

# 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。水面上的漂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，威胁运作安全性；阻拦船只出航，威胁航运业安全性；破坏生态环境，威胁生活饮水安全。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 and 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学习中的实现原理：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。

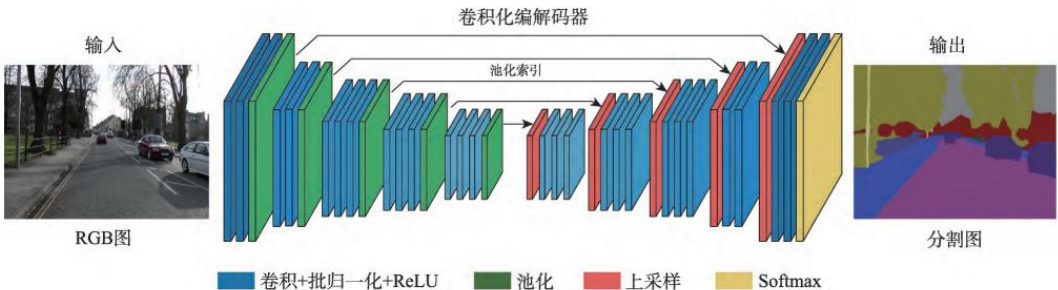


图 2：图像语义分割原理示意图

## 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。

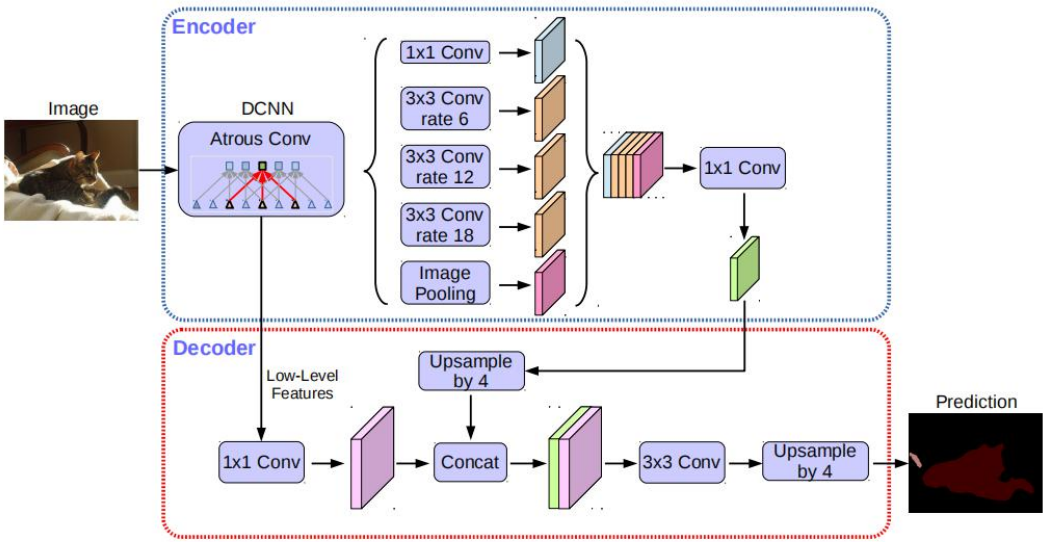


图 3：DeepLab V3+网络结构

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  $1 \times 1$  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  $1 \times 1$  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  $3 \times 3$  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，同时也可以保障特征表达能力的前提下减少模型大小，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 BatchNorm 和 Relu，然后利用 3\*3 深度可分离卷积进行特征提取，再进行 BatchNorm 和 Relu，然后再利用 1\*1 卷积降维，再进行一次 BatchNorm。右边是残差边部分，输入和输出直接相接。

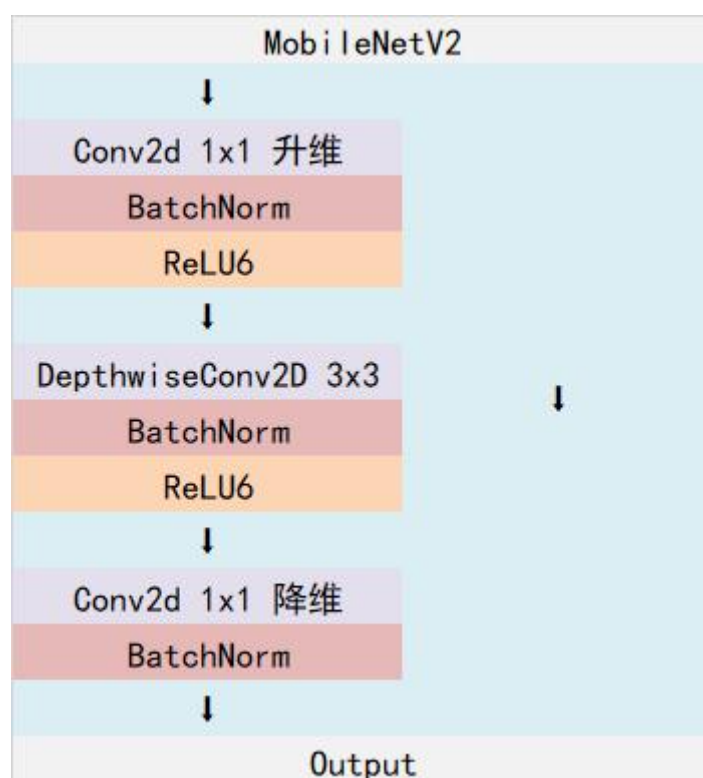


图 4: Inverted resblock

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。

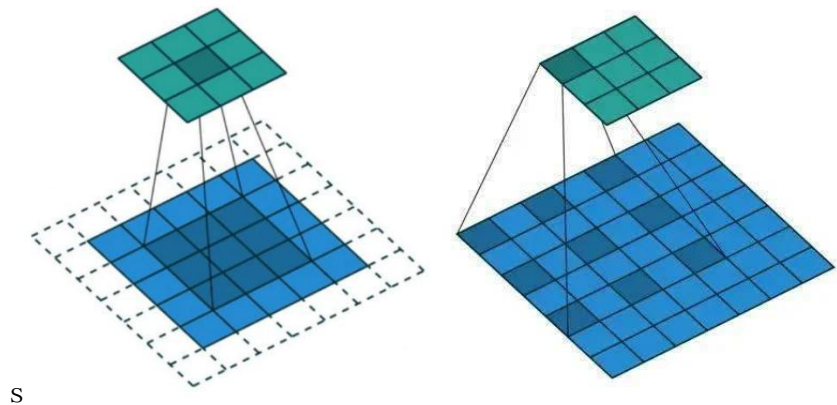


图 5: 普通卷积（左）和膨胀卷积（右）

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

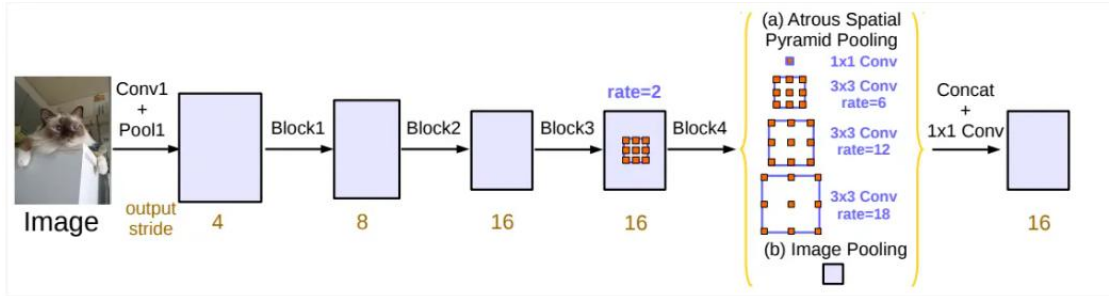


图 6: ASPP 结构



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：

分类结果混淆矩阵		
真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

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。

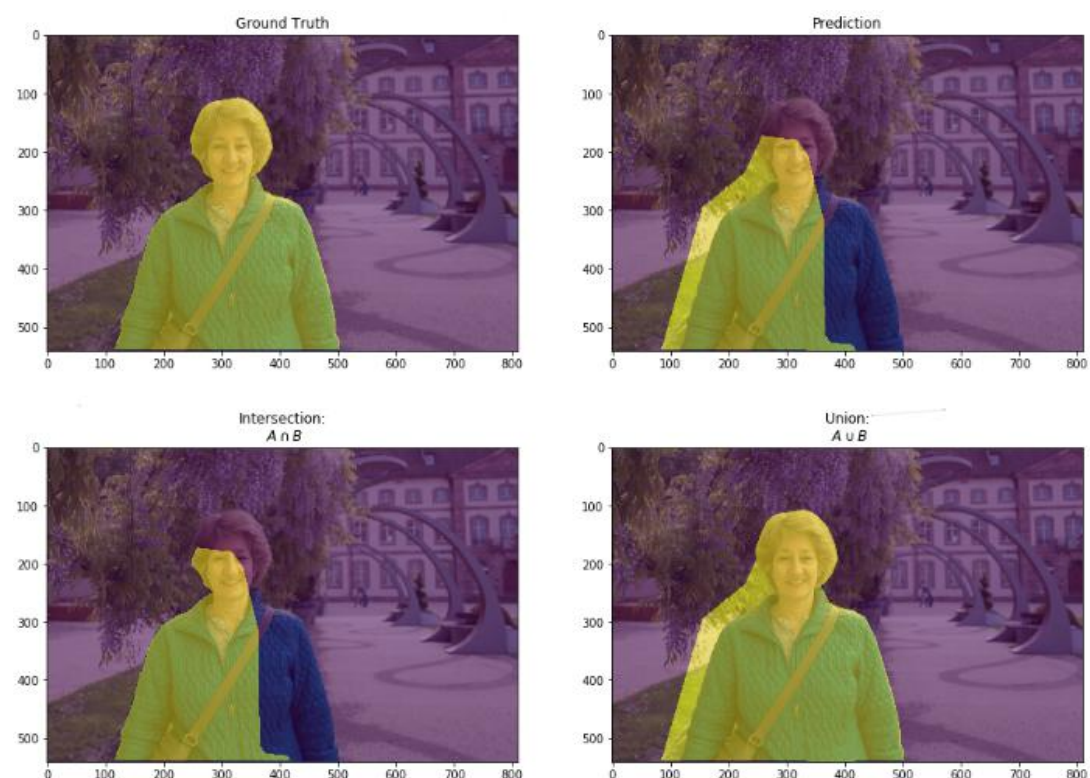


图 8：MIOU 示意图

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

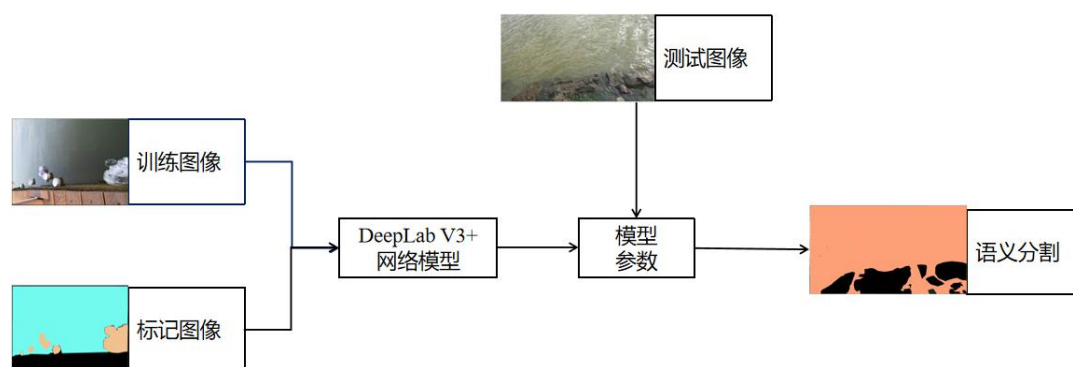


图 9：实验流程图

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。

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 GITLENS JUPYTER
[root@deeplabv3-plus-pytorch]$ python predict.py
/project/train/models/best_epoch_weights.pth model, and classes loaded.
Configurations:
-----
|          keys |          values|
|-----|-----|
| model_path | /project/train/models/best_epoch_weights.pth|
| num_classes | 6|
| backbone | mobilenet|
| input_shape | [512, 512]|
| downsample_factor | 16|
| mix_type | 0|
| cuda | True|
-----
Input image filename: /home/data/1945/ZDSfloating_objects20230206_V3_train_lakes_7_000012.jpg
-----
|          Key |          Value |          Ratio|
|-----|-----|-----|
|          background |          967281 |          46.65%|
|-----|-----|-----|
|          water |          1106319 |          53.35%|
|-----|-----|-----|
classes_nums: [ 967281.    0.    0.    0. 1106319.    0.]
<PIL.Image.Image image mode=L size=1920x1080 at 0x7FDD3B9F04D0>

```

图 12：在线实例预测单张图片





图 13 ： 被预测图片

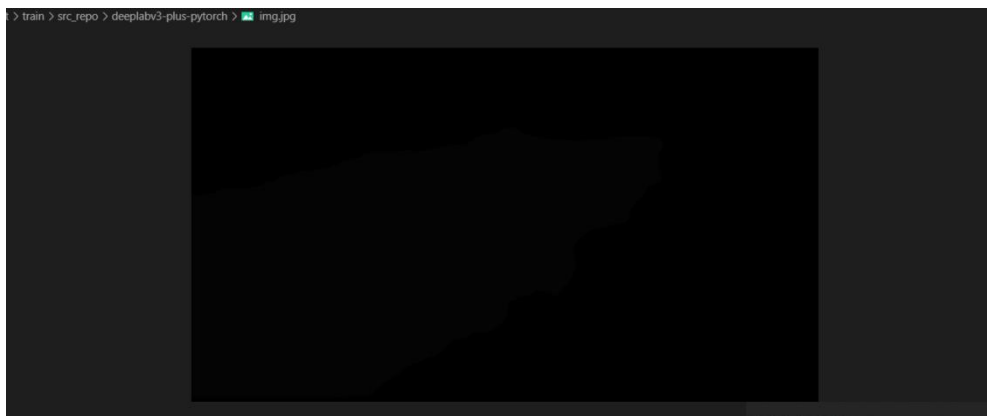


图 14 预测结果

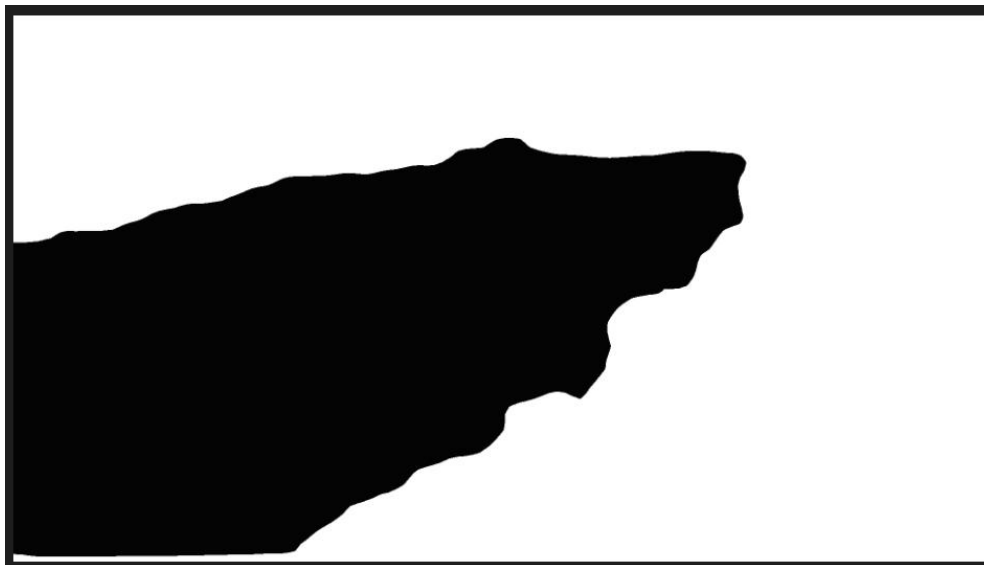


图 15： 处理后的预测结果图片

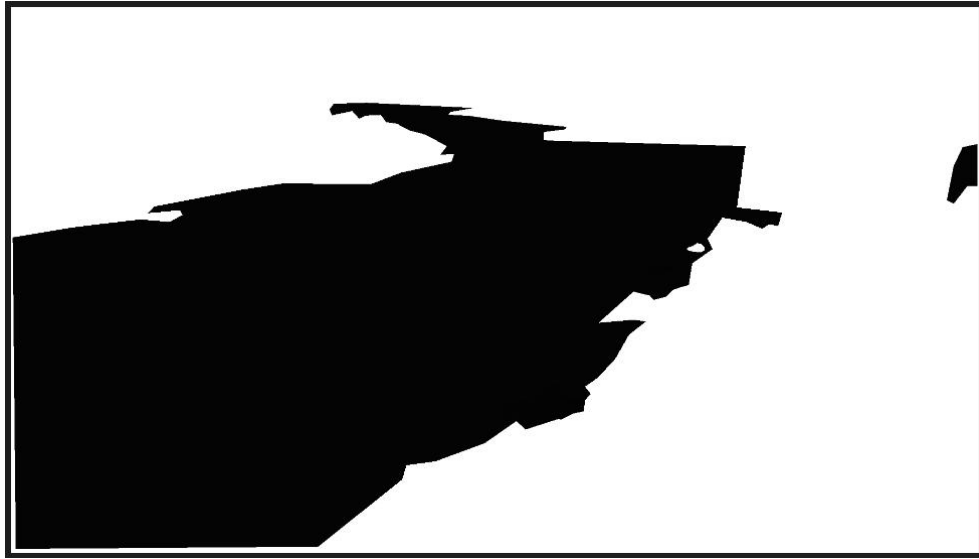


图 16：平台所给的对应标注文件进行相同处理的结果

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_BGR2RG
B))

    pred_mask_per_frame = handle.detect_image(image, count=True, name_class
es=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。

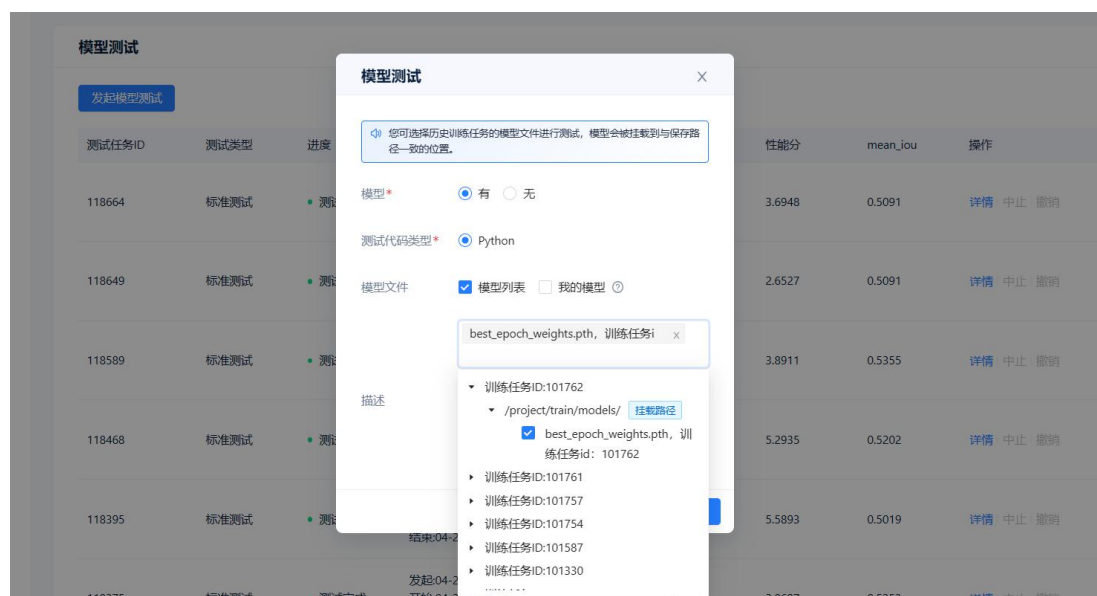


图 17：新建测试任务

## 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。

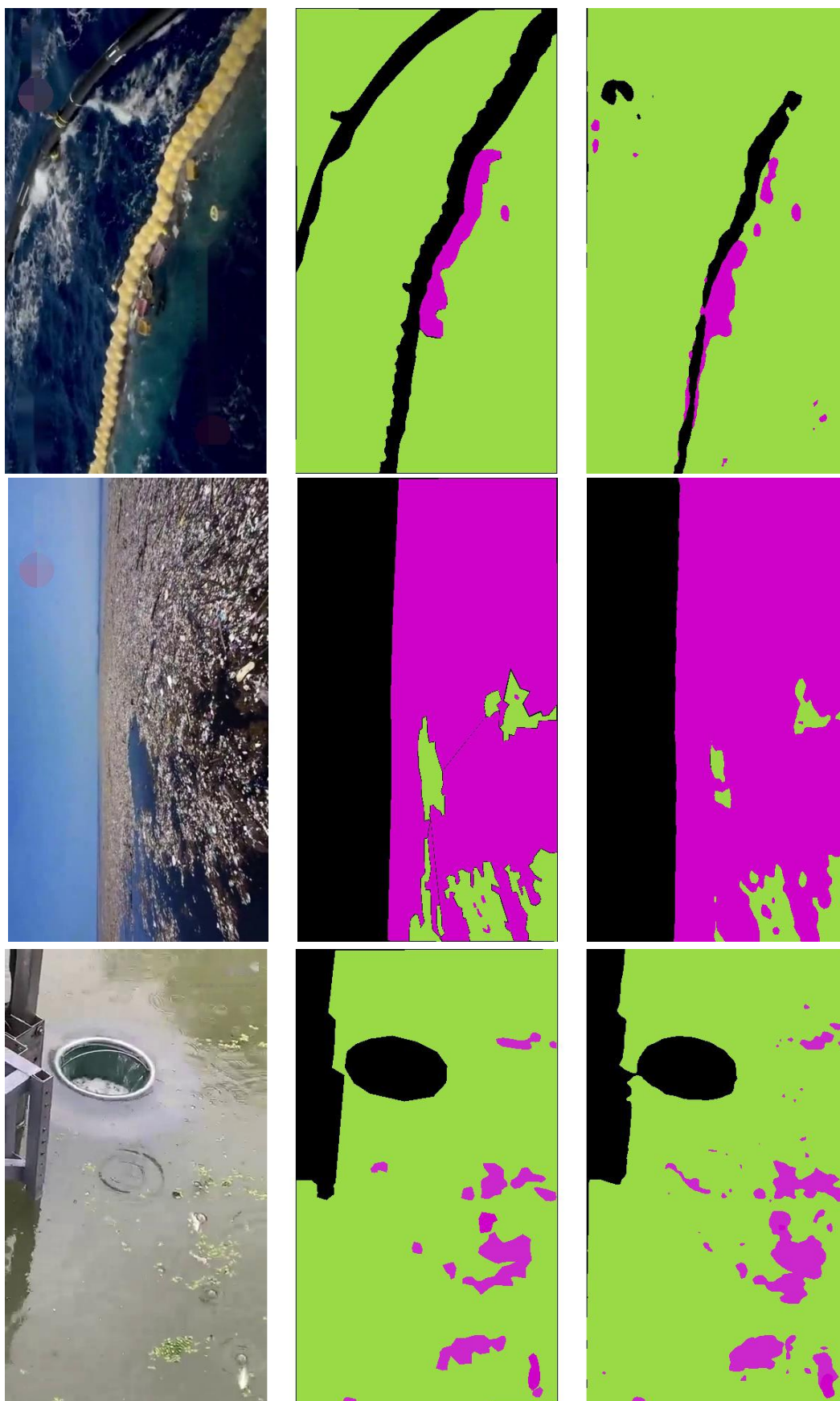


图 18：部分样例集推理结果

## 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：

04		coco1 T5	0.5911	0.6464	9.384
05		clwclw T5	0.5727	0.5966	35.8063
06		aer T4	0.5624	0.5613	57.318
07		zhangzhen T5	0.5209	0.5199	53.0225
08		很硬气的毕业 T1	0.4952	0.5483	1.7365
09		JinH T1	0.4912	0.5389	6.1988
10		jimfeng T1	0.4821	0.5309	4.3347

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是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。

训练					
新建训练任务					
训练任务ID	进度	发起时间	开始训练时间	结束训练时间	操作
102409	训练中	2023-04-25 19:29:26	2023-04-25 20:23:28		实时日志 中止 撤销
102213	训练完成	2023-04-25 11:42:11	2023-04-25 12:36:09	2023-04-25 19:24:14	详情 中止 撤销
101883	训练完成	2023-04-24 13:30:44	2023-04-24 13:33:49	2023-04-24 22:39:42	详情 中止 撤销
101762	训练完成	2023-04-24 02:41:24	2023-04-24 02:45:36	2023-04-24 11:49:21	详情 中止 撤销
101761	训练完成	2023-04-24 02:24:24	2023-04-24 02:26:32	2023-04-24 02:39:44	详情 中止 撤销
101757	训练完成	2023-04-24 02:02:35	2023-04-24 02:03:40	2023-04-24 02:12:36	详情 中止 撤销
101754	训练完成	2023-04-24 01:50:28	2023-04-24 01:51:43	2023-04-24 02:01:09	详情 中止 撤销
101587	训练完成	2023-04-23 16:28:25	2023-04-23 16:32:50	2023-04-24 00:26:56	详情 中止 撤销
101330	训练完成	2023-04-23 05:57:05	2023-04-23 06:01:42	2023-04-23 15:08:36	详情 中止 撤销
101273	训练完成	2023-04-23 00:21:04	2023-04-23 00:23:13	2023-04-23 03:37:03	详情 中止 撤销

共27条

< 1 2 3 > 10 条/页 跳至 页

图 20：极市平台训练记录

水务漂浮物识别							文档 jinh
模型测试							
发起模型测试							
测试任务ID	测试类型	进度	时间	消耗积分	性能分	mean_iou	操作
119209	标准测试	测试完成	发起:04-26 10:23:29 开始:04-26 10:32:15 结束:04-26 11:59:05	83	4.8051	0.5389	<a href="#">详情</a> <a href="#">中止</a> <a href="#">撤销</a>
119163	标准测试	测试完成	发起:04-26 07:58:00 开始:04-26 08:00:08 结束:04-26 09:04:30	59	6.1988	0.5389	<a href="#">详情</a> <a href="#">中止</a> <a href="#">撤销</a>
119115	标准测试	测试完成	发起:04-25 23:45:36 开始:04-25 23:46:21 结束:04-26 01:07:02	81	4.6279	0.5389	<a href="#">详情</a> <a href="#">中止</a> <a href="#">撤销</a>
119065	标准测试	测试完成	发起:04-25 20:58:10 开始:04-25 21:13:56 结束:04-25 22:38:35	79	5.1018	0.5389	<a href="#">详情</a> <a href="#">中止</a> <a href="#">撤销</a>
119026	标准测试	测试完成	发起:04-25 19:25:10 开始:04-25 19:26:22 结束:04-25 20:57:05	85	2.8532	0.5389	<a href="#">详情</a> <a href="#">中止</a> <a href="#">撤销</a>
118767	标准测试	测试完成	发起:04-24 22:40:24 开始:04-24 22:56:13 结束:04-25 00:12:02	73	3.9101	0.5363	<a href="#">详情</a> <a href="#">中止</a> <a href="#">撤销</a>

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。

排名	打榜人	成绩分	f-score	性能分	累计奖励
1	 sam007 T5	0.8499	0.8421	113.5333	
2	 jujiangluck T5	0.8357	0.8271	113.2651	100元
3	 aer T4	0.8147	0.8083	93.6581	100元
04	 zhangzhen T5	0.8095	0.8195	62.012	
05	 SemiLab T5	0.8029	0.8193	49.1346	
06	 coco1 T5	0.7854	0.8213	10.4796	
07	 AL_mzq T5	0.7845	0.7734	99.6465	
08	 xuehao101 T1	0.7435	0.7417	77.7827	
09	 JinH T1	0.7207	0.7245	64.8576	
10	 麦田里的守望者 T5	0.7015	0.7169	40.9462	

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

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

博客地址：[https://blog.csdn.net/weixin\\_44791964/article/details/120113686](https://blog.csdn.net/weixin_44791964/article/details/120113686)

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

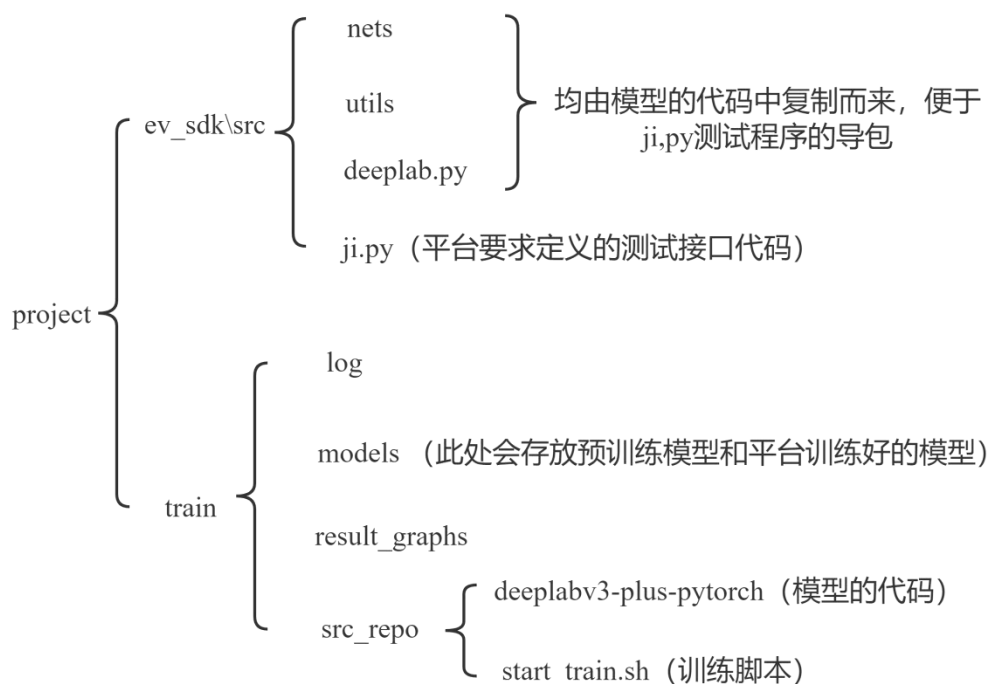
## 六、附 1:参考文献

- [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] 孟琰,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，这就需要调用模型中给定义的一些包或者代码。而模型的目录与 `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。

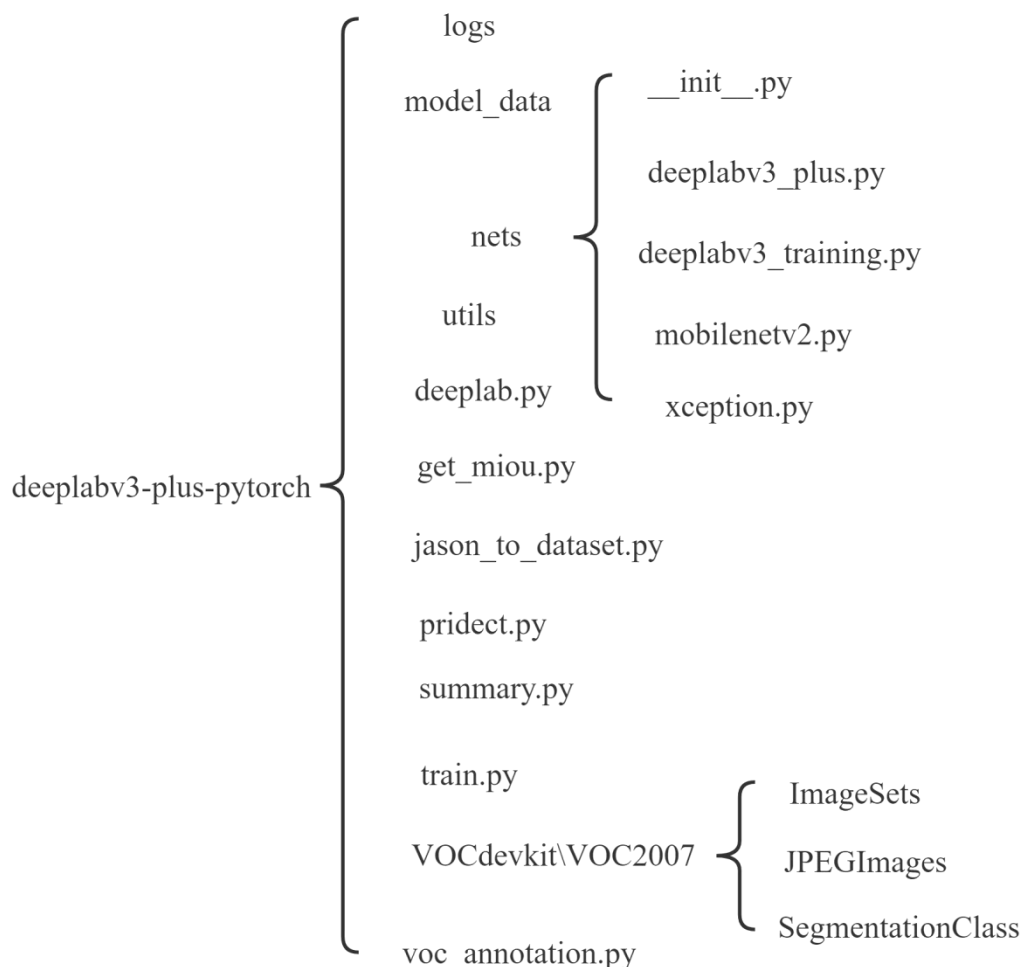
（2）`/project/train/models` 目录下需要预先放置一个预训练模型，否则测试脚本在本地检测会不通过，导致无法进一步在平台上发起完整数据集的训练任务。

（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。