习神经网络中，超参数的调整至关重要，通过观察在训练过程中的监测指标如损失loss和准确率来判断当前模型处于什么样的训练状态，及时调整超参数从而更科学地训练模型能够提高训练效率和资源利用率。以下将分别对YOLACT算法使用的不同超参数及其影响进行介绍。

（1）学习率（learning rate或作lr）。学习率是指在优化算法中更新网络权重的幅度大小。学习率可以是恒定的、逐渐降低的，基于动量的或者是自适应的。不同的优化算法决定不同的学习率，学习率过大时则可能导致模型不收敛，损失loss不断上下震荡；学习率过小则可能导致模型收敛速度偏慢，需要花更长的时间进行训练。

（2）批次大小（batch\_size）。批次大小是指每一次训练神经网络送入模型的样本数，在卷积神经网络中，大批次通常可以使网络更快收敛，但由于内存资源的限制，批次过大可能会导致内存不够用或者程序内核崩溃。

（3）迭代次数。迭代次数是指整个训练集输入到神经网络进行训练的次数，当测试错误率和训练错误率相差较小时，可认为当前迭代次数合适；当测试错误率先变小后变大时则说明迭代次数过大了，需要减小迭代次数，否则容易出现过拟合。

（4）激活函数。在神经网络中，激活函数的作用是给神经网络加入一些非线性因素，从而使得神经网络可以更好地解决较为复杂的问题。

1. **图像语义分割的实际应用与细节优化**

**4.1 针对嵌套圆管的数据集制作**

数据集图像为每张包括多个目标对象的圆管堆图像，由80%训练集和20%验证集组成，测试图像为训练图像经过图像数据增强得到。本实验使用labelme图像标注软件对原始图像进行实例分割和分类标注处理，转化为COCO数据集格式后，作为第一阶段的数据输入，处理后图像示例如图4-1所示：

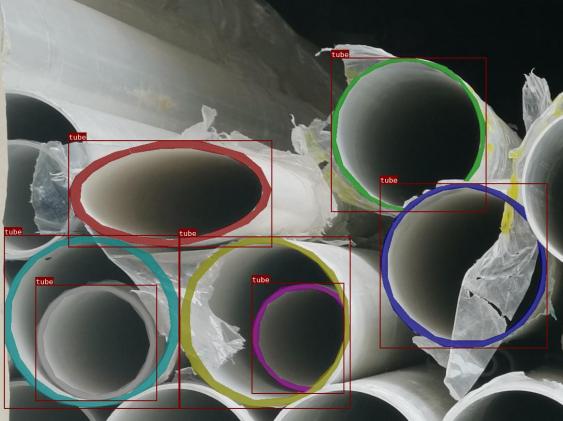


图4-1 原始图像（左），实例标注可视化图像（右）

**4.2 实验环境和参数设置**

本文实验环境基于Ubuntu 18.04.3 LTS，内核为GNU/Linux 4.15.0-171-generic x86\_64，使用Anaconda封装环境，python版本为3.7.11，深度学习库pytorch版本为1.11.0，torchvision版本为0.12.0，GPU驱动cudatoolkit版本为11.3.1。任何版本上的更改都会导致难以预料的兼容性问题，因此实验复现需要严格参照上述第三方库版本。

参数设置方面允许对数据集--dataset，批次大小--batch\_size，学习率--learning\_rate，循环周期--epoch等参数进行调整。此外，在读取图片的同时对其进行了次序相同、参数相等的增强处理，变换包括平移、旋转、缩放等基本变化，以及锐化、灰度增强、光度扭曲等增强方法。

**4.3 应用过程中的细节优化**

使用第一轮的数据集，训练迭代到达30000次时参数基本收敛不变，中止训练保存参数得到一个权重文件。用此权重文件对一些图像进行评估，输出图像如图4-2所示。



（a） （b）



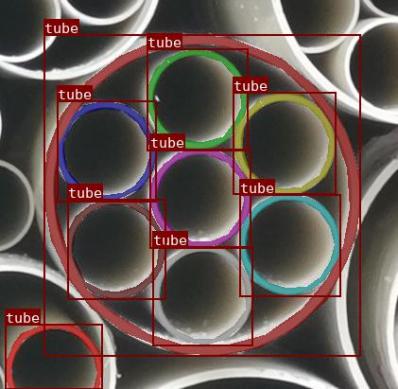
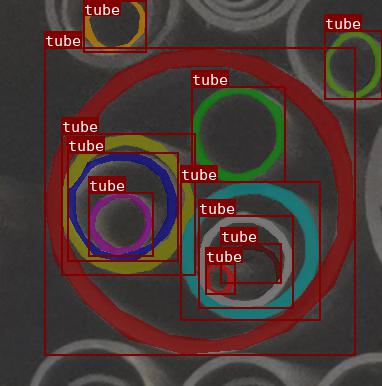
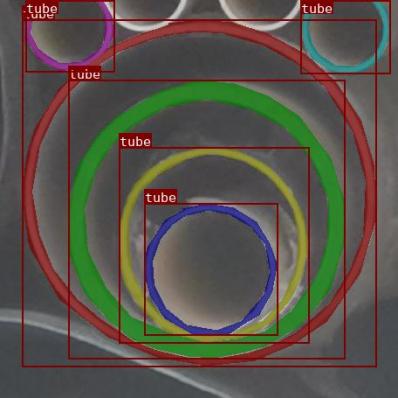
（c） （d）

图4-2 第一轮训练结果测试图

第一轮训练结果基本上可以把无嵌套的圆管检测出来，少数漏掉的实例存在一定程度的遮挡掩盖，极个别无遮挡却检测不出来值得在后续训练轮次特别关注一下。而对出现嵌套情况的圆管的检测则不是很理想，除开只套着一个的比较容易区分的情况之外，一旦出现大圆管内嵌套多个小圆管的情况，要么是大圆管检测不出来，要么是部分小圆管检测不出来。

将图4-2中（a）（b）两张图片与（c）（d）两张图片进行横向对比，可以发现（a）（b）两张图片效果要好于（c）（d）两张图片。分析两者的区别，注意到（a）（b）两张图片比起（c）（d）两张图片，其像素比较小，每个圆管个体在图中占的比例比较大，而且圆管的分布比较均匀，图片的亮度也比较高。

针对以上第一轮训练结果出现的问题，初步分析推断可能是训练数据集中对特殊情况的截取标注不够，包括但不仅限于实例出现遮挡掩盖、复杂嵌套等情况，图片出现像素较大、亮度较暗等情况。因此在第二轮训练的数据集准备上，针对特殊情况进行单独截取标记，并把第一轮测试中未能成功识别的例子进行标注投入训练。第二轮数据集标注的可视化效果图如图4-3所示。



1. （b） （c）



（d） （e）

图4-3 第二轮训练数据集（部分）实例标注可视化图像

**4.4 实验结果展示与分析**

第二轮训练在用时约15小时的60000次迭代之后，效果如图4-4所示。



1. （b）



（a） （b）

图4-4 第二轮训练结果测试图

与第一次训练结果进行对比，可见对特殊情况的检测成功率有较大提升。

用于评估语义分割算法性能的标准指标是平均 IOU（Intersection Over Union，即交并比），IOU 定义如下：

这样的评价指标可以判断目标的捕获程度（使预测标签与标注尽可能重合），也可以判断模型的精确程度（使并集尽可能重合）。

**第五章 总结与展望**

**5.1 工作总结**

**5.2 未来展望**

参考文献：

1. <https://arxiv.org/pdf/1311.2524.pdf>
2. <https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf>
3. <https://arxiv.org/pdf/1506.01497.pdf>

<https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf>