无线网络大实验报告

dz1933026 王国畅

主要成果：

- 在iOS下实现了数据集采集和手势识别应用
- 在Flask上实现在线服务，用于数据处理和模型调用
- 在自定义数据集上训练audio_cnn模型进行分类

文件说明：

提交文件中包括两个部分：

- zc_gesture: iOS应用的Xcode项目源代码
- zc_python: python部分源代码，包含ZC信号处理代码，Flask应用源码，模型文件和训练代码，数据集

其中zc_python下：app/ 为Flask应用源代码目录，wavGenerator为ZC信号调制实现，model.pth为模型文件，audiocnn.py为模型训练脚本，mydataset目录为数据集

运行说明：

- iOS应用：在Xcode下导入zc_gesture进行编译运行。
- Flask服务：在zc_python目录下依据requirements.txt安装依赖，flask run —host=0.0.0.0 启动服务。
- 模型训练：在zc_python目录下python audiocnn.py启动训练脚本。

实验参数：

硬件：iPhoneXr，MacBookPro

开发配置：Xcode11.5，swift 5, iOS13

深度学习框架：PyTorch

iOS界面：

应用包含两个Tab页面，collect为数据集收集页面，detect为手势识别页面。

数据收集页面逻辑：点击Dataset旁的按钮可切换训练集和测试集。滑动选择scroll up/scroll down/double click可选择数据标签。点击左下方播放按钮开始播放ZC调制信号，开始手势录入。点击右下方停止按钮结束录入。结束录入时数据被提交至Flask服务，并保存为wav文件，录入数据集。数据集组织为：

Root:

- Train
  - 0
    - sample.wav
  - 1
  - 2

- Val

手势检测页面逻辑：点击播放按钮播放ZC调制信号，开始手势录入，点击停止按钮结束录入并将数据上传至Flask服务，获取预测结果后切换图片表示预测结果。

信号播放逻辑：通过python方法生成ZC调制信号，并存储为wav文件，通过AVAudioEngine框架进行循环播放。

信号接收逻辑：通过AVAudioEngine框架将接收到的音频数据上传到Flask服务，通过python代码进行数据处理。

Flask应用：

Flask应用部分提供训练集和验证集的提交接口和手势检测接口，接口设计如下：

```
@app.route('/Train/<label>', methods=[‘POST’])
@app.route('/Val/<label>', methods=[‘POST'])
@app.route('/detect',methods=['POST'])
```

其中label指iOS前端为数据选择的标签。

模型训练：

本实验中将手势检测建模为简单的音频分类任务，并通过简单的audio_cnn模型进行训练和预测。

模型架构如下：

```
class AudioCNN(nn.Module):
    def __init__(self, num_classes):
        super(AudioCNN, self).__init__()

        self.main = nn.Sequential(
            nn.Conv1d(1, 64, 80, 4, 2),
            nn.BatchNorm1d(64),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv1d(64, 128, 80, 4, 2),
            nn.BatchNorm1d(128),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv1d(128, 256, 80, 4, 2),
            nn.BatchNorm1d(256),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv1d(256, 512, 80, 4, 2),
```

```
            nn.BatchNorm1d(512),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv1d(512, 512, 40, 4, 2),
            nn.BatchNorm1d(512),
            nn.LeakyReLU(0.2, inplace=True),
        )
        self.classifier = nn.Sequential(nn.Linear(14848,
num_classes), nn.Softmax())

    def forward(self, tensor):
        batch_size = tensor.size(0)
        hidden = self.main(tensor)
        # print(hidden.size())
        hidden = hidden.view(batch_size, -1)
        # print(hidden.size())
        return self.classifier(hidden)
```

实验结果：

训练集大小：74KB（每个样本）x 50（每个标签样本数）x 3 （标签数）

验证集大小：74KB（每个样本）x 20（每个标签样本数）x 3 （标签数）

模型大小：97.2MB

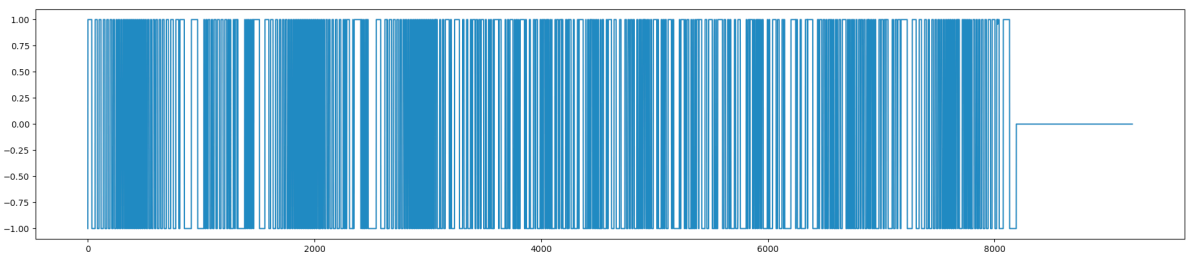
平均预测时间：0.56s

验证集准确率：34%

误差分析：

本次实验模型准确率很低，其误差来源可能来自两方面：

第一，声音样本质量低。由于本次实验使用的AVAudioEngine格式不支持48kHz的采样率，因此播音的音频和收音处理都使用了44.1kHz的采样率。由此得到的声音样本在经过@万浩然同学在课程群中提供的ZC_Process方法进行解调以后，经过差分和正则化得到的声音样本十分不理想，例如本实验中收集的数据集中mydataset/Train/0/1.wav图示如下：



波形频繁地在上下极限间振荡，此问题可能来自基于ZC_Process进行扩展的数据处理逻辑中，由于缺乏信号处理相关知识，这部分内容无法进行良好的debug。

第二，声音样本数量少。本次实验中只采集了少量的数据，模型训练过程的loss曲线没有很好的收敛，同时也有不少调优的空间。由于缺乏深度学习调优的相关知识，模型性能优化方面少有进展。

实验总结：

总体而言本次实验在iOS平台搭建了完整的超声波手势识别工作流。美中不足的是在数据处理和模型性能方面没有得到很好的结果。

实验中的大量时间花费在iOS的音频处理过程，由于iOS音频处理框架和科学计算框架选择较多且文档较少，试错过程占据了绝大部分iOS开发时间，最终仍然没有得到合适的科学计算框架足以进行ZC信号调制中的复数fft和ifft运算，因此选择通过python实现ZC信号调制和解调，并通过Flask框架提供在线服务。

本次实验中学习了iOS开发和Flask应用构建并进行了深度学习训练实践，技术而言收获颇丰。

特别鸣谢@万浩然同学在信号处理方面的悉心指导。同时感谢老师们一学期的知识分享和教学活动安排，让我对无线网络相关领域有了初步了解，受益匪浅。