



模式识别与机器学习

实验报告

班级：10012003

姓名：王旭辉

学号：2020302709

目录

一、 实验目的	3
1.1 赛题背景	3
1.2 赛题数据说明	3
二、 实验原理	3
2.1 DeepLabv3+	3
2.1.1 图像语义分割	3
2.1.2 空洞卷积	3
2.1.3 空洞卷积空间金字塔池化（ASPP）	4
2.1.4 DeepLabv3+处理过程	5
2.2 MobileNet V2	6
2.3 MIoU	7
三、 实验步骤和程序流程	8
3.1 思路分析	8
3.2 DeepLabv3+项目在线配置	8
3.3 数据处理和数据集划分	8
3.3.1 数据集划分	8
3.3.2 数据处理	9
3.4 训练参数设置和训练	10
3.4.1 训练参数设置	10
3.4.2 训练	10
3.5 模型测试	11
3.5.1 测试接口编写	11
3.5.2 发起测试	12
3.6 流程图	13
四、 实验结果	14
4.1 平台测试结果	14
五、 评价分析	14
5.1 实验分工	14
5.2 分析与总结	15
六、 实验总结	16
6.1 实验感想	16
6.2 最终提交文件说明	16
附 1:参考文献	18
附 2: 代码	19

一、 实验目的

建立模型对水体和水体上的漂浮物进行分割。

1.1 赛题背景

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。

1.2 赛题数据说明

赛题提供的数据集使用 0、1、2、3、4 的像素值作为特征标注,标注格式为 8bit 单通道 png 图片,所以看起来 mask 是纯黑的,无法用肉眼识别。需要分类的类别有:

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)

二、 实验原理

2.1 DeepLabv3+

2.1.1 图像语义分割

图像语义分割是图像理解的基石性技术,在自动驾驶系统(具体为街景识别与理解)、无人机应用(着陆点判断)以及穿戴式设备应用中举足轻重。

图像是由许多像素构成的,图像中的各个特征也由不同的像素点构成,因此将图像中的每个像素点分配到特定的类别中就提取出了图像中包含的特征,这就是语义分割所做的工作。下面以赛题数据集为例进一步说明:



图 1: 数据集样例

上图用红色表示 water 特征,用黑色表示 background 特征,语义分割即本题需要做的任务便是提取出图片中包含的特征。

2.1.2 空洞卷积

空洞卷积是 DeepLab 模型的关键之一在传统的分割模型中采用的是完全卷积的神经网络,这导致在是非常深的网络中图片经过累次的池化和下采样分辨率大大降低,从而失去了很多特征。空洞卷积的提出解决了这一问题。

空洞卷积也叫扩张卷积或者膨胀卷积,简单来说就是在卷积核元素之间加入一些空格(零)来扩大卷积核的过程。用空洞率表示表示卷积核在卷积操作的输入

上的取样步长。如下图所示，（a）就是空洞率为 1 时卷积核在图片的映射，（b）为空洞率为 2 时的卷积核在图片上的映射，（c）为空洞率为 3 时的卷积核在图片上的映射。

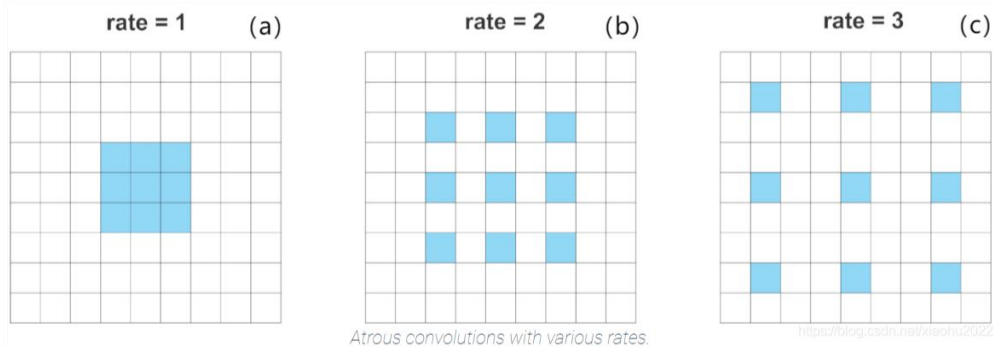


图 2 空洞卷积演示图

在一个二维卷积中，对于卷积输出的特征 y 上的每个位置 i 以及对应的卷积核 w ，对于输入 x ，空洞卷积的计算如下所示：

$$y[i] = \sum_{k=0}^{Kernel_size} x[i + r \cdot k]w[k]$$

上式中 r 为空洞率，表示卷积核在卷积操作的输入 x 上的取样步长； k 表示卷积核参数的位置，例如卷积核尺寸为 3，则 $k = 0,1,2$ ； $Kernel_size$ 表示卷积核尺寸。

通过空洞卷积与普通卷积的对比，不难看出，空洞卷积增大了感受野的同时并没有像下采样和池化那样丢失了图片的分辨率。感受野的增大可检测、分割大的目标，高分辨率则可精确定位目标，这十分适用于检测、分割任务中。

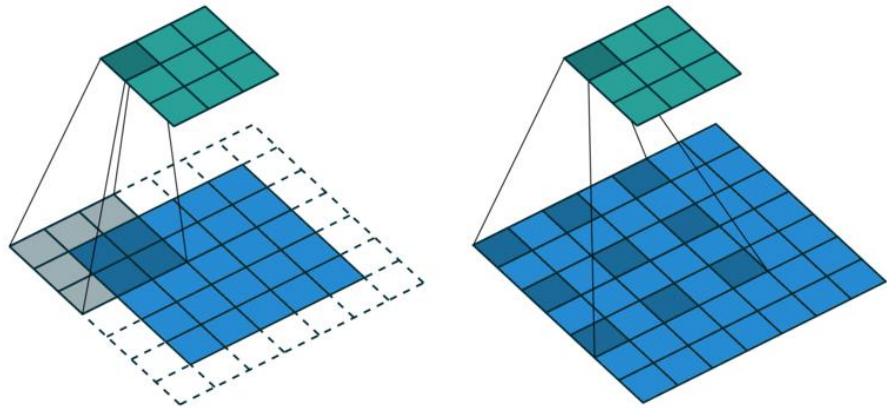


图 3 空洞卷积（右 1）与普通卷积（右 2）对比图

2.1.3 空洞卷积空间金字塔池化（ASPP）

在传统的神经网络中，输入的图片都是固定尺寸的，当图片尺寸不符合要求时需要进行 **resize**，但是 **resize** 之后会导致图片变形，因此特征信息会变形，限制了精度。空间金字塔池化（SPP）提出通过多种方式分别对图像进行池化，然后将多种池化结果进行拼接得到固定的大小，不必对图像进行 **resize** 也不必限制图像大小。如下图所示，将输入的图像进行三份处理，第一份直接进行池化，得

到 1×256 的向量，第二份分成 2×2 份，得到 4×256 的向量，第三份分成16份，得到 16×256 的向量，将最终得到的三份池化结果进行拼接得到了 21×256 的向量，这期间与图片的输入尺寸无关。

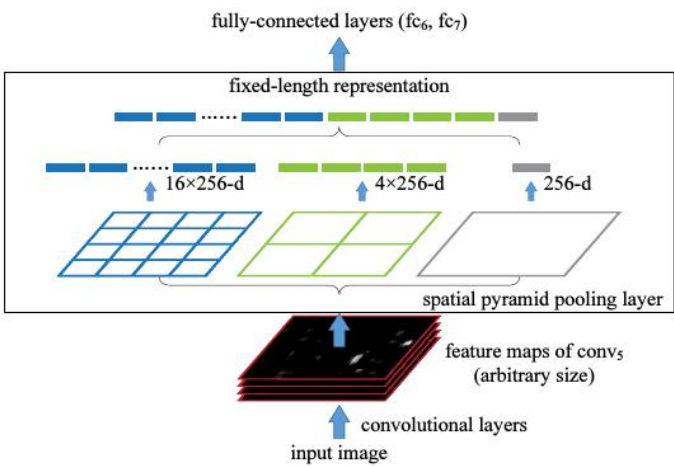


图 4 SPP 处理过程

由于在深层次神经网络中，大量池化会导致图像分辨率下降，丢失特征信息。所以 ASPP 提出使用空洞卷积替代 SPP 中的池化层，如下图所示，在 DeepLabv3+ 模型中，ASPP 采用不同采样率的空洞卷积捕捉特征信息，使其可以在不同尺度上捕获上下文信息，卷积完成后将得到的特征信息进行特征融合，拼接起来获得了更好的性能，实现了多尺度的特征提取。

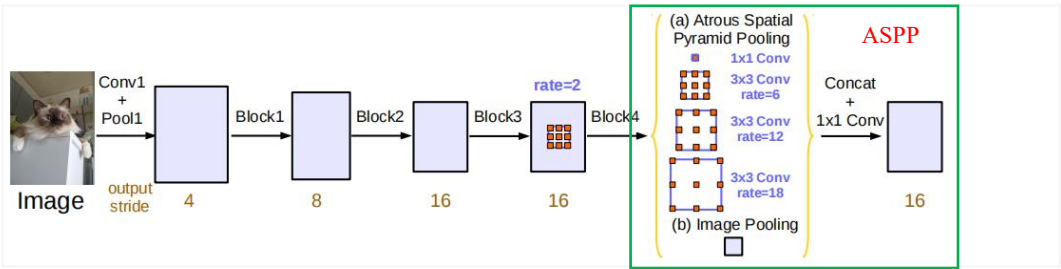


图 5 DeepLabv3+中的 ASPP

2.1.4 DeepLabv3+处理过程

DeepLabv3+在语义分割中拥有很好的效果，该模型采用了 Encoder-Decoder 结构，利用 Encoder 结构可以获得输入图片丰富的语义特征信息，获得信息后输入到 Decoder 里进行解码以获得特征分类结果。

下图表示了 DeepLabv3+模型的构建过程。

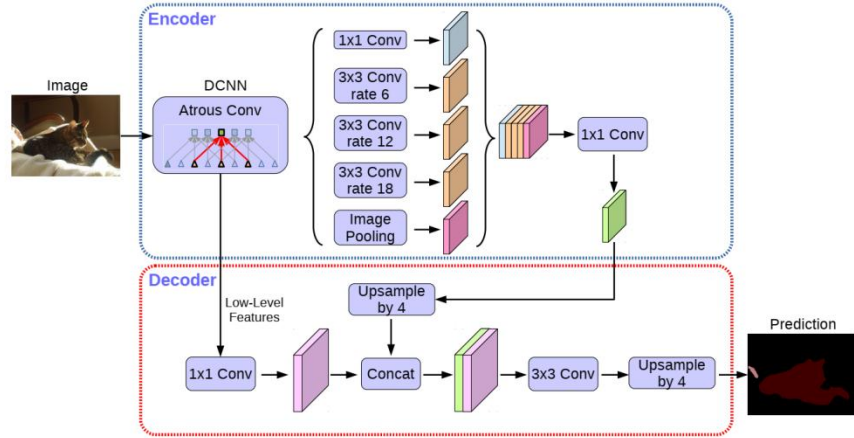


图 6: DeepLabv3+模型架构

在 Encoder 模块，将 DeepLabv3 作为 Encoder，这时 feature 通常为 16，深度卷积神经网络（DCNN）中采用了 stride=2 的卷积来实现下采样，在不损失信息的情况下，加大了感受野，让每个卷积输出都包含较大范围的信息。在获得了初步的有效特征层后，使用 ASPP 加强特征提取网络，在 ASPP 中并行采用不同空洞率的空洞卷积进行特征提取再结合图像全局池化进行特征融合，完成后采用 1×1 的卷积操作进行特征层的降维，降维后就得到了一个具有高语义信息的有效特征层。最后将在 Encoder 获得的有效特征层经过 4 倍的双线性插值上采样输入到 Decoder 中。

在 Decoder 模块，将 DCNN 获得的较浅的特征层，经过 1×1 的卷积将维度调整到和 Encoder 模块上采样后获得的特征层相同，减少浅层特征层的通道数目，这个较浅的特征层由于经过了较少的下采样，具有较大的高和宽，而 Encoder 获得的特征层经过了多个下采样，具有较小的高和宽。将这两部分特征层进行特征融合，经过 3×3 卷积的特征提取就得到了输入图片的特征浓缩。提取出特征后，将特征层的维度经过 1×1 的卷积降维到特征数即 Num_Classes，最后再经过 4 倍的双线性插值上采样将输出图像调整到输入图片大小。DeepLabv3+ 引入的 Decoder 结构有助于更好的恢复物体的分割细节。

2.2 MobileNet V2

本实验使用 MobileNet V2 网络作为 DCNN 的主干提取网络。MobileNet V2 是 MobileNet 的升级版，是一种轻量级网络，它具有一个非常重要的特点就是使用了 Inverted resblock，整个 mobilenetv2 都由 Inverted resblock 组成。

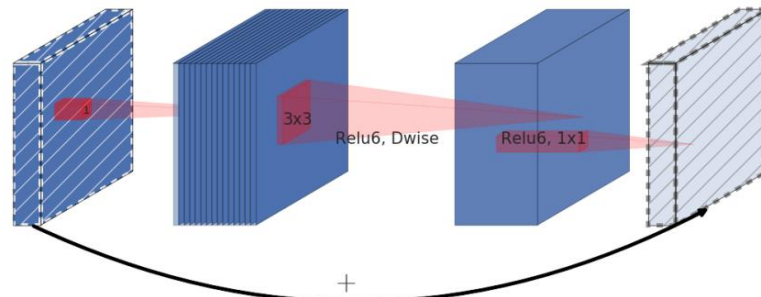


图 7 Inverted resblock 组成

如上图所示，nverted resblock 可以分为两个部分：第一部分，首先利用1×1卷积进行升维，然后利用3×3深度可分离卷积进行特征提取，然后再利用1×1卷积降维。第二部分是残差边部分，输入和输出直接相接。下图所示的操作为具体的网络结构，先经过1×1卷积升维，然后经过 Batchnorm 和 Relu 激活函数，经过深度可分离卷积进行特征提取后再用1×1卷积降维，最后输出。

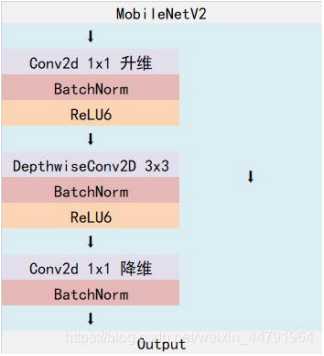


图 8 MobileNet V2 结构

深度可分离卷积通过分解卷积操作，分解为深度卷积和逐点卷积有效地减少了参数数量和计算复杂度，常用于轻量级神经网络的设计，以在计算资源有限的设备上实现高效的性能。例如对于 3 通道的卷积，先对 3 个通道分别卷积，得到 3 个数再用一个1×1的卷积核合为一个数，在保证精度的同时，降低了计算复杂度。

2.3 MIoU

本实验采用 MIoU 作为模型评价指标。MIoU (Mean Intersection over Union) 是语义分割的一个评价指标，表示平均交并比，即数据集上每一个类别的 IoU 值的平均。在下图中，左边的圆表示某个类的真实标签，右边的圆表示其预测结果，其中 *TP* 表示真实值为 Positive，预测为 Positive；*FN* 表示真实值为 Positive，预测为 Negative；*FP* 表示真实值为 Negative，预测为 Positive，称作是错误的 Positive；*TN* 表示真实值为 Negative，预测为 Negative，称作是正确的 Negative。

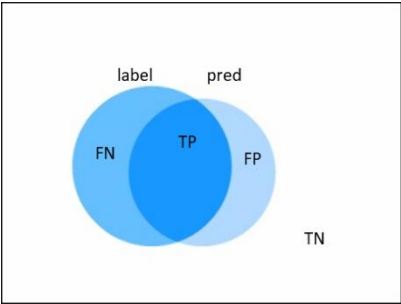


图 9 IoU 交并比

中间的 *TP* 部分即为真实值和预测值的交，*FN + FP + TP* 部分即为真实值和预测值的并，IoU 就是交与并的比值：

$$IoU = \frac{TP}{FN + FP + TP}$$

则 MIoU，平均交并比的具体计算公式如下：

设 *i* 表示真实值，*j* 表示预测值 P_{ij} 表示将 *i* 预测为 *j*

$$MIoU = \frac{1}{k+1} \sum_{i=0}^k \frac{P_{ii}}{\sum_{j=0}^k P_{ij} + \sum_{j=0}^k P_{ji} - P_{ii}}$$

分母中 $\sum_{j=0}^k P_{ij}$ 表示 i 的所有预测结果，即 $FN + TP$ ， $\sum_{j=0}^k P_{ji}$ 表示所有预测为 i 的值，即 $FP + TP$ ，最后减去一个重合的 P_{ii} 即 TP ，所以计算公式也可写为：

$$MIoU = \frac{1}{k+1} \sum_{i=0}^k \frac{TP}{FN + FP + TP}$$

在本实验中 P_{ij} 表示将 i 预测为 j 的像素数量。

三、实验步骤和程序流程

3.1 思路分析

在确定题目后，我们在网上查找了相关的分类模型，我们小组首先考虑的模型是采用 YOLOv5 进行目标识别，在做的过程中发现赛题的数据集格式无法满足 YOLOv5，需要采用 xml 格式而赛题给出的是 png 格式的图像。由于我们之前没有接触过用像素值表示图像特征的数据集标注格式，我们小组一度怀疑赛题给出的数据集格式出现了问题，是纯黑的。之后我们在网上查找了语义分割模型的相关知识，了解了语义分割模型的数据集标注格式和相关方法，才了解到 mask 肉眼无法分辨是因为用 0, 1, 2, 3, 4 的像素值表示特征，图像为 8 位单通道格式。

在解决了数据集格式问题后，我们小组查找了相关模型，查找了 FCN，DeepLabV 系列等模型，最终看到了 DeepLabv3+ 的相对于其他模型的优越性能，我们敲定使用 DeepLabv3+ 模型作为我们的语义分割模型。

3.2 DeepLabv3+ 项目在线配置

由于本次实验的数据集在极市平台上，我们无法进行下载，所以所有的项目配置都在平台提供的在线编码上完成。

使用 git 将项目克隆到平台上的
/project/train/src_repo/deeplabv3_plus-pytorch 目录下：

```
[root@src_repo]$ git clone https://github.com/bubbliiii/deeplabv3-plus-pytorch.git
```

图 10 项目克隆

在线平台提供了 pytorch 和 cuda 环境，所以不用配置相关 python 环境。

3.3 数据管理和数据集划分

3.3.1 数据集划分

为了满足 DeepLabv3+ 的数据集格式要求，需要将原图放到 JPEGImages 文件夹下且数据集格式为 jpg 格式，将 mask 放到 SegmentationClass 文件夹下且格式为 png 格式 8 位的单通道图片。在平台训练时，使用指令：

```
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
```

图 11 数据集构建

在平台上将 home/data 文件夹下的图片和 mask 分别放到模型对应的文件夹下。完成后开始训练集和测试集的划分，本模型要求在 ImageSets/Segmentation

下存放 train.txt 和 val.txt, 这两个文件里分别存放着训练集的图片名和测试集的图片名, 编写代码 voc_annotation.py 对数据集进行划分, 我们设置训练集和测试集之比为 9:1, trainval_percent =1, train_percent=0.9。相关代码如下所示:

```

38     num     = len(total_seg)
39     list     = range(num)
40     tv       = int(num*trainval_percent)
41     tr       = int(tv*train_percent)
42     trainval= random.sample(list,tv)
43     train    = random.sample(trainval,tr) #随机生成训练集编号
44
45     print("train and val size",tv)
46     print("traub suze",tr)
47     ftrainval = open(os.path.join(saveBasePath,'trainval.txt'), 'w')
48     # trainval.txt存放所有图片名
49     ftest      = open(os.path.join(saveBasePath,'test.txt'), 'w')
50     # 将测试集当作验证集使用, 不单独划分测试集
51     ftrain     = open(os.path.join(saveBasePath,'train.txt'), 'w')
52     fval       = open(os.path.join(saveBasePath,'val.txt'), 'w')
53     # 划分测试集和训练集
54     for i in list: # i存在哪个列表就将图片放到哪个列表
55         name = total_seg[i][:-4]+'\\n'
56         if i in trainval:
57             ftrainval.write(name)
58         if i in train:
59             ftrain.write(name)
60         else:
61             fval.write(name)
62         else:
63             ftest.write(name)

```

图 12 数据集划分 voc_annotation.py 部分代码

最终得到的数据集部分展示如下:

```

1 ZDSfloating_objects20230206_V3_train_sea_1_001461
2 ZDSfloating_objects20230206_V3_train_sea_1_000720
3 ZDSfloating_objects20230206_V3_train_lakes_7_000012
4 ZDSfloating_objects20230206_V3_train_sea_1_001080
5 ZDSfloating_objects20230206_V3_train_rivers_36_204
6 ZDSfloating_objects20230206_V3_train_sea_1_011323
7 ZDSfloating_objects20230206_V3_train_sea_1_005268
8 ZDSfloating_objects20230206_V3_train_sea_1_005376
9 ZDSfloating_objects20230206_V3_train_sea_1_009526
10 ZDSfloating_objects20230206_V3_train_rivers_24_000374
11 ZDSfloating_objects20230206_V3_train_sea_1_004712

```

图 13 train.txt

```

1 ZDSfloating_objects20230206_V3_train_sea_1_003521
2 ZDSfloating_objects20230206_V3_train_sea_12_95
3 ZDSfloating_objects20230206_V3_train_sea_1_006987
4 ZDSfloating_objects20230206_V3_train_rivers_37_1_1988
5 ZDSfloating_objects20230206_V3_train_sea_1_007946
6 ZDSfloating_objects20230206_V3_train_sea_1_010508
7 ZDSfloating_objects20230206_V3_train_rivers_37_1_1917
8 ZDSfloating_objects20230206_V3_train_rivers_22_000026
9 ZDSfloating_objects20230206_V3_train_sea_1_000678
10 ZDSfloating_objects20230206_V3_train_sea_1_000173

```

图 14 val.txt

voc_annotation.py 还对输入图片格式做了输入检查, 如果输入的图片不是 jpg 格式或者 mask 不是 8 位单通道 png 图像, 会进行报错提示检查图片格式。

3.3.2 数据处理

为了实现不失真的 **resize**, 我们给图像增加灰度条, 在预测完成后, 去掉灰度条即可。

```

15 #-----#
16 # 对输入图像进行resize
17 #-----#
18 def resize_image(image, size):
19     iw, ih = image.size
20     w, h = size
21
22     scale = min(w/iw, h/ih)
23     nw = int(iw*scale)
24     nh = int(ih*scale)
25
26     image = image.resize((nw,nh), Image.BICUBIC)
27     new_image = Image.new('RGB', size, (128,128,128))
28     new_image.paste(image, ((w-nw)//2, (h-nh)//2))
29
30     return new_image, nw, nh

```

图 15 resize_image, 不失真 resize

3.4 训练参数设置和训练

3.4.1 训练参数设置

在 train.py 里可以设置训练参数，设置模型需要分类的个数 num_classes=5，所使用的主干网络为 mobilenet，下采样倍数 downsample_factor=8，epoch 和 batchsize 为 100 和 4 并使用 sgd 优化器进行优化。所有参数设置如下图所示：

Configurations:

keys	values
num_classes	5
backbone	mobilenet
model_path	
input_shape	[512, 512]
Init_Epoch	0
Freeze_Epoch	50
UnFreeze_Epoch	30
Freeze_batch_size	8
Unfreeze_batch_size	4
Freeze_Train	False
Init_lr	0.007
Min_lr	7.000000000000001e-05
optimizer_type	sgd
momentum	0.9
lr_decay_type	cos
save_period	5
save_dir	logs
num_workers	1
num_train	90
num_val	10

图 16 模型参数设置

3.4.2 训练

参数设置完成后，在终端中跳转到 /project/train/src_repo/deeplabv3-plus-pytorch 目录下，使用命令 python train.py 即可运行。运行过程如下图所示：

```
Finish Validation
Epoch:3/30
Total Loss: 0.891 || Val Loss: 0.873
Save best model to best_epoch_weights.pth
Start Train
Epoch 4/30: 100%|████████████████████████████████████████| 22/22 [00:10<00:00, 2.03it/s, f_score=0.728, lr=0.00175, total_loss=0.962]
Finish Train
Start Validation
Epoch 4/30: 100%|████████████████████████████████████████| 2/2 [00:01<00:00, 1.41it/s, f_score=0.832, lr=0.00175, val_loss=1.03]
```

图 17 在线运行过程

在训练过程中，每 5 次 epoch 会保存一次最好的权重模型文件到 /project/train/models/best_epoch_weights.pth 下

在线编码成功后需要在极市平台上使用完整数据集训练，为了满足平台训练要求，我们将训练文件，模型文件放到了指定文件夹下并编写了 start_train.sh 文件，存放了训练需要执行的步骤，shell 文件如下所示：

```

1 CURRENT_DIR=$(cd $(dirname $0)/deeplabv3-plus-pytorch; pwd)
2
3 echo "${CURRENT_DIR}"
4 cd "${CURRENT_DIR}"
5
6
7 # 将/home/data文件夹下的文件放到模型指定的文件夹下，划分原图和mask图
8 cp /home/data/1945/*.jpg /project/train/src_repo/deeplabv3-plus-pytorch/VOCdevkit/VOC2007/JPEGImages
9 cp /home/data/1945/*.png /project/train/src_repo/deeplabv3-plus-pytorch/VOCdevkit/VOC2007/SegmentationClass
10
11
12 # 运行数据集划分文件，将数据集划分为训练集，测试集和验证集
13 python /project/train/src_repo/deeplabv3-plus-pytorch/voc_annotation.py
14
15 # 运行train.py进行训练 You, now * Uncommitted changes
16 python /project/train/src_repo/deeplabv3-plus-pytorch/train.py
17

```

图 18 start_train.sh 文件

以上工作做完后，在极市平台发起训练，训练部分过程如下图所示：

```

100%|██████████| 1638/1644 [02:42<00:00, 9.15it/s]
100%|██████████| 1640/1644 [02:42<00:00, 9.77it/s]
100%|██████████| 1643/1644 [02:43<00:00, 12.16it/s]
100%|██████████| 1644/1644 [02:43<00:00, 10.07it/s]
Calculate miou.
Num classes 5
==> mIoU: 35.98; mPA: 41.16; Accuracy: 90.01
Get miou done.
Epoch:5/100
Total Loss: 0.877 || Val Loss: 0.823
Save best model to best_epoch_weights.pth
Start Train

Epoch 6/100: 0%|          | 0/3697 [00:00<?, ?it/s<class 'dict'>]
Epoch 6/100: 0%|          | 0/3697 [00:00<?, ?it/s, f_score=0.783, lr=0.00175, total_loss=0.717]
Epoch 6/100: 0%|          | 1/3697 [00:00<53:07, 1.16it/s, f_score=0.783, lr=0.00175, total_loss=0.717]
Epoch 6/100: 0%|          | 1/3697 [00:01<53:07, 1.16it/s, f_score=0.842, lr=0.00175, total_loss=0.667]
Epoch 6/100: 0%|          | 2/3697 [00:01<33:24, 1.84it/s, f_score=0.842, lr=0.00175, total_loss=0.667]
Epoch 6/100: 0%|          | 2/3697 [00:01<33:24, 1.84it/s, f_score=0.822, lr=0.00175, total_loss=0.687]
Epoch 6/100: 0%|          | 3/3697 [00:01<28:42, 2.14it/s, f_score=0.822, lr=0.00175, total_loss=0.687]
Epoch 6/100: 0%|          | 3/3697 [00:02<28:42, 2.14it/s, f_score=0.772, lr=0.00175, total_loss=0.678]
Epoch 6/100: 0%|          | 4/3697 [00:02<34:58, 1.76it/s, f_score=0.772, lr=0.00175, total_loss=0.678]
Epoch 6/100: 0%|          | 4/3697 [00:02<34:58, 1.76it/s, f_score=0.778, lr=0.00175, total_loss=0.702]

```

图 19 训练日志

3.5 模型测试

3.5.1 测试接口编写

模型训练结束后需要在平台上发起测试，为了满足平台测试要求，我们编写了 ji.py 用于测试，ji.py 里包含对模型的初始化和输入图片的预测。为了成功发起测试，我们将 nets 文件夹和 utils 文件夹一并复制到 ev_sdk/src 文件夹下。这里由于平台提供的图片读取方式是 cv2，我们代码中读取方式是 PIL，所以这里做了 image = Image.fromarray(cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB))操作，将格式转换为 PIL。

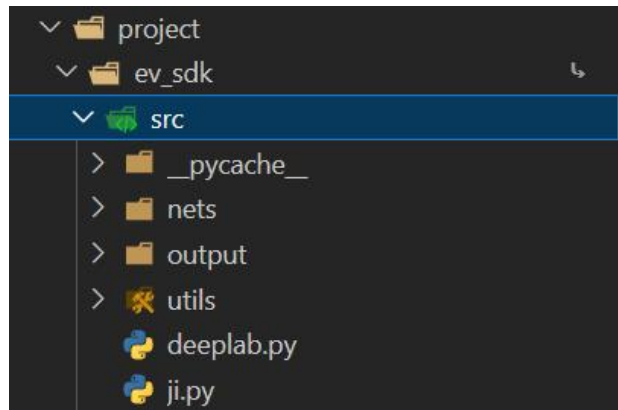


图 20 ev_sdk/src 文件夹

```

1  import json
2  from PIL import Image
3  import numpy as np
4  import cv2
5
6  from deeplab import DeeplabV3
7
8  def init():
9      # 初始化模型
10     deeplab = DeeplabV3()
11     return deeplab
12
13  def process_image(handle=None, input_image=None, args=None, **kwargs):
14      # 将单张图片输入到模型中
15      name_classes = ["background", "algae", "dead_twigs_leaves", "garbage", "water"]
16      # 需要分类的名称
17      args = json.loads(args)
18      mask_output_path = args['mask_output_path']
19
20      # 由于平台输入的图片是用cv2库，模型中使用PIL库，这里将cv2转换成PIL
21      image = Image.fromarray(cv2.cvtColor(input_image, cv2.COLOR_BGR2RGB))
22      # 以json文件的格式输出
23      # Process image here
24      # Generate dummy mask data
25      # 对输入的图片进行模型预测
26      pred_mask_per_frame = handle.detect_image(image, count=True, name_classes=name_classes)
27      pred_mask_per_frame.save(mask_output_path)
28      # 返回json格式的输出
29      return json.dumps({'mask': mask_output_path}, indent=4)

```

图 21 ji.py

3.5.2 发起测试

接口编写完成后，在平台选择训练好的 best_epoch_weights.th 模型进行预测，预测部分日志如图：

background	7907	0.38%	0
algae	0	0.00%	1
dead_twigs_leaves	5038	0.24%	2
garbage	0	0.00%	3
water	2060655	99.38%	4
classes_nums: [7907. 0. 5038. 0. 2060655.]			

图 22 测试日志

平台的预测结果 json 输出如下：

```
{
  "mask":
"/tmp/tmpd9sv7ojc/sample_out/1945/ZDSfloating_objects20230206_V3_train_sea_1_001743.png"
}
```

预测结果可视化如图，第一张为原图，第二张为正确分类的 mask，第三张为预测输出：



图 23 预测结果可视化

为了获取更好的模型和测试效果，需要不断在平台上迭代测试。

3.6 流程图

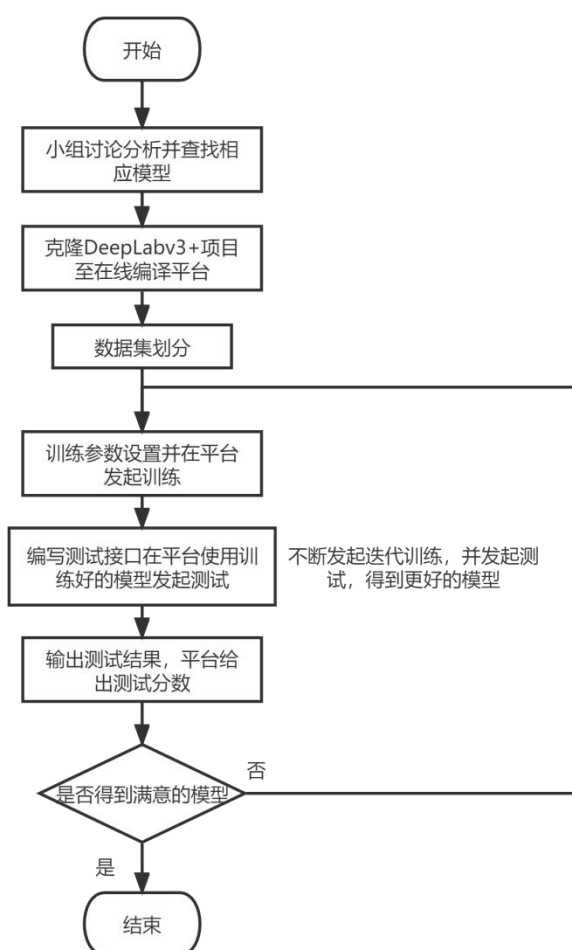


图 24 实验流程图

四、实验结果

4.1 平台测试结果

我们小组在极市平台发起测试的最好结果是成绩分为 0.4901，mean_iou 为 0.5389，性能分为 5.1018，排名第 9 名。


09	 JinH T1	0.4912	0.5389	6.1988
----	---	--------	--------	--------

图 25 打榜最好结果

测试任务ID	测试类型	进度	时间	消耗积分	性能分	mean_iou	操作
119127	标准测试	● 测试完成	发起:04-26 00:23:52 开始:04-26 01:08:10 结束:04-26 02:45:30	96	0.9701	0.5093	详情 中止 撤销
118573	标准测试	● 测试完成	发起:04-23 23:01:18 开始:04-23 23:02:30 结束:04-24 00:25:30	81	2.9028	0.4858	详情 中止 撤销
118430	标准测试	● 测试完成	发起:04-23 11:12:49 开始:04-23 11:13:43 结束:04-23 12:35:00	80	3.6086	0.4868	详情 中止 撤销
118244	标准测试	● 测试完成	发起:04-22 12:44:15 开始:04-22 12:45:07 结束:04-22 13:49:31	64	4.6501	0.1312	详情 中止 撤销
118235	标准测试	● 测试完成	发起:04-22 11:41:59 开始:04-22 11:42:51 结束:04-22 12:42:01	58	4.3205	0.1312	详情 中止 撤销
118183	标准测试	● 测试完成	发起:04-21 23:10:42 开始:04-21 23:12:09	82	3.1396	0.4628	详情 中止 撤销

图 26 部分测试结果

下图展示了部分预测结果对比：

第一张图片为原图，第二张图片为正确 mask 图，第三张为模型预测结果图，从三者对比可以看出，模型的预测性能还是比较优秀的。

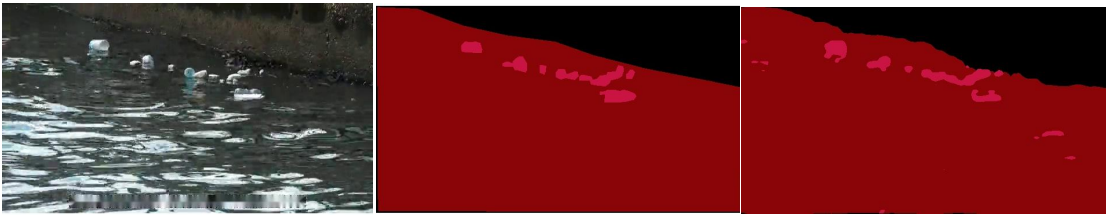


图 27 预测结果

五、评价分析

5.1 实验分工

由于本次实验我们小组成员都是第一次接触语义分割模型，所以并没有明确的分工，从讨论思路到确定模型再到成功测试得到分数，都是一起讨论，一起接力。一开始我们走向了错误的方向，一直在考虑使用 Yolov5 进行目标检测，这导致我们走了很多弯路；一开始对于赛题提供的数据集格式也不清楚，甚至以为提供了错误的数据集。之后我们一起在宿舍大厅查阅资料，耗费了两个晚上决定了使用 DeepLabv3+进行语义分割。在确定模型后需要发起训练，训练脚本的编

写也花费了我们一定的时间。最头疼的部分是测试接口的编写，由于是第一次使用极市平台，很多操作不熟悉，对于接口的许多定义不知道如何写，我们不断查找资料，阅读文档，读代码，最终接力完成了 `ji.py` 的编写，下面是我们测试和训练的一些截图，失败了太多次：

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
102245	● 训练完成	2023-04-25 13:07:56	2023-04-25 13:29:09	2023-04-26 00:23:19	详情 中止 撤销
101483	● 训练完成	2023-04-23 12:43:13	2023-04-23 12:45:26	2023-04-23 22:41:32	详情 中止 撤销
101478	● 已中止	2023-04-23 12:37:07	2023-04-23 12:39:28	2023-04-23 12:42:06	详情 中止 撤销
101142	● 已中止	2023-04-22 16:23:22	2023-04-22 16:24:42	2023-04-23 04:30:17	详情 中止 撤销
101069	● 已中止	2023-04-22 11:29:33	2023-04-22 11:31:14	2023-04-22 16:22:45	详情 中止 撤销
100941	● 已中止	2023-04-22 00:37:44	2023-04-22 00:40:30	2023-04-22 11:26:31	详情 中止 撤销
100624	● 已中止	2023-04-21 11:04:58	2023-04-21 11:07:09	2023-04-21 23:10:13	详情 中止 撤销
100563	● 训练完成	2023-04-21 00:54:47	2023-04-21 00:58:21	2023-04-21 01:40:36	详情 中止 撤销
100544	● 训练完成	2023-04-20 23:23:54	2023-04-20 23:58:26	2023-04-21 00:33:38	详情 中止 撤销
100144	● 已中止	2023-04-20 01:25:34	2023-04-20 01:27:59	2023-04-20 13:30:12	详情 中止 撤销

共18条

< 1 2 > 10条/页 跳至 页

图 28 模型训练部分截图

117858	标准测试	● 测试失败	发起:04-20 13:37:48 开始:04-20 13:38:49 结束:04-20 13:41:17	2	详情 中止 撤销
117856	标准测试	● 测试失败	发起:04-20 13:25:45 开始:04-20 13:27:12 结束:04-20 13:30:06	3	详情 中止 撤销
117837	标准测试	● 已中止	发起:04-20 11:44:51 开始:04-20 11:46:06 结束:	12	详情 中止 撤销
117828	标准测试	● 测试失败	发起:04-20 11:10:16 开始:04-20 11:11:09 结束:04-20 11:13:26	1	详情 中止 撤销
117823	标准测试	● 测试失败	发起:04-20 10:50:48 开始:04-20 10:52:05 结束:04-20 10:56:11	2	详情 中止 撤销
117785	标准测试	● 测试失败	发起:04-20 00:20:15 开始:04-20 00:21:49 结束:04-20 00:27:17	5	详情 中止 撤销

共34条

< 1 2 3 4 > 10条/页 跳至 页

图 29 模型测试部分截图

5.2 分析与总结

这个赛题是一个标准的语义分割模型，主要的任务就是识别图片中包含的水务信息，区分背景，垃圾，水藻，枯枝败叶和水体。我们小组采用了 DeepLabv3+ 模型进行训练，用训练得到的模型在平台上发起测试得到了比较不错的结果。

在实验过程中，由于实验样本很大，DeepLabv3+ 模型又需要更复杂的卷积运算和模型结构，所以每次训练的时间很长很长，每 12h 只能训完 30-40 轮次的任务，平台提供了迭代训练，可以采用每次训练得到的最好模型接着训练，在不断的迭代中，通过观察日志，miou 和准确率不断上升，这说明网络特征提取和

特征融合的结果越来越准确，最终测试结果达到了第 9 名，mean_iou 达到了 0.5389。

DeepLabv3+模型的优点有：

- ❖ 具有较强的边界检测能力，可以检测出更多的细节特征。
- ❖ 具有较强的泛化能力，可以在不同的数据集上获得较好的性能。
- ❖ DeepLabV3+模型具有较强的特征提取能力，可以提取出更多的细节特征。

但是 DeepLabv3+模型的缺点也在训练中暴露出来，因为它使用了更多的卷积层和更复杂的模型结构，所以需要更多的计算资源和时间。我们在训练时 30 轮次的训练一般需要 12 小时。此外，它还需要更多的数据来训练模型，我们使用 100 张样例数据进行训练时的准确率和 mIoU 并不好，用完整数据集训练的模型效果较好。

在模型训练到一定精度后，无法再进行精进，我们小组通过分析发现可能是数据集的原因，有些数据集的标注和原图的质量较差，严重影响了训练。例如下面这张数据集，第一张为原图，第二张在没有垃圾的地方进行了错误标注，导致预测结果和标注结果不符。



图 30 部分较差数据集

如果数据集质量更好的话我相信我们的模型会有更好的分类结果。

六、 实验总结

6.1 实验感想

语义分割模型是我们小组第一次接触，这次实验相较于上次实验难度提升很大，我们开始甚至数据集都无法读懂。在查阅资料后我们发现，DeepLabv3+模型非常适合这种语义分割任务。于是我们从零开始学习了语义分割模型，学习了 DeepLabv3+模型，了解了空洞卷积的原理和作用，了解了 ASPP 的原理和作用，了解了 1×1 卷积的升维降维原理和 MobileNet 网络结构等，真的令我受益匪浅。虽然和同组的同学熬夜找模型，debug 甚至到晚上四五点，但是看到接力完成的模型运行结果和平台上不错的跑分成绩，意识到这段时间的学习真的难忘。在平台测试期间，我们也失败了很多很多次，但集结三人的力量最终攻克了下来。每次进行模型训练都要等待 12 小时，测试又需要一个半小时，等待的时间是漫长的但是又充满期待的，当看到平台上不断进步的成绩时，颇有成就感。

本学期的两次大作业，都让我对模式识别与机器学习这个领域有了更深的了解，也对计算机视觉这个方向有了一定的认知，我也对此产生了非常浓厚的兴趣，希望在以后的学习过程中能够不断地努力和深入。

这里非常感谢 B 站 UP 主 Bubbliiiiing，将自己的代码分享在了 github 上并制作了讲解视频。

6.2 最终提交文件说明

在提交的 project 文件中，log 文件夹存放模型训练时的日志模型，由于平台训练出的模型无法下载，这里没有给出；models 下存放的是训练出的最好模型；result-graphs 下存放的是输出的可视化图片；src_repo 下的 start_train.sh 文件是启

动训练的 shell 文件，环境配置完成后，点击此文件即可运行代码；在 **deeplabv3-plus-pytorch** 文件夹下，**nets** 存放的是模型网络构建代码，**utils** 文件夹下是训练中用到的各种参数计算，保存模型，数据处理代码，**VOCdevkit** 下存放的是数据集，目录下直接存放的 **python** 代码为训练, 数据集划分和预测代码。

附 1:参考文献

- [1] 周志华, 机器学习
- [2] Chen, L. C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv:1706.05587 (2017)
- [3] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in Proceedings of the European conference on computer vision (ECCV), 2018, pp. 801-818.
- [4] Mark Sandler Andrew Howard Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen: MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv:1801.04381 [cs.CV]
- [5] [憨批的语义分割重制版 9——Pytorch 搭建自己的 DeeplabV3+语义分割平台 Bubbliiiiing 的博客-CSDN 博客](#)
- [6] [语义分割的评价指标——MIoU 金渐层猫的博客-CSDN 博客](#)
- [7] [ASPP 学习笔记 Clark-dj 的博客-CSDN 博客](#)
- [8] [如何理解空洞卷积 \(dilated convolution\)? - 知乎 \(zhihu.com\)](#)

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

附 2：代码

模型代码较长，故报告中不再赘述，具体代码见提交文档。

我们将项目传到了 gitee 上，具体可至我们的主页，

<https://gitee.com/kylo-ren17/deeplab.git>

也上传至 github 上，可至我们的主页，

https://github.com/guadawcz/deeplabv3_plus.git