目录

[一、 模型思路与建模过程 3](#_Toc185424156)

[**采用CNN+LSTM模型的原因和思路：** 3](#_Toc185424157)

[**展示代码并说明：** 3](#_Toc185424158)

[二、 实验结果与分析 11](#_Toc185424159)

[三、 实验总结 13](#_Toc185424160)

报告正文

1. 模型思路与建模过程

**采用CNN+LSTM模型的原因和思路：**

首先我的主题是ECG的心律失常分类，每条数据有188位数值，前187位都是心电波形散点幅值，心电波形复杂，异常心电波形更是出现位置很难预判，而且数据维度大，而CNN的优势在于可以局部特征提取，CNN善于捕捉ECG信号中的局部形态特征；并且CNN具有平移不变性，无论心跳特征出现在信号的哪个位置，CNN都能够识别；而且CNN的降维能力强，对于心电图这样庞大的数据，可以通过卷积和池化层逐步减少数据维度，提取关键特征。

对于LSTM，LSTM是具有记忆能力和上下文理解能力的，LSTM能够捕捉ECG信号中的长期时序依赖关系；而LSTM的记忆能力可以记住之前的心跳模式，有助于识别异常心律；LSTM的上下文理解能够理解心跳序列的上下文信息。

而我将CNN和LSTM结合起来，就是想利用CNN和LSTM的互补性，CNN处理局部特征，LSTM处理时序关系，CNN首先提取低层次的形态特征，LSTM在此基础上学习高层次的时序模式，两者结合提供了更全面的特征理解。

**展示代码并说明：**

首先导入库，其中的torch.optim是优化器模块，包含各种优化算法，pandas用来数据处理和CSV文件读取，sklearn.metrics用来用于模型评估，其中的confusion\_matrix是生成混淆矩阵，classification\_report生成包含精确率、召回率等详细分类指标的报告。Seaborn和matplotlib是用来绘制图表：

文本

描述已自动生成

接下来是配置中文字体，因为如果不配置好的话，很可能出现图表中的中文字体乱码，再就是设置随机种子和设备，设置随机种子是为了使实验结果可以复现，因为为了提高模型准确率，进行过多次调参数，随机种子可以帮我复现结果最好的模型。这里设备使用的是GPU：

文本

描述已自动生成

自定义数据集类用于批量加载数据，由于数据量很大，使用默认的数据加载是一次性加载，会导致内存不够用。这里将x表示为前187位，y表示为最后一位的标签，len用来返回数据集大小，getitem通过idx索引获取数据集中的某一个样本：

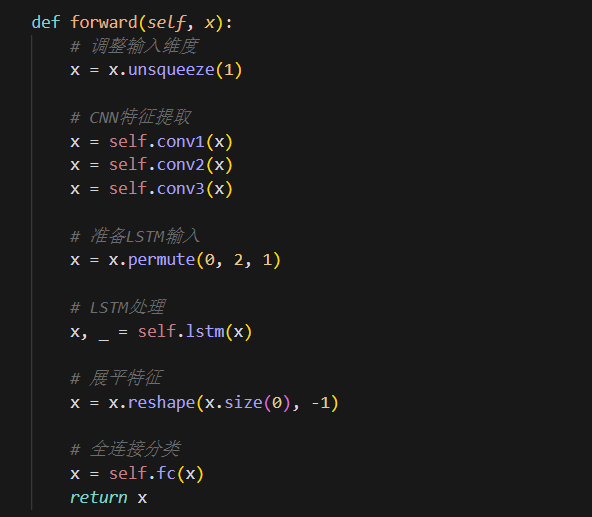
文本

描述已自动生成

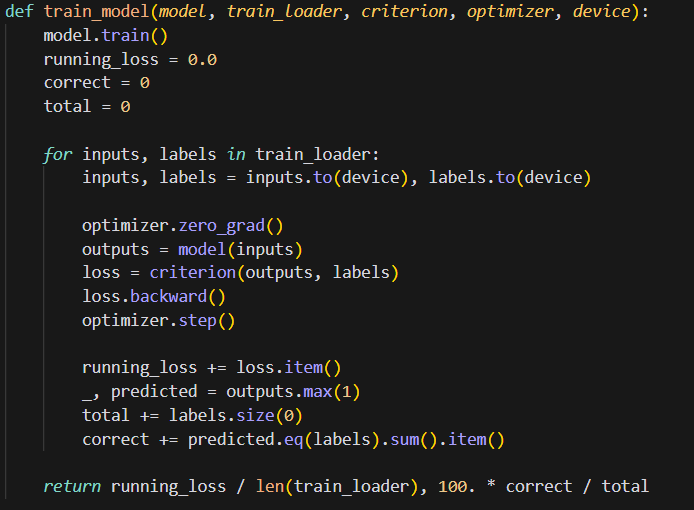
定义CNN+LSTM模型，结构是三层卷积层，两层LSTM，两层全连接层。第一层卷积层输入通道数为1，输出通道数为64，卷积核大小为3，填充为1，激活函数为ReLU，池化为2，后面两层只有通道数的区别，LSTM输入特征维度为256，隐藏层数为256，第一全连接层输入为256\*23，输出为256，第二全连接层为输入256，输出5。Forward的顺序就是先经过三层CNN进行特征提取，然后进入LSTM处理序列数据，捕获时序特征，最后由全连接层进行分类：

文本

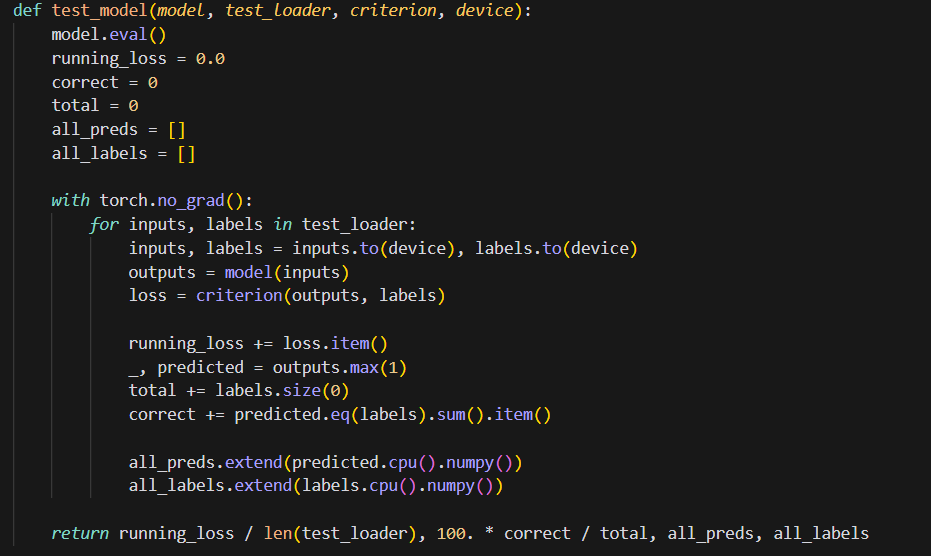
描述已自动生成



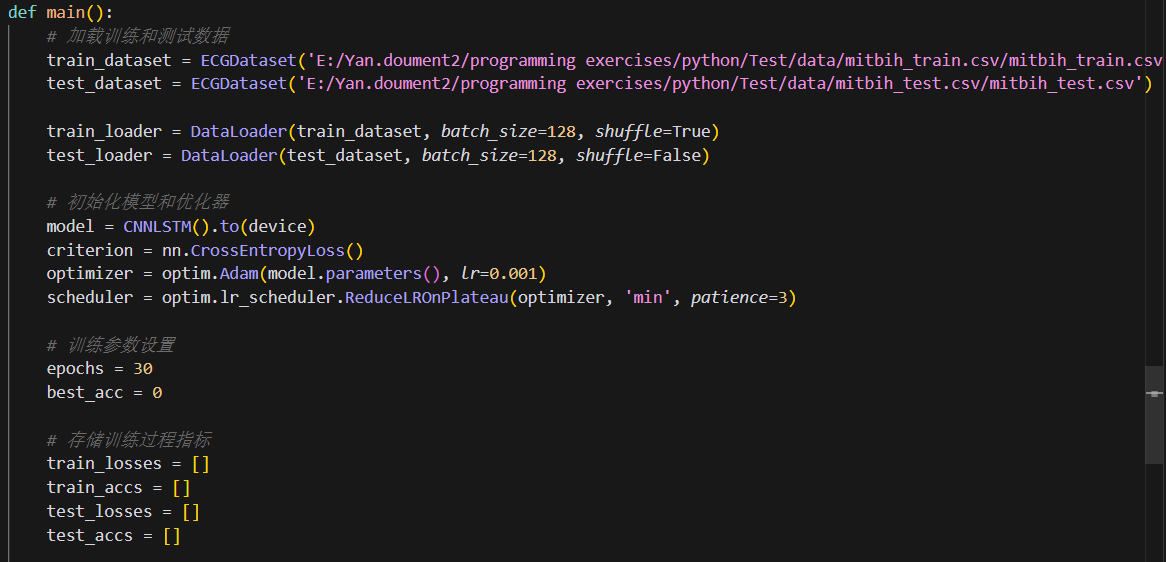
再定义模型训练函数，这个训练函数在每个训练周期（epoch）中遍历数据集，通过前向传播计算损失，反向传播更新模型参数，并返回平均损失值和准确率：



然后是模型测试函数，这个测试函数在评估模式下，不计算梯度，对测试数据集进行前向传播，计算损失值和准确率，同时收集所有预测结果和真实标签，用于后续的模型性能分析。：



接下来就是主函数的编写，首先加载训练和测试函数，并且要分批次处理，因为数据量庞大，一次性加载会导致内存不够。数据准备完毕后，对模型进行初始化，移动至GPU，设置损失函数交叉熵损失函数，适用于分类问题，配置优化器，Adam优化器，设置学习率调度器，等待3个epoch损失都没改善才调整学习率。这里设置30轮，因为我发现在25轮附近准确率就趋于稳定了，并且设置存储训练过程指标，方便最后绘制图表：



开始训练测试模型，每次循环开始，先通过train\_model返回训练损失和准确率，通过test\_model在测试集上评估，返回测试损失、准确率、预测值和真实标签，然后将每个epoch的指标添加到列表中，用于后续绘图，然后根据测试损失调整学习率，输出本轮相关信息，保存最佳模型结果：

文本

描述已自动生成

最后就是绘制图表过程，绘制了训练损失和测试损失随轮次的变化，还有训练准确率和测试损失率随轮次的变化，还有混淆矩阵，以及分类报告展示混淆矩阵参数结果：

文本

描述已自动生成

文本

描述已自动生成

1. 实验结果与分析

下面展示训练和测试结果：

首先是损失和准确率随轮次的变化：

左图（损失随轮次的变化）

初始阶段（0-5轮）：

训练损失（蓝线）从0.30快速下降到约0.05

测试损失（红线）从0.17快速下降到约0.07

特点：模型学习速度最快，损失值显著降低

中期阶段（5-15轮）：

训练损失继续缓慢下降

测试损失趋于平稳，在0.06-0.07之间波动

特点：学习速度放缓，模型开始趋于稳定

后期阶段（15-30轮）：

训练损失继续下降至接近0

测试损失略有上升并稳定在约0.07

特点：出现轻微过拟合迹象

右图（准确率随轮次的变化）：

初始阶段（0-5轮）：

训练准确率（蓝线）和测试准确率（红线）都从92%快速上升到98%

特点：模型快速学习，性能显著提升

中期阶段（5-15轮）：

训练准确率继续上升到99%以上

测试准确率缓慢提升到98.5%左右

特点：性能提升放缓，但仍在改善

后期阶段（15-30轮）：

训练准确率达到接近100%

测试准确率稳定在98.8%左右

特点：模型基本收敛：

图表, 折线图

描述已自动生成

最终结果如图，最佳测试准确率为98.85%

图片包含 图形用户界面

描述已自动生成

然后是混淆矩阵，可以看到类别0：18068个正确预测，类别1：462个正确预测，类别2：1397个正确预测，类别3：126个正确预测，类别4：1588个正确预测，类别1和类别3的识别准确率相对较低，其中类别3的识别准确率最低：

日历

描述已自动生成

这里可以看见，类别0准确率在0.99，类别1准确率在0.93，类别2准确率在0.97，类别3准确率最低为0.88，类别4准确率最高达到了1.00，同样也有召回率和F1分数的显示，也是类别3最低，类别1其次。简单平均准确率达到0.95，召回率在0.91，F1分数在0.93。加权平均值，因为类别0数量最多，准确率在0.99，召回率在0.99，F1分数在0.99：

日历

描述已自动生成

1. 实验总结

本次实验花费我时间最多的一个活动就是调超参数使模型准确率尽可能的提高，我最初始的模型是两层卷积，其他跟现在的模型一模一样，后来我发现三层的卷积可以提高准确率，因为三层的卷积可以更全面准确的提取特征，在每层的输入和输出通道数上也是不停的修改，最终发现相比较小的通道数，较大的通道数效果更好，但是训练时间也同样更长。

这个模型虽然最终准确率为98.85%，但是结果显示在类别1和3上还是不够表现好，这是这个模型的缺点，我也对模型的改进有一些以下建议：

针对数据不平衡的问题，由于数据集中类别0这一类是最多的，所有可以对少数类进行过采样或数据增强的方法。也可以使用类别权重。

模型优化可以增加正则化（L1/L2），尝试不同的网络架构，考虑使用注意力机制。