



模式识别与机器学习

实验报告

班级：10042001

姓名：王旭升

学号：2020302708

目录

一、 实验目的	3
1.1 需求边界定义：水务场景下的垃圾	3
二、 实验原理	3
2.1 数据说明	3
2.2 图像语义分割	4
2.3 DeeplabV3+模型	4
2.3.1 整体架构	4
2.3.2 空洞卷积	5
2.3.3 空间金字塔池化（ASPP）	6
2.3.4 主干特征提取网络	6
三、 实验步骤和程序流程	7
3.1 流程图	7
3.2 数据预处理	7
3.3 数据增强	8
3.4 网络结构	9
3.5 训练结果预测	10
四、 实验结果	12
4.1 输出示例	12
4.2 测试结果	12
4.3 最终排名	13
五、 评价分析	13
5.1 实验分工与总结	13

附录

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。

- **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

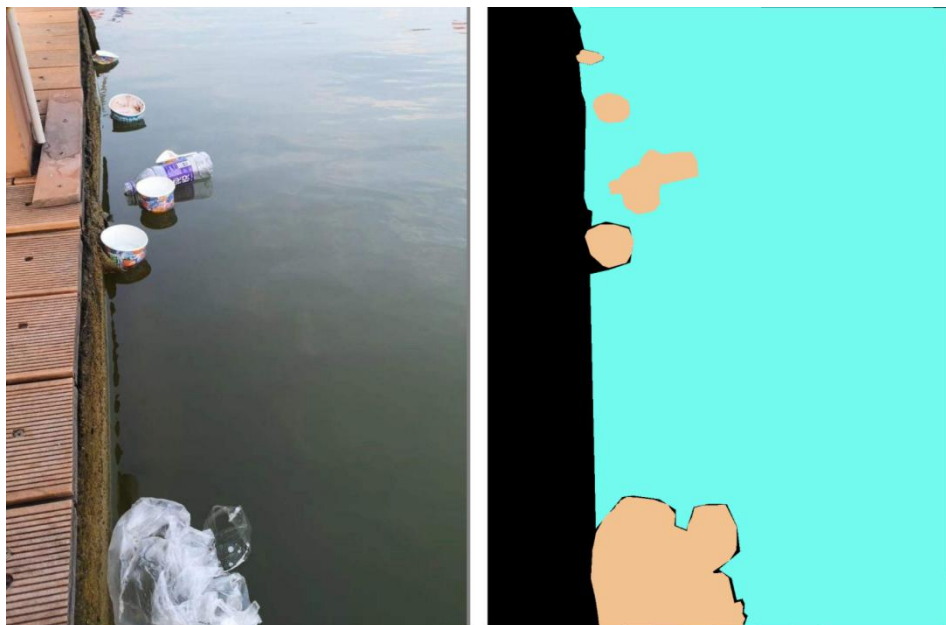
- **需求边界定义：**水务场景下的垃圾

- **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

- **识别场景：**水域

- **识别对象：**水体、垃圾

- **环境要求：**室外-白天



二、 实验原理

2.1 数据说明

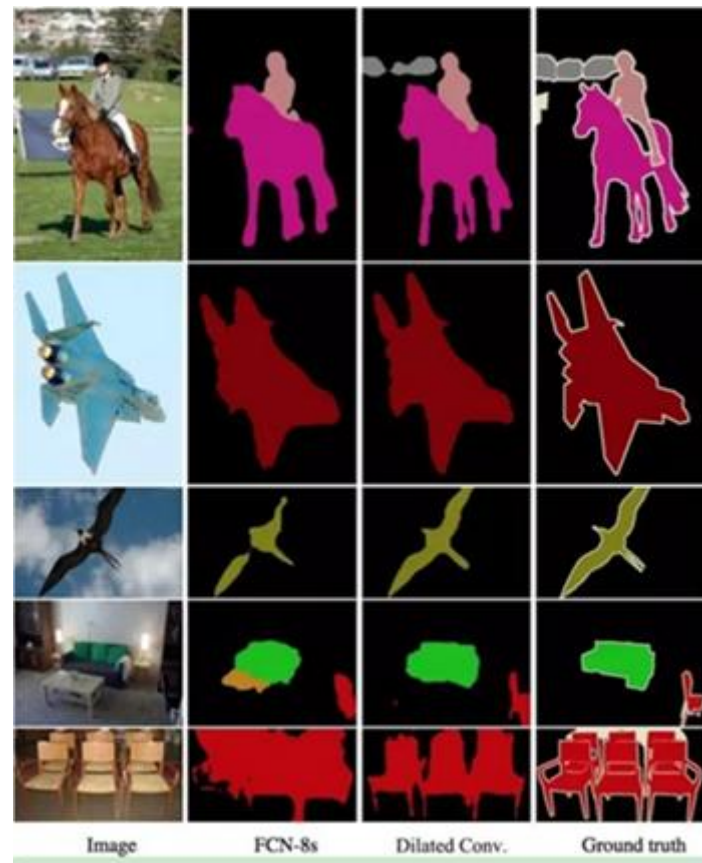
Mask

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

图像分割是计算机视觉中除了分类和检测外的另一项基本任务，它意味着要将图片根据内容分割成不同的块。相比图像分类和检测，分割是一项更精细的工作，因为需要对每个像素点分类，由于对每个像素点都分类，物体的轮廓是精准勾勒的，而不是像检测那样给出边界框。



上图为语义分割的一个实例，其目标是预测出图像中每一个像素的类标签。

图像语义分割是图像处理和计算机视觉技术中关于图像理解的重要的一环。语义分割对图像中的每一个像素点进行分类，确定每个点的类别（如属于背景、边缘或身体等）

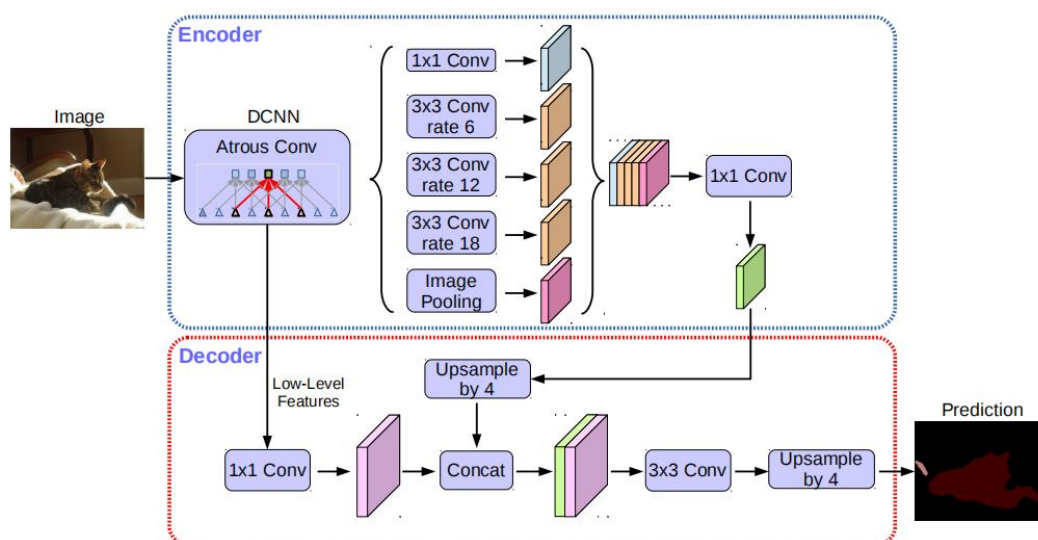
需要和实例分割区分开来。语义分割没有分离同一类的实例；它关心的只是每个像素的类别，如果输入对象中有两个相同类别的对象，则分割本身不会将它们区分为单独的对象。

2.3 DeeplabV3+模型

2.3.1 整体架构

DeeplabV3+模型的整体架构如下图所示，它的 Decoder 的主体是带有空洞卷积的 DCNN，可以采用常用的分类网络如 ResNet，然后是带有空洞卷积的空间金字塔池化模块 (Atrous Spatial Pyramid Pooling, ASPP)，主要是为了引入多尺度信息；相比 DeeplabV3+，V3+引入了 Decoder 模块，其将底层特征与高层特征进一步融合，提升分割边界准确度。从某种意义上看，DeeplabV3+在 DilatedFCN

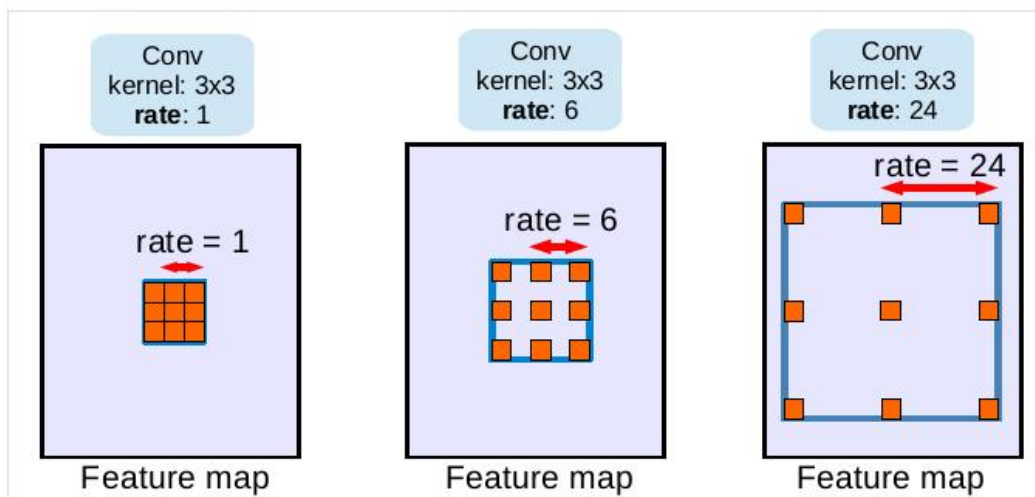
基础上引入了 EcdoderDecoder 的思路。



- 对于编码器部分，实际上就是 DeeplabV3+网络。图像进入主干网络后，获得两个特征层，浅层的特征层直接进入 decoder 中进行 1*1 卷积进行通道压缩，目的是减少低层级的比重。深层特征则在 encoder 编码器中进入 ASPP 模块，论文认为编码器得到的 feature 具有更丰富的信息，所以编码器的 feature 应该有更高的比重。
- 在 ASPP 模块中，包括了 1 个 1*1 卷积、3 个 3*3 的 Atrous convolution 分别比率为 6、12、18，以及一个图像全局的 Pooling 操作，这些操作过后是 1 个 1*1 卷积。
- 对于解码器部分，直接将编码器的输出上采样 4 倍，使其分辨率和低层级的 feature 一致。将两个特征层连接后，再进行一次 3×3 的卷积（细化作用），然后再次上采样就得到了像素级的预测。
- 1*1 卷积的作用是升维或降维，使其与要结合的特征层保持一致。

2.3.2 空洞卷积

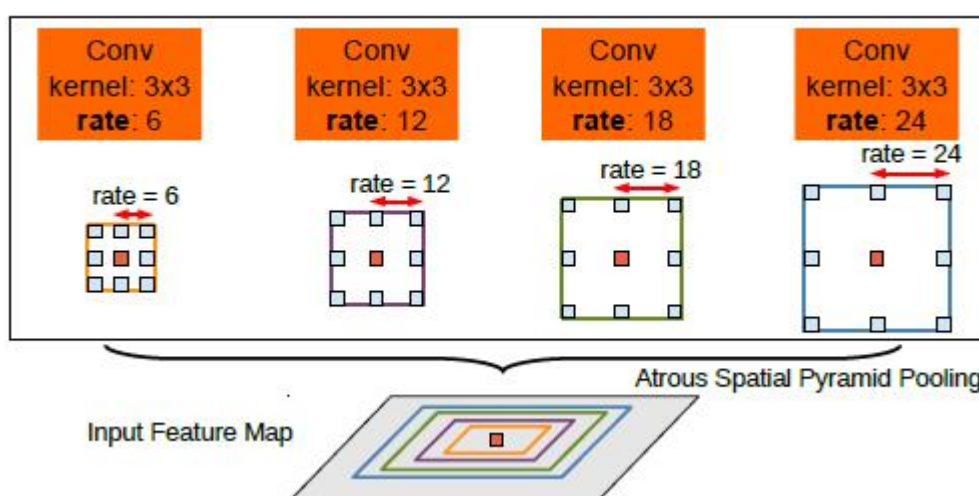
空洞卷积 (Dilated/Atrous Convolution) 也叫扩张卷积或者膨胀卷积，字面意思上来说就是在卷积核中插入空洞，起到扩大感受野的作用。



空洞卷积是 DeepLab 模型的关键之一，它可以在不改变特征图大小的同时控制感受野，这有利于提取多尺度信息。空洞卷积如上图所示，其中 rate (r) 控制着感受野的大小， r 越大感受野越大。

2.3.3 空间金字塔池化 (ASPP)

在 DeepLab 中，采用空间金字塔池化模块来进一步提取多尺度信息，这里是采用不同 rate 的空洞卷积来实现这一点。ASPP 模块主要包含以下几个部分：



- (1) 一个 1×1 卷积层，以及三个 3×3 的空洞卷积，对于 $\text{output_stride}=16$ ，其 rate 为 (6, 12, 18)，若 $\text{output_stride}=8$ ，rate 加倍（这些卷积层的输出 channel 数均为 256，并且含有 BN 层）；
- (2) 一个全局平均池化层得到 image-level 特征，然后送入 1×1 卷积层（输出 256 个 channel），并双线性插值到原始大小；
- (3) 将 (1) 和 (2) 得到的 4 个不同尺度的特征在 channel 维度 concat 在一起，然后送入 1×1 的卷积进行融合并得到 256-channel 的新特征。

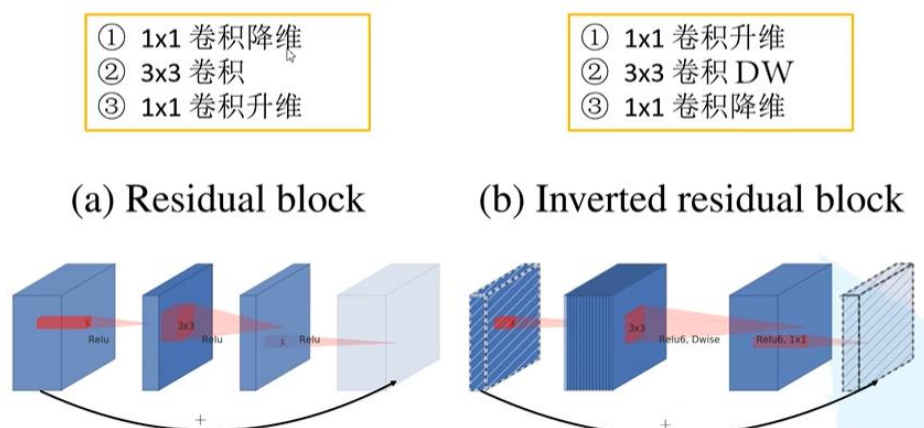
2.3.4 主干特征提取网络

DeeplabV3+在论文中采用的是 Xception 系列作为主干特征提取网络，但是由

于算力限制，我们采用了 MobileNet V2 作为主干特征提取网络。

MobileNet V2 网络是由 Google 团队在 2018 年提出的，相比 MobileNet V1 网络，准确率更高，模型更小。

网络中的亮点就是使用了 Inverted resblock（倒残差结构）

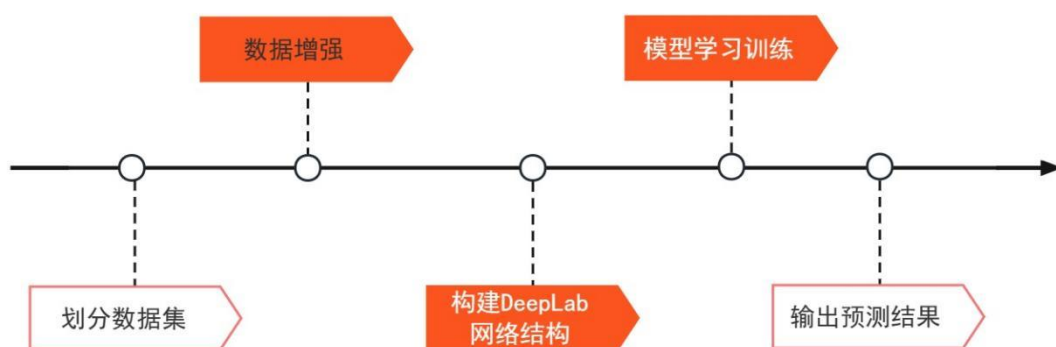


ResNet 残差结构是先用 1x1 的卷积降维，再升维的操作；而在 MobileNetV2 中，是先升维，在降维的操作。所以对于 ResNet 残差结构是两头大，中间小。而对于 MobileNetV2 结构是中间大，两头小的结构。

三、 实验步骤和程序流程

- 环境：Python3.7.5 + VSCode
- 框架：PyTorch1.10

3.1 流程图



3.2 数据预处理


```

48         for i in list:
49             name = total_seg[i][:-4]+'\\n'
50             if i in trainval:
51                 ftrainval.write(name)
52                 if i in train:
53                     ftrain.write(name)
54             else:
55                 fval.write(name)
56         else:
57             ftest.write(name)

```

首先将数据划分为训练集与验证测试集，将每个集中的图像名称写入相应的文本文件，以便后续调用。然后，循环遍历所有分割图像并检查它们是否存在、是灰度或 8 位彩色图像以及是否具有有效的像素值来检查数据集的格式。

```

34         #-----#
35         # 数据增强
36         #-----#
37         # 训练时候对数据进行随机增强，但验证时不进行
38         jpg, png = self.get_random_data(jpg, png, self.input_shape, random = self.train)
39         # 将图片转换成神经网络所能接受的格式，即将图片转换成矩阵，并且把颜色通道放在第一维
40         jpg = np.transpose(preprocess_input(np.array(jpg, np.float64)), [2,0,1])
41         png = np.array(png)
42         png[png >= self.num_classes] = self.num_classes
43         #-----#
44         # 转化成one_hot的形式
45         # 在这里需要+1是因为voc数据集有些标签具有白边部分
46         # 我们需要将白边部分进行忽略，+1的目的是方便忽略。
47         #-----#
48         seg_labels = np.eye(self.num_classes + 1)[png.reshape([-1])]
49         seg_labels = seg_labels.reshape((int(self.input_shape[0]), int(self.input_shape[1]), self.num_classes + 1))

```

训练时候对数据进行随机增强，但验证时不进行。并将图片转换成神经网络所能接受的格式，即将图片转换成矩阵，并且把颜色通道放在第一维。最后，将标签图片转化成 one-hot 编码的形式，方便计算交叉熵损失。

3.3 数据增强

数据增强常用方法：

- 缩放，进行长和宽的扭曲
- 左右翻转
- 旋转
- 高斯模糊
- HSV 色域变换

针对图片缩放产生的失真问题，我们通过给图像增加灰条的方式，以实现不失真的 resize。


```

#-----#
# 对输入图像进行resize
#-----#
def resize_image(image, size):
    iw, ih = image.size
    w, h = size

    scale = min(w/iw, h/ih)
    nw = int(iw*scale)
    nh = int(ih*scale)

    image = image.resize((nw,nh), Image.BICUBIC)
    new_image = Image.new('RGB', size, (128,128,128))
    new_image.paste(image, ((w-nw)//2, (h-nh)//2))

    return new_image, nw, nh

```

先得到 scale 这个较小的宽高比，然后对图片进行缩放，new_image 是一张灰色的新图，然后将缩放好的图像粘贴到灰色图像上，覆盖掉一部分，那么剩下的部分就是灰条。

这种方法在保证图像质量的同时，可以抑制因缩放引起的锐化等失真问题。

3.4 网络结构

```

169 def forward(self, x):
170     H, W = x.size(2), x.size(3)
171     #-----#
172     # 获得两个特征层
173     # low_level_features: 浅层特征-进行卷积处理
174     # x : 主干部分-利用ASPP结构进行加强特征提取
175     #-----#
176     low_level_features, x = self.backbone(x)
177     x = self.aspp(x)
178     low_level_features = self.shortcut_conv(low_level_features)
179
180     #-----#
181     # 将加强特征边上采样
182     # 与浅层特征堆叠后利用卷积进行特征提取
183     #-----#
184     x = F.interpolate(x, size=(low_level_features.size(2), low_level_features.size(3)), mode='bilinear', align_corners=True)
185     x = self.cat_conv(torch.cat((x, low_level_features), dim=1))
186     x = self.cls_conv(x)
187     x = F.interpolate(x, size=(H, W), mode='bilinear', align_corners=True)
188     return x

```

首先，获取输入张量 x 的尺寸，即图像的高度和宽度。然后，通过调用 self.backbone 函数将输入张量 x 传入主干部分，得到浅层特征 low_level_features 和加强特征 x。

```

128 elif backbone=="mobilenet":
129     #-----#
130     # 获得两个特征层
131     # 浅层特征 [128,128,24]
132     # 主干部分 [30,30,320]
133     #-----#
134     self.backbone = MobileNetV2(downsample_factor=downsample_factor, pretrained=pretrained)
135     in_channels = 320
136     low_level_channels = 24
137 else:
138     raise ValueError('Unsupported backbone - `{}`, Use mobilenet, xception.'.format(backbone))
139
140 #-----#
141 # ASPP特征提取模块
142 # 利用不同膨胀率的膨胀卷积进行特征提取
143 #-----#
144 self.aspp = ASPP(dim_in=in_channels, dim_out=256, rate=16//downsample_factor)

```

我们使用 MobileNet 作为神经网络的主干部分，网络的深度为 22 层，并

且由于 MobileNet 的特殊结构,网络能够使用较少的参数来实现相当好的效果。

```
146 #-----#
147 # 浅层特征边
148 #-----#
149 self.shortcut_conv = nn.Sequential(
150     nn.Conv2d(low_level_channels, 48, 1),
151     nn.BatchNorm2d(48),
152     nn.ReLU(inplace=True)
153 )
```

接下来,利用 ASPP 模块对加强特征进行加强,利用 shortcut 模块对浅层特征进行卷积处理,使其通道数变为 48。之后,将加强特征通过上采样操作,使其与浅层特征形状一致。

```
155 self.cat_conv = nn.Sequential(
156     nn.Conv2d(48+256, 256, 3, stride=1, padding=1),
157     nn.BatchNorm2d(256),
158     nn.ReLU(inplace=True),
159     nn.Dropout(0.5),
160
161     nn.Conv2d(256, 256, 3, stride=1, padding=1),
162     nn.BatchNorm2d(256),
163     nn.ReLU(inplace=True),
164
165     nn.Dropout(0.1),
166 )
167 self.cls_conv = nn.Conv2d(256, num_classes, 1, stride=1)
```

此外,在上采样时,使用了双线性插值(mode='bilinear')的方式,以保持图像的平滑性。再将加强特征和浅层特征在通道维度上连接起来,并使用 cat_conv 对连接后的特征进行特征提取。

最后,对输出特征进行上采样操作,使其恢复为输入张量的原始尺寸,并使用一个 $1 \times 11 \times 1$ 的卷积层 cls_conv 将特征映射到指定的分类数目上。

整个网络结构包括主干部分、ASPP 模块、shortcut 模块和分类模块,通过这些模块逐步提取和融合多尺度的特征,最终输出精细的图像分割结果。

3.5 训练结果预测

由于平台需要,我们按照要求实现如下了函数接口(ji.py):

input_image (numpy.ndarray): image to be process, format: (h, w, c), BGR

在 OpenCV 中,它默认使用的颜色空间是 BGR,而在 PIL 中,它默认使用的颜色空间是 RGB。因此,在进行图像处理时,我们需要先进行类型转换,以保证颜色空间一致。

```

9  def init():
10     deeplab = DeeplabV3()
11     return deeplab
12
13  def process_image(handle=None, input_image=None, args=None, **kwargs):
14     name_classes = ["background", "algae", "dead_twigs_leaves", "garbage", "water"]
15
16     args = json.loads(args)
17     mask_output_path = args['mask_output_path']
18
19     # Process image here
20     # Generate dummy mask data
21     image = Image.fromarray(cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB))
22     pred_mask_per_frame = handle.detect_image(image, count=True, name_classes=name_classes)
23     pred_mask_per_frame.save(mask_output_path)
24     return json.dumps({'mask': mask_output_path}, indent=4)
25

```

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，并输出 MASK（PNG 图像）和 json（PNG 图像的存储路径）。

PyTorch 的输入格式为 (batch_size, channels, height, width)

```

#-----#
#  添加batch_size维度
#-----#
image_data = np.expand_dims(np.transpose(preprocess_input(np.array(image_data, np.float32)), (2, 0, 1)), 0)

```

- preprocess_input 函数：对输入图像数据归一化
- numpy.transpose ：重新排列维度顺序
- numpy.expand_dims ：在第 0 维增加一个维度

在深度学习中，输入数据通常采用 batch 形式进行处理，即同时处理多个输入样本。因此，输入数据通常需要增加一个 batch 维，也就是在数据的最前面增加一维。

```

#-----#
#  图片传入网络进行预测
#-----#
pr = self.net(images)[0]
#-----#
#  取出每一个像素点的种类
#-----#
pr = F.softmax(pr.permute(1,2,0),dim = -1).cpu().numpy()
#-----#
#  将灰条部分截取掉
#-----#
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)]
#-----#
#  进行图片的resize
#-----#
pr = cv2.resize(pr, (orininal_w, orininal_h), interpolation = cv2.INTER_LINEAR)
#-----#
#  取出每一个像素点的种类
#-----#
pr = pr.argmax(axis=-1)
#-----#
#  转换成Image的形式
#-----#
image = Image.fromarray(np.uint8(pr))

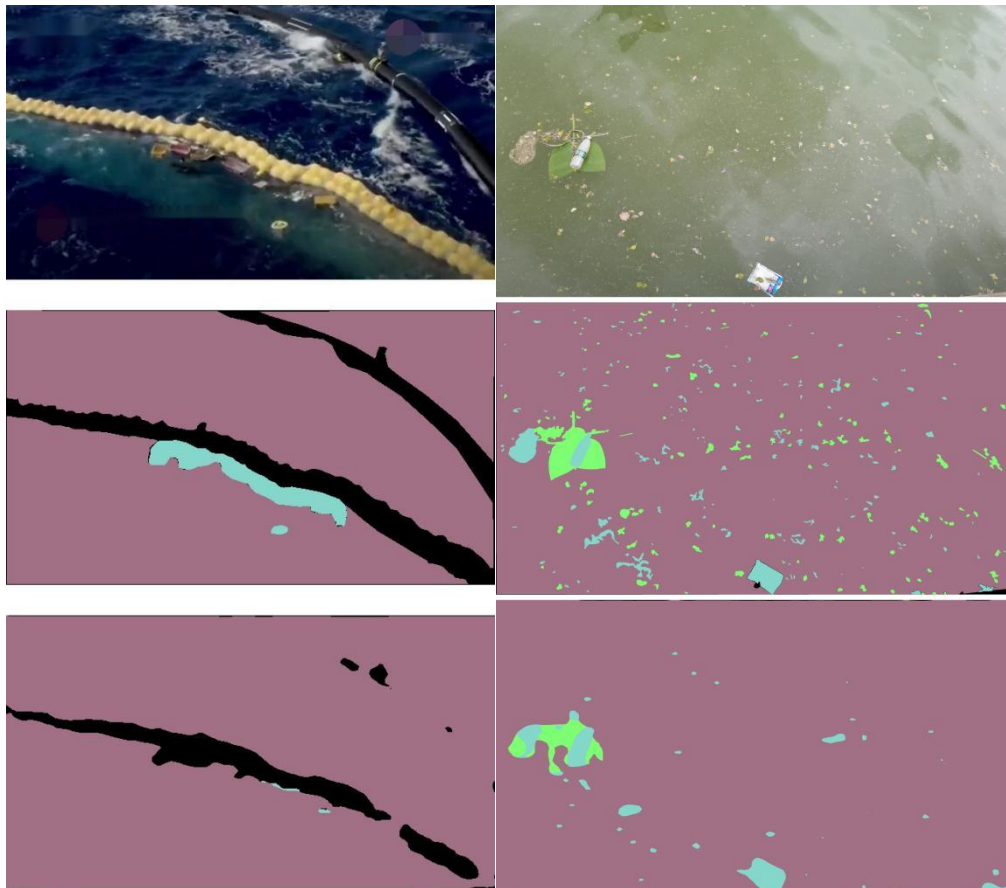
```

其中，detect_image 函数将图片传入网络进行预测，取每一个像素点的种类并转为 Image（PNG 图像）输出。

四、 实验结果

4.1 输出示例

通过极市工作台，我们可以查看样例集的推理结果：



4.2 测试结果

极市

AI开发工作台

水务源污物识别

文档

Kylo_Ren

打标详情

模型开发

实例

训练

模型列表

模型测试

算法开发

资产管理

101889

训练完成

2023-04-24 14:06:45

2023-04-24 14:09:50

2023-04-24 14:16:03

详情

中止

删除

101753

训练完成

2023-04-24 01:21:40

2023-04-24 01:22:57

2023-04-24 01:30:09

详情

中止

删除

101745

训练完成

2023-04-24 00:19:41

2023-04-24 00:21:04

2023-04-24 00:28:18

详情

中止

删除

101742

训练完成

2023-04-23 23:56:47

2023-04-24 00:02:08

2023-04-24 00:15:58

详情

中止

删除

101723

训练完成

2023-04-23 22:15:56

2023-04-23 22:19:29

2023-04-23 22:26:35

详情

中止

删除

101712

训练完成

2023-04-23 21:46:16

2023-04-23 21:49:23

2023-04-23 21:54:19

详情

中止

删除

101276

训练完成

2023-04-23 00:39:56

2023-04-23 00:41:14

2023-04-23 08:23:13

详情

中止

删除

101130

训练完成

2023-04-22 15:50:34

2023-04-22 15:52:13

2023-04-23 00:37:47

详情

中止

删除

100965

训练完成

2023-04-22 02:47:03

2023-04-22 02:51:00

2023-04-22 13:11:33

详情

中止

删除

100961

训练完成

2023-04-22 02:19:01

2023-04-22 02:22:24

2023-04-22 02:41:56

详情

中止

删除

100955

训练完成

2023-04-22 01:44:42

2023-04-22 02:06:48

2023-04-22 02:13:08

详情

中止

删除

100896

已中止

2023-04-21 22:54:17

2023-04-21 23:09:11

2023-04-22 01:43:21

详情

中止

删除

100655

已中止

2023-04-21 13:13:51

2023-04-21 13:14:51

2023-04-21 19:31:03

详情

中止

删除

100643

训练完成

2023-04-21 12:17:20

2023-04-21 12:18:39

2023-04-21 12:27:43

详情

中止

删除

极市

AI开发工作台

水务源污物识别

文档

Kylo_Ren

打标详情

模型开发

实例

训练

模型列表

模型测试

算法开发

资产管理

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

119076

标准测试

测试完成

发起:04-25 21:15:11
开始:04-25 21:21:23
结束:04-25 22:39:34

76

3.5752

0.519

详情

中止

删除

118788

标准测试

测试完成

发起:04-25 04:35:40
开始:04-25 04:39:13
结束:04-25 05:51:30

69

3.773

0.5185

详情

中止

删除

118758

标准测试

测试完成

发起:04-24 21:45:24
开始:04-24 21:46:30
结束:04-24 22:30:13

42

8.3471

0.5218

详情

中止

删除

118416

标准测试

测试完成

发起:04-23 10:15:15
开始:04-23 10:16:49
结束:04-23 11:17:01

58

7.0143

0.5218

详情

中止

删除

118378

标准测试

测试完成

发起:04-23 00:39:38
开始:04-23 00:40:44
结束:04-23 01:51:01

69

4.9257

0.5042

详情

中止

删除

118262

标准测试

测试完成

发起:04-22 13:38:26
开始:04-22 14:02:37
结束:04-22 14:43:31

39

6.5741

0.4849

详情

中止

删除

118057

标准测试

测试完成

发起:04-21 12:17:34
开始:04-21 12:19:49
结束:04-21 14:16:31

115

7.5851

0.3404

详情

中止

删除

极市

AI开发工作台


水务源污物识别

文档

Kylo_Ren

经过多次训练及测试，个人模型最优平均交并比 mean_iou 为 0.5218

4.3 最终排名

09	 JinH T1	0.4912	0.5389	6.1988
10	 jimfeng T1	0.4821	0.5309	4.3347
11	 Kylo_Ren T1	0.4779	0.5218	8.3471
12	 xh0511 T1	0.4593	0.5093	0.9701

三名队员分别排在第 9、11、12 位，最终本队模型最优平均交并比 mean_iou 为 0.5389，榜单排名第 9

五、 评价分析

5.1 实验分工与总结

本次实验我们小组并没有特别明确的分工，小组成员在前期工作中相互讨论，

共同完成。我们在宿舍大厅讨论了两晚最终敲定方案，确定使用 DeepLab V3+模型来进行建模。并从 Github 上找到了 DeepLab V3+模型相关的开源项目，将其应用于本次水务漂浮物识别任务中。由于是第一次接触极市平台，对平台需求的接口定义比较陌生，除了训练模型以外，本次实验大部分工作量在于定义测试接口方面。后面我们便各自调整参数，训练并测试模型。

注：

完整代码见：<https://gitee.com/kylo-ren17/deeplab>

附 1:参考文献

- [1] Bubbliiiiing. 憋批的语义分割重制版 9——Pytorch 搭建自己的 DeeplabV3+ 语义分割平台, from https://blog.csdn.net/weixin_44791964/article/details/120113686
- [2] 王潇棠, 闫河, 刘建骐等. 一种边缘梯度插值的双分支 deeplabv3+语义分割模型[J/OL]. 智能系统学报:1-9[2023-04-25]. <http://kns.cnki.net/kcms/detail/23.1538.tp.20230320.1616.012.html>.
- [3] 唐璐, 万良, 王婷婷等. DECANet: 基于改进 DeepLabv3+的图像语义分割方法[J]. 激光与光电子学进展, 2023, 60(04):92-100.
- [4] DeepLabv3+论文地址: <http://export.arxiv.org/abs/1802.02611>
- [5] DeepLabv3 论文地址: <http://arxiv.org/abs/1706.05587>
- [6] MobileNet V2 论文地址: <https://arxiv.org/pdf/1801.04381.pdf>
- [7] 空洞卷积论文地址: <https://arxiv.org/pdf/1511.07122.pdf>
- [8] ASPP 论文地址: <https://export.arxiv.org/pdf/1606.00915>

附 2：代码

注：详细代码见压缩包内的项目文件。