机器学习课程设计报告：糖尿病视网膜病变分级

徐浩淞20226446
1593120349@qq.com

摘要

针对分级任务，主要目的是分析2D眼底图像的临床数据，根据视觉特征（病灶信息）区分糖尿病视网膜病变的严重程度。主要包括糖尿病视网膜病变以及糖尿病黄斑水肿。

关键词： Diabetic retinopathy ； Diabetic macular edema ； Clinical data analysis

XuHaosong
1593120349@qq.com

Abstract

The main objective of the hierarchical task is to analyze clinical data from 2D fundus images and classify the severity of diabetic retinopathy based on visual features (lesion information). This includes diabetic retinopathy as well as diabetic macular edema.

Keywords: Diabetic retinopathy , Diabetic macular edema , Clinical data analysis

1 实验内容与具体要求

针对分级任务，主要目的是分析2D眼底图像的临床数据，根据视觉特征（病灶信息）区分糖尿病视网膜病变的严重程度。主要包括糖尿病视网膜病变以及糖尿病黄斑水肿。采用F1-score Macro评价指标来评价提交结果。

2 实验过程

2.1 训练集图像和标签预处理

设计数据集类，用pil库打开图像，转换成rgb格式，对训练集图像进行预处理，转换成后续resnet50模型需要的格式。

```python
class RetinopathyDataset(Dataset):#数据预处理
    def __init__(self, csv_file, img_dir, transform=None):
        self.data = pd.read_csv(csv_file)
        self.img_dir = img_dir
        self.transform = transform

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        img_name = os.path.join(self.img_dir, f"{self.data.iloc[idx, 0]}.jpg")#图像名称
        image = Image.open(img_name).convert('RGB')
        label = self.data.iloc[idx, 1]#病变等级

        if self.transform:
            image = self.transform(image)

        return image, label
```

Figure 1: 训练集图像和标签预处理

2.2 resnet50模型

使用预训练的resnet50模型，对全连接层进行修改，变成2分类任务，具体分类是正常和病变。

```python
class RetinopathyModel(nn.Module):#模型
    def __init__(self):
        super(RetinopathyModel, self).__init__()
        self.resnet = resnet50(pretrained=True)
        num_ftrs = self.resnet.fc.in_features #获取全连接层的输入特征数
        self.resnet.fc = nn.Sequential(
            nn.Linear(num_ftrs, 256),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear(256, 2)
        )

    def forward(self, x):
        return self.resnet(x)
```

Figure 2: Enter Caption

2.3 训练模型

先预测结果，然后利用交叉熵损失函数计算损失，反向传播计算梯度，然后更新参数。

```python
def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs):
    device = torch.device("cuda" )
    model = model.to(device)

    scheduler = StepLR(optimizer, step_size=10, gamma=0.1)#学习率调度器
    best_val_loss = float('inf')#最佳验证损失初始无穷大

    for epoch in range(num_epochs):#训练模型
        model.train()
        running_loss = 0.0

        for inputs, labels in train_loader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()#更新参数

            running_loss += loss.item()
        scheduler.step()#更新学习率
```

Figure 3: 训练模型

2.4 验证并保存模型

在每一轮训练中，同时添加验证，计算平均验证损失，如果验证损失更低，才保存模型。

```python
    with torch.no_grad():
        for inputs, labels in val_loader:
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            val_loss += loss.item()

            _, predicted = torch.max(outputs.data, 1)#每行的预测结果
            total += labels.size(0)
            correct += (predicted == labels).sum().item()#正确预测数量

    epoch_val_loss = val_loss / len(val_loader)
    accuracy = 100 * correct / total

    print(f'轮数: [{epoch+1}/{num_epochs}]')
    print(f'准确率: {accuracy:.2f}%')

    if epoch_val_loss < best_val_loss:#如果当前验证损失更好
        torch.save(model.state_dict(), os.path.join(BASE_DIR, 'model','final_model.pth'))
```

Figure 4: 验证并保存模型

2.5 训练模型

对图像进行处理，转化为张量并正则化，对数据集划分，其中的百分之八十用作训练集，百分之二十用作验证集，使用交叉熵损失函数计算损失，使用adamw梯度优化器，训练三十轮。

```python
def train():
    transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomHorizontalFlip(),#水平
        transforms.RandomVerticalFlip(),#垂直
        transforms.RandomRotation(20),#旋转
        transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406],
                             std=[0.229, 0.224, 0.225])
    ])

    dataset = RetinopathyDataset(
        csv_file=os.path.join(BASE_DIR, 'Mess1_annotation_train.csv'),
        img_dir=os.path.join(BASE_DIR, 'train'),
        transform=transform
    )

    train_size = int(0.8 * len(dataset))
    val_size = len(dataset) - train_size
    train_dataset, val_dataset = torch.utils.data.random_split(dataset, [train_size, val_size])#随机划分数据集

    train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)  # 训练数据加载器
    val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)  # 验证数据加载器

    model = RetinopathyModel()
    criterion = nn.CrossEntropyLoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=0.0001)

    train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs=30)
```

Figure 5: 训练模型

2.6 测试模型并输出结果

对图片进行处理，然后按id排列，依次预测结果，用panda库创建dataframe对象存储结果，然后转换成规定的csv格式。

```python
def predict(model, image_path, transform):
    device = torch.device("cuda")
    model = model.to(device)
    model.eval()

    image = Image.open(image_path).convert('RGB')
    image = transform(image).unsqueeze(0).to(device)#增加批次维度

    with torch.no_grad():
        outputs = model(image)
        _, predicted = torch.max(outputs.data, 1)

    return predicted.item()

def main():
    model = RetinopathyModel()
    model.load_state_dict(torch.load(os.path.join(BASE_DIR, 'model', 'final_model.pth')))

    transform = transforms.Compose([#图像处理
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(mean=[0.485, 0.456, 0.406],
                             std=[0.229, 0.224, 0.225])
    ])


    test_dir = os.path.join(BASE_DIR, 'test')
    predictions = []
    image_ids = []

    for image_name in sorted(os.listdir(test_dir), key=lambda x: int(x.split('.')[0])):#按图像ID排序
        if image_name.endswith('.jpg'):
            image_path = os.path.join(test_dir, image_name)
            image_id = image_name.split('.')[0]#获取id

            prediction = predict(model, image_path, transform)#进行预测

            image_ids.append(image_id)
            predictions.append(prediction)

    submission = pd.DataFrame({
        'Image': image_ids,
        'Predict': predictions
    })

    submission.to_csv(os.path.join(BASE_DIR, 'submission.csv'), index=False)
    print("预测完成")
```

Figure 6: 测试模型并输出结果

3 实验结果

3.1 输出结果

内容过多，仅展示部分结果。

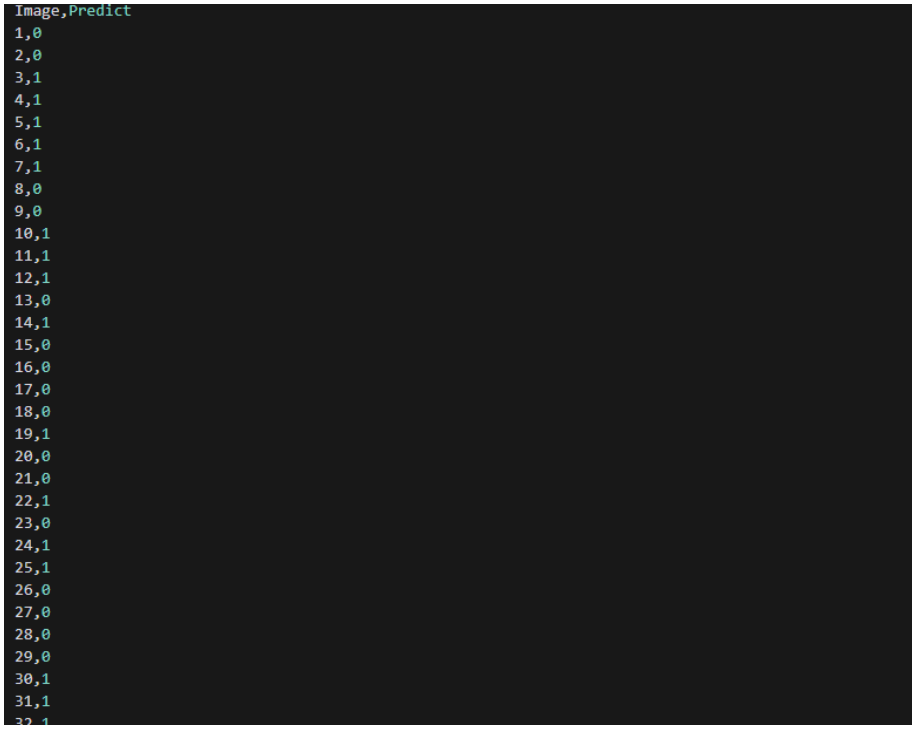

Figure 7: 输出结果

3.2 得分

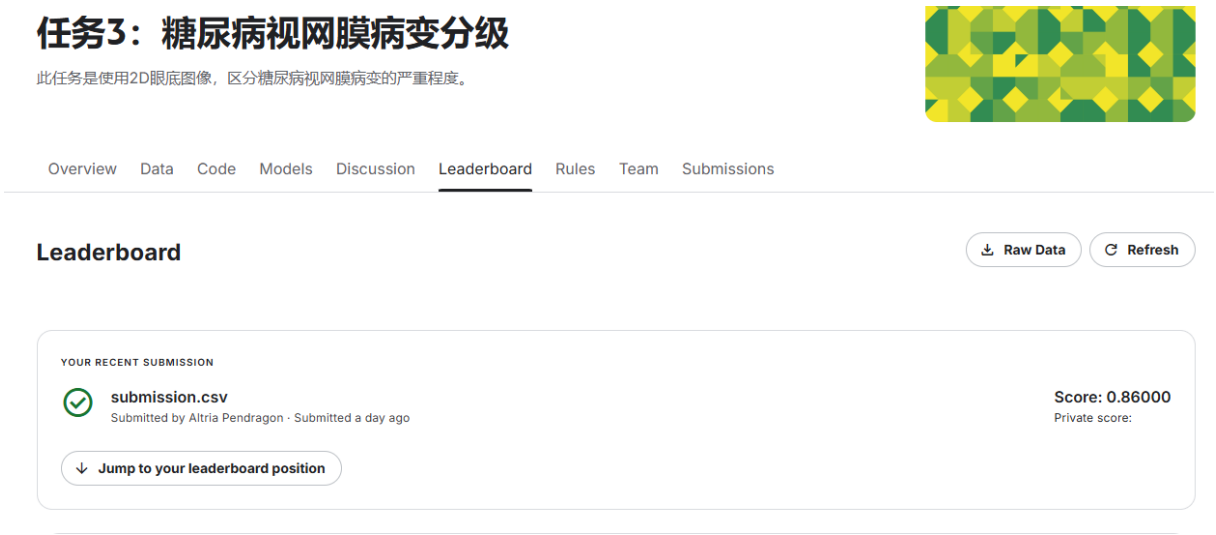

Figure 8: 得分

4 实验心得

通过第三个课程设计任务，我学习了使用预训练的resnet50模型来进行糖尿病视网膜病变的预测，在这个任务的具体应用里，需要把全连接层修改，变成2分类任务，对应此任务中的正常“0”和病变“1”。并且学习了使用学习率调度器，将训练集随机划分为训练集和验证集，当且仅当平均验证损失更低时才保留模型结果，还学习了随机处理图像，包括随机水平、垂直反转、旋转、颜色调整等增加模型的通用性，并且对使用panda库的dataframe对象创建结果文件，再转换为csv文件这个过程更加熟悉了。