课程实践 - Project1

语义分割 - 地表建筑物识别

复旦大学计算与智能创新学院

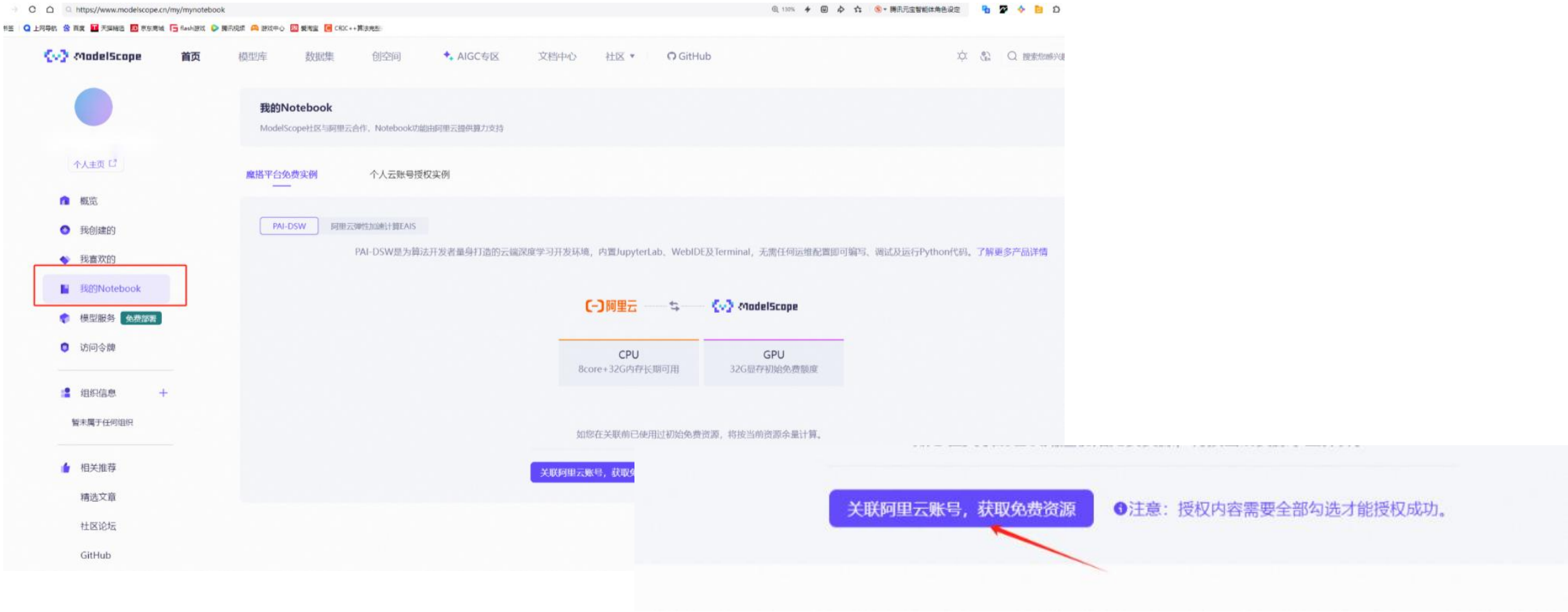
陈智能

2025-9-29

免费GPU服务器使用教程

1.登录ModelScope：https://www.modelscope.cn/

2.关联阿里云账号



免费GPU服务器使用教程

3.选择环境并启动（注意PAI-DSW环境下可以将数据/代码放到/mnt/workspace/下进行长期存放，而阿里云弹性加速计算EAIS环境下除了.ipynb文件外不会自动存储！）

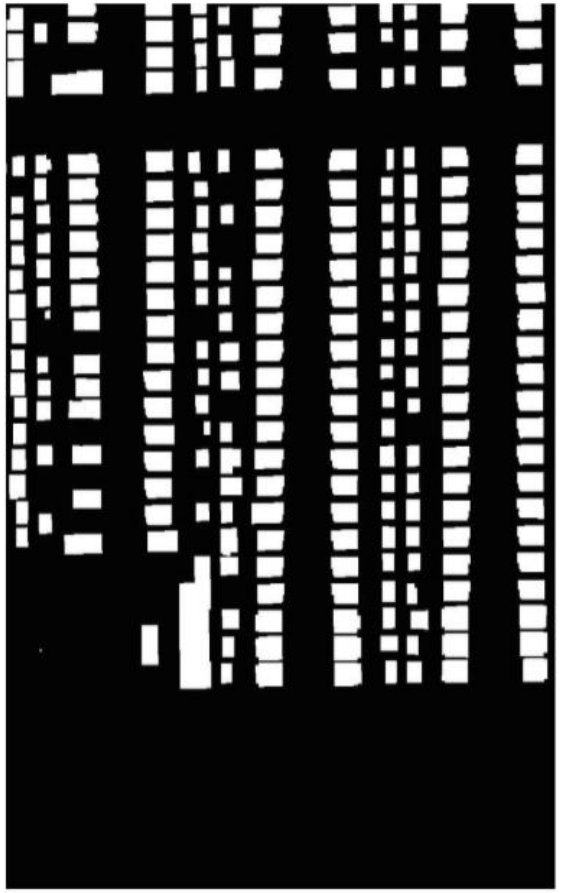
任务介绍

无人机技术高速发展的今天，面对海量的航拍图像，AI如何识别地表建筑物呢?

赛题以计算机视觉为背景，要求选手使用给定的航拍图像训练模型并完成地表建筑物识别任务。

赛题链接：【AI入门系列】地球观察员：建筑物识别学习赛_学习赛_天池大赛-阿里云天池的赛制

Chicago



Chicago - reference

数据集介绍

赛题数据来源（Inria Aerial Image Labeling），并进行拆分处理。赛题数据为航拍图，需要参赛选手识别图片中的地表建筑具体像素位置。

具体的标签为图像像素类别，在赛题数据中像素属于2类（无建筑物和有建筑物），因此标签为有建筑物的像素。原始图片为jpg格式，标签为RLE编码的字符串。

FileName	Size	含义
test_a.zip	314.49MB	测试集A榜图片
test_a_samplesubmit.csv	46.39KB	测试集A榜提交样例
train.zip	3.68GB	训练集图片
train_mask.csv.zip	97.52MB	训练集图片标注

79LA4JOU0T.jpg 69358 27 69870 27 70382 27 70894 27 71406 27 … …

采用RLE编码，RLE是一种压缩格式，通过记录“连续相同值的起始位置 + 长度” 来表示二值掩码，比如69358 27：表示从一维掩码的 第69358个位置开始，有27个连续的建筑物像素。

Baseline

1.下载完整数据集、初始权重，放入对应文件夹进行解压（./data/）

2.建议使用wget命令下载

FileName	Size	Link
test_a.zip	314.49MB	http://tianchi-competition.oss-cn-hangzhou.aliyuncs.com/531872/%E5%9C%B0%E8%A1%A8%E5%BB%BA%E7%AD%91%E7%89%A9%E8%AF%86%E5%88%AB/test_a.zip
test_a_samplesubmit.csv	46.39KB	http://tianchi-competition.oss-cn-hangzhou.aliyuncs.com/531872/%E5%9C%B0%E8%A1%A8%E5%BB%BA%E7%AD%91%E7%89%A9%E8%AF%86%E5%88%AB/test_a_samplesubmit.csv
train.zip	3.68GB	http://tianchi-competition.oss-cn-hangzhou.aliyuncs.com/531872/%E5%9C%B0%E8%A1%A8%E5%BB%BA%E7%AD%91%E7%89%A9%E8%AF%86%E5%88%AB/train.zip
train_mask.csv.zip	97.52MB	http://tianchi-competition.oss-cn-hangzhou.aliyuncs.com/531872/%E5%9C%B0%E8%A1%A8%E5%BB%BA%E7%AD%91%E7%89%A9%E8%AF%86%E5%88%AB/train_mask.csv.zip

Baseline

2. RLE与图片之间进行转换

```python
# RLE编码：对掩码图像进行RLE编码
def rle_encode(im):
    # 将二维掩码图像im按列优先 order='F' 展平为一维数组 pixels
    pixels = im.flatten(order='F')
    # 在序列首尾添加0作为哨兵
    pixels = np.concatenate([[0], pixels, [0]])
    # 记录所有像素值发生变化的位置
    runs = np.where(pixels[1:] != pixels[:-1])[0] + 1
    # 将“跳变点位置”转换为“起始位置 + 连续长度”的编码格式
    runs[1::2] -= runs[::2]
    # 将runs数组中的整数转换为字符串，用空格拼接成最终的RLE编码字符串
    return ' '.join(str(x) for x in runs)


# RLE编码：将RLE编码解码为掩码图像
def rle_decode(mask_rle, shape=(512, 512)):
    # 如果输入的是空字符串，表示该图像没有建筑物，直接返回全为0的掩码图像
    if not mask_rle:
        return np.zeros(shape, dtype=np.uint8)
    # 将RLE字符串按空格拆分为列表
    s = mask_rle.split()
    # 将字符串列表转换为整数数组
    starts, lengths = [np.asarray(x, dtype=int) for x in (s[0::2], s[1::2])]
    # 调整起始位置并计算结束位置
    starts -= 1
    ends = starts + lengths
    # 生成掩码图像
    img = np.zeros(shape[0] * shape[1], dtype=np.uint8)
    for lo, hi in zip(starts, ends):
        img[lo:hi] = 1
    # 重塑为shape对应的二维数组
    return img.reshape(shape, order='F')
```

将掩码图像编码为rle格式

将rle格式进行解码为掩码图像

Baseline

3. 构建数据集

读取数据

简单数据增强

```python
class TianChiDataset(D.Dataset):
    def __init__(self, paths, rles=None, transform=None, test_mode=False):
        self.paths = paths
        self.rles = rles if not test_mode else ['' for _ in paths]
        self.transform = transform
        self.test_mode = test_mode

        # 图像转张量: 转换为PIL→尺寸调整→转为张量→标准化
        self.to_tensor = T.Compose([
            T.ToPILImage(),
            T.Resize(IMAGE_SIZE),
            T.ToTensor(),
            T.Normalize([0.625, 0.448, 0.688], [0.131, 0.177, 0.101]),
        ])

    def __getitem__(self, index):
        # 读取图像并转换为RGB
        img = cv2.imread(self.paths[index])
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        if not self.test_mode:
            # 训练模式: 加载掩码并增强
            mask = rle_decode(self.rles[index])
            augments = self.transform(image=img, mask=mask)
            return self.to_tensor(augments['image']), augments['mask'][None]
        else:
            # 测试模式: 仅返回图像
            return self.to_tensor(img), ''

    def __len__(self):
        return len(self.paths)
```

Baseline

4. 定义模型

```python
# 构建FCN-ResNet50分割模型
def get_model():
    # 初始化模型
    model = torchvision.models.segmentation.fcn_resnet50(weights=None)

    # 修改输出层为1个通道（二分类）
    model.classifier[4] = nn.Conv2d(512, 1, kernel_size=(1, 1), stride=(1, 1))
    return model.to(DEVICE)
```

初始化fcn_resnet50

修改最后一层

Baseline

5. 定义损失函数

使用Dice coefficient来衡量结果与真实标签的差异性，Dice coefficient可以按像素差异性来比较结果的差异性。

另外，采用组合损失函数，通过加权融合 BCE 损失和 Dice 损失，兼顾像素级分类准确性和区域级分割完整性。

Dice coefficient的具体计算方式如下：

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

```python
# Dice损失计算
class SoftDiceLoss(nn.Module):

    def __init__(self, smooth=1., dims=(-2, -1)):
        super().__init__()
        # 平滑项，防止分子 / 分母为 0 导致的计算错误
        self.smooth = smooth
        # 对高度（H）和宽度（W）维度求和，得到每个样本、每个类别的重叠度
        self.dims = dims

    def forward(self, x, y):
        # 计算True Positive: 预测和真实掩码都为1的区域
        tp = (x * y).sum(self.dims)
        # 计算False Positive: 预测为1但真实为0的区域
        fp = (x * (1 - y)).sum(self.dims)
        # 计算False Negative: 预测为0但真实为1的区域
        fn = ((1 - x) * y).sum(self.dims)
        # 计算Dice系数
        dc = (2 * tp + self.smooth) / (2 * tp + fp + fn + self.smooth)
        # 返回Dice损失
        return 1 - dc.mean()


# 组合损失函数: 80%BCE + 20%Dice
def get_loss_fn():
    # 初始化BCE损失与Dice损失
    bce_fn = nn.BCEWithLogitsLoss()
    dice_fn = SoftDiceLoss()

    # 加权融合两种损失函数
    def loss_fn(y_pred, y_true):
        return 0.8 * bce_fn(y_pred, y_true) + 0.2 * dice_fn(y_pred.sigmoid(), y_true)
    return loss_fn
```

Baseline

6. 训练代码

```python
# 模型训练
def train_model(train_loader, valid_loader):
    # 初始化模型、优化器、损失函数
    model = get_model()
    optimizer = torch.optim.AdamW(model.parameters(), lr=1e-4, weight_decay=1e-3)
    loss_fn = get_loss_fn()
    best_loss = float('inf')

    # 输出训练日志
    print("\n" + r"Epoch |  Train Loss |  Valid Loss | Time(m)")
    print("-" * 40)

    for epoch in range(1, EPOCHES + 1):
        start_time = time.time()
        model.train()
        train_losses = []

        # 训练轮次
        for img, target in tqdm(train_loader, desc=f"Epoch {epoch}"):
            img, target = img.to(DEVICE), target.float().to(DEVICE)
            optimizer.zero_grad()
            output = model(img)['out']
            loss = loss_fn(output, target)
            loss.backward()
            optimizer.step()
            train_losses.append(loss.item())

        # 验证轮次
        model.eval()
        valid_losses = []
        with torch.no_grad():
            for img, target in valid_loader:
                img, target = img.to(DEVICE), target.float().to(DEVICE)
```

数据加载
模型加载
优化器
损失函数

Baseline

7. 预测代码

```python
# 模型测试
def predict_test(model):
    """预测测试集并生成提交文件"""
    # 读取测试集信息
    test_mask = pd.read_csv(TEST_CSV, sep='\t', names=['name', 'mask'])

    # 拼接测试图像完整路径
    test_mask['name'] = test_mask['name'].apply(lambda x: os.path.join(TEST_IMG_DIR, x))

    # 构建测试数据集
    test_dataset = TianChiDataset(
        paths=test_mask['name'].values,
        test_mode=True
    )

    # 预测并编码
    model.eval()
    submissions = []
    for idx, (img, _) in tqdm(enumerate(test_dataset), total=len(test_dataset)):
        with torch.no_grad():
            # 模型预测
            img = img.to(DEVICE)[None]
            pred = model(img)['out'][0][0].sigmoid().cpu().numpy()
            # 二值化+恢复尺寸
            pred_mask = (pred > 0.5).astype(np.uint8)
            pred_mask = cv2.resize(pred_mask, (512, 512))
            # RLE编码
            rle = rle_encode(pred_mask)
            # 提取文件名
            img_filename = os.path.basename(test_mask['name'].iloc[idx])
            submissions.append([img_filename, rle])
```

提交结果

日期: 2025-09-28 21:50:08

分数: 0.8124

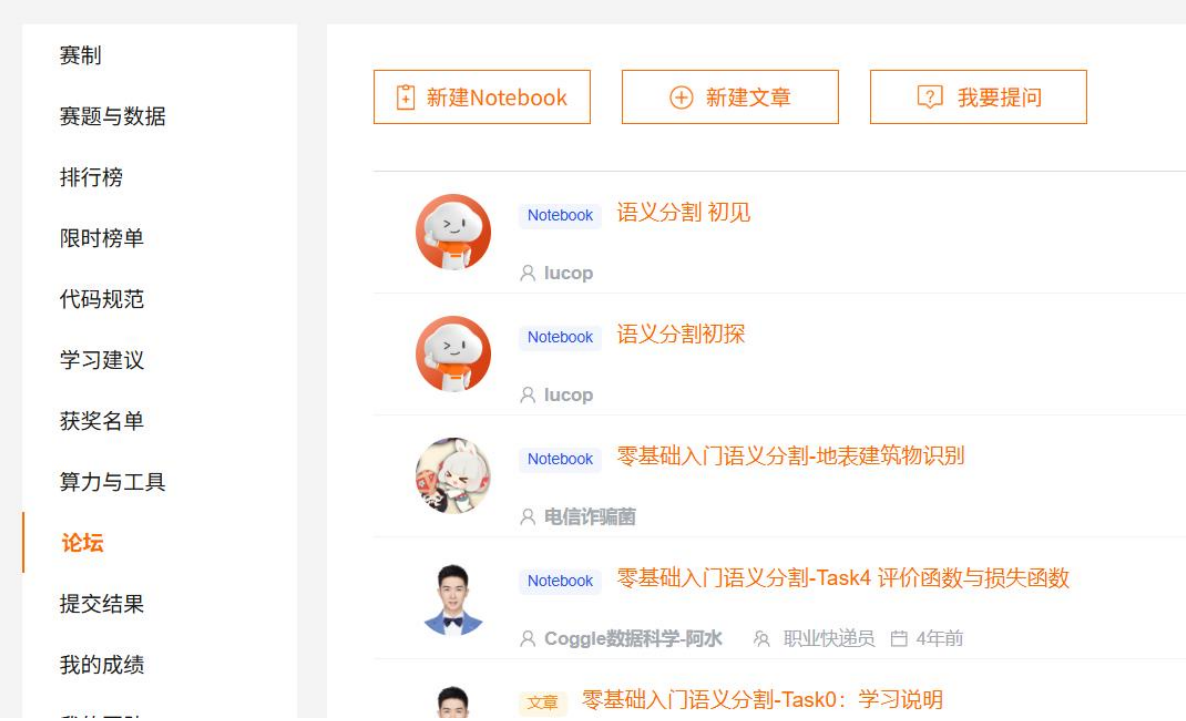
改进思路

1. 在Baseline基础上改进，包括但不限于选用更深更先进的CNN网络或其它模型，调整数据增强方案（可增加空间变换、像素变换、混合增强，覆盖更多真实场景变异），调整损失函数，调整超参数等；

2. 针对数据集，进行数据预处理优化和后处理优化（分割结果修正）等，比如进行图像噪声去除，进行图像标准化等；

3. 采用模型集成方案，综合考虑不同模型的预测结果，对概率分布或最终预测结果等进行加权投票；

参考资源



论坛里有一些高分实现方案可以作为直接参考，直接复现就能得到不错的结果。若参考了论坛里的实现，要求必须在报告中引用链接，并要求在复现其结果的基础上有一定程度的修改（变好或变差都行）

[milesial/Pytorch-UNet: PyTorch implementation of the U-Net for image semantic segmentation with high quality images]

[say4n/pytorch-segnet: SegNet implementation in Pytorch framework]

选择图像分割任务中表现优异的，且有完整训练代码的模型，将数据处理成其要求格式，重新训练或微调模型

实验要求

1. 官网提交测试集结果进行评测，实验结果可复现（切忌手动修改预测结果） - 10分

   超过0.70得2分　超过0.75得4分　超过0.80得6分

   超过0.85得8分　超过0.90得10分

2. 报告中介绍**实现方案**，包括运行环境说明，模型设计，损失函数设计等 - 6分

3. 报告中介绍涨点所做的数据增强、调参等**工作**；介绍实现方案的**创新性**，或

   在开源方案基础上做的修改；介绍实验中遇到的困难及对应的**解决方案** - 6分

提交完整实现代码（不含数据）+ 方案报告（附结果截图）

至elearning，命名为“学号_姓名_PJ1.zip”　　总分：20分

最后得分为：min(20, s1+s2+s3)　　截止日期: 2025年10月26日23:59 !!

THANKS

人工智能前沿探索实践-2025年秋

陈智能