

**模式识别与机器学习**

**实验报告**

**班级：10012003**

**姓名： 徐金海**

**学号：2020302703**

**目录**

一、实验目的  **1**

二、实验原理 1

2.1 数据来源 1

2.2 图像语义分割简述 1

2.3 DeepLab V3+模型解析 2

2.4 模型评价指标MIOU 6

三、实验步骤与程序流程 7

3.1 实验步骤整体流程 7

3.2 VOC数据集格式详解 8

3.3 部分训练参数的解析和数据集的训练 9

3.4 利用训练好的模型预测（在线实例） 10

3.5 发起测试任务 13

四、实验结果 14

4.1 部分样例集推理结果展示 14

4.2 最终得分排名展示 16

五、评价分析 16

5.1 实验分工 16

5.2 分析与总结 17

六、附1：参考文献 18

七、附2：代码 19

7.1 水务识别模型整个项目结构 19

7.2 DeepLab V3+模型整个项目结构 20

**一、实验目的**

**项目背景：**水环境治理难题一直遭受世界各国人们的高度关注。水面上的飘浮废弃物不仅仅会造成消极的视觉冲击，还会继续常常造成水质问题；河面漂浮物危害水口，威协运作安全性；阻拦船只出航，威协航运业安全性；破坏生态环境，威协生活饮水安全。所以需要实时监测水面上的白色垃圾、生活垃圾等，以便及时处理。

**需求边界定义：**水务场景下的垃圾

**算法报警的业务逻辑：**识别水域场景中的垃圾，如果符合面积定义则报警

**项目算法要达到的目的：**通过对水体和水体上的漂浮物进行分割，并按照面积阈值判断并输出报警消息。

**二、实验原理**

2.1数据来源

极市平台在线项目中提供了100组.jpg格式的图片和与其对应的根据目标类型标注的像素点值的.png格式图片，注意png图片是8位单通道，且像素点的值均较小，故点开图片查看时，看到的是一张纯黑的图片，并非数据图片错误。以下是平台对数据集的一些说明：  
 对水体和垃圾等进行分割标注，类别有：

background：背景(像素值：0)；algae：水藻(像素值：1)；dead\_twigs\_leaves：枯枝败叶(像素值：2)；garbage：垃圾(像素值：3)；water：水体(像素值：4)

2.2图像语义分割简述

对于一张实物彩色图片，我们仅用肉眼便可判断出图片中的各个部分分别代表什么物体，是什么形状，有多大。而对于计算机而言，图片仅是一个个像素点构成的集合，如果想要找到这些像素点分别属于那些物体，这便是语义分割所需要做的事情。图1展示了本次实验中最终水务识别的一个结果样例，其中黑色部分代表background，橙红色部分代表water。

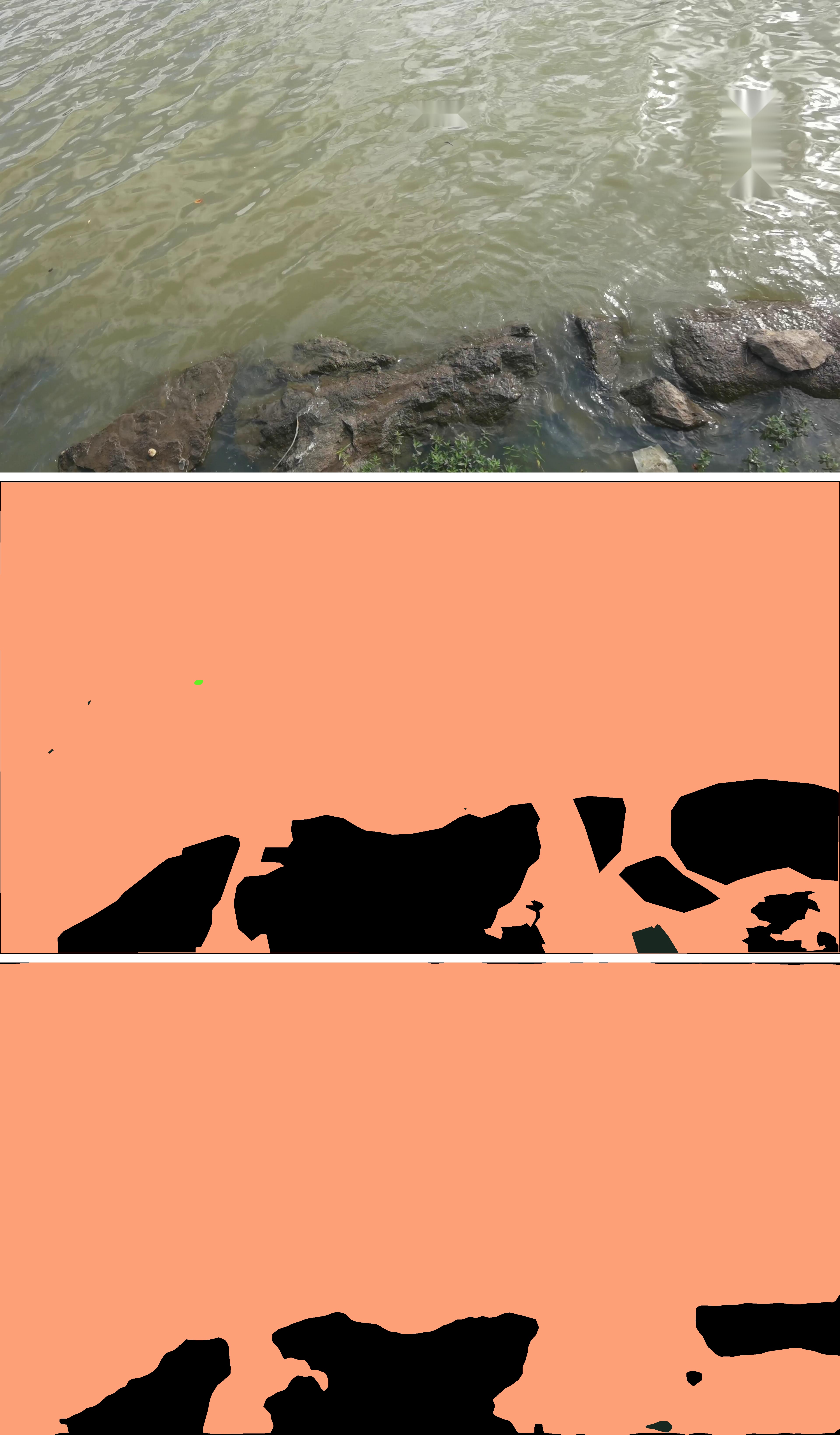
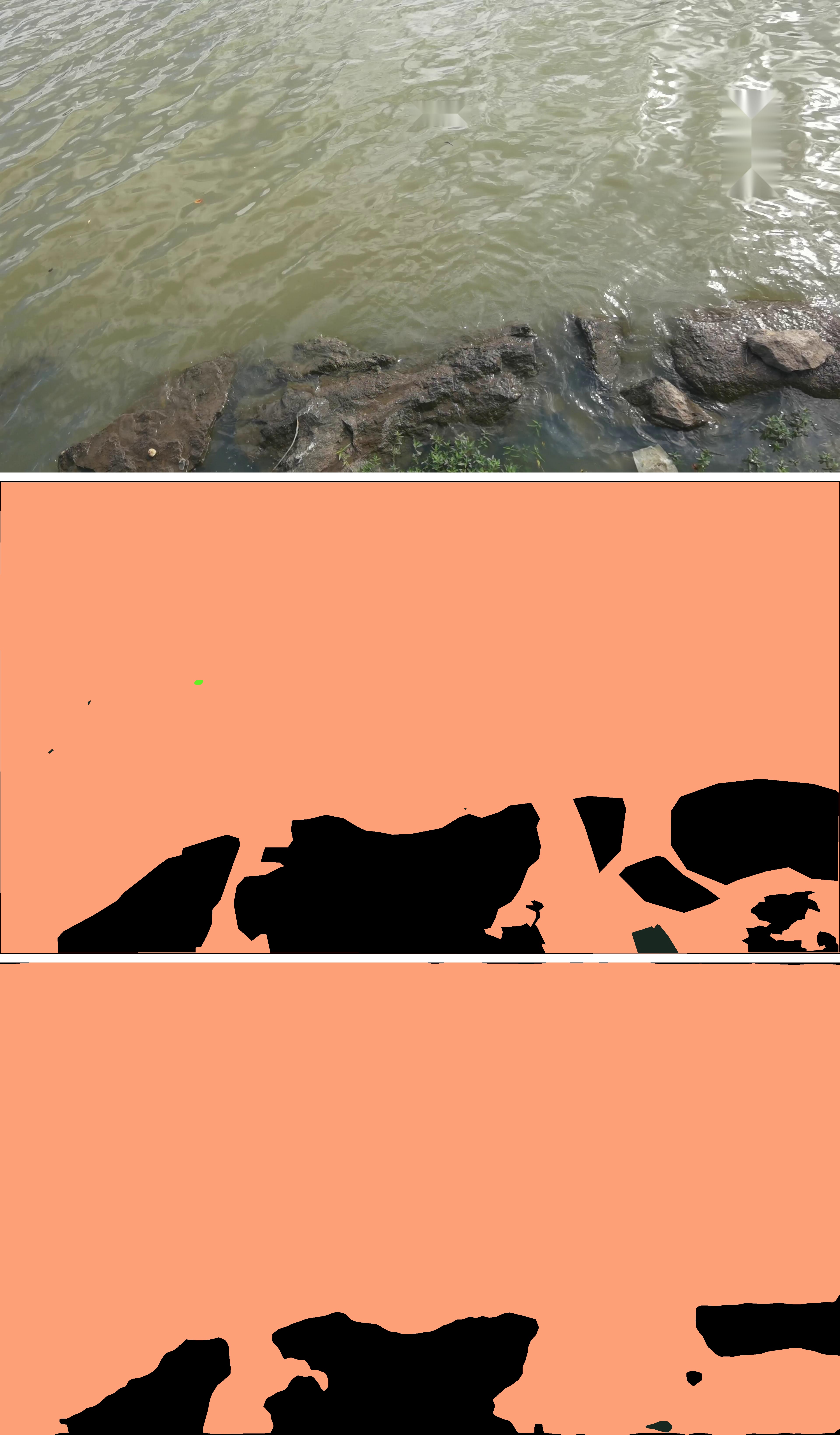


图1：水务识别语义分割样例图

图2展示了语义分割在深度神经网络中的实现原理：卷积神经网络可以进行特征提取，所以我们可以使用一堆又一堆卷积神经网络进行特征提取，在获得足够的特征之后，便可以通过反卷积操作一层一层地将图像进行放大，最后得到分割图。其中编码器Encoder用于提取特征，而解码器Decoder用于恢复原图。

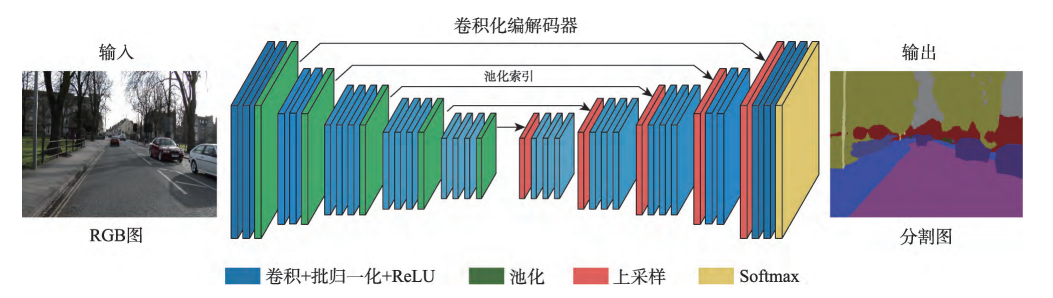


图2：图像语义分割原理示意图

2.3 DeepLab V3+模型解析

2.3.1 DeepLab V3+整体结构

DeepLab V3+被认为是语义分割模型的新高峰，其主要是在模型的结构上做改进，引入了可任意控制编码器提取特征的分辨率，通过空洞卷积平衡精度和耗时。

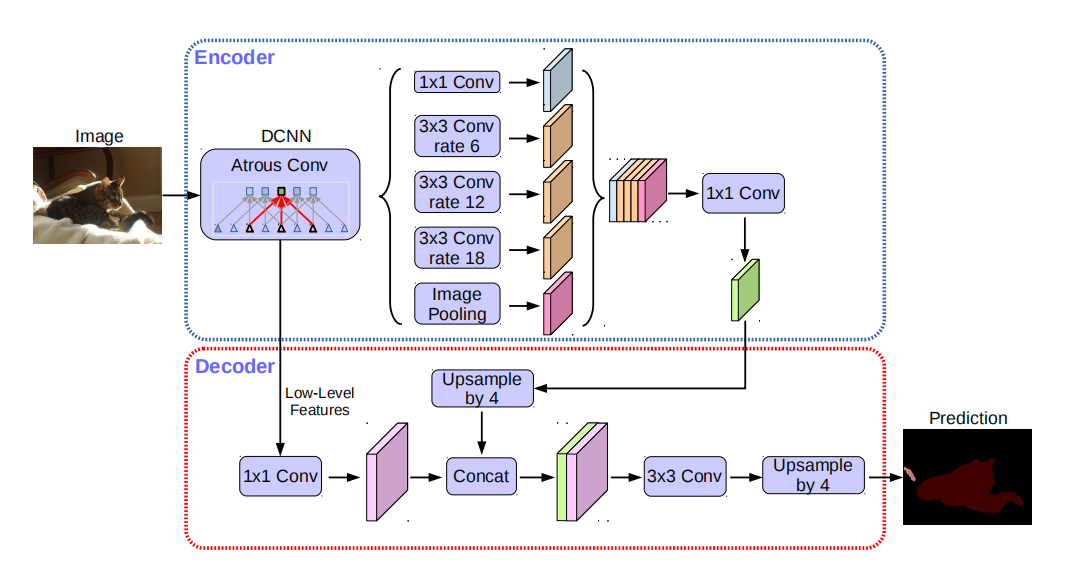


图3：DeepLab V3+网络结构

图3很好地展示了DeepLab V3+的结构，我们可以看出DeepLab V3+可以分为两个部分——Encoder和Decoder。简单地说，这个模型其实就是一个先编码再解码的过程，利用Encoder可以获取到输入图片的特征情况，利用Decoder可以将我们得到的特征进行解码，获取到我们预测的结果，这个预测结果就是我们输入的图片每一个像素点所对应的种类。

当由一幅图片输入到DeepLab V3+模型时，首先会传入到DCNN（深度卷积神经网络）中，利用DCNN我们可以获得两个有效特征层。其中一个为比较浅的有效特征层，它的高和宽会大一些，另一个为比较深的有效特征层，它所进行的下采样会更多一些，所以其高和宽会小一些。

我们会对比较深的有效特征层使用不同膨胀率的膨胀卷积进行特征提取，以此提高网络的感受野，使得网络有不同的特征的感受情况，在完成特征提取之后，会将获取到的特征层进行堆叠，随后会利用一个 1\*1 的卷积来进行通道数的调整，这样便得到一个新的特征层。到此Encoder部分基本结束。

在Decoder中我们会将Encoder中获取到的两个特征层传入，我们首先对较深的特征层经过膨胀卷积后得到的新的特征层进行上采样，再将其与较浅的特征层利用 1\*1 卷积后的结果及进行堆叠，这就相当于进行了一个特征融合的过程。然后将堆叠后的特征层经过一个 3\*3 的卷积进行特征提取，然后最终将输出图像调整为与输入图像相同大小的结果，这样输出结果就代表输入图像每一个像素点所属于的种类了。

与其他图像分割模型相比，DeepLab V3+ 具有以下优点：

1. 对于对象边界的检测更加准确。
2. 能够捕捉到图像中的小物体。

（3）在多个数据集上的效果表现出色。

因此，DeepLab V3+适用于各种CV领域的应用，如自动驾驶、人脸分析、物体检测等。

2.3.2 MobileNet V2

本次实验中我们采用的DeepLab V3+以MobileNet V2为主干特征提取网络（DCNN），这是Google在2018年提出的一种轻量级卷积神经网络，用于解决在移动设备和嵌入式设备上进行实时图像识别任务的问题。

MobileNet V2网络引入了Depthwise Separable Convolution技术。MobileNet V2采用了深度可分离卷积来减少计算量。深度可分离卷积将标准卷积分为深度卷积和逐点卷积两个部分。首先将输入张量的所有通道单独卷积（深度卷积），然后再将每个通道的结果合并（逐点卷积）。这种方法比标准卷积要快得多。

MobileNet V2的一个非常重要的特点就是使用了Inverted Residual Block，整个mobilenetv2都由Inverted Residual Block组成。Inverted Residual Block，中文名称为倒残差块，是一种被广泛应用于移动端深度学习模型中的基础网络模块。它是在ResNet的基础上进一步优化的，相较于ResNet，Inverted Residual Block 采用了Depthwise Conv + Pointwise Conv的结构，减少了计算量，同时也可以在保障特征表达能力的情况下减少模型大小，加速模型推理速度。初步应用在MobileNetV2中，达到了不错的效果。如图4所示，Inverted resblock可以分为两个部分。左边是主干部分，首先利用1\*1卷积进行升维，再进行BacthNorm和Relu，然后利用3\*3深度可分离卷积进行特征提取，再进行BacthNorm和Relu，然后再利用1\*1卷积降维，再进行一次BacthNorm。右边是残差边部分，输入和输出直接相接。

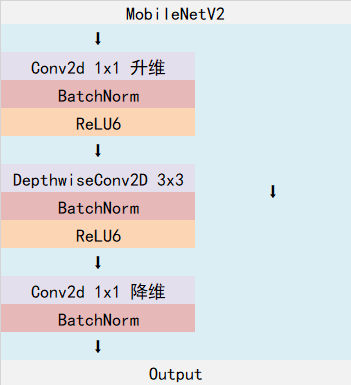


图4：Inverted resblock

需要注意的是，在Deeplab V3+当中，一般不会5次下采样，可选的有3次下采样和4次下采样，本实验中使用的4次下采样。这里所提到的下采样指的是不会进行五次长和宽的压缩，通常选用三次或者四次长和宽的压缩。

2.3.3 膨胀卷积

膨胀卷积（Dilated Convolution），又称为空洞卷积（Atrous Convolution），它是卷积神经网络中一种特殊的卷积方式，通过在卷积核中添加间隔空洞（dilation rate）的方式来增加每个像素的有效感受野，以及在不改变卷积核大小和参数个数的情况下，增加网络的感受野，从而提高了网络的特征提取和感知能力。

膨胀卷积与传统卷积的不同之处在于，传统卷积在每个卷积核位置只考虑了该位置上的像素，而膨胀卷积在卷积核中插入了一定数量的间隔（由 dilation rate 决定），使得卷积核在每个位置上对应的感受野变大，即能够看到更多的上一层特征。

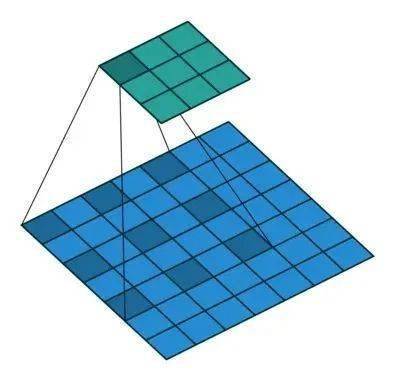
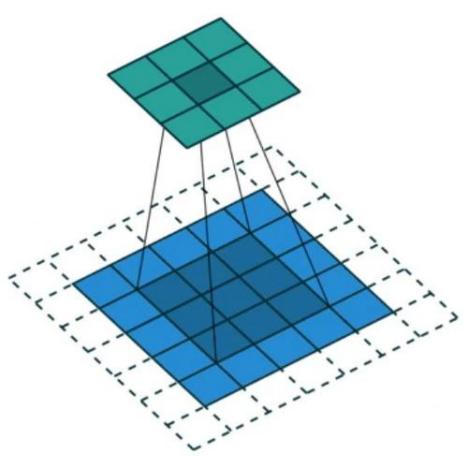
s

图5:普通卷积（左）和膨胀卷积（右）

膨胀卷积在语义分割模型中的应用有很多好处：

（1）提高感受野：语义分割模型需要考虑整张图像的上下文信息，在卷积操作过程中，膨胀卷积可以增加感受野，使得每个像素能够获得更多的上下文信息，这有助于提高语义分割的精度。

（2）减少参数：由于膨胀卷积的参数共享性质，相比于传统卷积，可以在不增加参数的情况下，实现更大的感受野，减轻网络的计算负担，缩短训练时间。

（3）处理多尺度特征：语义分割任务需要对不同尺度的图像特征进行处理，而膨胀卷积可以灵活调整膨胀率，实现多尺度特征的处理。

（4）保持空间分辨率：在传统卷积下采样（pooling池化）时，空间分辨率会失真给模型带来不利影响，而膨胀卷积不需要下采样，可以保持空间分辨率，增强模型的判别能力。

2.3.4 ASPP特征提取模块

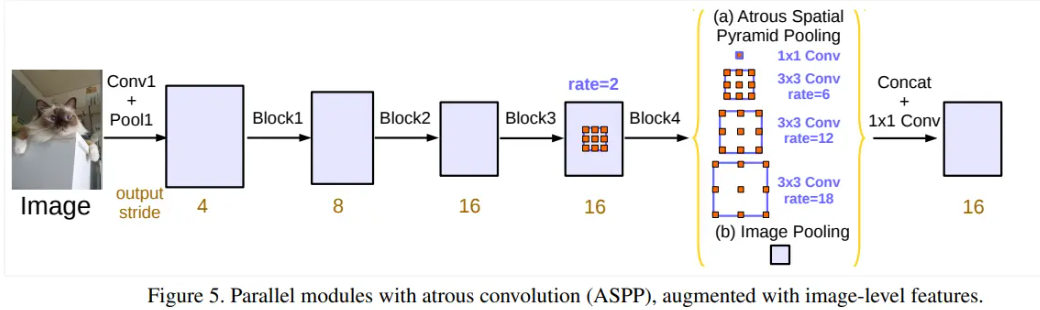


图6：ASPP结构

ASPP这个结构用于对DCNN获得的深层特征层进行加强特征提取。如图6所示，在Deeplab V3+中，其可以分为5个部分，第一个部分为一个 1\*1 的卷积，第二个部分为膨胀率为6的 3\*3 卷积，第三个部分为膨胀率为12的 3\*3 卷积，第四个部分为膨胀率为18的 3\*3 卷积，第五个部分是对输入的特征层进行池化。然后会对这五个特征层进行堆叠，再利用一个 1\*1 的卷积调整通道数，得到一个新的特征层作为Decoder部分的输入。

2.3.5浅层特征和深层特征的融合

如图3所示，Decoder部分进行浅层特征和深层特征的融合。首先对具有高语义信息的深层特征层进行上采样，上采样完成后与具有低语义信息的浅层特征通过一个1\*1卷积的结果进行特征融合。在完成特征融合之后，再通过一个3\*3的卷积进行特征提取。

2.3.6利用加强特征获得预测结果

我们的最终目的是将所有的像素点进行分类，上面我们已经完成了浅层特征与深层特征的融合，接下来需要利用特征获得预测结果。

如图3所示，这个过程可以分为两步，首先利用一个1x1卷积进行通道调整，调整成Num\_Classes，即每个特征点所属的种类。然后再利用resize进行上采样使得最终输出层，宽高和输入图片一样，这样最终的预测结果才能代表输入图片每个像素点的种类。

2.4 模型评价指标MIOU

MIOU，即Mean Intersection over Union，是一种用于计算语义分割模型性能的常见指标。它基于像素级别的准确率计算方式，以二分类语义分割模型为例，其计算公式为：

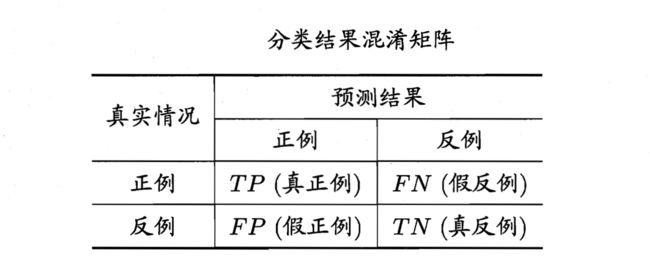


图7：二分类结果混淆矩阵

MIOU = (TP/(TP + FP + FN))

其中，TP是True Positive，FP是False Positive，FN是False Negative。TP表示模型预测正确的正样本数量，FP表示模型把负样本预测成正样本的数量，FN表示模型把正样本预测成负样本的数量。

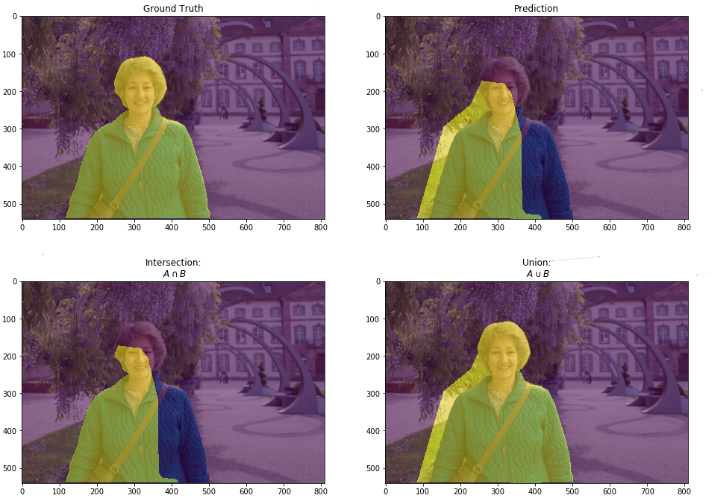


图8：MIOU示意图

MIOU可以理解为两个集合A和B的交集与并集的比值，其中A表示模型预测的对象，B表示真实的对象。如果模型准确地将预测的对象与真实的对象匹配，交集就越大，MIOU得分就越高，性能就越好。

**三、实验步骤和程序流程**

3.1实验步骤整体流程

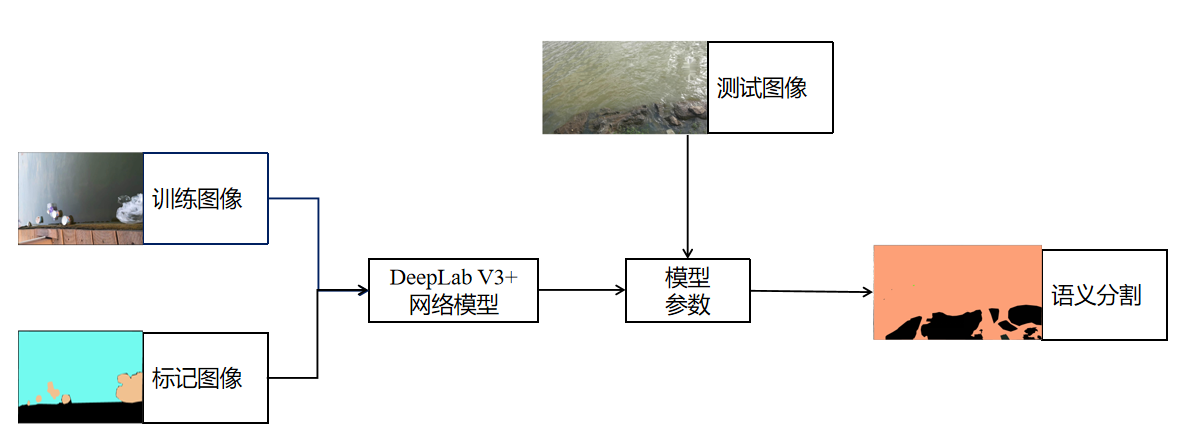


图9：实验流程图

在本次实验中，我们可以将原始数据分为训练图像（.jpg文件）和标记图像（.png文件），将其代入DeepLab V3+模型中进行训练，得到模型文件，再将需要测试的jpg图像代入预测，便可得到对应图像语义分割的结果。整体流程图如图9所示。

3.2 VOC数据集格式详解

对于本次实验，jpg文件就是实物彩色图片，对应的同名png文件是与之对应的做好像素点分类的图片文件，一共5类，像素值分别为0，1，2，3，4。



图10：RGB原图和标注图片

我们需要使用VOC格式的数据集进行训练。在本次实验中，将数据分别放到/project/train/src\_repo/deeplabv3-plus-pytorch/VOCdevkit/VOC2007目录下的SegmentationClass、JPEGImages和ImageSets/Segmentation文件夹下。其中：  
 JPEGImages目录下存放的是普通的RGB图片，也就是jpg格式的原图；

SegmentationClass目录下存放的是标签文件，即png格式的图片，其与原图一一对应；

ImageSets/Segmentation文件夹下存放了4个txt文件，其分别指向对应的图片文件，其中train.txt文件存放了用于训练的图片的名称，val.txt文件中存放了用于验证的图片的名称，trainval.txt文件存放了用于计算模型性能指标的图片文件的名称，test.txt文件中存放了测试图片的名称

注意，以上txt文件均不是手动生成，而是通过运行项目中的voc\_annotation.py文件生成。

另外需要注意的是，极市平台将数据存放在了home/data目录下，而并非在我们之前所提到的文件夹中，所以在编写strat\_train.sh脚本训练时，在开始训练之前，要将对应的图片文件拷贝到上述文件夹下，具体实现命令如下：

cp /home/data/1945/\*.jpg /project/train/src\_repo/deeplabv3-plus-pytorch/VOCdevkit/VOC2007/JPEGImages

cp /home/data/1945/\*.png /project/train/src\_repo/deeplabv3-plus-pytorch/VOCdevkit/VOC2007/SegmentationClass

至此，相关数据集的操作大概完成。

3.3 训练参数的解析和数据集的训练

3.3.1部分训练参数解析

模型训练的参数定义在train.py文件中，下面将对一些重要参数进行说明并初始化。

Cuda = True # 使用GPU训练

num\_classes = 6 # 需要的分类个数+1

backbone = "mobilenet" # 使用MobileNet V2作为主干特征提取网络

model\_path = "/project/train/models/best\_epoch\_weights.pth" # 预训练权重模型

downsample\_factor = 8 # 下采样的倍数（三次下采样）

input\_shape = [512, 512] # 输入图片的大小

VOCdevkit\_path = 'VOCdevkit' # 数据集路径

optimizer\_type = "sgd" # 优化器

momentum = 0.9 # 优化器内部使用到的momentum参数

weight\_decay = 1e-4 # 权值衰减，可防止过拟合

其余参数相对不重要，代码中均有注释，故不在此处说明。

3.3.2 数据集的训练

在线实例中在终端使用python voc\_annotation.py命令生成对应的txt文件之后，在输入命令python train.py即可进行模型的训练，但是在在线实例中，平台仅提供了一小部分数据，无法完成所有数据集的训练，只有建立模型训练才能使用平台的所有数据集进行训练。但我们可以根据上面的思路编写一个平台要求的训练脚本，通过平台检测后建立训练任务进行训练。start\_train.sh文件的内容如下：

CURRENT\_DIR=$(cd $(dirname $0)/deeplabv3-plus-pytorch; pwd)

echo "${CURRENT\_DIR}"

cd "${CURRENT\_DIR}"

cp /home/data/1945/\*.jpg /project/train/src\_repo/deeplabv3-plus-pytorch/VOCdevkit/VOC2007/JPEGImages

cp /home/data/1945/\*.png /project/train/src\_repo/deeplabv3-plus-pytorch/VOCdevkit/VOC2007/SegmentationClass

python /project/train/src\_repo/deeplabv3-plus-pytorch/voc\_annotation.py

python /project/train/src\_repo/deeplabv3-plus-pytorch/train.py

cp /project/train/src\_repo/deeplabv3-plus-pytorch/logs/best\_epoch\_weights.pth /project/train/models



图11：本地自测成功

脚本文件编写完成后，需要通过bash /project/train/src\_repo/start\_train.sh命令进行本地检测，通过后便可发起训练。

3.4利用训练好的模型预测（在线实例）

在训练好了一个模型后，我们便可以对输入的RGB图片进行预测，得到其语义分割结果。

predict.py通过调用deeplab.py中定义的预测detect\_image( )方法对输入图片实现预测。相关方法的定义代码为：

def detect\_image(self, image, count=False, name\_classes=None):

image = cvtColor(image)

old\_img = copy.deepcopy(image)

orininal\_h = np.array(image).shape[0]

orininal\_w = np.array(image).shape[1]

image\_data, nw, nh = resize\_image(image, (self.input\_shape[1],self.input\_shape[0]))

image\_data = np.expand\_dims(np.transpose(preprocess\_input(np.array(image\_data, np.float32)), (2, 0, 1)), 0)

with torch.no\_grad():

images = torch.from\_numpy(image\_data)

if self.cuda:

images = images.cuda()

pr = self.net(images)[0]

pr = F.softmax(pr.permute(1,2,0),dim = -1).cpu().numpy()

pr = pr[int((self.input\_shape[0] - nh) // 2) : int((self.input\_shape[0] - nh) // 2 + nh), \

int((self.input\_shape[1] - nw) // 2) : int((self.input\_shape[1] - nw) // 2 + nw)]

pr = cv2.resize(pr, (orininal\_w, orininal\_h), interpolation = cv2.INTER\_LINEAR)

pr = pr.argmax(axis=-1)

if count:

classes\_nums = np.zeros([self.num\_classes])

total\_points\_num = orininal\_h \* orininal\_w

print('-' \* 63)

print("|%25s | %15s | %15s|"%("Key", "Value", "Ratio"))

print('-' \* 63)

for i in range(self.num\_classes):

num = np.sum(pr == i)

ratio = num / total\_points\_num \* 100

if num > 0:

print("|%25s | %15s | %14.2f%%|"%(str(name\_classes[i]), str(num), ratio))

print('-' \* 63)

classes\_nums[i] = num

print("classes\_nums:", classes\_nums)

image = Image.fromarray(np.uint8(pr))

return image

在实例中，我们可以通过在终端输入命令：python predict.py，再在命令行中输入需要预测的图片的路径，即可对该图片进行预测，相关运行情况如图12所示，其中原图为图13。

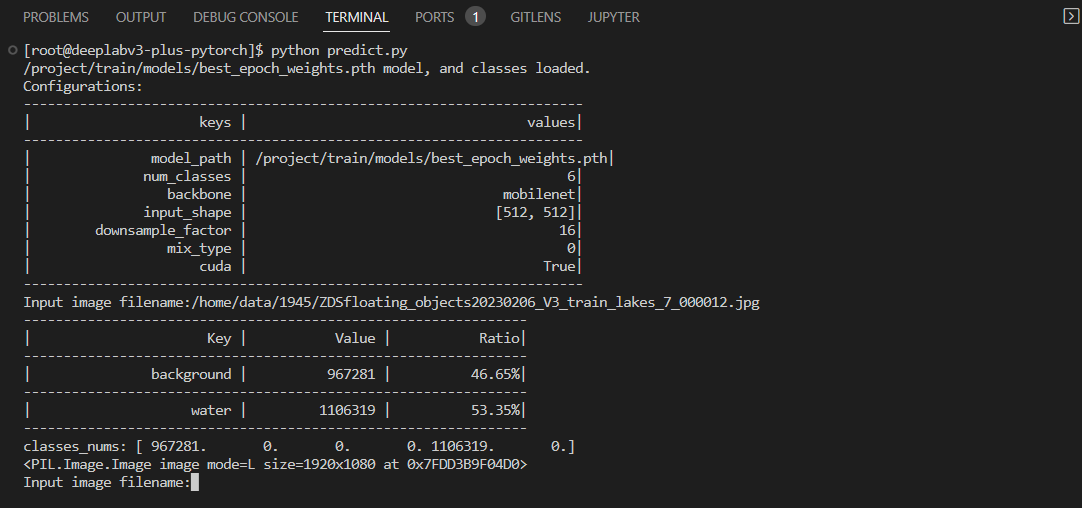


图12：在线实例预测单张图片



图13 ：被预测图片



图14 预测结果

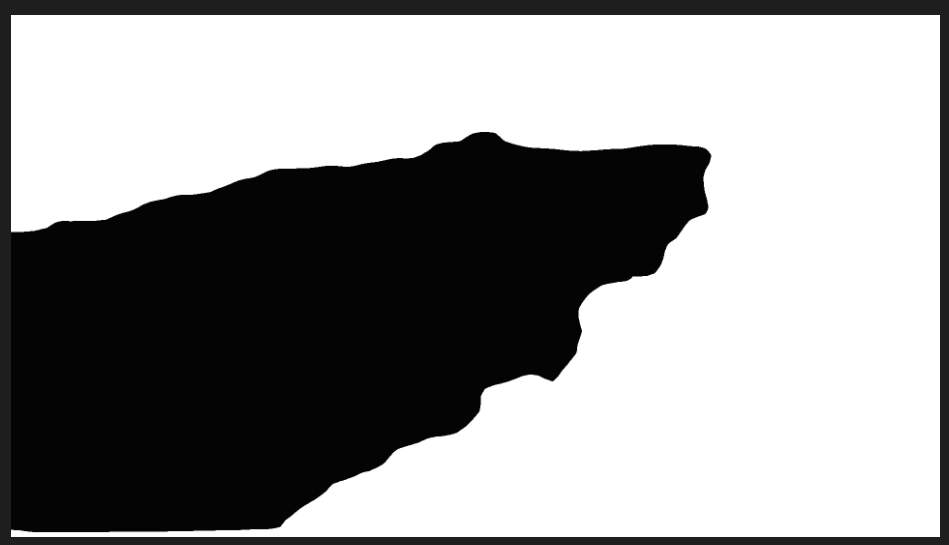


图15：处理后的预测结果图片

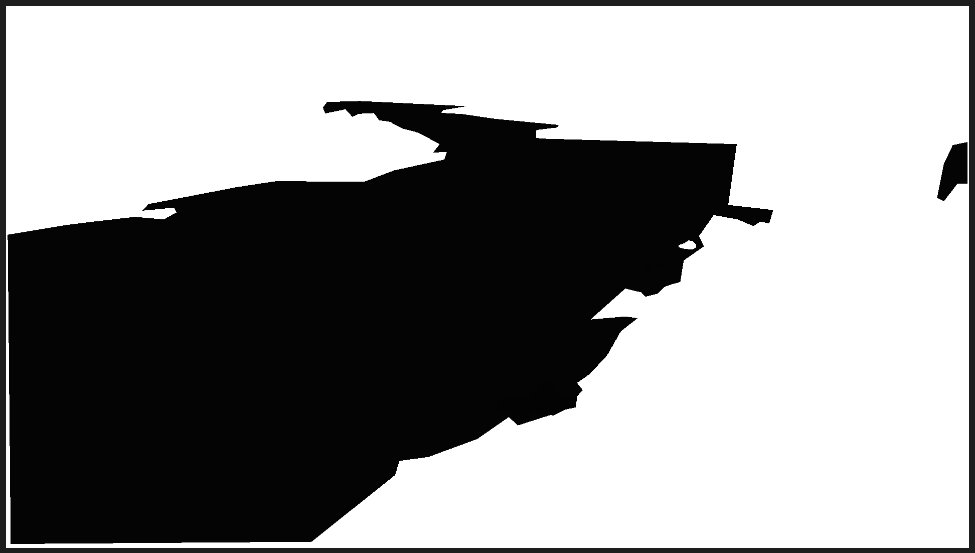


图16：平台所给的对应标注文件进行相同处理的结果

图13的预测结果如图14所示，由于像素值大小的原因，我们肉眼无法看出，于是根据图13的输出结果编写了一段代码对预测结果图像进行了处理，方便肉眼查看预测结果，处理后的图片如图15所示，图16展示了平台所提供的对应标注文件进行相同处理的结果。

通过图15和图16可以看出，模型的预测结果具有一定的可靠性和准确性，值得注意的是，这只是相当于在本地训练的模型的预测结果，而不是发起训练任务训练好的模型的预测结果。

3.5 发起测试任务

为了让平台测试训练好的模型，我们需要编写一个ji.py文件，平台发起测试时，系统会调用文件/project/ev\_sdk/src/ji.py，并将测试图片逐次送入process\_image接口，并将输出MASK输出到给定的路径，按照平台定义的相关接口代码如下：

import json

from PIL import Image

import numpy as np

import cv2

from deeplab import DeeplabV3

def init():

deeplab = DeeplabV3()

return deeplab

def process\_image(handle=None,input\_image=None,args=None, \*\*kwargs):

name\_classes = ["background","algae","dead\_twigs\_leaves","garbage","water"]

args =json.loads(args)

mask\_output\_path =args['mask\_output\_path']

# Process image here

# Generate dummy mask data

image = Image.fromarray(cv2.cvtColor(input\_image, cv2.COLOR\_BGR2RGB))

pred\_mask\_per\_frame = handle.detect\_image(image, count=True, name\_classes=name\_classes)

pred\_mask\_per\_frame.save(mask\_output\_path)

return json.dumps({'mask': mask\_output\_path}, indent=4)

与本地预测的predect.py一样，其本质均是通过创建一个DeepLab V3+模型类的对象，再调用deeplab.py中定义的detect\_image( )方法对输入图片实现预测。

ji.py编写完成后，便可通过加载训练任务训练好的模型新建测试任务，测试成功完成后，系统会给出评分。

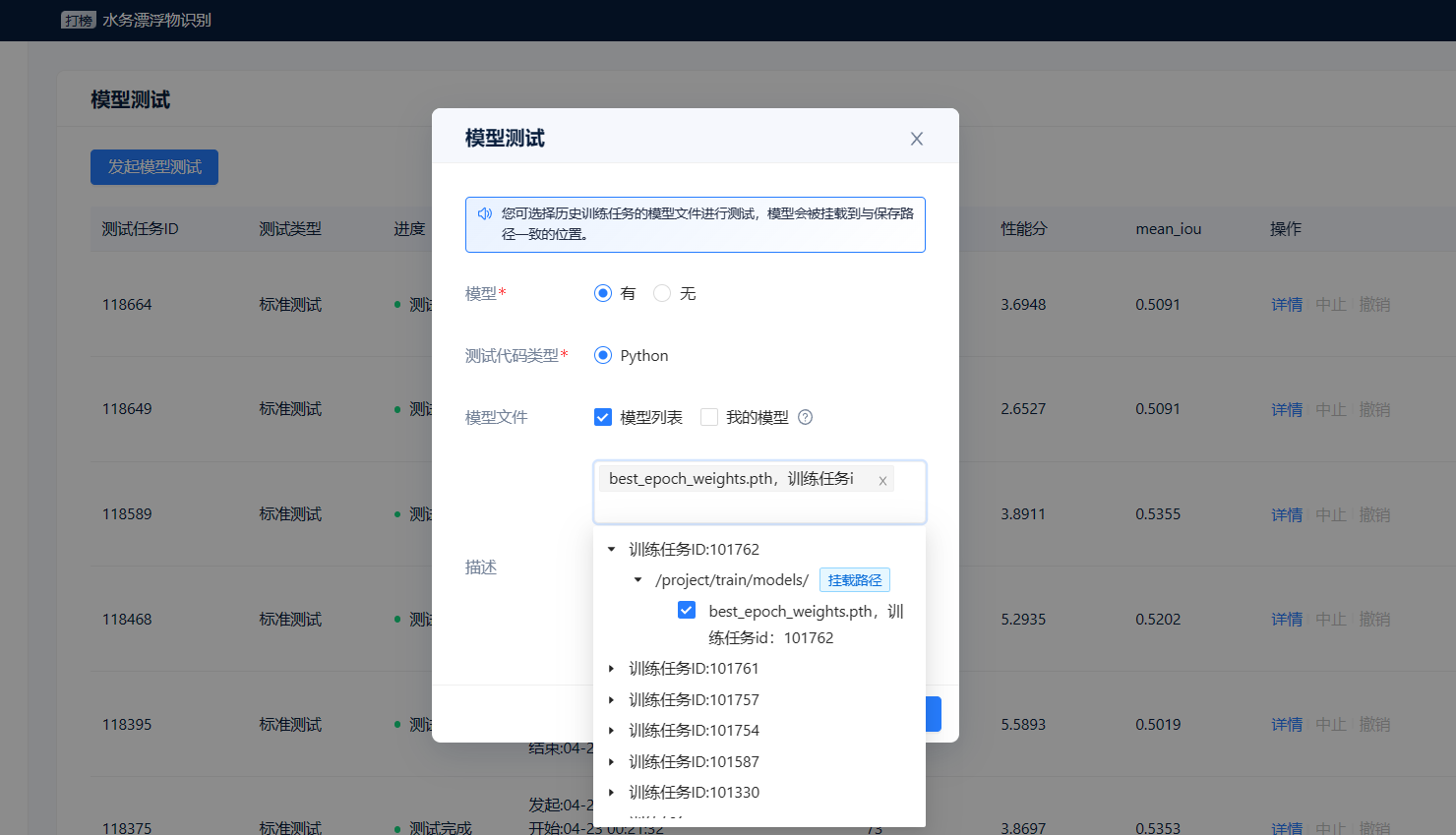
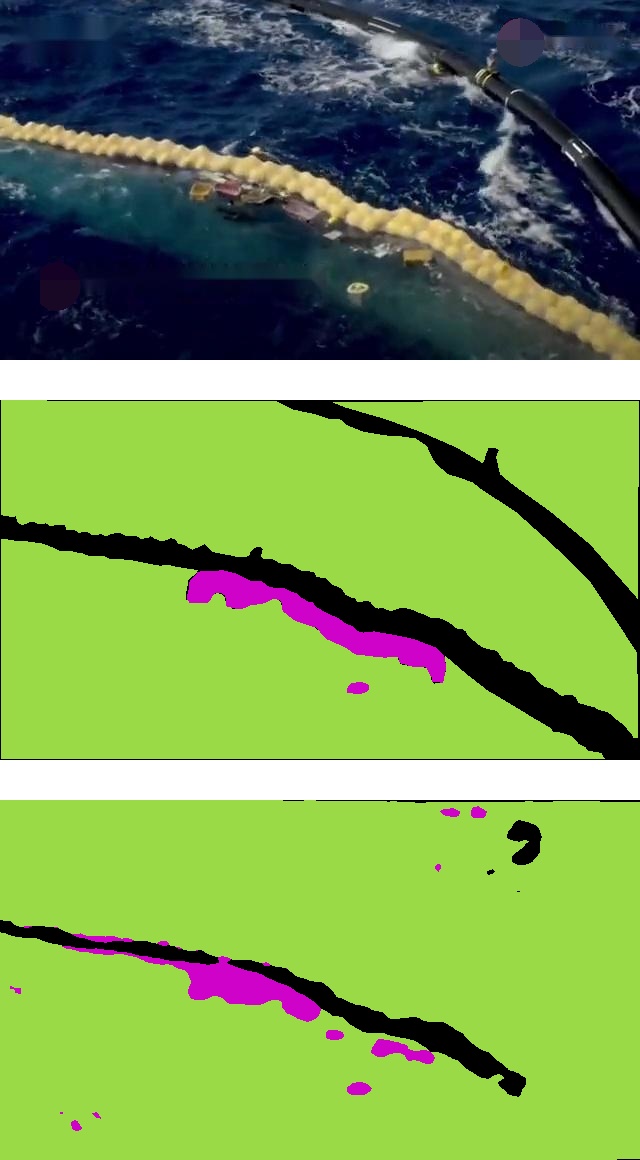


图17：新建测试任务

**四、实验结果**

4.1 部分样例集推理结果展示

通过极市平台的测试任务，我们可以查看样例集推理结果，如图18所示。



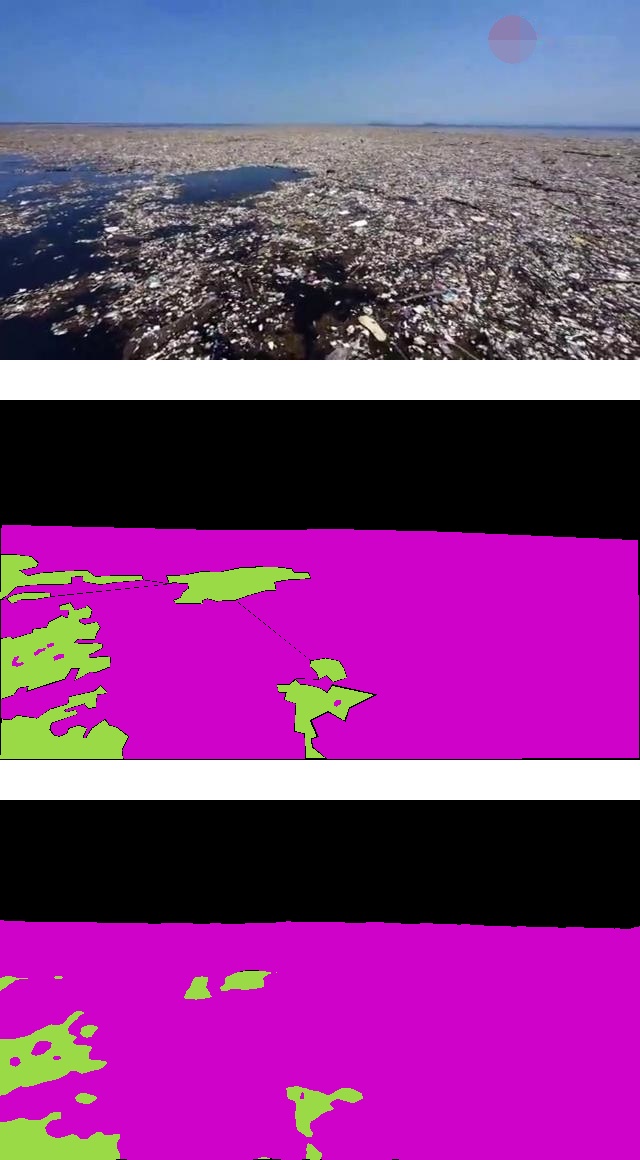




图18：部分样例集推理结果

4.2 最终得分排名展示

经过多轮训练和测试，截止2023年4月27日，我们组最终成绩得分及排名如图19所示：



图19：最终成绩得分及排名

**五、评价分析**

5.1实验分工：

本次实验我们小组并没有特别明确的分工，从开始到结束一直都是在一起完成，一起讨论，从开始的错误以为这个任务市一个目标检测任务，到确定使用DeepLab V3+模型来进行图像语义分割建模，我们在宿舍大厅讨论了两晚最终敲定方案。然后我们一起查阅资料，找到了一个开源代码，并将其应用到这次实验任务中。由于对接口定义比较陌生，除了训练模型以外，本次实验中最困难的部分便是定义测试接口，这花费了我们挺多时间。后面我们便各自调整参数，训练并测试模型。图20和图21是我的一些平台截图记录。

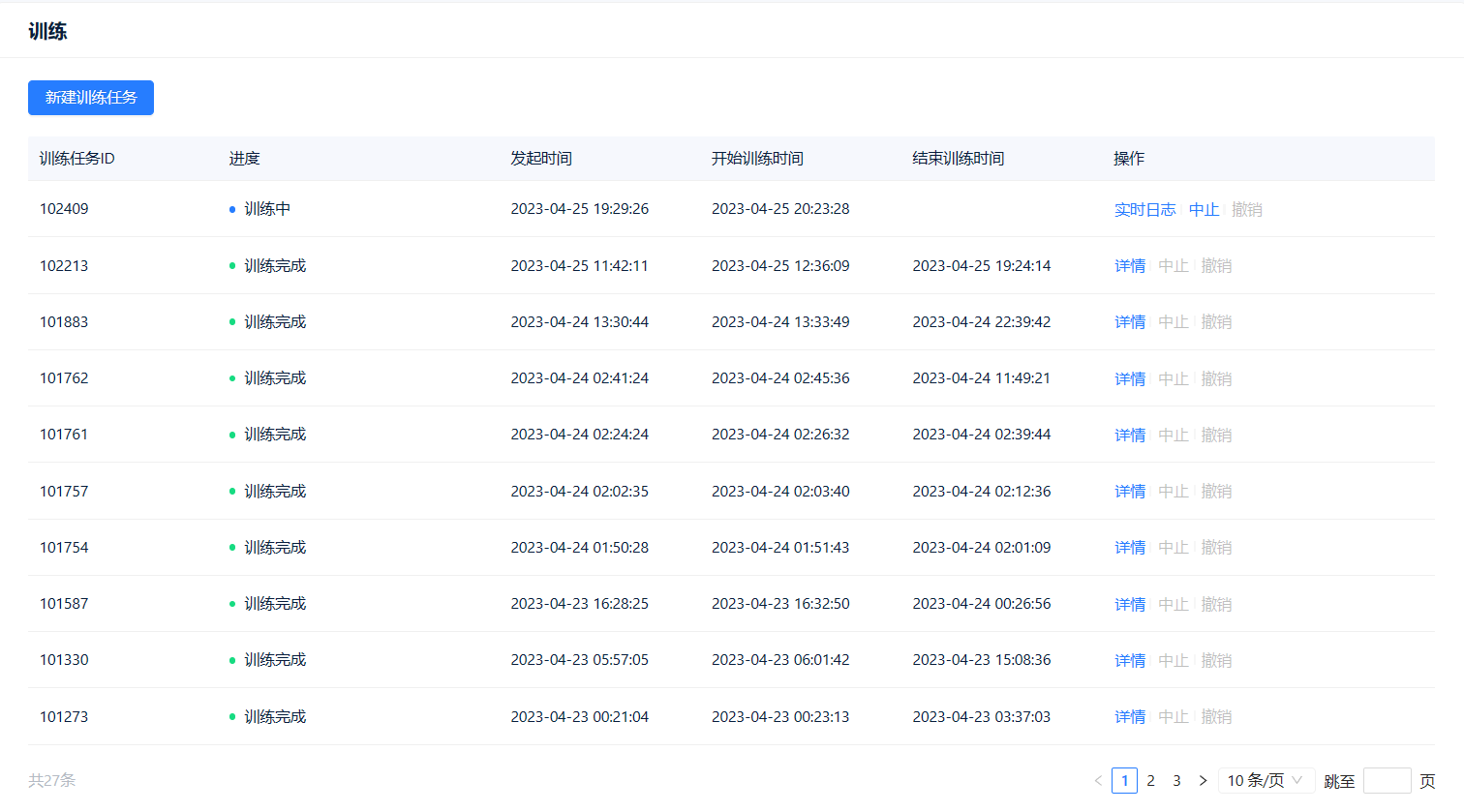


图20：极市平台训练记录

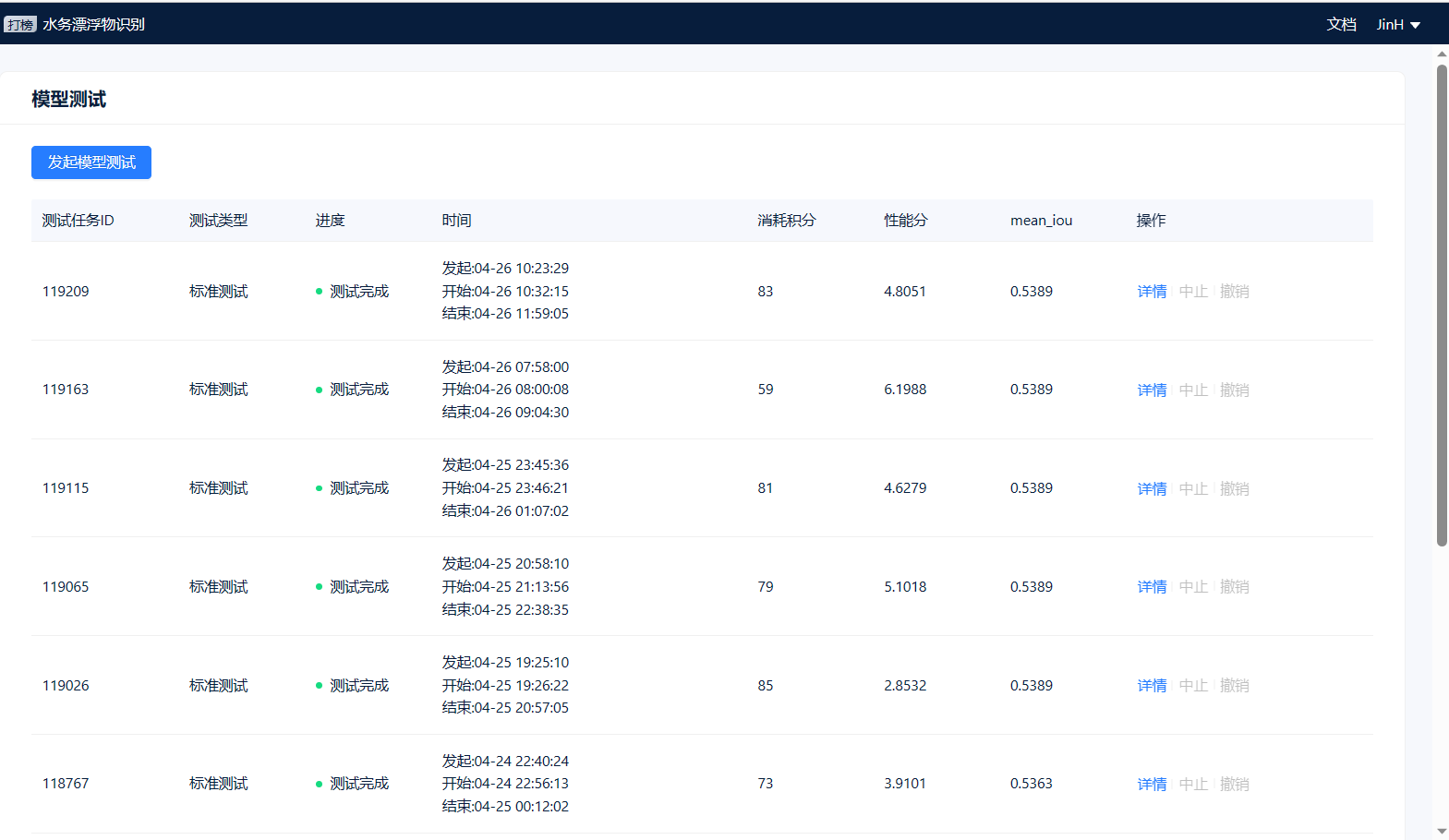


图21：极市平台测试记录

5.2分析与总结：

首先，本次实验让我对目标识别和语义分割的区别有了较为深刻的认识。因为在一开始，受极市平台提供的训练套件的影响，考虑使用yolov5模型，但这需要xml格式的标注文件，而平台没有提供，后来才发现这是一个语义分割任务。对于开源的代码，调试跑通并不难，解读理解也不是一个很大的问题。

本实验的难点在于将模型搬到平台上经过平台的训练和测试。开始由于不熟悉，测试接口经过好长时间交流试错才完成编写，后来发现是我们把问题考虑复杂了，很多我们考虑的乱七八糟的东西平台测试时都会自己完成，并非需要我们人为地去定义，我们只需要按照说明并结合自身项目的实际情况简单加以修改即可。

通过本次实验，我也对DeepLab V3+模型有了清晰的认知，在调通代码后，一边训练测试一边学习模型基本知识。在看到自己的成绩处于班级前列时，也是十分高兴，有成就感。同时，与好朋友们一起一步步解决问题的过程也十分有趣，是一次不错的经历。

其实在我们小组那两天卡在ji.py测试接口的定义时，我也尝试了另外一个任务：工服识别。这个模型利用平台提供的训练套件即可轻松解决，难度并不是很大，目前仅训练出来一个偏上的成绩（下图中排名第9），还没有深入了解代码逻辑。



**注：**本次实验使用的代码来源于csdn博主Bubbliiiing的开源资料。

源码地址：<https://github.com/bubbliiiing/deeplabv3-plus-pytorch>

博客地址：<https://blog.csdn.net/weixin_44791964/article/details/120113686>

DeepLab V3+论文下载地址：<https://arxiv.org/pdf/1802.02611.pdf>

**六、附1:参考文献**

1. 王蓝玉. 基于Deeplab V3+网络的遥感地物图像语义分割研究[D].哈尔滨工业大学,2020.DOI:10.27061/d.cnki.ghgdu.2020.001166.
2. 陆妍如,毛辉辉,贺琰等.基于Deeplab v3+的高分辨率遥感影像地物分类研究[J].地理空间信息,2022,20(06):1-6.
3. 孟琭,徐磊,郭嘉阳.一种基于改进的MobileNetV2网络语义分割算法[J].电子学报,2020,48(09):1769-1776.
4. 谢林江. 基于MobileNetV2和DeepLabV3+的Android人像背景虚化系统[D].西安电子科技大学,2020.DOI:10.27389/d.cnki.gxadu.2020.000711.
5. 刘文雅,岳安志,季珏等.基于DeepLabv3+语义分割模型的GF-2影像城市绿地提取[J].国土资源遥感,2020,32(02):120-129.
6. 杨威. 基于卷积神经网络的高效语义分割方法研究[D].中国科学院大学(中国科学院光电技术研究所),2019.

**七、附2：代码**

注：由于代码量较大，故不在此处粘贴代码了，详细代码见压缩包内的项目文件。下面会说明项目结构（仅详细说明本次实验所用到的内容）。

7.1水务识别模型整个项目结构：

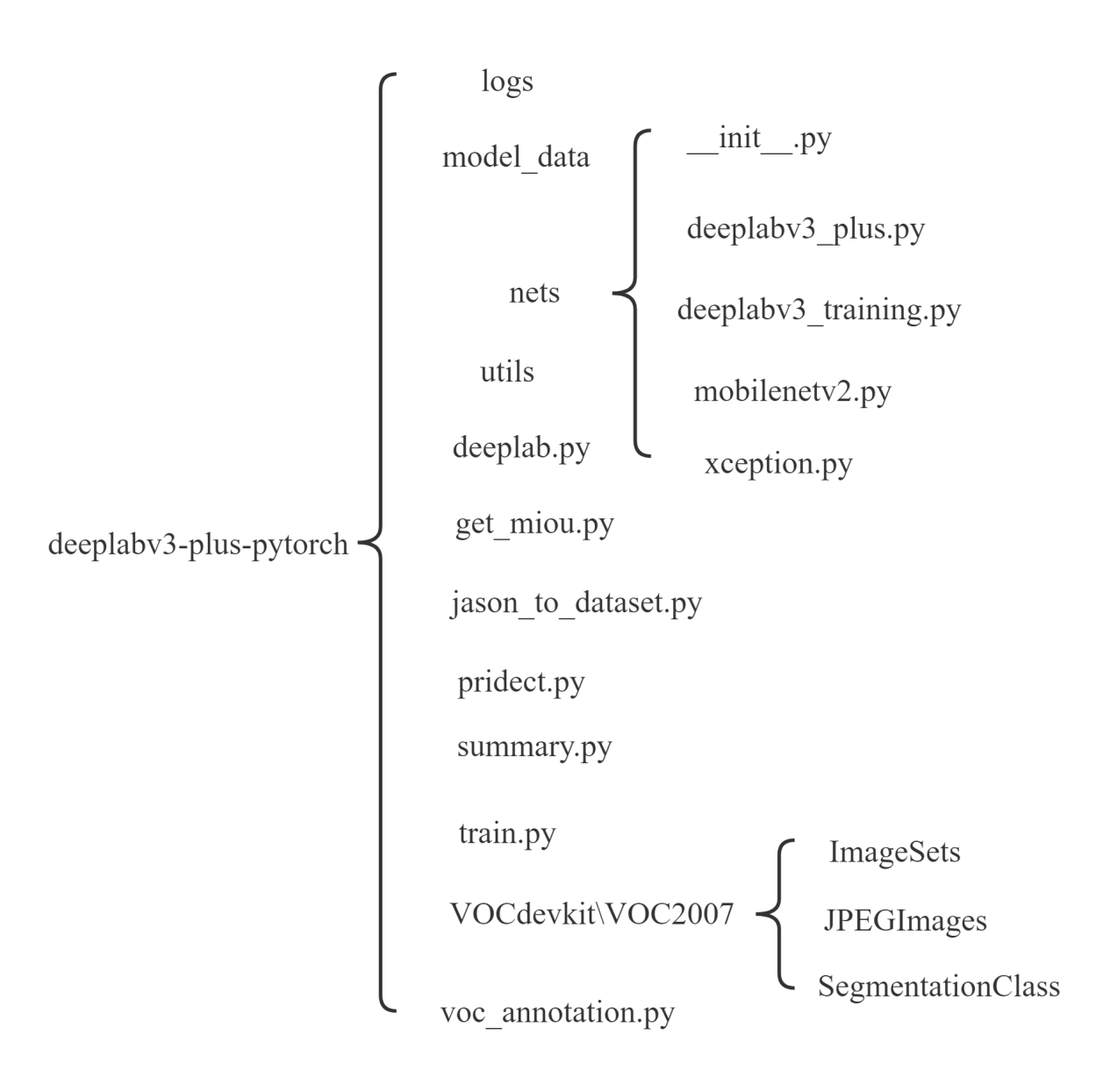


附图1： 平台整个project结构图

对于上述结构图，以下有几点说明：

1. project/ev\_sdk/src目录与平台实例中的/usr/local/ev\_sdk/src共享，内容完全一致，且目录中含有测试接口程序ji.py的定义，由于该接口的定义中存在模型初始化定义和调用deeplab.py中定义的预测方法，这就需要台哦用模型中给定义的一些包或者代码。而模型的的目录与ji.py测试程序的目录并不一致，为了ji.py成功调用模型的包，故将/project/train/src\_repo/deeplabv3-plus-pytorch目录下定义的一些内容直接复制到ji.py所在的目录下。
2. /project/train/models目录下需要预先放置一个预训练模型，否则测试脚本在本地检测会不通过，导致无法进一步在平台上发起完整数据集的训练任务。
3. 其余未提到的包或程序均有平台提供或者作用不大，此处不再赘述。
4. 有关模型所有代码均在/project/train/src\_repo/deeplabv3-plus-pytorch目录下，接下来我会详细说明其逻辑结构。

7.2 DeepLab V3+模型整个项目结构：



附图2： DeepLab V3+模型代码结构图

以下是对上述模型代码结构的一些详细说明：

1. logs目录下存放的是训练过程中产生的日志文件和保存的模型。
2. model\_data本实验中没有实际作用，此处不再说明。
3. nets目录下主要是DeepLab V3+模型的网络定义，其中deeplabv3\_plus.py中定义了主体模型；deeplabv3\_training.py中定义了模型训练部分；Mobilenetv2.py中定义了DeepLab V3+模型的DCNN部分，采用的是Mobilenetv2网络结构；xception.py中定义了另一种DCNN的网络结构，采用的是Xception网络结构，本次实验中并没有使用。
4. utils目录下存放了训练和测试有关的一些工具和函数的代码，此处不再详细说明。
5. deeplab.py中初始化了模型类并实现了预测方法detect\_image( )的定义，便于其他程序的调用。
6. get\_miou.py中定义的内容用于本地对测试集进行miou评价指标的计算，与平台后台测试计算的miou指标会稍有不同。
7. jason\_to\_dataset.py中用于对自己制作的VOC数据集进行转换，但本次实验中，由平台提供了标注好的数据集，故在实验中并无作用。
8. pridect.py通过调用deeplab.py中定义的detect\_image( )方法对输入图片实现预测，仅适用于本地预测，测试时预测调用的是ji.py中定义的接口。
9. summary.py用于查看网络结构。
10. train.py中初始化了一些模型超参数、优化器以及学习率调整器等信息，并在start\_train.sh脚本文件中通过python train.py对模型发起训练。
11. VOCdevkit/VOC2007中存放的是VOC格式的数据集，关于数据集详细说明在报告中“3.2 VOC数据集格式详细说明”中已经提到，此处不再赘述。
12. voc\_annotation.py文件是用于生成关于voc数据集指定图片的相关txt文件，并将生成的文件存放在VOCdevkit/VOC2007/ImageSets目录下，用于后续训练使用。值得注意的是，这个操作必须进行在train.py之前，故须在start\_train.sh脚本文件中通过python train.py对模型发起训练之前需要预先执行python voc\_annotation.py命令。