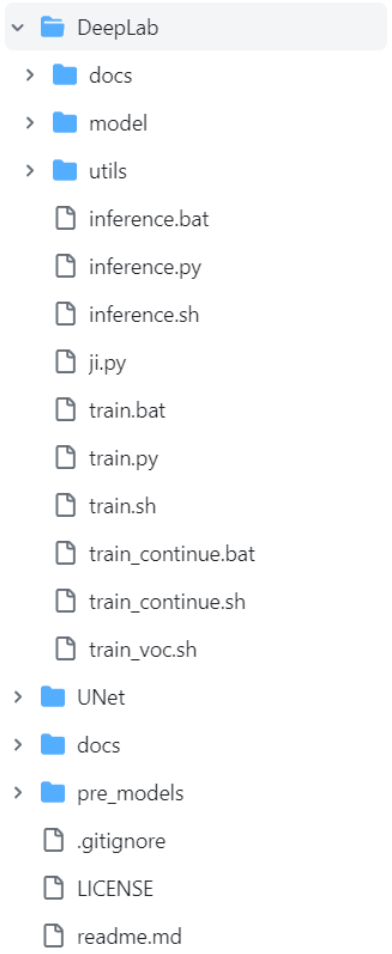
水务漂浮物识别报告

第12组 孟祥宇 舒适

框架搭建

```
# paths
train_path=/project/train/src_repo/UNet/train.py
dataset_root=/home/data/
dataset_dir=1945/
save_root=/project/train/
save_dir=models/unet/
log_root=/project/train/
log_dir=log/unet/train/

# training methods
loss=CrossEntropy
lr=0.0001
batch_size=2
epoch=100
image_width=512
image_height=512
optimizer=RMSprop

# showing
log_visual=true
use_tqdm=false
only_best=true
```

train.sh 中的超参数调节



文件结构

Linux (cvmart.net competition)

setup see here

UNet

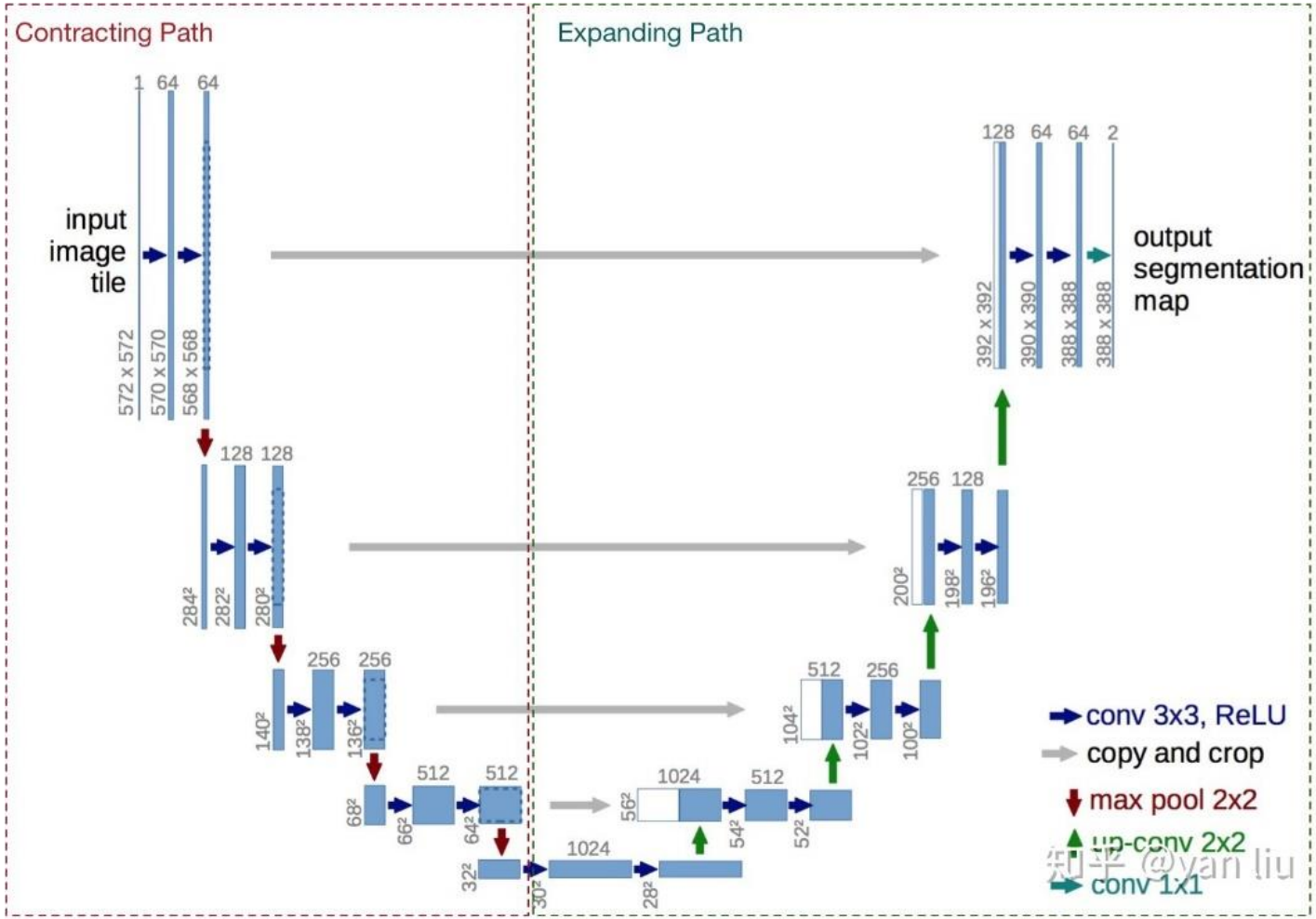
```
bash /project/train/src_repo/UNet/train.sh
bash /project/train/src_repo/UNet/inference.sh
```

DeepLabV3+

```
bash /project/train/src_repo/DeepLab/train.sh
bash /project/train/src_repo/DeepLab/inference.sh
```

运行命令

UNet


Contracting Path
Expanding Path
input image tile
output segmentation map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1
1 64 64
572 x 572
570 x 570
568 x 568
128 128
284²
282²
280²
256 256
140²
138²
136²
512 512
68²
66²
64²
32²
1024
30²
28²
1024
56²
512
54²
52²
512 256
104²
102²
100²
256 128
200²
198²
196²
128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388

模型代码与训练参数

```
You, 上周 | 1 author (You)
class UNet(nn.Module):
    def __init__(self, n_channels, n_classes, bilinear=False):
        super(UNet, self).__init__()
        self.n_channels = n_channels
        self.n_classes = n_classes
        self.bilinear = bilinear

        self.inc = (DoubleConv(n_channels, 64))
        self.down1 = (Down(64, 128))
        self.down2 = (Down(128, 256))
        self.down3 = (Down(256, 512))
        factor = 2 if bilinear else 1
        self.down4 = (Down(512, 1024 // factor))
        self.up1 = (Up(1024, 512 // factor, bilinear))
        self.up2 = (Up(512, 256 // factor, bilinear))
        self.up3 = (Up(256, 128 // factor, bilinear))
        self.up4 = (Up(128, 64, bilinear))
        self.outc = (OutConv(64, n_classes))

    def forward(self, x):
        x1 = self.inc(x)
        x2 = self.down1(x1)
        x3 = self.down2(x2)
        x4 = self.down3(x3)
        x5 = self.down4(x4)
        x = self.up1(x5, x4)
        x = self.up2(x, x3)
        x = self.up3(x, x2)
        x = self.up4(x, x1)
        logits = self.outc(x)
        return logits
```

标准UNet

```
# training methods
loss=CrossEntropy
lr=0.0001
batch_size=2
epoch=100
image_width=512
image_height=512
optimizer=RMSprop

# showing
log_visual=true
use_tqdm=false
only_best=true
```

RMSprop 更新法
Loss: CrossEntropy

加载数据集

```
def __getitem__(self, index) -> Tuple[Union[np.ndarray, torch.Tensor], Union[np.ndarray, torch.Tensor]]:
    """
    Input: image [H, W, C (RGB)]
    Train or Test: image [C (RGB), H, W], label [H, W]
    Note: label will be [H, W, C (Classes)] if one_hot is True
    """

    image = Image.open(os.path.join(self.path, self.image_paths[index]))
    image = image.resize((self.image_width, self.image_height), Image.BILINEAR)
    image = np.array(image)
    image = np.transpose(image, [2, 0, 1])

    label = Image.open(os.path.join(self.path, self.label_paths[index]))
    label = label.resize((self.image_width, self.image_height), Image.BILINEAR)
    label = np.array(label)

    if self.one_hot:
        label_one_hot = np.zeros((label.shape[0], label.shape[1], self.class_num))
        for i in range(self.class_num):
            label_one_hot[:,:,i] = (label == i)
        if self.to_torch:
            image = torch.from_numpy(image).float()
            label_one_hot = torch.from_numpy(label_one_hot).long()
        return image, label_one_hot
    else:
        if self.to_torch:
            image = torch.from_numpy(image).float()
            label = torch.from_numpy(label).long()
        return image, label
```

dataset.py数据集加载模块

参数选择

```python
def get_parser():
    import argparse

    parser = argparse.ArgumentParser(description='Train')
    parser.add_argument("--dataset", type=str, default="Kitti", help="dataset to use")
    parser.add_argument("--num_classes", type=int, default="34", help="number of classes")
    parser.add_argument("--data_root", type=str, default="./", help="data directory root path (where training/ testing/ or *.png is in)")
    parser.add_argument("--data_dir", type=str, default="dataset/", help="directory where data are saved")
    parser.add_argument("--save_root", type=str, default="./", help="save directory root path (where models/ is in)")
    parser.add_argument("--save_dir", type=str, default="models/", help="directory where models are saved")
    parser.add_argument("--log_root", type=str, default="./", help="log directory root path (where logs/ is in)")
    parser.add_argument("--log_dir", type=str, default="log/train/", help="directory where logs are saved")
    parser.add_argument("--loss", type=str, default="CrossEntropy", help="loss function to use")
    parser.add_argument("--lr", type=float, default="0.001", help="initial learning rate")
    parser.add_argument("--batch_size", type=int, default="2", help="size to train each batch")
    parser.add_argument("--epoch", type=int, default="10", help="train epochs")
    parser.add_argument("--image_width", type=int, default=640)
    parser.add_argument("--image_height", type=int, default=640)
    parser.add_argument("--optimizer", type=str, default="AdamW", help="optimizer to use")
    parser.add_argument("--log_visual", type=str2bool, default=True, help="save visualized picture while training")
    parser.add_argument("--use_tqdm", type=str2bool, default=True)
    parser.add_argument("--only_best", type=str2bool, default=True, help="only save best .pt")

    args = parser.parse_args()

    return args
```

设置lr, epoch等参数

```
python $train_path \
    --dataset $dataset_name \
    --num_classes $num_classes \
    --data_root $dataset_root \
    --data_dir $dataset_dir \
    --save_root $save_root \
    --save_dir $save_dir \
    --log_root $log_root \
    --log_dir $log_dir \
    --loss $loss \
    --lr $lr \
    --batch_size $batch_size \
    --epoch $epoch \
    --image_width $image_width \
    --image_height $image_height \
    --optimizer $optimizer \
    --log_visual $log_visual \
    --use_tqdm $use_tqdm \
    --only_best $only_best
```

训练脚本

训练代码

每个batch的操作

```
for iteration, (data, label) in enumerate(train_loader):

    data, label = data.cuda(), label.cuda()
    # label: N, H, W; pred_label: N, C, H, W
    pred_label = train_model(data)

    if args.loss == "CrossEntropy":
        # N, C, H, W => C, N*H*W
        pred_label = pred_label.contiguous().permute(0, 2, 3, 1)
        pred_label = pred_label.reshape(-1, pred_label.size(3))
        # N, C, H, W => C*N*H*W
        label = label.view(-1)
    else:
        pass
    loss = criterion(pred_label, label)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    # copy the tensor to host memory first
    t_pred_label = pred_label.cpu().detach().numpy()
    t_label = label.cpu().detach().numpy()
    # get max arg as output label
    t_pred_label = np.argmax(t_pred_label, axis=1)
    if args.loss == "CrossEntropy":
        pass
    elif args.loss == "Focal":
        t_label = np.transpose(t_label, [0, 3, 1, 2]).argmax(axis=1)
    # update accuracy
    acc = np.sum(t_label == t_pred_label) / np.prod(t_label.shape)

    if args.use_tqdm:
        pbar.set_description(f"Epoch {epoch+1}/{init_epoch} loss: {loss:.4f} train_acc: {acc:.4f}")
        pbar.update(1)
    else:
        ave_loss += loss / len(train_loader)
        ave_acc += acc / len(train_loader)
```

训练过程与结果截图

```
Epoch 9/100 loss: 0.2132 train_acc: 0.9460: 58%|#####7 | 4258/7396 [47:38<36:13, 1.44it/s]
```

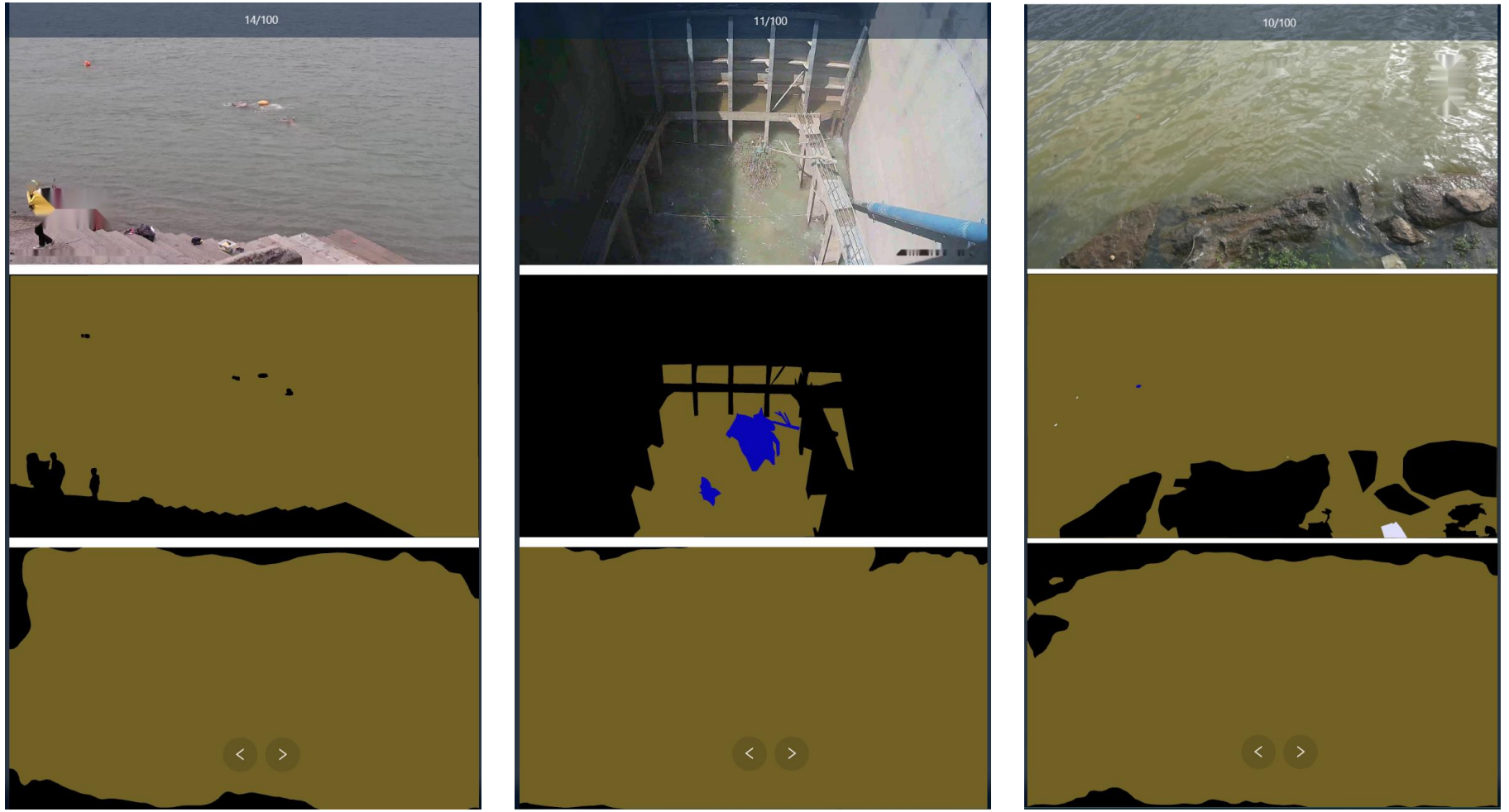
```
2023-04-29-01-42-28_epoch-100_lr-0.0001_loss-CrossEntropy_optim-RMSprop_best_acc-0.9133.pt[119M]
2023-04-29-01-42-28_epoch-100_lr-0.0001_loss-CrossEntropy_optim-RMSprop_best_acc-0.9072.pt[119M]
2023-04-29-01-42-28_epoch-100_lr-0.0001_loss-CrossEntropy_optim-RMSprop_best_acc-0.9011.pt[119M]
```

120525	标准测试	发起:2023-04-30 06:33:16 开始:2023-04-30 06:34:44 结束:2023-04-30 07:22:02	测试完成	0.1201	10.0447	0.1181	暂无排名	¥ -

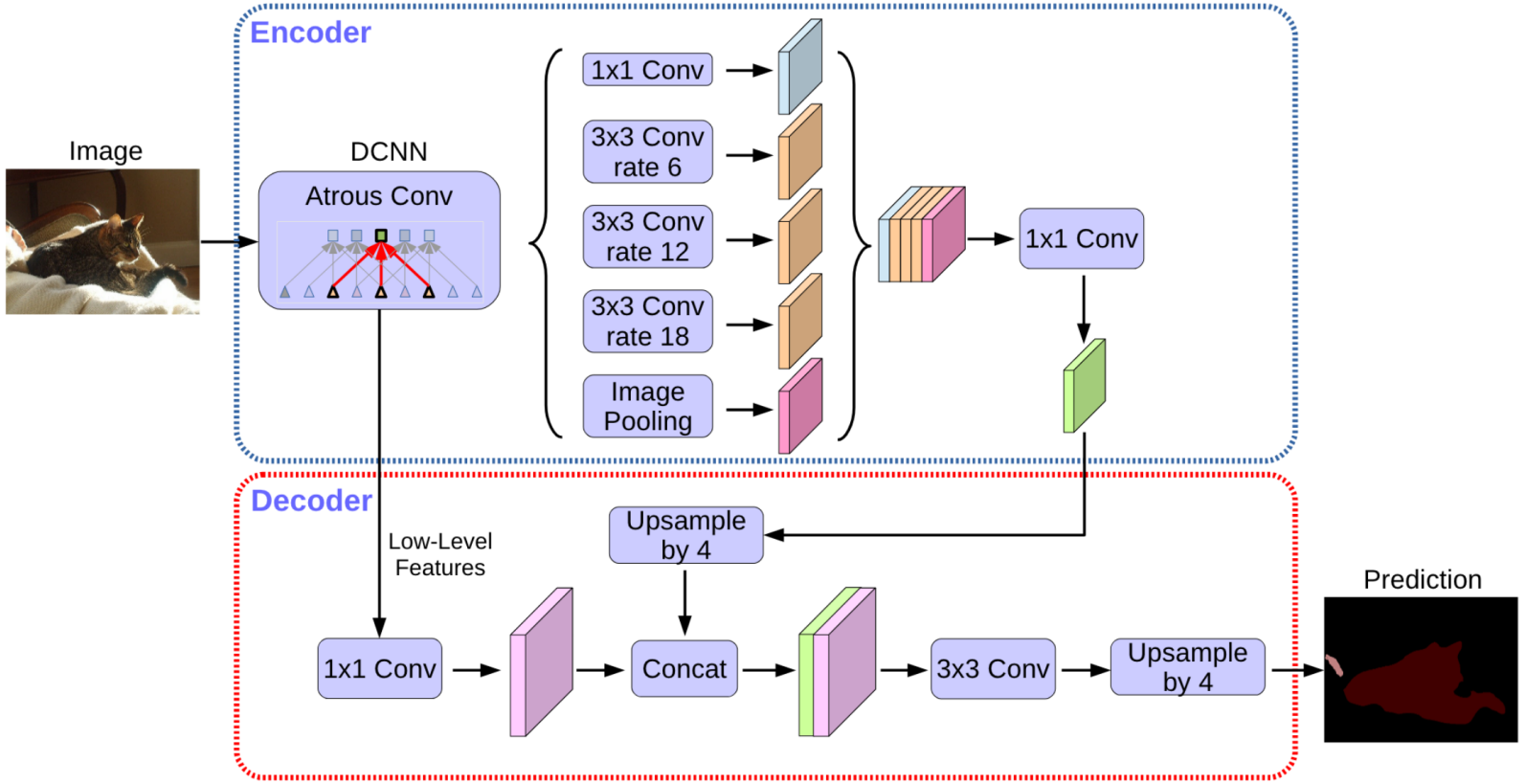
MIoU = 0.1201

输出样例

MIoU = 0.1201

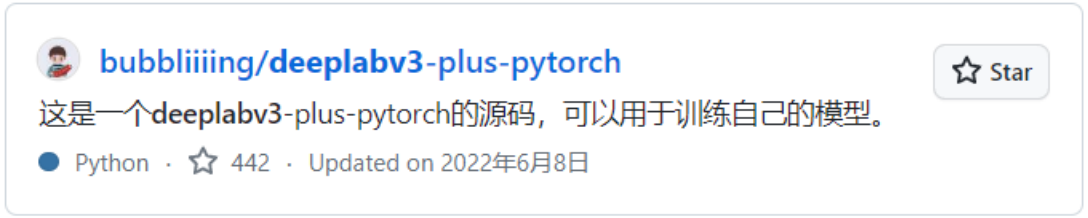


DeepLabv3+



网络模型

模型代码




```python
def forward(self, x):
    # x: C, H, W
    H, W = x.size(2), x.size(3)

    #-----------------------------------------#
    #   获得两个特征层
    #   low_level_features: 浅层特征-进行卷积处理
    #   x : 主干部分-利用ASPP结构进行加强特征提取
    #-----------------------------------------#
    low_level_features, x = self.backbone(x)
    x = self.aspp(x)
    low_level_features = self.shortcut_conv(low_level_features)

    #-----------------------------------------#
    #   将加强特征边上采样
    #   与浅层特征堆叠后利用卷积进行特征提取
    #-----------------------------------------#
    x = F.interpolate(x, size=(low_level_features.size(2), low_level_features.size(3)), mode='bilinear', align_corners=True)
    x = self.cat_conv(torch.cat((x, low_level_features), dim=1))
    x = self.cls_conv(x)
    x = F.interpolate(x, size=(H, W), mode='bilinear', align_corners=True)
    return x
```

该仓库下
的模型源码

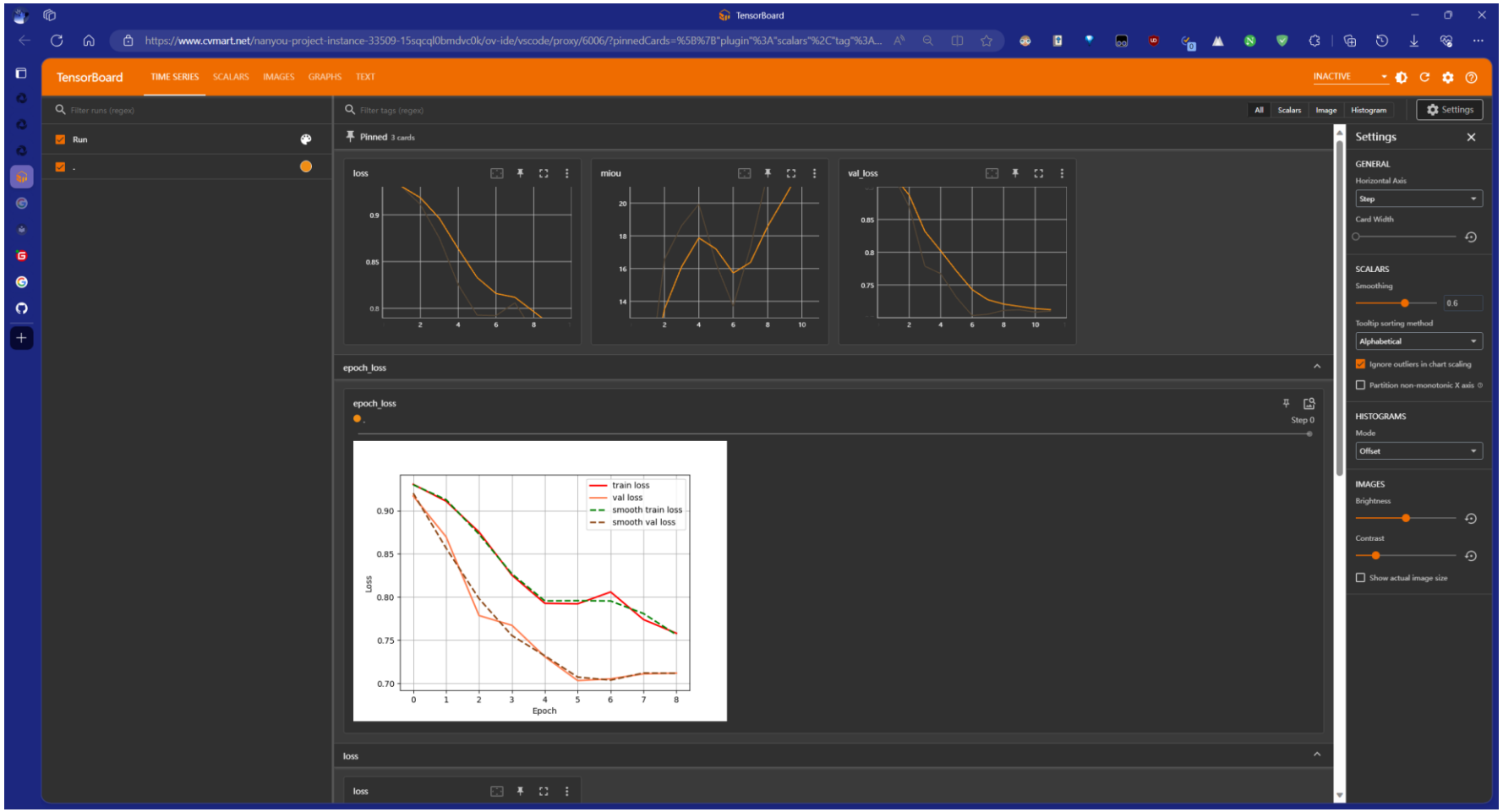
数据集预处理

```python
def build_dir_structure(data_path, root_path):

    """
    .
    ├── ImageSets
    ├── JPEGImages
    ├── SegmentationClass
    """

    # Make Directories
    os.makedirs(
        os.path.join(root_path, "ImageSets", "Segmentation"), exist_ok=True
    )
    os.makedirs(
        os.path.join(root_path, "JPEGImages"), exist_ok=True
    )
    os.makedirs(
        os.path.join(root_path, "SegmentationClass"), exist_ok=True
    )

    file_list = os.listdir(data_path)
    file_list = [ x[:-4] for x in file_list if x.endswith('.jpg') ]

    for x in file_list:
        copyfile(
            os.path.join(data_path, x + ".jpg"),
            os.path.join(root_path, "JPEGImages", x + ".jpg")
        )
        copyfile(
            os.path.join(data_path, x + ".png"),
            os.path.join(root_path, "SegmentationClass", x + ".png")
        )

    split(file_list, 0.9, os.path.join(root_path, "ImageSets", "Segmentation"))

    os.system(f"tree -d {root_path}")
```

```
/home/data/1945
├── ImageSets
│   └── Segmentation
├── JPEGImages
└── SegmentationClass
```

建立对应PascalVOC的目录架构

Tensorboard训练过程可视化



实时输出loss, miou等信息

输出优化

```
===> Epoch 41 / 50 starts
+ Start Train
- Finish Train
+ Start Validate
+ Get miou
+ Read val_images and Calculate miou
+ Calculate miou
Num classes 6
===> mIoU: 22.51; mPA: 24.79; Accuracy: 82.82
- Get miou done
===> In Unfrozen Epoch: 41 / 50
===> Total Loss: 0.722 || Val Loss: 0.701
- Best model stays at 25.2964
total 225M
drwxr-xr-x 1 root root 4.0K May  3 14:50 .
drwxr-xr-x 1 root root 4.0K Apr 30 19:09 ..
-rw-r--r-- 1 root root  23M May  3 14:48 best_epoch_weights.pth
-rw-r--r-- 1 root root  23M May  3 14:47 ep005-loss0.793-val_loss0.731.pth
-rw-r--r-- 1 root root  23M May  3 14:47 ep010-loss0.785-val_loss0.708.pth
-rw-r--r-- 1 root root  23M May  3 14:48 ep015-loss0.758-val_loss0.724.pth
-rw-r--r-- 1 root root  23M May  3 14:48 ep020-loss0.751-val_loss0.691.pth
-rw-r--r-- 1 root root  23M May  3 14:49 ep025-loss0.744-val_loss0.686.pth
-rw-r--r-- 1 root root  23M May  3 14:49 ep030-loss0.751-val_loss0.697.pth
-rw-r--r-- 1 root root  23M May  3 14:50 ep035-loss0.722-val_loss0.698.pth
-rw-r--r-- 1 root root  23M May  3 14:50 ep040-loss0.744-val_loss0.709.pth
-rw-r--r-- 1 root root  23M May  3 14:50 last_epoch_weights.pth
===> Epoch 41 / 50 cost 5.79 seconds
```

支持不使用tqdm输出训练进度

重写测试函数

```
# 添加灰边
scale = min(w / iw, h / ih)
nw = int(iw * scale)
nh = int(ih * scale)

image = cv2.resize(image, (nw, nh), interpolation=cv2.INTER_CUBIC)
new_image = np.full((h, w, 3), 128, dtype=np.uint8)
new_image[int((h-nh)/2):int((h-nh)/2)+nh, int((w-nw)/2):int((w-nw)/2)+nw] = image

                np.array(new_image, np.float32) / 255.0,

#---------------------------------------------------#
#    取出每一个像素点的种类
#---------------------------------------------------#
pred_label = F.softmax(pred_label.permute(1,2,0),dim = -1).cpu().numpy()
```

适应新代码框架重写测试函数

训练-测试-调优

120574	标准测试	● 测试完成	发起:04-30 11:15:45 开始:04-30 11:45:45 结束:04-30 12:50:00	56	5.6038	0.3002

冻结训练25epoch，解冻训练10epoch，验证集miou=0.4

训练-测试-调优

存在问题：分割准确率不高 => 继续训练

训练-测试-调优

120697	标准测试	● 测试完成	发起:04-30 19:40:26 开始:04-30 19:41:47 结束:04-30 20:44:05	62	4.4259	0.3196

冻结训练25epoch，解冻训练20epoch，验证集miou=0.42

存在问题：解冻训练提升较少，权重不合适

=> 修改权重，冻结训练

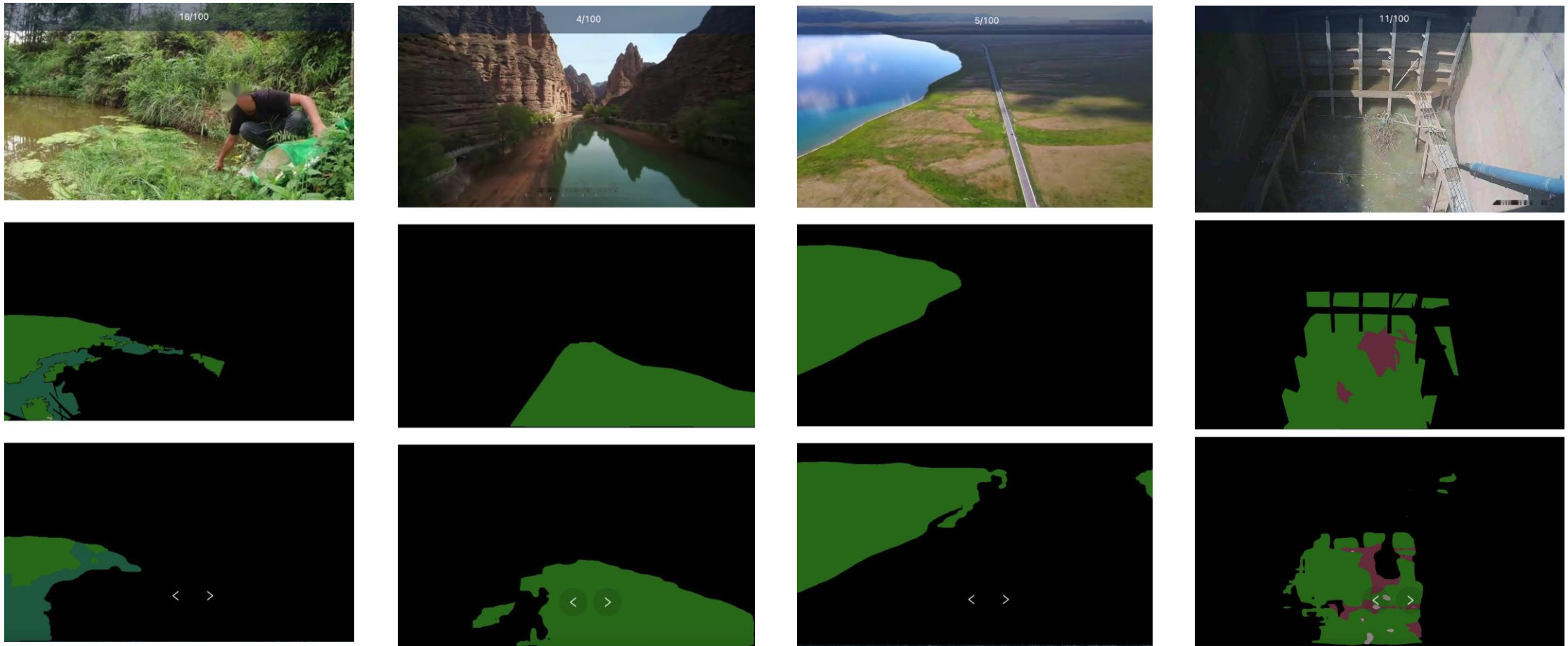
```
--focal_loss \
--dice_loss \
--class_weights 1 2.3697 787.4016 628.9308 270.2703 1.7953
```

权重=1/数据量比例*重要程度

训练-测试-调优

120727	标准测试	● 测试完成	发起:04-30 22:01:41 开始:04-30 22:03:13 结束:04-30 23:02:01	59	5.2393	0.3683

冻结训练25epoch，解冻训练10epoch，冻结训练20epoch，验证集miou=0.45



未来改进方向

- 模型始终没有达到过拟合，可以继续进行更多的训练让模型达到收敛，确定模型的最佳性能
- 类别权重需要继续调整，由于水草、垃圾等权重过高，在某些图片中错误地将背景预测成水草、垃圾等
- Backbone可以尝试使用Xception模型，从而可以获得更高的miou

感 谢 聆 听

第12组 孟祥宇 舒适