修改1

原文代码

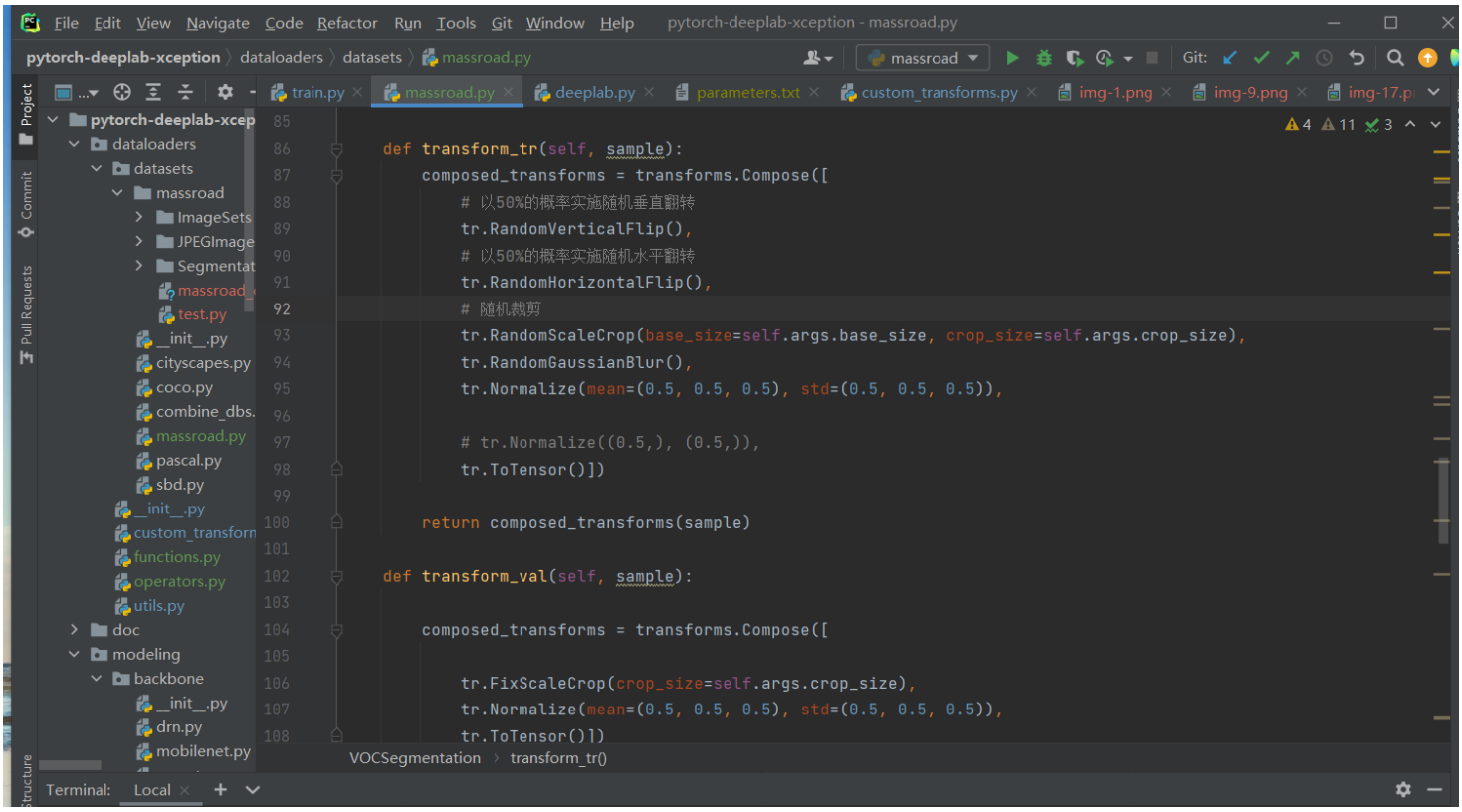
启动前应补充包

```
pip install scipy

pip install pycocotools

pip install tensorboardX
```

数据预处理

参考方法

仿照VOC数据集来读取马萨诸塞州道路数据集，数据预处理文件 `massroad.py` ,其中为了防止过拟合，用了50%随机垂直翻转和水平翻转和随机裁剪，又用了归一化到-1到1



仿VOC格式

文件结构

```
- ImageSets
        - Segmentation
                - train.txt
                - trainval.txt
                - val.txt
- JPEGImages
- SegmentationClass
```

JPEGImages 放原图

SegmentationClass里面放对应的mask图片png格式,和JPEGImages里的图片一一对应

ImageSets/ Segmentation的txt文件中放去掉后缀的图片名

修改代码

1.在mypath.py中添加自己的数据集名称与路径

```python
class Path(object):
    @staticmethod
    def db_root_dir(dataset):
        if dataset == 'pascal':
            return '/path/to/datasets/VOCdevkit/VOC2012/'  # folder that co
        elif dataset == 'sbd':
            return '/path/to/datasets/benchmark_RELEASE/'  # folder that co
        elif dataset == 'cityscapes':
            return '/path/to/datasets/cityscapes/'     # foler that contain
        elif dataset == 'coco':
            return '/path/to/datasets/coco/'
        elif dataset =='massroad':
            return '/path/to/datasets/massroad'
        else:
            print('Dataset {} not available.'.format(dataset))
            raise NotImplementedError
```

2.在同级目录中修改train.py约185行添加自己数据集的名称

mypath.py × mobilenet.py × resnet.py × parameters.txt × train.py × coco.py ×

```python
            self.best_pred = new_pred
            self.saver.save_checkpoint({
                'epoch': epoch + 1,
                'state_dict': self.model.module.state_dict(),
                'optimizer': self.optimizer.state_dict(),
                'best_pred': self.best_pred,
            }, is_best)

def main():
    parser = argparse.ArgumentParser(description="PyTorch DeeplabV3Plus Trainin
    parser.add_argument('--backbone', type=str, default='resnet',
                        choices=['resnet', 'xception', 'drn', 'mobilenet'],
                        help='backbone name (default: resnet)')
    parser.add_argument('--out-stride', type=int, default=16,
                        help='network output stride (default: 8)')
    parser.add_argument('--dataset', type=str, default='pascal',
                        choices=['pascal', 'coco', 'cityscapes','massroad'],
                        help='dataset name (default: pascal)')
    parser.add_argument('--use-sbd', action='store_true', default=True,
                        help='whether to use SBD dataset (default: True)')
    parser.add_argument('--workers', type=int, default=4,
                        metavar='N', help='dataloader threads')
    parser.add_argument('--base-size', type=int, default=513,
                        help='base image size')
    parser.add_argument('--crop-size', type=int, default=513,
                        help='crop image size')
    parser.add_argument('--sync-bn', type=bool, default=None,
```

3.在dataloaders目录下修改__init__.py

在第一行添加数据集名称，复制'pascal'数据集描述，把名称修改为自己数据集的名字

```python
        val_set = pascal.VOCSegmentation(args, split='val')
        if args.use_sbd:
            sbd_train = sbd.SBDSegmentation(args, split=['train', 'val'])
            train_set = combine_dbs.CombineDBs([train_set, sbd_train], excluded=[val_

        num_class = train_set.NUM_CLASSES
        train_loader = DataLoader(train_set, batch_size=args.batch_size, shuffle=True
        val_loader = DataLoader(val_set, batch_size=args.batch_size, shuffle=False, *
        test_loader = None

        return train_loader, val_loader, test_loader, num_class
    elif args.dataset == 'massroad':
        train_set = massroad.VOCSegmentation(args, split='train')
        val_set = massroad.VOCSegmentation(args, split='val')
        # if args.use_sbd:
        #     sbd_train = sbd.SBDSegmentation(args, split=['train', 'val'])
        #     train_set = combine_dbs.CombineDBs([train_set, sbd_train], excluded=[va

        num_class = train_set.NUM_CLASSES
        train_loader = DataLoader(train_set, batch_size=args.batch_size, shuffle=True
        val_loader = DataLoader(val_set, batch_size=args.batch_size, shuffle=False, *
        test_loader = None

        return train_loader, val_loader, test_loader, num_class

    elif args.dataset == 'cityscapes':
        train_set = cityscapes.CityscapesSegmentation(args, split='train')
```

4.修改dateloaders目录下utils.py

设置每一类别的颜色显示

```python
def get_massroad_labels():
    return np.array([
        [0,0,0],[255,255,255]
    ])
```

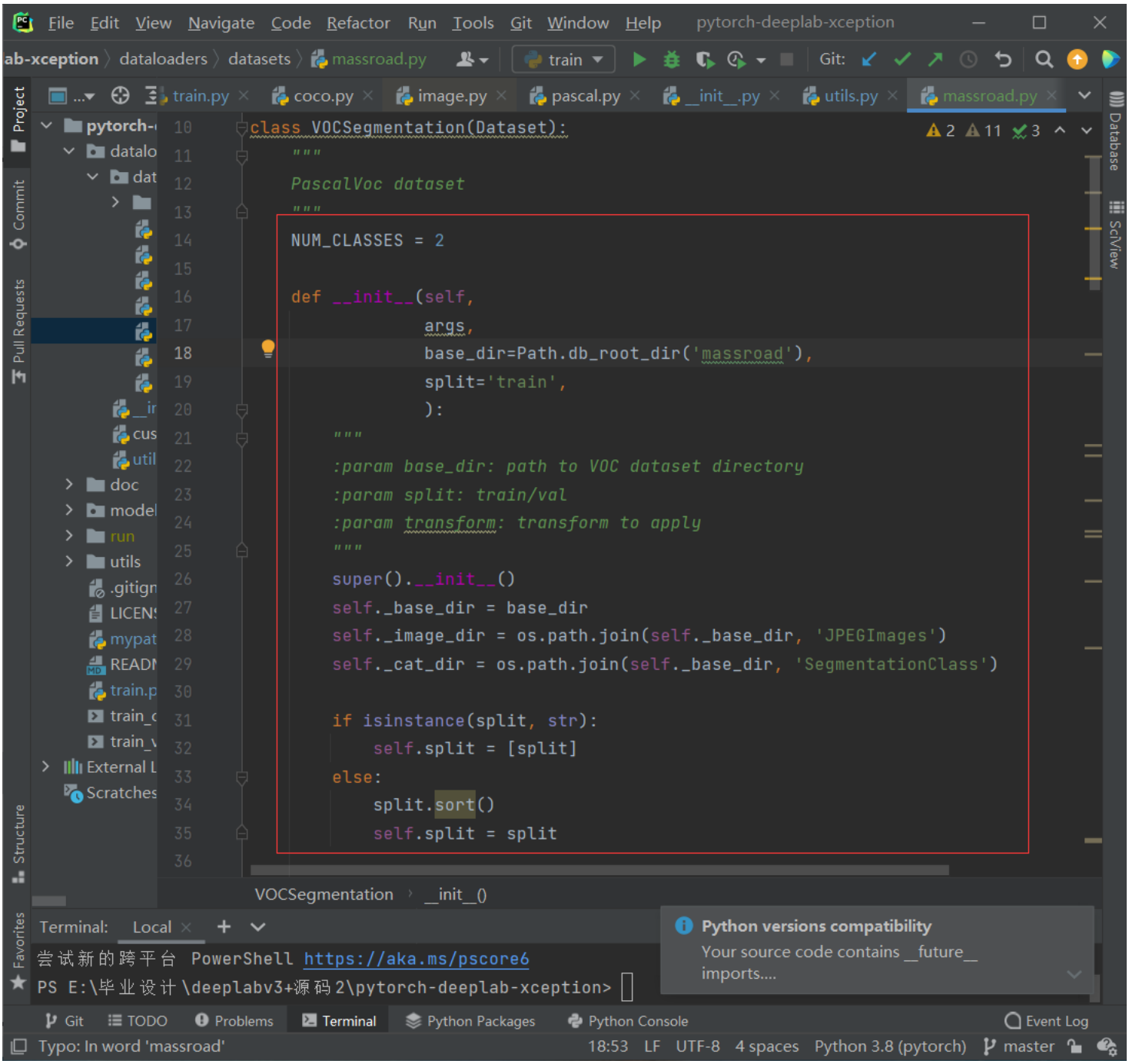
添加分割类别数

train.py | coco.py | image.py | pascal.py | __init__.py | utils.py | massro

```python
    return rgb_masks


def decode_segmap(label_mask, dataset, plot=False):
    """Decode segmentation class labels into a color image
    Args:
        label_mask (np.ndarray): an (M,N) array of integer values denoting
          the class label at each spatial location.
        plot (bool, optional): whether to show the resulting color image
          in a figure.
    Returns:
        (np.ndarray, optional): the resulting decoded color image.
    """
    if dataset == 'pascal' or dataset == 'coco':
        n_classes = 21
        label_colours = get_pascal_labels()
    elif dataset == 'cityscapes':
        n_classes = 19
        label_colours = get_cityscapes_labels()
    elif dataset == 'massroad':
        n_classes=2
        label_colours =get_massroad_labels()
    else:
        raise NotImplementedError

    r = label_mask.copy()
    g = label_mask.copy()
```

```python
    return rgb_masks


def decode_segmap(label_mask, dataset, plot=False):
    """Decode segmentation class labels into a color image
    Args:
        label_mask (np.ndarray): an (M,N) array of integer values denoting
          the class label at each spatial location.
        plot (bool, optional): whether to show the resulting color image
          in a figure.
    Returns:
        (np.ndarray, optional): the resulting decoded color image.
    """
    if dataset == 'pascal' or dataset == 'coco':
        n_classes = 21
        label_colours = get_pascal_labels()
    elif dataset == 'cityscapes':
        n_classes = 19
        label_colours = get_cityscapes_labels()
    elif dataset == 'massroad':
        n_classes=2
        label_colours =get_massroad_labels()
    else:
        raise NotImplementedError

    r = label_mask.copy()
    g = label_mask.copy()
```

定义massroad.py文件



模型测试

参考方法

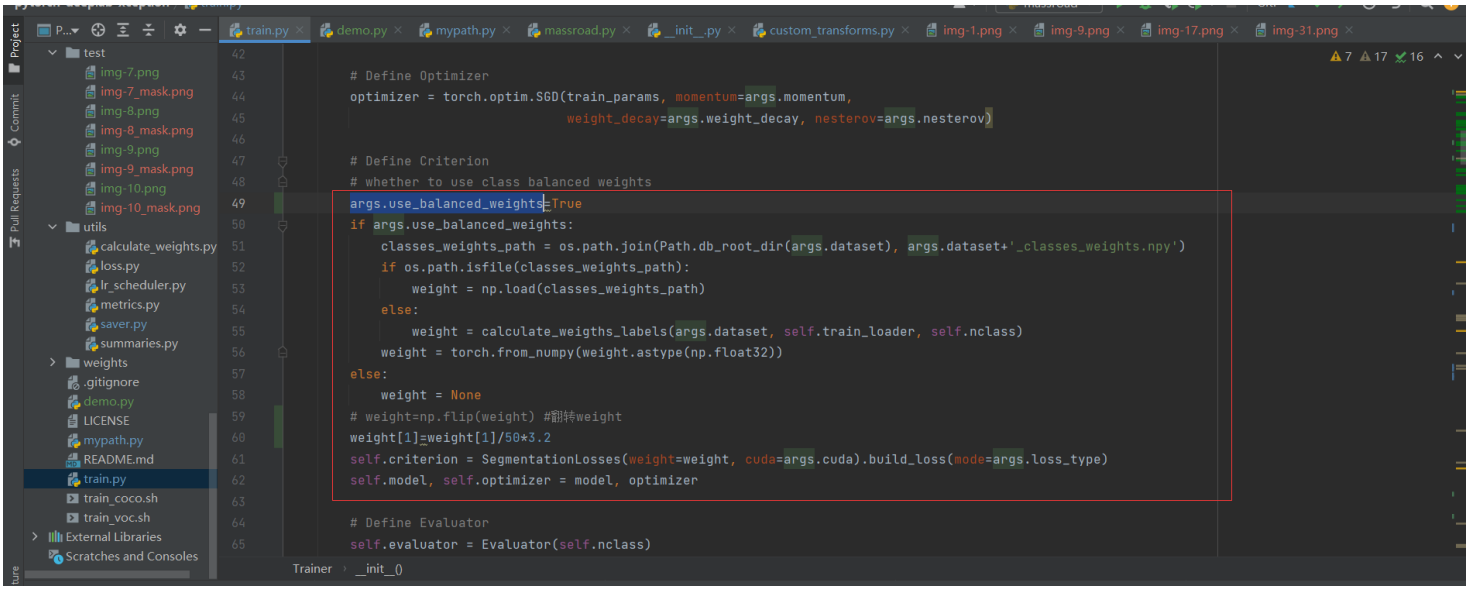
测试指令

```
python demo.py --in-path ./test --ckpt run/massroad/deeplab-resnet/model_best.pth.tar --backbone resnet --dataset massroad
```

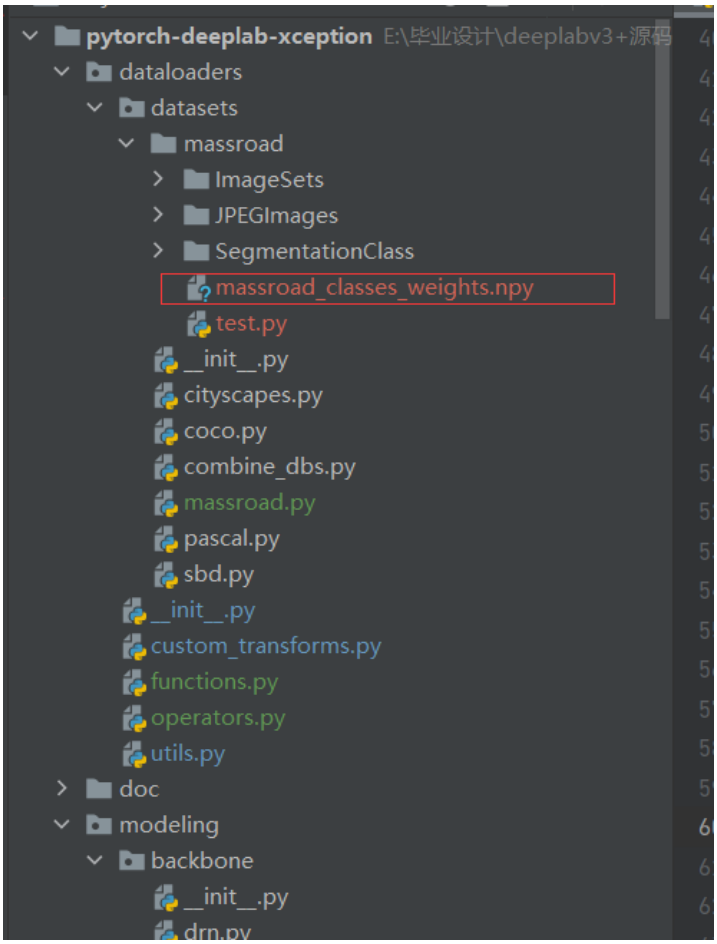
修改测试数据集参数，修改批量归一化到（-1，1）

测试图片放到test文件夹下

启用默认参数（args.use_balanced_weights）



默认情况下该参数是关闭的，但是我们知道数据集中道路标签相比背景占比过低，导致训练的图像道路总是缺损。启用该参数后，会生成一个权重文件。



通过 `weight[1]=weight[1]/50*3.1` ,可以通过该语句修改权重比达到更好的mIoU，但是达到最好的mIoU时（`weight[1]=weight[1]/50*3.1`），主干道路本身会有缺损，而且会有一定的在标签图不存在的细小道路被标注出来

保存更多的实验结果

原代码只能保留最多十个实验结果，修改代码后可以保留无上限，但是缺点是必须预先生成目录

```python
class Saver(object):

    def __init__(self, args):
        self.args = args
        self.directory = os.path.join('run', args.dataset, args.checkname)
        self.runs = sorted(glob.glob(os.path.join(self.directory, 'experiment_*')))
        # run_id = int(self.runs[-1].split('_')[-1]) + 1 if self.runs else 0
        max = 0
        for dir in os.listdir(self.directory):
            s1 = re.sub('\D', '', dir)
            if (s1 != ""):
                if max < int(s1):
                    max = int(s1)
        run_id=max+1
        self.experiment_dir = os.path.join(self.directory, 'experiment_{}'.format(str(run_id)))
        if not os.path.exists(self.experiment_dir):
            os.makedirs(self.experiment_dir)

    def save_checkpoint(self, state, is_best, filename='checkpoint.pth.tar'):
        """Saves checkpoint to disk"""
        filename = os.path.join(self.experiment_dir, filename)
        torch.save(state, filename)
        if is_best:
```

网络结构试调整

解决忽略白色像素点问题

在VOC数据集中，标签标注的标签类边缘用255（白色像素点）填充，也就是默认是忽略白色像素点，但是我们的massroad数据集就是用白色标签来标注道路的。为了解决这个问题，我们采取如下策略：直接除以255，将其换为1，而原图经过批量规范化后范围也是（-1，1）

```python
def training(self, epoch):
    train_loss = 0.0
    self.model.train()
    tbar = tqdm(self.train_loader)
    num_img_tr = len(self.train_loader)
    for i, sample in enumerate(tbar):
        image, target = sample['image'], sample['label']/255
        # grid_image = make_grid(target,
        #                        3, normalize=False, range=(0, 255))
        # save_image(grid_image, "./test/test.png")
        # grid_image = make_grid(image,
        #                        3, normalize=False, range=(0, 255))
        # save_image(grid_image, "./test/test1.png")
        if self.args.cuda:
            image, target = image.cuda(), target.cuda()
        self.scheduler(self.optimizer, i, epoch, self.best_pred)
        self.optimizer.zero_grad()
        output = self.model(image)
        loss = self.criterion(output, target)
        loss.backward()
        self.optimizer.step()
        train_loss += loss.item()
        tbar.set_description('Train loss: %.3f' % (train_loss / (i + 1)))
```

```python
    def validation(self, epoch):
        self.model.eval()
        self.evaluator.reset()
        tbar = tqdm(self.val_loader, desc='\r')
        test_loss = 0.0
        for i, sample in enumerate(tbar):
            image, target = sample['image'], sample['label']/255
            if self.args.cuda:
                image, target = image.cuda(), target.cuda()
            with torch.no_grad():
                output = self.model(image)
            loss = self.criterion(output, target)
            test_loss += loss.item()
            tbar.set_description('Test loss: %.3f' % (test_loss / (i + 1)))
            pred = output.data.cpu().numpy()
            target = target.cpu().numpy()
            pred = np.argmax(pred, axis=1)
            # Add batch sample into evaluator
            self.evaluator.add_batch(target, pred)
```

Trainer › __init__()

尝试在ASPP模块中加入注意力模块

参考代码1

参考代码2

改进主要在 `aspp.py` 中,试了三个模块，加上之后反而有一定程度下滑，从0.6948将为0.687

注意力模块

```python
class SE_Block(nn.Module):                        # Squeeze-and-Excitation block
    def __init__(self, in_planes):
        super(SE_Block, self).__init__()
        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
        self.conv1 = nn.Conv2d(in_planes, in_planes // 16, kernel_size=1)
        self.relu = nn.ReLU()
        self.conv2 = nn.Conv2d(in_planes // 16, in_planes, kernel_size=1)
        self.sigmoid = nn.Sigmoid()


    def forward(self, x):
        x = self.avgpool(x)
        x = self.conv1(x)
        x = self.relu(x)
        x = self.conv2(x)
        out = self.sigmoid(x)
        return out

class ECALayer(nn.Module):
    def __init__(self, k_size=3):
        super(ECALayer, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.conv = nn.Conv1d(1, 1, kernel_size=k_size, padding=(k_size - 1) // 2, bias=False)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        # x: input features with shape [b, c, h, w]
        b, c, h, w = x.size()
        # feature descriptor on the global spatial information
        y = self.avg_pool(x)
        # Two different branches of ECA module
        y = self.conv(y.squeeze(-1).transpose(-1, -2)).transpose(-1, -2).unsqueeze(-1)
        # Multi-scale information fusion
        y = self.sigmoid(y) ## y为每个通道的权重值
        return x * y.expand_as(x) ##将y的通道权重一一赋值给x的对应通道

class SELayer(nn.Module):
    def __init__(self, channel, reduction=16):
        super(SELayer, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.fc = nn.Sequential(
            nn.Linear(channel, channel // reduction, bias=False),
            nn.ReLU(inplace=True),
            nn.Linear(channel // reduction, channel, bias=False),
            nn.Sigmoid()
        )

    def forward(self, x):
        b, c, _, _ = x.size()
        y = self.avg_pool(x).view(b, c)
        y = self.fc(y).view(b, c, 1, 1)
        return x * y.expand_as(x)
```

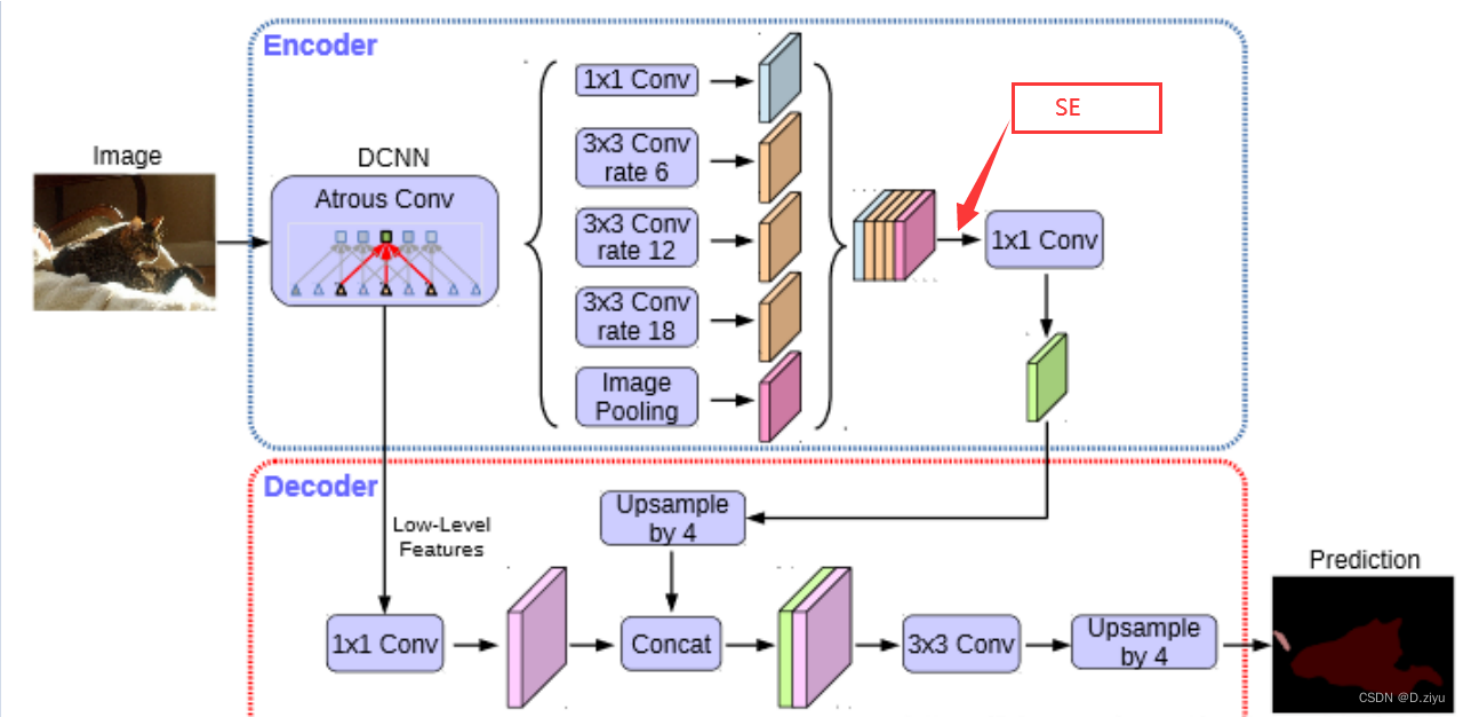
具体加入地方

train.py × aspp.py × deeplab.py × mypath.py × custom_transforms.py × utils.py × saver.py × img-7_mask.png × img-7.png ×

```python
        self.conv1 = nn.Conv2d(1280, 256, 1, bias=False)
        self.bn1 = BatchNorm(256)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.5)
        self._init_weight()
        # SE_Block
        self.conv_cat = nn.Sequential(
            nn.Conv2d(256 * 5, 256, 1, 1, padding=0, bias=True),
            nn.BatchNorm2d(256, momentum=0.1),
            nn.ReLU(inplace=True),
        )
        self.senet = SE_Block(in_planes=256 * 5)
        # ECA
        k_size = 3
        self.eca = ECALayer(k_size)
        #
        """se"""
        reduction = 16  #
        self.se1 = SELayer(256*5, reduction)  # 我的c1_in_channels为256，所以直接就是10*256=2560，记得修改
```

```python
        x5 = F.interpolate(x5, size=x4.size()[2:], mode='bilinear', align_corners=True)
        x = torch.cat((x1, x2, x3, x4, x5), dim=1)
        #
        # seaspp1 = self.senet(x)
        # se_feature_cat = seaspp1 * x
        # result = self.conv_cat(se_feature_cat)
        # return result
        # return self.dropout(result)
        # aspp_outs = self.eca(x)
        # return aspp_outs
        """SE"""
        aspp_outs = self.se1(x)
        # return aspp_outs

        x = self.conv1(aspp_outs)
        x = self.bn1(x)
        x = self.relu(x)

        return self.dropout(x)
```

对应网络结构



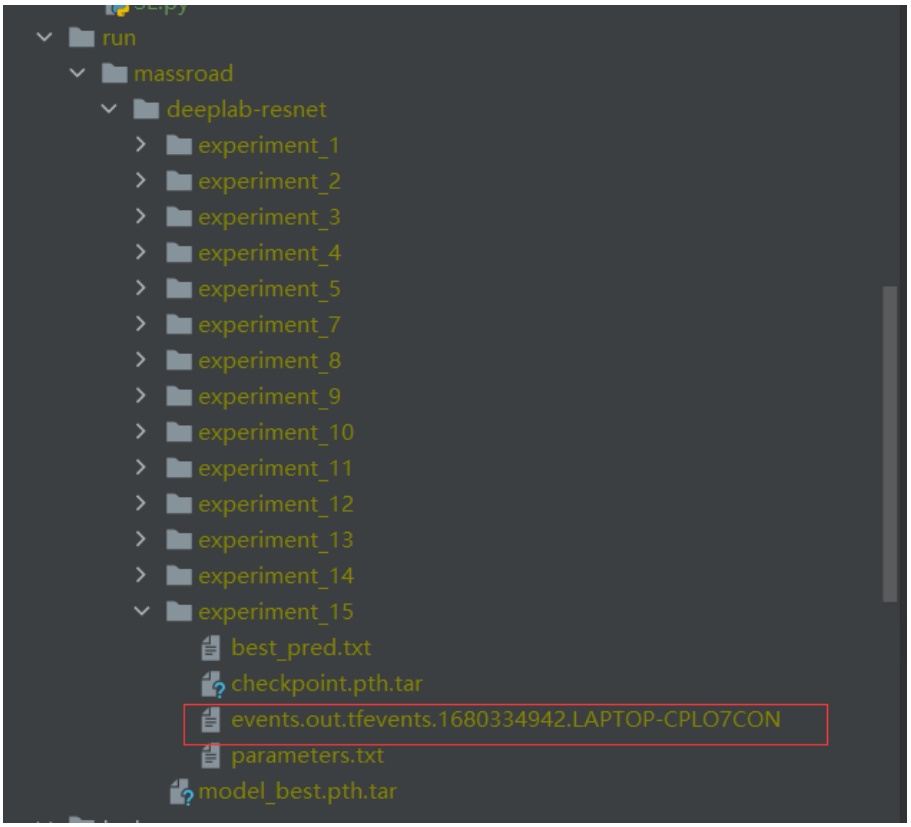
其他

尝试在 `deeplab.py` 中修改前向传播参数，没啥用

```
def forward(self, input):
    x, low_level_feat = self.backbone(input)
    x = self.aspp(x)
    x = self.decoder(x, low_level_feat)
    # x = F.interpolate(x, size=input.size()[2:], mode='bilinear', align_corners=True)
    x = F.interpolate(x, size=input.size()[2:], mode='bilinear', align_corners=False)

    # x=255*x
    return x


def freeze_bn(self):
    for m in self.modules():
```

使用内置tensorboard查看训练验证情况

用自带的tensorboard查看实验产生日志就行了



训练指令

```
python train.py --dataset massroad --backbone resnet --lr 0.01 --workers 1 --epochs 100 --batch-size 4 --gpu-ids 0
```

增加epoch，增加初始化学习率和修改batch_size确实会对评价指标产生影响，多次测试得到如下结果

实验编号	深度学习框架	backone	lr(初始学习率)	epochs	batch_size	weight[1]	mIoU	图像后处理	注意力机制
1	pytorch	mobilnet	0.007	30	8	1	0.5943	无	无
2	pytorch	mobilnet	0.007	30	4	1	0.6104	无	无
3	pytorch	resnet_100	0.007	30	4	1	0.6206	无	无
4	pytorch	resnet_100	0.01	100	4	1	0.6396	无	无
5	pytorch	resnet_100	0.01	100	4	2.5	0.6923	无	无
6	pyrorch	resnet_100	0.01	100	4	3	0.6946	无	无
7	pytorch	resnet_100	0.01	100	4	3.1	0.6944	无	无
8	pytorch	resnet_100	0.01	100	4	3.2	0.6948	无	无
9	pytorch	resnet_100	0.01	100	4	3.3	0.6946	无	无
10	pytroch	resnet_100	0.01	100	4	3.4	0.6944	无	无
11	pytroch	resnet_100	0.01	100	4	3.5	0.6946	无	无
12	pytroch	resnet_100	0.01	100	4	5	0.6654	无	无
13	pytroch	resnet_100	0.01	100	4	10	0.6408	无	无
14	pytroch	resnet_100	0.01	100	4	3.4	0.5383	无	无

实验编号	深度学习框架	backone	lr(初始学习率)	epochs	batch_size	weight[1]	mIoU	图像后处理	注意力机制
15	paddlepaddle	resnet_50-vd	0.007	100	8	--	0.7803	--	无
16	paddlepaddle	resnet_50-vd	0.007	100	4	--	0.7874	--	无

备注：

weight[1]可以调整（指定）白色（道路）标签权重

paddlepaddle未启用指定权重，白色（道路）与黑色（背景）的权重会随着模型训练而发生改变

--：代表不确定

个人研究认为，paddlepaddle使用的paddlers的deeplab v3+和基于pytorch的deeplab v3+的网络结构是基本一致的。个人认为区别为在模型结构上，paddlers使用ResNet50-vd代替标准的ResNet50作为backbone，同时使用了具有更好指标的ImageNet预训练权重，这方面与原始DeepLab V3+可能有一些区别。

Others

也添加了一些无用类，在project中没啥用，本来想删忘删了

可能还有其他改进，暂时忘了