上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

项目报告

PROJECT REPORTS

题目： 心脏生理信号监测诊断系统的开发

指导老师： 涂 圣 贤

参与学生：张露雨 519130910018

郭焘玮 520021910866

张弼弘 520021910492

目录

心脏生理信号监测诊断系统的开发

1. 背景介绍

据《中国心血管健康与疾病报告 2021》报道，中国心血管病患病率处于持续上升阶段。推算心血管病现患病人数 3.3 亿，心脏疾病占心血管疾病死亡总数的六成，据统计，72%的患者在心脏骤停前会有明显不适，其中 70%的患者预警症状持续 15 min 以上[1]。心脏生理信号监测作为有效的心脏疾病监测和预防手段，在心血管医学领域意义重大。

心脏电信号是由于心脏在心动周期中，心肌细胞共同活动产生的电信号，电活动的强弱、参与收缩的心肌细胞的数量和位置，共同决定了心电信号的幅度和方向[2]。心电信号是很微弱的低频信号，幅值通常为 0.5~4mV，频率在 0.05~20Hz 之间。

心音信号由心肌收缩、心脏瓣膜关闭和血液撞击心室壁、大动脉壁等引起的机械波现象所产生，能够反映心脏瓣膜活动和血液流动状况。它包含的生理信息特征，如心率（HR）、心脏舒张期与心脏收缩期的时限比（D/S）、第一心音和第二心音的幅值比（S1/S2），对心血管疾病的预防和诊断有重要的参考价值。

心脏疾病是目前全球范围内导致死亡和疾病负担的主要原因之一，早期诊断和治疗对于预防和治疗心脏疾病具有至关重要的作用。心电图是临床上用以诊断心血管疾病最常用的一种方式。心脏在跳动的过程中产生电脉冲，而由此产生的电脉冲会引起人体体表的电势发生变化，心电图记录下了这样的变化[3]。但是心电图的诊断由于受到医生的个人经验和主观判断的影响容易出现偏差，如何对心脏疾病进行正确的早筛早查以及全流程的管理正在成为越来越重要的问题。

基于此、本课题将从服务端、客户端、模型算法三个方向进行心电监测平台的开发。

2. 研究内容及重难点

2.1 服务端

2.1.1 服务端研究内容

- 与客户端进行对接，完成数据传输的双向传输
- 医生、患者的管理，数据库的存储
- 调用模型分析患者的心电心音信号，并返回给客户端
- 软件 license 管理，包括服务及数量

2.1.2 服务端研究重难点

（1）高效的数据传输和处理：在与客户端进行对接时，实现高效且稳定的双向数据传输是一个重大的挑战。服务端需要处理大量的并发请求，同时确保数据的完整性和安全性。处理数据的延迟和阻塞问题，以及进行有效的负载均衡也是研究重难点。

（2）复杂的用户和数据库管理：医生和患者的管理，以及数据库的存储需要复杂的权限和访问控制。设计和实施一个可扩展的数据库架构，以及有效的数据备份和恢复策略，是服务

端开发的重要挑战。

（3）调用模型分析和结果反馈：服务端需要调用模型来分析患者的心电心音信号，然后将结果返回给客户端。这涉及到处理大量的数据，以及实现实时或近实时的分析，这是一个重大的技术挑战。此外，如何准确解释和反馈模型分析的结果，也是一个研究重难点。

2.2 客户端

2.2.1 客户端研究内容

- 从蓝牙获取病人的心音数据，对数据进行实时展示
- 根据上述接口和传输要求上传至服务器端，接收并展示处理后的数据。
- 对病人与医生账户的管理。

2.2.2 客户端研究重难点

（1）心电信号的采集的便携性。从用户预防和随时监测的需求出发，心脏信号采集传感器应当是小巧而便携的，方便患者随身携带，不影响日常生活和工作。因此心脏信号采集手机客户端应当支持以无线方式与这些传感器进行通讯。

（2）数据传输和存储：心电信号采集后需要通过手机客户端进行传输，数据则需要保存在云端的服务器上，因此需要考虑与服务器的通信协议问题。

2.3 算法

2.3.1 算法研究内容

由于是对心血管病患者的心脏生理信号诊断，算法的准确性尤为重要。此外，数据是直接由客户端采集，算法的输入应该不需要进行人工处理。最后，因为模型需要配合服务器端进行部署，选用的算法在保证准确性的前提下不应有过大的计算量和复杂度。

综上，我们需要研究决策树、深度学习、小波分析、聚类分析等算法，并从中找出分类准确性高、稳定性好、无需人工干预的一类算法。

2.3.2 算法研究难点

最终使用深度学习的方法对心脏生理信号进行分类，由于此前从未学习过机器学习相关的课程，所有需要从零开始学习相关知识，对于从未接触过的新手来说，即使是复现论文中的方法也存在一定困难。

此外，由于对服务器操作的不熟悉，因此选择在本地进行模型训练。在显存有限的情况下，训练数据量和 batch size 都受到了很大的限制，这为训练出准确且高效的模型带来了一定的困难。

2.3.3 算法研究创新点

在原有的算法基础上增加可视化模块，能够比较直观地显示数据处理的过程。

增加适应于客户端采集信号的数据处理模块，使算法能够更好地配合客户端使用

3. 研究方法与方案

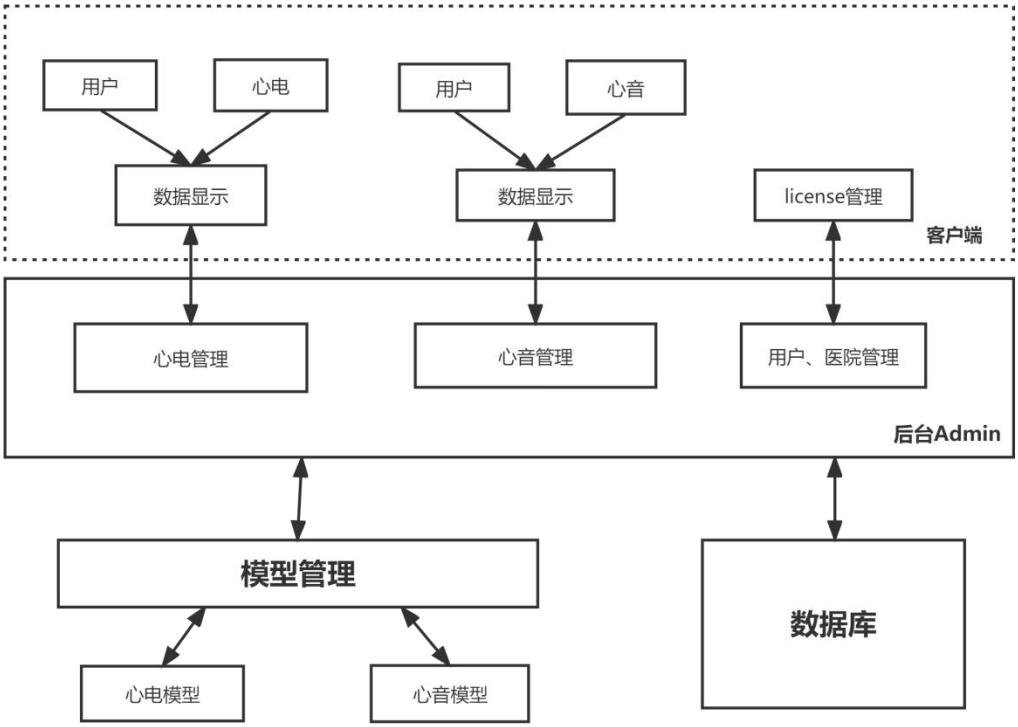


图 3.1 总体方案图

我们的技术路线包含服务端、客户端、模型三部分组成。

在完成用户心电或心音的数据采集后，数据将被传输到我们的客户端。客户端首先负责展示这些数据，让用户可以实时地看到他们的心电或心音数据。这个展示功能不仅可以帮助用户理解他们的数据，也能让用户感到更加参与和控制他们的健康状况。然后，客户端将这些数据传输到我们的服务端。

在服务端，我们有一个强大的模型库，可以对接收到的数据进行深度分析。这些模型经过了大量的训练和优化，可以在短时间内对大量的数据进行处理，而且准确性极高。服务端的数据处理功能还包括数据清洗、数据整合和数据解析等。数据清洗可以去除数据中的噪声和异常值，数据整合可以把来自不同源的数据进行融合，数据解析可以把原始数据转化成有意义的信息。

最后，服务端会将分析结果返回到客户端进行展示。返回的结果包括具体的分析数据，以及是否正常，帮助用户理解和使用这些数据。这个结果展示功能也采用了用户友好的设计，让用户可以方便地查看和理解结果。

4. 服务端

4.1 技术路线介绍

服务端技术路线：Python+Django+DRF+SQLlite

接口规范：RESTful

4.1.1 Dango 框架介绍

Django 是一个基于 Python 的开源 Web 应用程序框架，它采用了 MVC（Model-View-Controller）的设计模式，旨在帮助开发者快速地构建高质量的 Web 应用程序。Django 具有许多优秀的特性和工具，例如 ORM（对象关系映射）、自动化管理后台、URL 路由、表单验证、安全性、缓存等等。以下是一些 Django 的主要特点：

高效快速的开发： Django 为 Web 开发者提供了许多内置的功能和库，大大降低了开发成本。例如，Django 的自动化管理后台、ORM 等功能可以使开发者快速构建出基本的 Web 应用程序。

内置 ORM：Django 的 ORM 功能可以让开发者直接操作数据库表格而无需编写 SQL 代码，同时也支持多种数据库，如 MySQL、PostgreSQL 等，极大地提高了开发效率。

安全性： Django 提供了内置的安全性控制，包括用户认证、跨站点请求伪造（CSRF）保护、XSS（跨站点脚本）保护等等，让开发者可以轻松地实现高安全性的 Web 应用程序。

可扩展性：Django 的应用程序是模块化的，可以非常容易地添加新的应用程序或插件。此外，Django 的设计理念是“松耦合”，应用程序之间不会互相影响，极大地提高了可扩展性。

结合本项目实际情况，我们将以 Django 框架为基础进行开发。

4.1.2 SQLlite 数据库介绍

SQLite 是一个开源的、嵌入式的、轻量级的数据库系统，它以独立性、零配置、跨平台兼容性、ACID 兼容性和丰富的 SQL 支持等特性著名。作为一个轻量级数据库系统，SQLite 只需要很少的磁盘和内存资源，因此非常适合小型设备，如嵌入式设备和移动设备。SQLite 的所有操作都是直接在磁盘文件上进行的，避免了在服务器和客户端之间传输数据的开销。此外，SQLite 不需要任何配置，也不需要数据库管理员，这使得 SQLite 非常易于安装和使用。它可以在各种操作系统上运行，包括 Windows、Linux、Mac OS X 等。

SQLite 遵循 ACID（原子性、一致性、隔离性、持久性）原则，能确保数据库的一致性和可靠性，支持完全的事务处理。尽管 SQLite 是一个小型数据库，但它支持大部分 SQL92 标准的 SQL 语法。此外，SQLite 是在公共领域下发布的，可以自由使用，包括在商业产品中。SQLite 数据库存储在单一的磁盘文件中，这使得 SQLite 数据库非常方便备份和传输。最后，SQLite 支持多种数据类型，包括 NULL、INTEGER、REAL（浮点数）、TEXT（文本）和 BLOB（二进制对象）。

4.2 项目展示

在进入后台 Admin 管理页面之后，用户首先会看到首页。此首页集中展示了所有的功能和信息，为用户提供便捷的管理和信息查询功能。

首页的左侧是一个精致设计的侧边栏，由医院、患者、模型、认证和授权等模块构成。这些模块是后台管理的重要组成部分，负责处理与各自模块相关的数据和业务。例如，医院模块负责管理医院的信息和数据，患者模块则负责管理和维护患者的信息。模型模块对数据模型进行管理和调整，认证和授权模块负责管理用户的权限和认证信息。

首页的右侧是主体部分，主要包含快捷操作和最近的修改操作记录两个功能。快捷操作功能提供了一种便利的方式，使用户能更快速地完成常用任务，如添加新的患者信息，修改医院的信息等。最近的修改操作记录功能则提供了一个可以追踪和查看网站最新变化的手段。所有对网站进行的修改，无论何时、由谁执行，都会被记录在此。通过查看这些记录，用户可以清晰了解网站在何时、由谁、进行了何种修改，从而更有效地追踪和控制网站的变化。

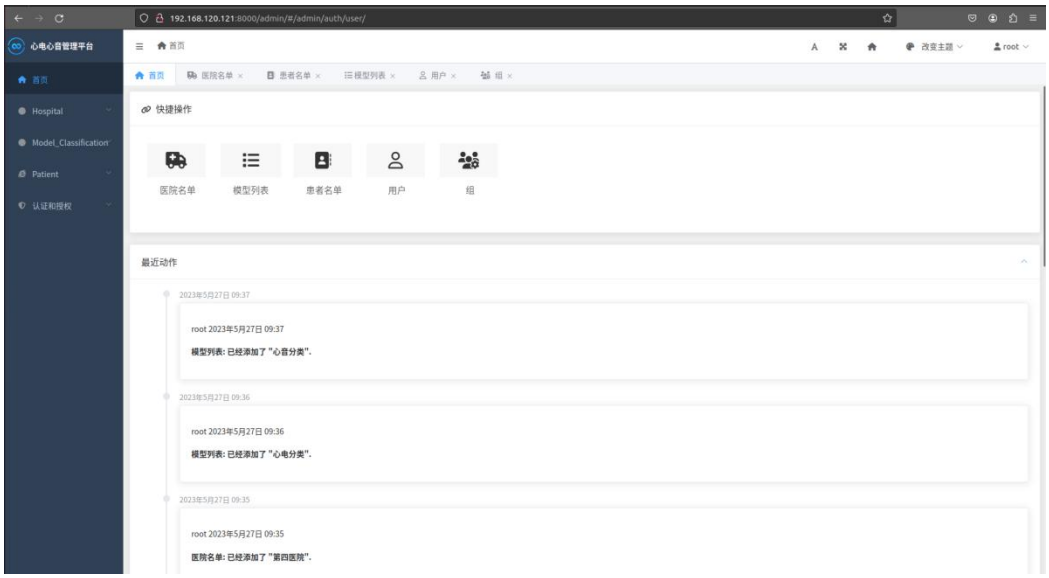


图 4.1 服务端后台 Admin 首页

以医院模块为例，用户进入该模块后，首先会看到一个详尽的医院信息列表。这个列表以清晰的方式展示了医院编号、医院名称、医院地址、医院介绍和 license 授权等关键信息。这些信息对于后台管理人员来说至关重要，因为它们不仅提供了全面的医院信息，也方便了数据的查找和管理。

医院编号是每个医院的唯一标识，通过这个编号，管理员可以迅速找到特定的医院信息。医院名称和地址则提供了医院的基本信息，这对于患者和医疗人员来说都是非常重要的。医院介绍部分则包含了医院的详细描述，这样的设计帮助用户了解医院的历史、规模、专科服务等情况。license 授权模块则显示了医院的运营和服务授权信息，为管理员提供了合规性和合法性的参考。

除了这些展示信息，医院模块上还设计了增加、删除、保存、导入导出等功能键。增加键允许管理员添加新的医院信息，而删除键则可以移除不再需要的信息。保存键则为管理员提供了保存当前修改的功能，确保信息的准确性和一致性。而导入导出键则为管理员提供了便捷的数据管理方式，用户可以方便地将数据导入到其他系统或从其他系统导入数据。

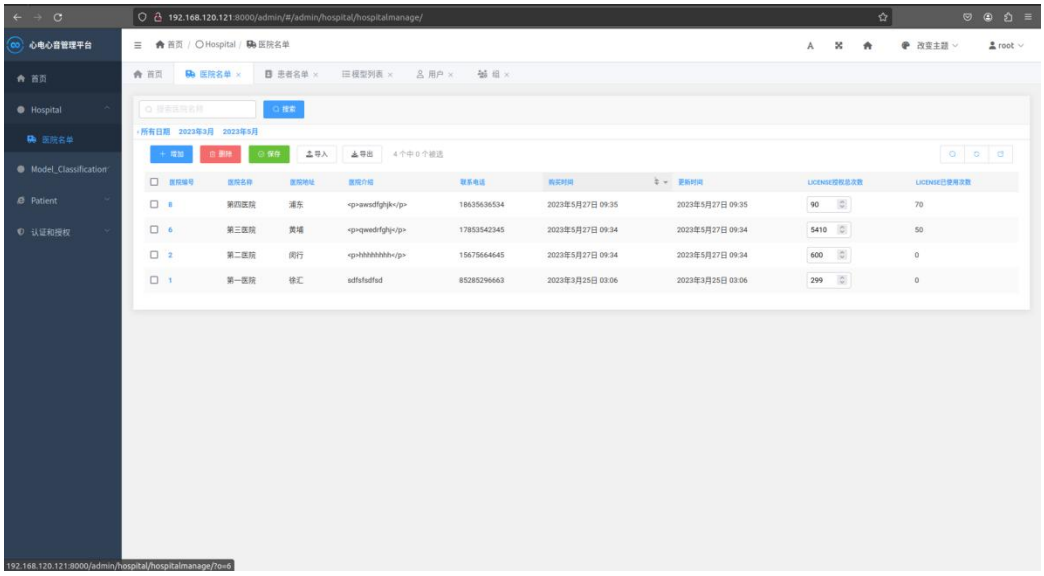


图 4.2 服务端医院管理模块页面

当用户点击医院模块中的“增加”按钮时，系统会提供一个新的医院记录表单供用户填写。这个表单设计得十分详细，包括必填条目和选填条目，以确保收集到的医院信息尽可能全面。

必填条目包括医院的基本信息，如医院编号、医院名称等。这些信息对于医院的识别和查找至关重要。医院编号是医院的唯一标识符，有助于在大量的医院信息中快速定位。医院名称是医院的基本属性，不仅方便了用户的查找，也保证了医院信息的完整性。

除了必填条目，表单中还包括一些选填条目，比如医院的联系方式、开放时间和其他附加信息等。这些信息虽然不是必须的，但如果提供的话，可以帮助用户更全面地了解医院的详细情况。

为了让用户在填写医院介绍时有更好的体验，我们在表单中融合了富文本编辑框。通过这个编辑框，用户可以方便地输入和格式化文本，添加图片，甚至插入链接等。这意味着在填写医院介绍时，用户不仅可以输入文本，还可以插入图片、表格和其他元素，使得介绍更加生动且具有吸引力。

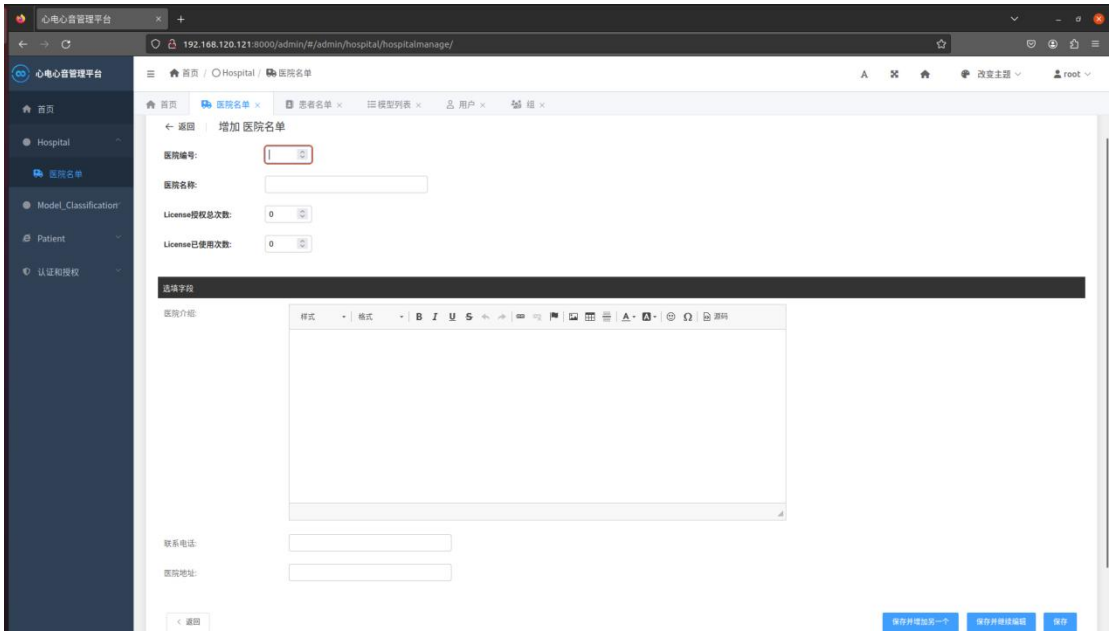


图 4.3 服务端医院管理模块增加板块

4.3 模型调用模块

模型调用的接口在 model_classification 文件夹下的 views.py 文件中。客户端使用 GET 请求传 data 给服务端，服务器调用模型进行分析，返回分析结果给客户端。

4.3.1 心电模型调用接口

```
from model_classification.ECG.ECGModelTest import ecg_inference
class ECGModelInferenceView(APIView):
    def get(self, request):
        #客户端将数据以{data:array}参数传入,通过 get 请求获取数据
        data_input = request.GET["data"]
        result = ecg_inference(data_input)
        return Response(result)
```

模型推理：

```
def ecg_inference(input_data):
    model = load_model('model_classification/ECG/ECGmodel_cp1.h5')
    pred = model.predict(input_data)
    pred_class = pred.argmax(axis=-1)
    result=pred_class[0]
    return result
```

4.3.2 心音模型调用接口

```
from model_classification.HS.HSModelTest import hs_inference
class HSModelInferenceView(APIView):
    def get(self, request):
        #客户端将数据以{data:array}参数传入，通过 get 请求获取数据
        data_input=request.GET["data"]
        result = hs_inference(data_input)
        return Response(result)
```

模型推理：

```
def hs_inference(input_data):
    model = load_model('model_classification/HS/HSmodel_cp01.h5')
    test_predictions = model.predict(input_data)
    result = np.argmax(test_predictions)
    return result
```

5. 客户端

5.1 客户端方案路线

总体采用 Android Studio 编写基于安卓操作系统的手机客户端。Android Studio 是谷歌推出的一款 Android 集成开发工具，基于 IntelliJ IDEA。它是用于开发 Android 应用的官方集成开发环境（IDE）。

5.1.1 用户界面设计。

用户界面设计是 android 应用程序开发的重要一环。从应用功能出发，如下是总结的用户界面设计的实验需求：

用户可以通过输入相关信息来采集心脏生理信号，并将其显示在展示栏中。同时，用户还可以通过上传/下载按钮来提交和下载信号数据。

界面的设计需要考虑到以下几个方面：

1. 用户信息的输入：需要提供若干文本输入框，用于填写用户的相关信息。
2. 信号接收和显示：需要提供开始/停止按钮，用于控制信号的接收和显示。
3. 信号上传和下载：需要提供上传/下载按钮，用于控制提交和下载信号数据。

5.1.2 便携式传感器的连接。

蓝牙作为一种短波高频无线通信标准，其传输距离适中，低功耗，低成本，易于广泛应用。因此选择蓝牙作为手机客户端与心脏信号传感器交互的接口。

5.1.3 数据传输与存储。

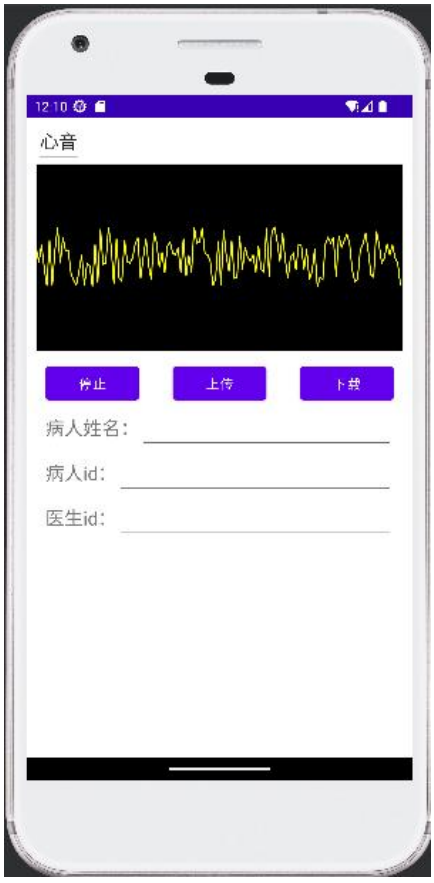
由于不同组员在各自领域分工合作，与服务器的交互事实上是设计一个数据接口并约定好传输参数。需要考虑与服务器的通信协议问题。http 协议中有 GET 和 POST 可以分别实现从服务器接收数据和向服务器上传数据。Java 语言中，有 json 数据交换格式可供选择，它采用了完全独立于语言的文本格式，可以在多种语言之间进行数据交换。

5.2 客户端实施进展

在手机客户端的开发设计方面，对应实验方案计划，实现了方案中的 5.1.1 和 5.1.3，部分实现了方案中的 5.1.2；具体实现了如下客户端功能：

（1）图像化界面和互动的应用。如下是实际实现的 android app 界面。

最上面一行是可编辑的文本框，默认显示“心音”字样，标识采集的心脏数据的类型，下面是显示波形的主要区域。在下方有三个按钮，点击即可开始采集、或向服务器上传数据、或从服务器下载数据。下方还有输入病人相关信息的文本输入框。



（2）模拟一个心音波形，展示在界面上。

由于小组人力有限，我们没有额外精力完成心脏生理信号采集传感器的工作，折中选择了模拟一个随机信号作为心音信号输入。

（3）与服务器端使用 GET 和 POST 方法接收和发送数据，POST 实现上先将 java 数据类通过 gson 提供的功能转化为一个 json 字符串，再通过 post 请求上传到服务器，而 GET 操作则相反，先向服务器提交 GET 请求，接收返回的 json 字符串，再通过 gson 的功能转化为 java 类，最后提取数据包中相应的信息。

6. 算法模型

6.1 心电分类方案设计

获取数据 —> 数据预处理 —> 数据扩增 —> 神经网络训练 —> 测试

6.2 心电分类具体实验方案

6.2.1 获取数据

使用 MIT-BIH 心电数据库，原始数据包括头文件、数据文件、标注等，如下图所示：

100	1992/7/29 21:21	ATR 文件	5 KB
100.dat	1992/7/30 1:36	DAT 文件	1,905 KB
100	1992/7/30 1:36	HEA 文件	1 KB
100.xws	1999/12/12 17:28	XWS 文件	1 KB
101	1992/7/30 1:36	ATR 文件	4 KB
101.dat	1992/7/30 1:36	DAT 文件	1,905 KB
101	1992/7/30 1:36	HEA 文件	1 KB
101.xws	1999/12/12 17:28	XWS 文件	1 KB

图 6.1 心电原始数据

6.2.2 数据预处理

由于原始数据是双通道采集，且不同通道的数据差异较小，因此仅选择一通道的数据。

此外，文件中标注的心拍种类达十余种，本项目中仅选取其中七种进行分类：A（房性早搏，atrial premature beat）、E（室性逸搏，ventricular escape beat）、L（左束枝传导阻滞心搏，left bundle branch block beat）、N（正常心搏，normal beat）、PB（起搏心搏，paced beat）、R（右束枝传导阻滞心搏，right bundle branch block beat）和 V（室性早搏，premature ventricular contraction）。

然后根据标注文件中的标注进行分割，图中代码展示的函数是分割一个心电数据中所有标注为 N（正常心拍）的心拍。

```
def segmentation(records):
    Normal = []
    for e in records:
        #print(e)
        signals, fields = wfdb.rdsamp(e, channels = [0])

        ann = wfdb.rdann(e, 'atr')
        good = ['N']
        ids = np.in1d(ann.symbol, good)
        imp_beats = ann.sample[ids]
        beats = (ann.sample)
        for i in imp_beats:
            beats = list(beats)
            j = beats.index(i)
            if(j!=0 and j!=(len(beats)-1)):
                x = beats[j-1]
                y = beats[j+1]
                diff1 = abs(x - beats[j])//2
                diff2 = abs(y - beats[j])//2
                Normal.append(signals[beats[j] - diff1: beats[j] + diff2, 0])
    return Normal
```

图 6.2 心拍分割函数

分割完成后，将数据转换为 128×128 的灰度图像，易于观察和保存。

图 6.3 一个正常心拍的 128×128 灰度图像

6.2.3 数据扩增

由于部分异常心拍的数量较少，为了保持数据集的平衡，需要对其进行扩增。具体来说，对需要扩增的图像进行左上、中上、右上、左中、中、右中、左下、中下和右下的平移，裁剪获得额外 9 个数据特征。

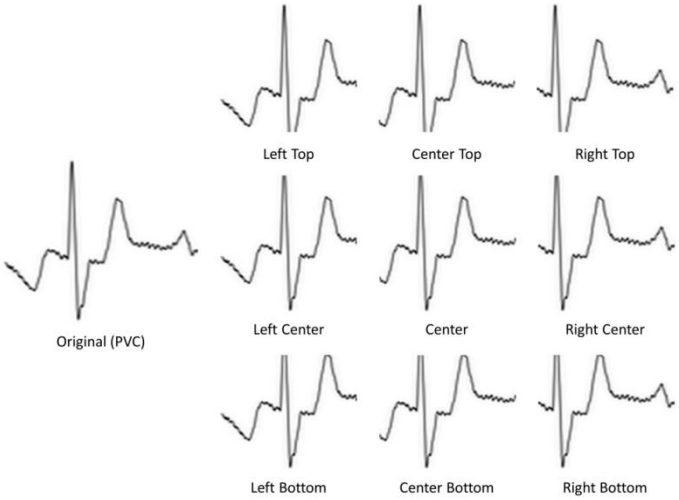


图 6.4[1] 原始 PVC 异常图像以及 9 张扩增图像

6.2.4 神经网络训练

网络结构见下图：

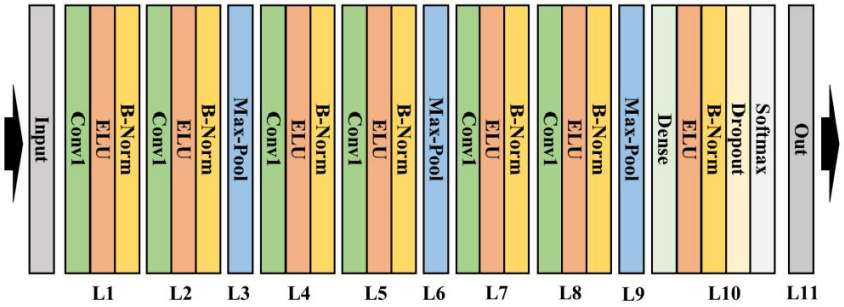


图 6.5[7] 心电分类 CNN 结构

```
inputdata = keras.Input(shape=(128, 128, 3))

final = Conv2D(64, (3,3),strides = (1,1), input_shape = (128, 128, 3), kernel_initializer='glorot_uniform')(inputdata)
final = keras.layers.ELU()(final)
final = BatchNormalization()(final)
final = Conv2D(64, (3,3),strides = (1,1),kernel_initializer='glorot_uniform')(final)
final = keras.layers.ELU()(final)
final = BatchNormalization()(final)
final = MaxPool2D(pool_size=(2, 2), strides= (2,2))(final)
final = Conv2D(128, (3,3),strides = (1,1),kernel_initializer='glorot_uniform')(final)
final = keras.layers.ELU()(final)
final = BatchNormalization()(final)
final = Conv2D(128, (3,3),strides = (1,1),kernel_initializer='glorot_uniform')(final)
final = keras.layers.ELU()(final)
final = BatchNormalization()(final)
final = MaxPool2D(pool_size=(2, 2), strides= (2,2))(final)
final = Conv2D(256, (3,3),strides = (1,1),kernel_initializer='glorot_uniform')(final)
final = keras.layers.ELU()(final)
final = BatchNormalization()(final)
final = Conv2D(256, (3,3),strides = (1,1),kernel_initializer='glorot_uniform')(final)
final = keras.layers.ELU()(final)
final = BatchNormalization()(final)
final = MaxPool2D(pool_size=(2, 2), strides= (2,2))(final)
final = Flatten()(final)
final = Dense(2048)(final)
final = keras.layers.ELU()(final)
final = BatchNormalization()(final)
final = Dropout(0.5)(final)
final = Dense(7, activation='softmax')(final)
model = keras.Model(inputs=inputdata, outputs=final)

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

图 6.6 模型具体实现代码

6.3 心音分类具体方案

6.3.1 获取数据

使用 2016 PhysioNet/CinC Challenge 所用数据集，原始数据如下图所示：

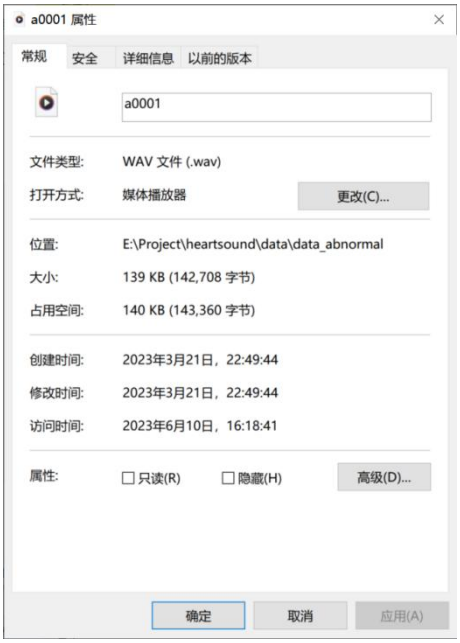


图 6.7 心音原始数据

6.3.2 数据预处理

使用 librosa 库读取原始音频数据后，使用二阶巴特沃斯中值滤波器。

```
def band_pass_filter(original_signal, order, fc1,fc2, fs):
    b, a = signal.butter(N=order, Wn=[2*fc1/fs,2*fc2/fs], btype='bandpass')
    new_signal = signal.lfilter(b, a, original_signal)
    return new_signal
```

图 6.8 滤波器代码

然后利用 samplerate 库将信号下采样到 1000hz 并进行归一化处理

```
down_sample_audio_data = samplerate.resample(audio_data.T, 1000 / fs, converter_type='sinc_best').T

down_sample_audio_data = down_sample_audio_data / np.max(np.abs(down_sample_audio_data))
```

图 6.9 下采样和归一化代码

最后以 2.5s 的长度切割音频，同时保留 50%的重叠

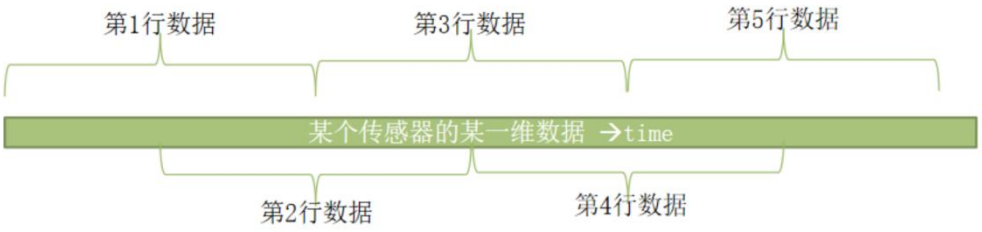


图 6.10[2] 保留 50%重叠示意图

6.3.3 特征提取

使用二阶谱分析法来进行特征提取，生成的特征图是 256×256 的二维矩阵。

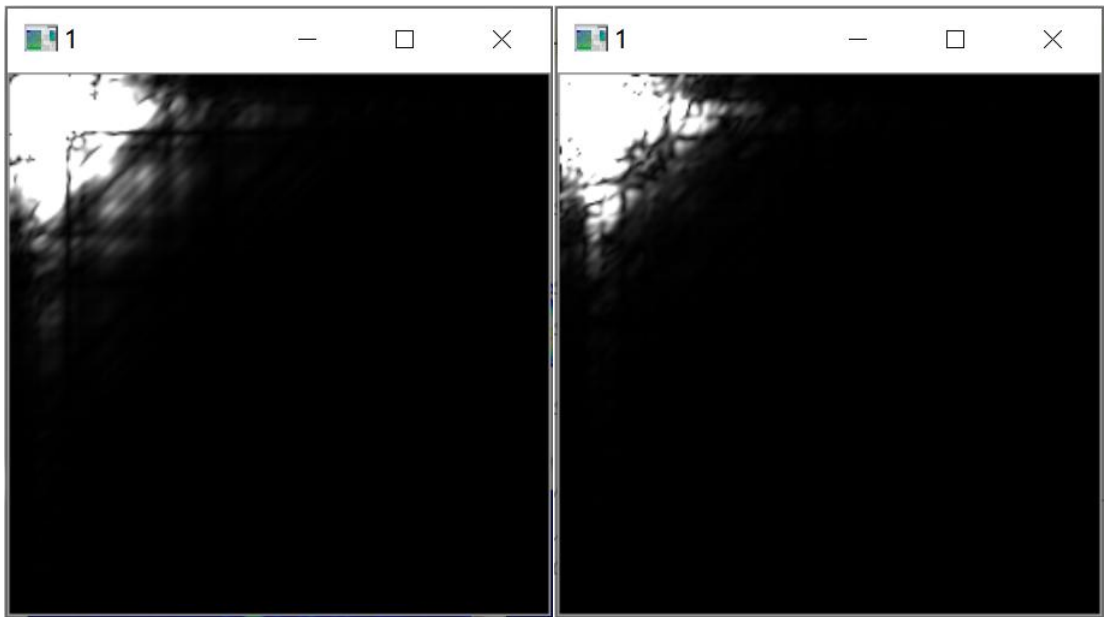


图 6.11 不同音频的特征图

6.3.4 神经网络训练

网络结构见下图：

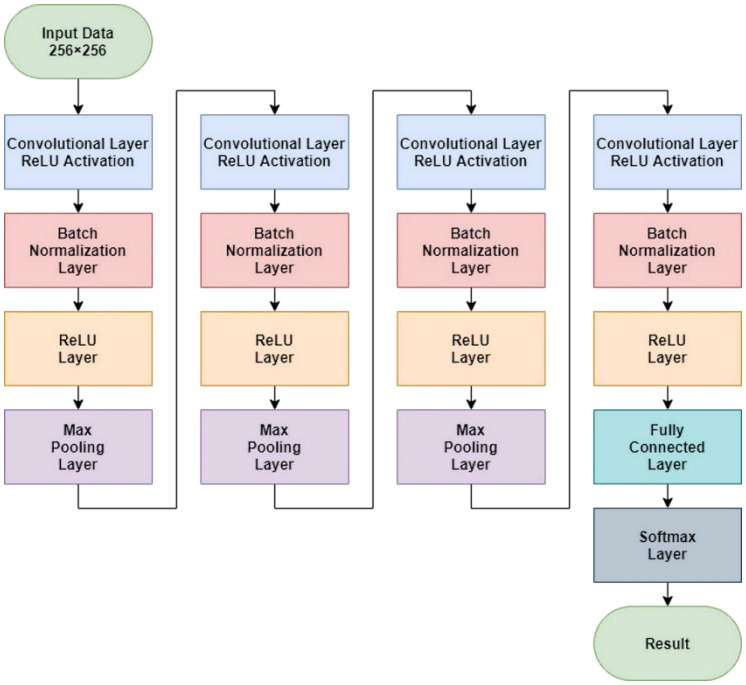


图 6.12[8] 心音分类 CNN 结构

```
inputdata = keras.Input(shape=(256, 256, 1))

final = keras.layers.Conv2D(64, (3, 3), padding="same",activation='relu')(inputdata)
final = keras.layers.BatchNormalization()(final)
final = keras.layers.ReLU()(final)
final = keras.layers.MaxPooling2D((2, 2), strides=(2, 2))(final)

final = keras.layers.Conv2D(32, (3, 3), padding="same",activation='relu')(final)
final = keras.layers.BatchNormalization()(final)
final = keras.layers.ReLU()(final)
final = keras.layers.MaxPooling2D((2, 2), strides=(2, 2))(final)

final = keras.layers.Conv2D(16, (3, 3), padding="same",activation='relu')(final)
final = keras.layers.BatchNormalization()(final)
final = keras.layers.ReLU()(final)
final = keras.layers.MaxPooling2D((2, 2), strides=(2, 2))(final)

final = keras.layers.Conv2D(32, (3, 3), padding="same",activation='relu')(final)
final = keras.layers.BatchNormalization()(final)
final = keras.layers.ReLU()(final)

final = keras.layers.Flatten()(final)
final = keras.layers.Dense(2)(final)
# final = keras.layers.BatchNormalization()(final)
final = keras.layers.Softmax()(final)

model = keras.Model(inputs=inputdata, outputs=final)
optimizer = keras.optimizers.Adam(
    lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None)
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])
return model
```

图 6.13 模型具体实现代码

6.4 实验结果

6.4.1 心电分类实施进展

已经完成了心电分类模型的训练与测试，并且成功部署到后端服务器。

模型在测试集上的准确率达到 94.26%，然而并没有达到论文中 98%左右的准确率，原因可能是训练使用数据量较小或是训练轮次较少。如果之后有机会，会继续研究并训练出更好的模型。

模型参数经多次调整，发现最终模型的准确率均不及调整之前。最终在模型文件大小和准确率两者之中，还是优先选择准确率，因此对模型参数并未做出改动，导致文件较大，达到了 877MB。

6.4.1.1 结果展示

心电分类模型的混淆矩阵如下，从图中可以看到虽然总体的准确率为 94.26%，但是对标记为 E 和 N 的心拍的识别率分别为 96.6%和 98.8%，而标记为 R 的心拍识别率仅有 91%，不同异常的识别差异较大。

而观察 N 列可以发现数字普遍较大，说明其他异常被识别为正常心拍的概率较高，可能是因为测试集中部分的异常心拍与正常心拍的波形差异较小，模型未能正确识别。

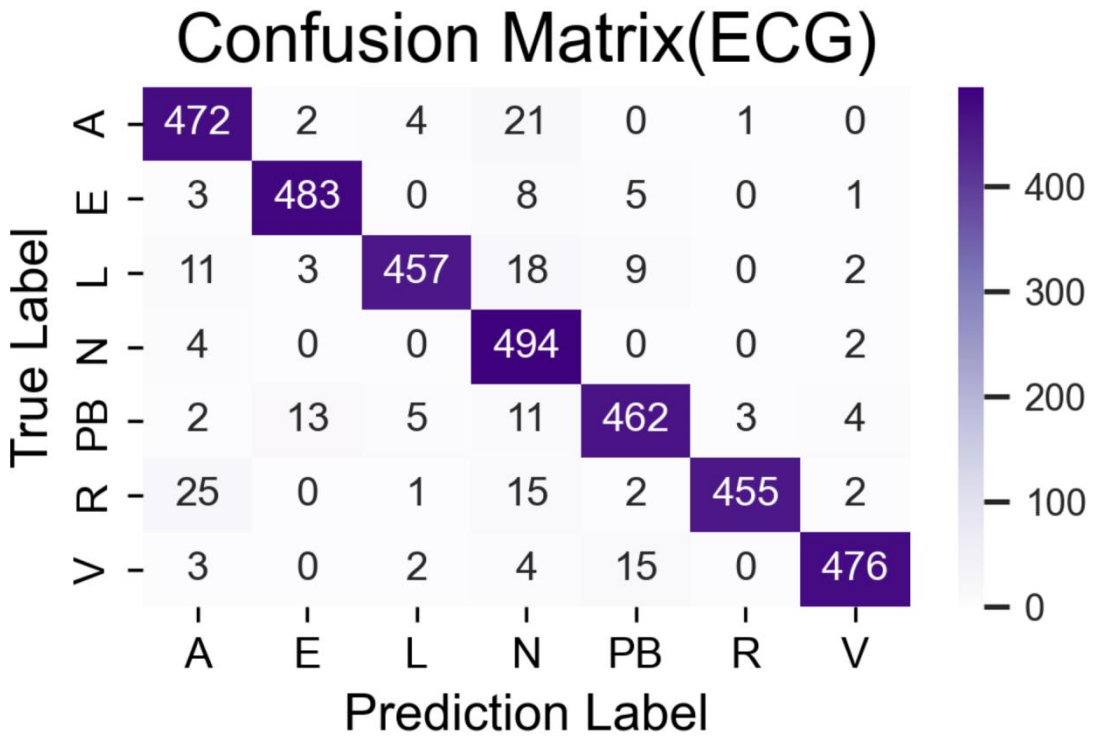


图 6.14 混淆矩阵（心电分类模型）

6.4.2 心音分类实施进展

已经完成了心音分类模型的训练与测试，并且成功部署到后端服务器。

模型在测试集上的准确率达到 92.475%，同样没有达到文章中 96.77%的准确率，原因应该也是训练使用数据量较小或是训练轮次较少。

6.4.2.1 结果展示

心音分类模型的混淆矩阵如下，可以看到模型识别正常和异常心音的准确率相差并不多。

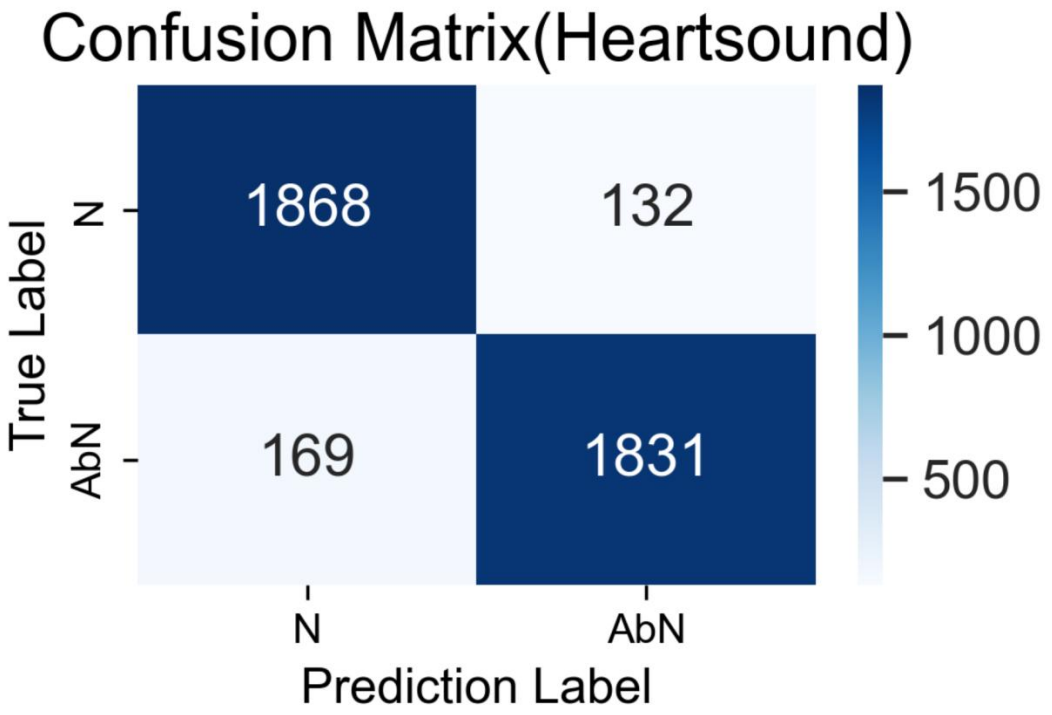


图 6.15 混淆矩阵（心音分类模型）

7. 总结与展望

在过去的一段时间里，我们的团队在客户端、服务端和算法三个方面进行了深入的工作。我们的客户端功能包括从蓝牙获取病人的心音数据并进行实时展示，将数据上传至服务器端，并接收并展示处理后的数据。此外，我们的客户端还负责医生和病人账户的管理。在服务端，我们实现了与客户端的数据传输对接，进行医生和患者的管理，以及数据库的存储。我们还实现了调用模型来分析患者的心电心音信号，并将结果返回给客户端。在算法方面，我们通过深度学习实现了心电信号和心音信号的分类，并实现了与服务器的数据传输。

展望未来，我们有许多的计划和期待。在客户端，我们计划进一步优化数据获取和展示的过程，以便更加有效地处理和展示心音数据。为了更好地满足医生和患者的需求，我们也将优化账户管理系统，增加多账号管理、密码处理、登录和忘记密码等功能。对于病人账户，我们计划建立一个与医生和检查结果之间的对应关系管理系统，使得医生和病人之间的多对多对应关系以及与检查结果的一对多对应关系得到有效管理。

在服务端，我们将持续改进数据传输系统，保证数据传输的双向性和稳定性，同时提高数据传输的效率。我们也将对医生和患者的管理进行更深层次的优化，提高数据库的存储能力和效率。另外，我们将进一步完善模型调用系统，使其能更准确、更快速地分析患者的心电心音信号，并将结果返回给客户端。

在算法方面，我们将继续探索和研究深度学习技术，以提高心电信号和心音信号的分类准确性。我们将进一步优化数据传输算法，提高数据传输的效率和安全性。此外，我们也将研究新的算法和技术，以便更好地处理和分析心音数据。

总的来说，虽然我们已经取得了一些成果，但我们也清楚地认识到，我们仍有许多工作

需要去做。我们将继续努力，不断改进和优化我们的系统，以满足医生和患者的需求，提供高效、准确、易用的医疗服务。我们坚信，只要我们不断努力，我们将能够打造出一个更加易用的医疗服务系统。

8. 小组分工与合作

张露雨：负责总体路线规划、服务端开发、报告撰写、整合
郭焘玮：负责客户端开发、报告撰写
张弼弘：负责算法模型训练、报告撰写

9. 参考文献

[1] X. Zhai and C. Tin, "Automated ECG Classification Using Dual Heartbeat Coupling Based on Convolutional Neural Network," in IEEE Access, vol. 6, pp. 27465-27472, 2018, doi: 10.1109/ACCESS.2018.2833841.
[2] Mathews SM, Kambhamettu C, Barner KE. A novel application of deep learning for single-lead ECG classification. Comput Biol Med. 2018 Aug 1;99:53-62. doi: 10.1016/j.compbiomed.2018.05.013. Epub 2018 Jun 4. PMID: 29886261.
[3] Lai LS, Redington AN, Reinisch AJ, Unterberger MJ, Schriefl AJ. Computerized Automatic Diagnosis of Innocent and Pathologic Murmurs in Pediatrics: A Pilot Study. Congenit Heart Dis. 2016 Sep;11(5):386-395. doi: 10.1111/chd.12328. Epub 2016 Mar 15. PMID: 26990211.
[4]谷晓芳,许伟,陈宽,张嵩.远程心电监测技术研究进展[J].中国医学装备,2022,19(11):207-213.
[5].《中国心血管健康与疾病报告2021》要点解读[J].中国心血管杂志,2022,27(04):305-318.
[6]李传鹏,郭兴明,张文英等.基于智能手机的心音监测系统的研究与设计[J].计算机工程与应用,2014,50(13):37-41.
[7] Jun T J, Nguyen H M, Kang D, et al. ECG arrhythmia classification using a 2-D convolutional neural network[J]. arXiv preprint arXiv:1804.06812, 2018.
[8] 知错就改. 心音（PCG）分类算法——基于深度学习-树莓派的心脏疾病听诊[EB/OL].（2020-12-8）[2023-5-5]. https://zhuanlan.zhihu.com/p/334586587.

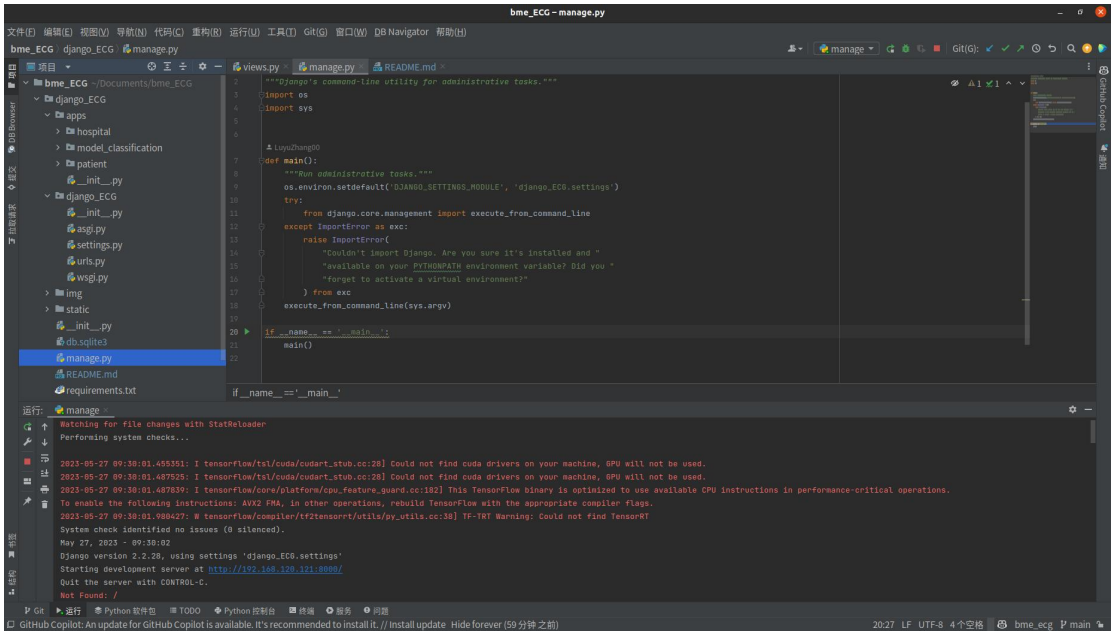
10. 附录

服务端开发环境：

主要依赖包：

- python==3.9.16
- django==2.2.28
- djangorestframework==3.14.0
- django-simpleui==2023.3.1
- sqlite3

Pycharm：



数据库管理 Dbeaver：