

results

以下划分数数据集的方式是前80%training set, 后20%是testing set。

```
=====Task: qd=====
```

```
F1 scores for each class:
```

```
Class 0: 0.8400
```

```
Class 1: 0.8992
```

```
Class 2: 0.7632
```

```
Confusion Matrix:
```

```
[[ 21   6   0]
```

```
 [  2 116   3]
```

```
 [  0  15  29]]
```

```
Epoch 30/30:
```

```
Train Loss: 0.0013, Train Acc: 1.0000
```

```
Val Loss: 0.5375, Val Acc: 0.8646, Precision: 0.8697, Recall: 0.8646, F1:  
0.8597
```

```
=====Task: sl=====
```

```
F1 scores for each class:
```

```
Class 0: 0.9810
```

```
Class 1: 0.5333
```

```
Confusion Matrix:
```

```
[[181   2]
```

```
 [  5   4]]
```

```
Epoch 30/30:
```

```
Train Loss: 0.0001, Train Acc: 1.0000
```

```
Val Loss: 0.1346, Val Acc: 0.9635, Precision: 0.9588, Recall: 0.9635, F1:  
0.9600
```

过采样之后:

```
-----  
F1 scores for each class:
```

```
Class 0: 0.9946
```

```
Class 1: 0.8750
```

```
Confusion Matrix:
```

```
[[183   0]
```

[2 7]]

Train Loss: 0.0045, Train Acc: 0.9986

Val Loss: 0.0239, Val Acc: 0.9896, Precision: 0.9897, Recall: 0.9896, F1: 0.9890

===== zjppt =====

全部过采样之后:

F1 scores for each class:

Class 0: 0.6300

Class 1: 0.8931

Class 2: 0.6619

Class 3: 0.9545

Confusion Matrix:

[[126 33 37 5]

[0 213 0 1]

[73 17 138 14]

[0 0 0 210]]

Train Loss: 0.5461, Train Acc: 0.7772

Val Loss: 0.5039, Val Acc: 0.7924, Precision: 0.7880, Recall: 0.7924, F1: 0.7824

只有training set过采样之后

F1 scores for each class:

Class 0: 0.3738

Class 1: 0.4407

Class 2: 0.8124

Class 3: 0.0952

Confusion Matrix:

[[20 12 43 2]

[2 13 1 0]

[7 18 184 13]

[1 0 3 1]]

Train Loss: 0.5286, Train Acc: 0.7967

Val Loss: 1.1354, Val Acc: 0.6813, Precision: 0.7291, Recall: 0.6813, F1: 0.6771

===== zyzg =====

全部过采样之后:

F1 scores for each class:

Class 0: 0.4463

Class 1: 0.2902

Class 2: 0.5382

Class 3: 0.7141

Confusion Matrix:

```
[[164  79 114  56]
```

```
 [119  84 103  87]
```

```
 [ 39  23 250 125]
```

```
 [  0   0  25 366]]
```

Train Loss: 0.9128, Train Acc: 0.5675

Val Loss: 0.9518, Val Acc: 0.5288, Precision: 0.5114, Recall: 0.5288, F1: 0.4974

只有training set过采样之后:

F1 scores for each class:

Class 0: 0.4818

Class 1: 0.2157

Class 2: 0.2599

Class 3: 0.1250

Confusion Matrix:

```
[[159 102  98  41]
```

```
 [ 80  37  44  33]
```

```
 [ 19   9  36  29]
```

```
 [  2   1   6   8]]
```

Epoch 26/30:

Train Loss: 0.8593, Train Acc: 0.5971

Val Loss: 1.6648, Val Acc: 0.3409, Precision: 0.4435, Recall: 0.3409, F1: 0.3706

代码能够处理任务类型 ('qd'、'sl'、'zjppt'、'zyzg'和'positioning' (但是最近太忙了positioning没时间调了不好意思QAQ所以zjppt和zyzg没有用positioning的信息，我感觉后两个任务应该还是大模型的结果比较好)):

数据处理与预处理

1. 数据加载与标签处理：

- 通过 load_labels 函数解析JSON格式的标签数据 (如果有traverse和sagittal的，就提取

的traverse的)

- 建立了图像路径与索引的双向映射关系
- 根据不同任务类型 ('qd'、'sl'、'zjppt'、'zyzg') 将字母标签 (A、B、C、D) 转换为数值标签
- 对于定位任务 ('positioning'), 从答案中提取边界框坐标

2. 图像预处理：

- 将图像调整为统一大小 (224×224)
- 应用标准化处理
- 转换为PyTorch张量格式

3. 数据不平衡处理：

- 使用 RandomOverSampler 对少数类别进行过采样，解决类别不平衡问题
- 分析并展示了过采样前后的类别分布情况

4. 数据集划分：

- 采用80%/20%的比例将数据集分为训练集和验证集
- 使用 train_test_split 函数确保随机性和可重复性 (随机种子设为42)

模型架构

1. 分类任务 ('qd'、'sl'、'zjppt'、'zyzg')：

- 基于预训练的ResNet50模型
- 冻结部分底层特征提取层以加速训练
- 修改了最终全连接层，添加了dropout (0.3) 以防止过拟合
- 最终分类层根据任务类别数量动态调整 ('qd': 3类, 'sl': 2类, 'zjppt'/'zyzg': 4类)

2. 分割任务 ('positioning')：

- 使用基于ResNet34骨干网络的U-Net架构
- 利用预训练权重 ('imagenet') 进行迁移学习
- 输出单通道分割掩码，使用sigmoid激活函数

训练策略

1. 损失函数：

- 分类任务：交叉熵损失 (CrossEntropyLoss)
- 分割任务：BCE Jaccard损失 (bce_jaccard_loss)，结合二元交叉熵和Jaccard指数

2. 优化器与学习率调度：

- 使用Adam优化器，初始学习率为0.001
- 实现了学习率自适应调整策略 (ReduceLROnPlateau)
- 当验证损失停止改善时，学习率降低50%

3. 训练过程监控：

- 记录并展示训练损失和验证损失
- 跟踪准确率、精确率、召回率和F1分数等多种评估指标
- 保存性能最佳的模型权重

4. 模型检查点：

- 实现了模型检查点保存和加载机制
- 支持训练中断后继续训练
- 记录训练元数据（如已训练轮次）