课程设计：睡眠质量助手（软件）

项目报告

2030416018 高歌

2022.12.25

# 目 录

[1 目 录 2](#_Toc122903041)

[1 系统概述 3](#_Toc122903042)

[1.1 背景介绍 3](#_Toc122903043)

[1.2 系统简介 3](#_Toc122903044)

[2 需求分析 3](#_Toc122903045)

[3 功能设计 4](#_Toc122903046)

[3.1 系统概观 4](#_Toc122903047)

[3.2 关键代码分析 5](#_Toc122903048)

[3.2.1 模型训练及部署 5](#_Toc122903049)

[3.2.2 应用开发 8](#_Toc122903050)

[4 系统实现 12](#_Toc122903051)

[4.1 睡眠记录界面 12](#_Toc122903052)

[4.2 睡眠报告界面 12](#_Toc122903053)

[5 项目总结 13](#_Toc122903054)

[5.1 小组成员分工 13](#_Toc122903055)

[5.2 系统总结与展望 13](#_Toc122903056)

睡眠质量助手（软件）

——物联网技术与应用课程设计报告

高 歌

（苏州大学计算机科学与技术学院，江苏苏州）

# 系统概述

## 背景介绍

随着智能穿戴设备的普及，睡眠监测功能逐渐在以智能手环为代表的智能设备上得到应用。本项目将睡眠监测功能集成到手机APP中，通过使用手机麦克风记录夜间睡眠情况，并根据睡眠过程中的打鼾情况、环境噪音及梦话次数给出综合睡眠报告。

## 系统简介

本系统形式为基于安卓系统的手机APP。睡前启动APP并将手机放置在枕边。在睡眠期间，APP监测环境音并分析环境噪声、呼噜声的持续时长及声音大小（分贝），并记录梦话与睡眠打断时长。在起床后APP将生成睡眠质量报告显示相关信息，并给出综合睡眠质量评分。

# 需求分析

* 环境噪声监测：记录睡眠期间的环境噪声变化情况，生成曲线图，并给出平均噪声大小（分贝）
* 鼾声检测：记录每次鼾声的起止时间并在睡眠报告中保存数个片段，生成分析图，并给出平均呼噜声大小
* 梦话记录：记录每一次梦话的起止时间及内容，可在睡眠报告中播放
* 入睡计时：在启动app睡眠记录功能后开始计时并分析环境声，判断入睡时间以及入睡时长

# 功能设计

## 系统概观

APP的目标平台为安卓手机端。但由于系统基于跨端框架Flutter开发，因此理论上同时兼容IOS平台、Windows/linux/Mac OS桌面端以及Web端。目前仅在安卓9以上版本测试可用。

应用使用Material Design设计风格。

应用开发框架为Flutter，开发语言为Dart。声音识别使用Tensorflow Lite框架构建，使用的深度学习框架为语音识别框架YAMNet，训练所使用数据集大部分为自行采集，鼾声数据集来源：https://www.kaggle.com/datasets/tareqkhanemu/snoring

所使用的Flutter第三方包包括：

cupertino\_icons: ^1.0.2

record: ^4.4.3

just\_audio: ^0.9.30

flutter\_bloc: ^8.1.1

permission\_handler: ^10.2.0

equitable: ^2.0.5

intl: ^0.17.0

tflite\_audio: ^0.3.0

除上述提及的第三方资源外，应用开发未涉及任何已有的第三方代码库。

大致来说，系统概观如下图所示：

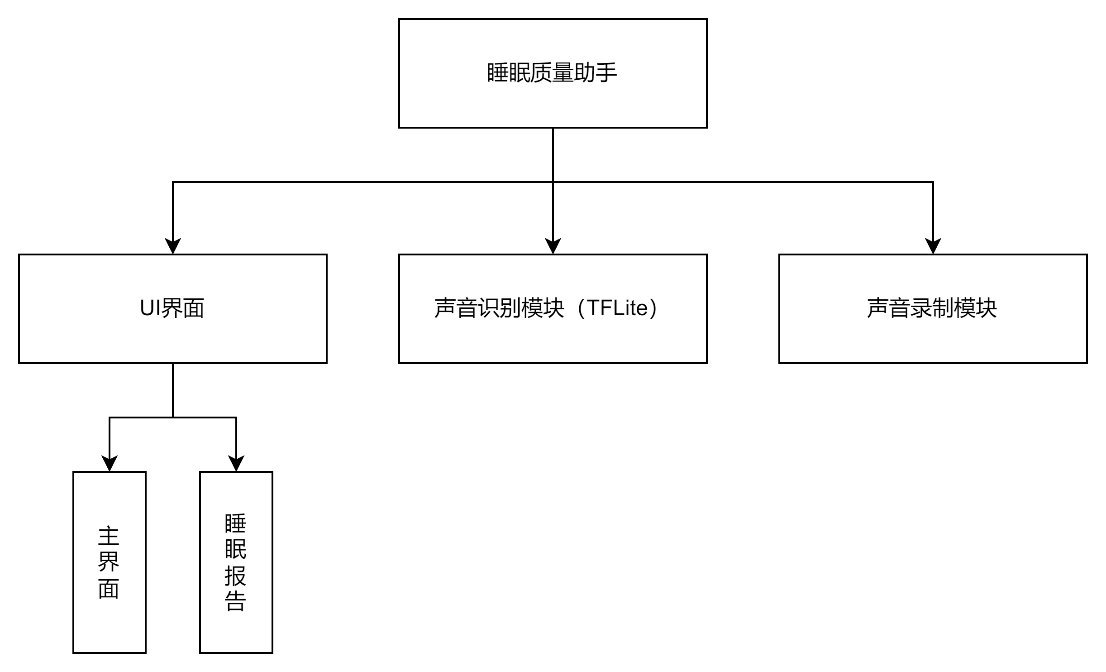


图 3‑1 系统概观

如图所示，系统主要分为三部分：前端UI界面、睡眠监测时对声音实时分析的声音识别模块（基于TFLite模型部署），以及在相应时间自动录制鼾声/梦话声的声音录制模块。

开始睡眠监测后，语音模型会实时读取麦克风获取的声音，并判断当前声音是环境噪声、鼾声或梦话。若判断当前声音是环境噪声，则记录当前噪声分贝大小，计入平均环境噪声；若判断当前声音是鼾声或梦话，则启动声音录制模块，录制当前声音并保存到手机存储中的相应位置，可在睡眠报告中播放查看。

## 关键代码分析

### 模型训练及部署

模型使用Tensorflow Lite官方的tflite-model-maker训练。

注：目前tflite-model-maker支持的最高Python版本仅为Python 3.8，且务必在Linux环境下训练。由于训练过程中会从谷歌源下载headers，因此请确保代码运行环境可以正常访问代理，否则训练将失败。对于Windows系统，建议在WSL2环境下安装Miniconda+Python 3.8版本，并安装必要的依赖，如tensorflow和matplotlib、jupyter等，然后在WSL2下启动终端并开启代理，然后通过命令行启动jupyter，或通过命令行输入code启动VSCode连接WSL2代理远程jupyter服务器。

代码及所使用数据集已打包到snoring\_classification.zip中，直接运行解压后运行snoring\_classificaton.ipynb即可。再次强调，请确保使用的Python版本低于或等于3.8，使用Linux系统（建议Ubuntu，否则请自行修改部分代码），且代码运行时可以正常访问代理。

Notebook运行结束后，模型将输出到./snoring\_models/ snoring\_model.tflite，同时还有附带的labels.txt文件，请将这两个文件一并拷贝到Flutter代码的assets目录下，并重命名snoring\_model.tflite为snoring\_classifier.tflite。所附代码已完成这一步，无需重复该步骤。

下面简单摘出训练过程中的关键几步做描述。

下为使用YAMNet构建模型的关键代码。模型将声音分为三类：背景噪声（background）、鼾声（snoring）以及梦话（speech）。

spec = audio\_classifier.YamNetSpec(

    keep\_yamnet\_and\_custom\_heads=False,

    frame\_step=audio\_classifier.YamNetSpec.EXPECTED\_WAVEFORM\_LENGTH,

    frame\_length=audio\_classifier.YamNetSpec.EXPECTED\_WAVEFORM\_LENGTH)

下面的代码进行数据集分割：

train\_data = audio\_classifier.DataLoader.from\_folder(

    spec, os.path.join(data\_dir, 'train'), cache=True)

train\_data, validation\_data = train\_data.split(0.8)

test\_data = audio\_classifier.DataLoader.from\_folder(

    spec, os.path.join(data\_dir, 'test'), cache=True)

下面为训练代码。这里将epochs设为25，batch\_size设为128

batch\_size = 128

epochs = 25

print('Training the model')

model = audio\_classifier.create(

    train\_data,

    spec,

    validation\_data,

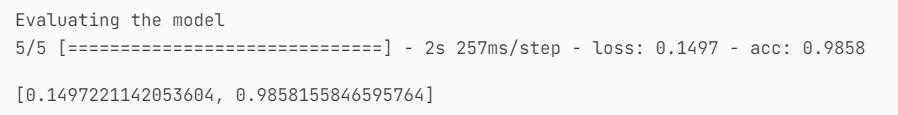
    batch\_size=batch\_size,

    epochs=epochs)

下为模型得分：

print('Evaluating the model')

model.evaluate(test\_data)



下为模型的Confusion Matrix

def show\_confusion\_matrix(confusion, test\_labels):

    """Compute confusion matrix and normalize."""

    confusion\_normalized = confusion.astype("float") / confusion.sum(axis=1)

    axis\_labels = test\_labels

    ax = sns.heatmap(

        confusion\_normalized, xticklabels=axis\_labels, yticklabels=axis\_labels,

        cmap='Blues', annot=True, fmt='.2f', square=True)

    plt.title("Confusion matrix")

    plt.ylabel("True label")

    plt.xlabel("Predicted label")

confusion\_matrix = model.confusion\_matrix(test\_data)

show\_confusion\_matrix(confusion\_matrix.numpy(), test\_data.index\_to\_label)

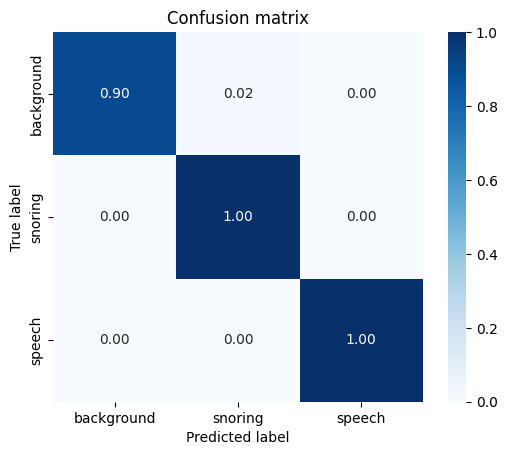
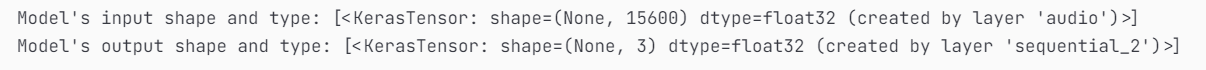


图 3‑2 Confusion Matrix

可以看到总的来说模型的准确性还是符合预期的。



输出模型应接受一个1×15600的声音向量，输出一个1×3的向量（对应三种声音类型）。

### 应用开发

首先是主界面，联动了两个子页面睡眠记录页（home）和睡眠报告页（report\_list）。下面摘取一段代码。

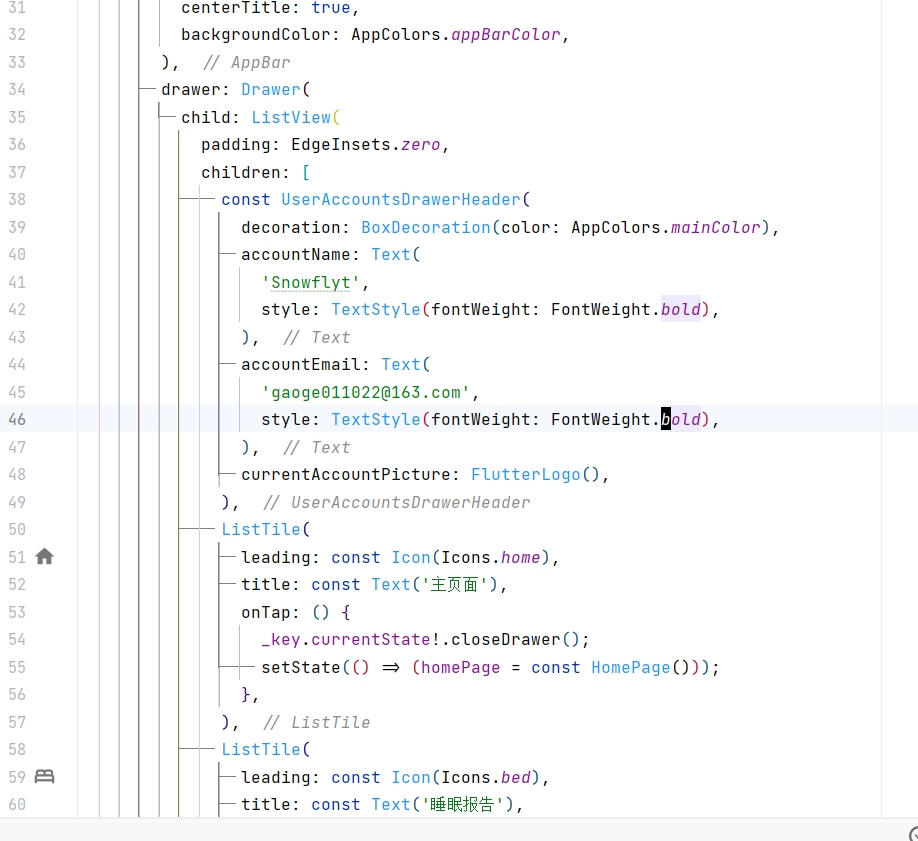


图 3‑3 主页

睡眠记录页在初始化时先加载TFLite模型。

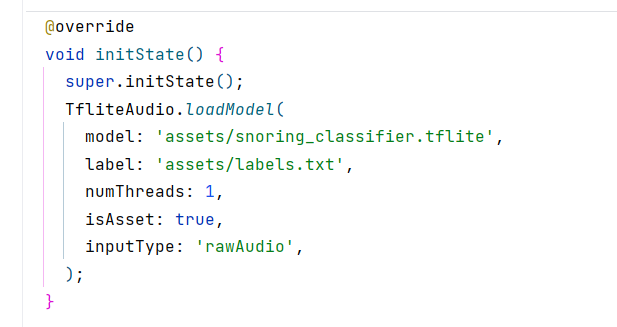


图 3‑4 加载模型

点击“开始记录”按钮后，首先判断权限是否获取成功，然后记录流逝时间，该值会参与平均背景噪音的计算中。



图 3‑5 点击开始按钮

\_recognize()函数的逻辑大致为模型不断监听麦克风，并判断当前声音类型，根据不同声音类型选择计算背景音或开始录制。



图 3‑6 判断声音类型

上图中的\_updateBgAmplitudeDb()函数更新平均背景噪声，而\_startRecording()函数开始记录声音（鼾声或梦话）并保存到手机存储的相应位置中。

点击“停止录制”按钮后，保存睡眠报告。



图 3‑7 点击停止录制

上图中的\_generateReport()函数即负责保存睡眠报告的逻辑。

录制界面的其余逻辑及查看睡眠报告的逻辑较为简单，则不再赘述。完整代码均已包含在附件中。

# 系统实现

## 睡眠记录界面



图 4‑1 睡眠记录界面

打开APP即进入主界面（即“睡眠记录”界面）。在主界面点击“开始记录”，即开始睡眠记录。在此过程中会显示记录时长以及实时显示当前声音类型。点击“结束记录”即结束当此睡眠记录，并生成睡眠报告。

## 睡眠报告界面



图 4‑2 睡眠报告界面

可从边栏切换到“睡眠报告”界面。该界面顶部可切换不同睡眠报告。对于每一份睡眠报告，都将显示其起止时间、持续时间（这里因为演示的时间太短，所以显示效果不好），睡眠记录过程中的平均背景噪声、打鼾次数、说话次数，以及每次打鼾和说话的起止时间。同时还可播放每次打鼾和说话的声音记录，用于查看睡眠状况。

# 项目总结

## 小组成员分工

高歌：单人完成APP的全部开发工作。

## 系统总结与展望

当前来说，本项目基于Tensorflow Lite和Flutter简单实现了一个能够通过手机麦克风记录睡眠情况并生成睡眠质量报告的手机APP。

未来可从两个方向着手继续优化APP。一是提升声音识别的准确度并增加可识别的声音种类。当前可以识别的声音仅有背景噪音、鼾声以及说话声，且模型还无法很好地区分鼾声及说话声。而睡眠过程中的声音常常出现起夜、重物落地（如床边小物件）等其他声音，将这些声音也识别为这三类是不太合适的，因此这方面有待加强。

二是优化UI界面，增加系统功能。当前系统已经可以采集整个睡眠过程中的鼾声情况、梦话声音等，可以在此基础上绘制睡眠周期图，并根据鼾声及梦话的频次判断哪个时段处于什么睡眠周期，并且对睡眠过程中的异常打断做判断，统计起夜次数。

总的来说，当前已经实现了一个简单而实用的睡眠质量监测APP，并且有待更进一步的完善。