实验六：语义分割

X

日期：July 15, 2022

摘　　要

本实验为模式识别与深度学习课程的实验六：语义分割。在本实验中，我们基于视网膜血管分布的 DRIVE 数据集，实现了一个简单的 UNet 语义分割模型，通过测试，我们的模型最终取得了良好的效果。

关键词：语义分割，UNet，DRIVE 数据集

1　选题说明及任务描述

1.1　选题说明

本次实验实现了一个语义分割程序，其可以根据 DRIVE 数据集中的视网膜图像，识别出其中的血管分布，有助于更好地实现疾病诊断。

1.2　任务描述

本实验所完成的计算机视觉任务为语义分割。语义分割是计算机视觉中的基本任务，在语义分割中，需要将视觉图像输入分为不同的语义可解释类别。如图1所示，是车辆行驶过程中语义分割任务的实现。

图 1: 语义分割任务示例

2 数据集描述

DRIVE 数据集是一个从视网膜图像中提取血管分布的医学数据集，适用于语义分割任务。该数据集由总共 40 张 JPEG 格式的彩色眼底图像组成；其中带有病理异常症状的有 7 例。这些图像来自荷兰的糖尿病视网膜病变筛查项目。数据集中每张图像的分辨率为 584*565 像素，有 RGB 三个通道。

图2为 DRIVE 数据集中的原始图像以及人工提取的血管分布：

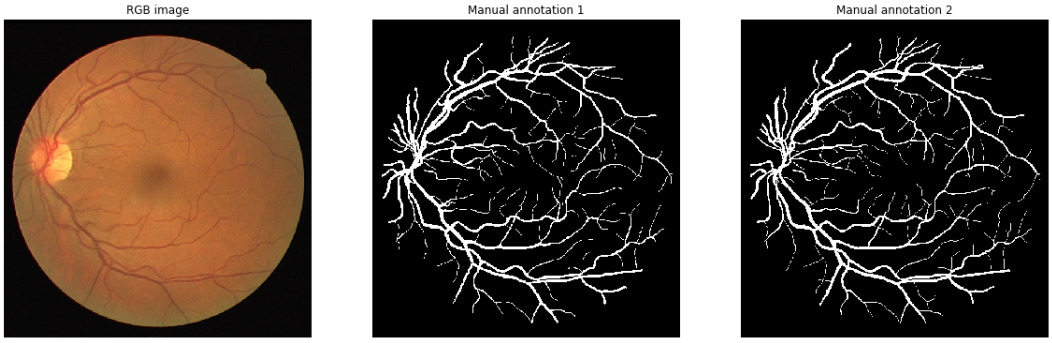

图 2: DRIVE 数据集中的 train image 及 annotation

3 方案设计

3.1 UNet 模型简介

本次实验中我们所实现的网络模型为 UNet。UNet 是一种经典的语义分割模型。整个 UNet 网络的结构类似于一个大型的字母 U，结构较为简单。在 UNet 中，encoder 下采样 4 次，一共下采样 16 倍，对称地，其 decoder 也相应上采样 4 次，将 encoder 得到的高级语义特征图恢复到原图片的分辨率。

就具体例子而言，UNet 先对图片进行卷积和池化（在 UNet 原始论文中进行了 4 次池化，如：原始图片尺寸是 224x224，则会生成 112x112，56x56,28x28,14x14 四个不同尺寸的特征）在此之后，对 14x14 的特征图做上采样或者反卷积，得到 28x28 的特征图，这个 28x28 的特征图与之前的 28x28 的特征图进行拼接，然后再对拼接之后的特征图做卷积和上采样，得到 56x56 的特征图，再与之前的 56x56 的特征拼接，卷积，再上采样，经过四次上采样可以得到一个与输入图像尺寸相同的 224x224 的预测结果。

UNet 相比更早提出的 FCN 网络更为轻量简单。它使用拼接来作为特征图的融合方式。这样可以形成更“厚”的特征，但也会更消耗显存。

图3是 UNet 的结构示意图。

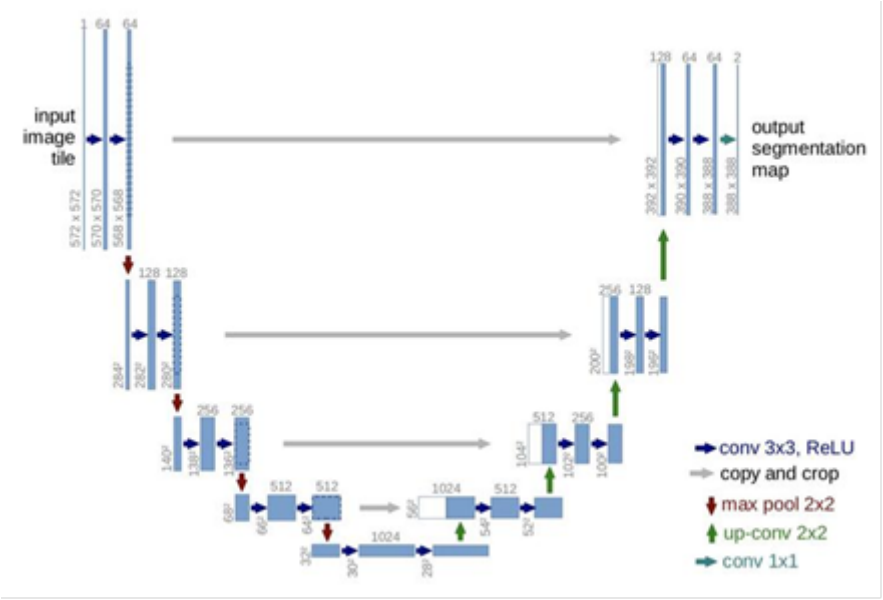

图 3: UNet 网络结构

图4为我们所实现的 UNet 部分结构。

3.2 损失函数与优化器等

在本实验中，我们使用 SGD 优化器，并使用了自定义的 loss 函数（如图5所示）

另外，我们通过 create_lr_scheduler 函数设定学习率每个 step 更新一次，而不是每个 epoch。

4 实验过程

4.1 数据预处理

本次实验使用的是 DRIVE 数据集，为了更方便的使用该数据集对模型进行训练和测试，在实验开始前，要对 DRIVE 数据集进行预处理，具体进行的预处理步骤如下：

- 使用给出的方差与均值进行归一化。
- 对图像使用 RandomCrop 进行随机剪裁。

完成之后，我们使用自定义的 drive_dataset 读取 DIRVE 数据集，并划分为训练集、验证集和测试集。

4.2 模型训练

由于本次实验中的 UNet 模型涉及大量的图片计算，因此本次实验在 Kaggle online notebook 上，使用 Kaggle 的免费算力完成。在训练过程中，将模型迁移至 GPU 上以通过 cuda 加速，提高训练效率。由于本次实验的模型涉及 R、G、B 多个通道数据，因此在计算模型的训练指标的时候，要对数据集的所有通道的数值进行综合统计，作为训练结果。

```python
class UNet(nn.Module):
    def __init__(self, in_channels=1, out_channels=2, channels=[64, 128, 256, 512]):
        super(UNet, self).__init__()
        self.in_channels = in_channels
        self.out_channels = out_channels
        self.channels = channels
        self.down_conv1 = Conv(in_channels, channels[0])
        self.down_conv2 = Conv(channels[0], channels[1])
        self.down_conv3 = Conv(channels[1], channels[2])
        self.down_conv4 = Conv(channels[2], channels[3])
        self.down_sample = nn.MaxPool2d(2, 2)

        self.bottleneck = Conv(channels[3], channels[3])

        self.up_sample = nn.Upsample(scale_factor=2, mode='bilinear', align_corners=True)

        self.up_conv1 = Conv(2 * channels[3], channels[2])
        self.up_conv2 = Conv(2 * channels[2], channels[1])
        self.up_conv3 = Conv(2 * channels[1], channels[0])
        self.up_conv4 = Conv(2 * channels[0], channels[0])

        self.out_conv = nn.Conv2d(channels[0], out_channels, 3, 1, 1)
```

图 4: 我们实现的 UNet 网络结构（部分）

```python
def criterion(inputs, target, loss_weight=None, num_classes: int = 2, dice: bool = True, ignore_index: int = -100):
    losses = {}
    for name, x in inputs.items():
        # 忽略target中值为255的像素，255的像素是目标边缘或者padding填充
        loss = nn.functional.cross_entropy(x, target, ignore_index=ignore_index, weight=loss_weight)
        if dice is True:
            dice_target = build_target(target, num_classes, ignore_index)
            loss += dice_loss(x, dice_target, multiclass=True, ignore_index=ignore_index)
        losses[name] = loss

    if len(losses) == 1:
        return losses['out']

    return losses['out'] + 0.5 * losses['aux']
```

图 5: 自定义 loss 函数

4.3 模型测试

为了更好的观察每个 epoch 后模型的泛化能力，在每个 epoch 训练结束后，都对模型进行了测试，利用相关测试指标（了 train loss，dice coefficient，global correct，IoU 等）对模型性能进行衡量。DRIVE 数据集的测试数据集中有两组 label，包括 1st_mannal 和 2n_mannal，两组 label 的区别在于 1st_mannal 的标记精度比 2nd_mannal 更为精确，本次实验在测试的时候选用的是 1st_mannal，以保证更为精确的测试结果。

5 实验结果

本次实验总共训练了 200 个 epoch。在完成训练后，模型对于测试集图片给出的语义分割结果如下：

可以看出，模型可以很好地提取出视网膜血管的分布，取得了很好的效果。

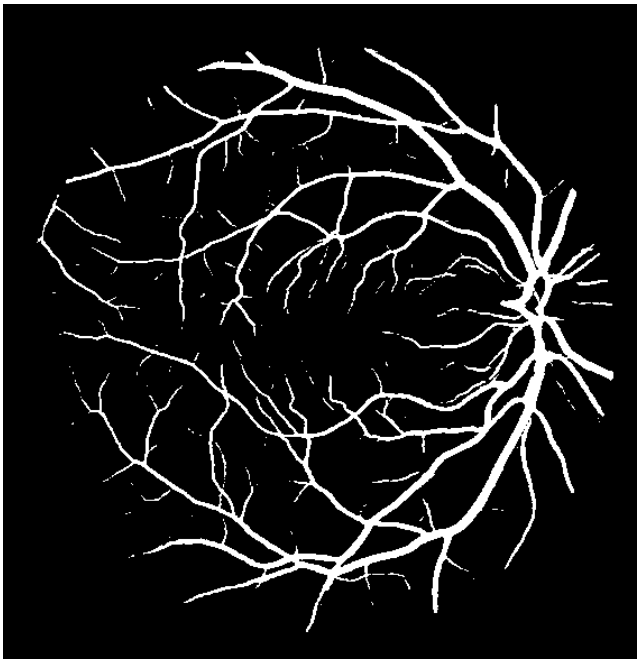

图 6: 语义分割结果

具体的结果以 log 记录的形式保存在./results.txt 中。图7给出了 Mean IoU，train loss，global correct 三个指标的可视化结果：

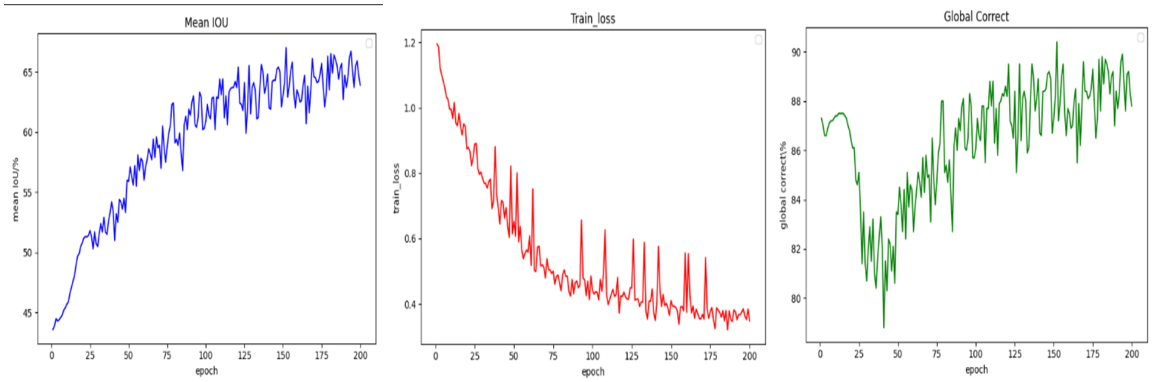

图 7: 可视化训练结果

6 方案评价

本次实验方案从以上实验结果中可以看出，train loss 随着 epoch 的增加呈持续下降的趋势，说明模型在训练集上的 loss 持续下降，对训练集的拟合程度持续增强；Mean IOU 随着 epoch 的增加，整体呈上升趋势，最高点大约出现在第 152 个 epoch，在这之后，Mean IOU 出现震荡，并最终趋于稳定；Global Correct 与 Mean IOU 呈现出几乎相同的结果，同样是在第 152 个 epoch 达到峰值，并在之后趋于稳定。Global Correct 与 Mean IOU 的这种高度相似性也从侧面验证了测试指标对于模型性能的可信度。

7 成员分工

- 姚舜宇：完成数据集切分与处理，模型搭建，以及测试部分。
- 江骏朋：完成数据集切分与处理，模型搭建，以及测试部分。
- 李旻翀：完成模型评估调优与报告撰写。
- 田轩：完成模型评估调优与报告撰写。