第九届湖南省研究生数学建模竞赛承诺书

我们仔细阅读了湖南省高校研究生数学建模竞赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们完全清楚，在竞赛中必须合法合规地使用文献资料和软件工具，不能有任何侵犯知识产权的行为。否则我们将失去评奖资格，并可能受到严肃处理。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们授权湖南省研究生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

所属学校和学院（请填写完整的全名）：国防科技大学智能科学学院

参赛队员（打印后签名）：1. 周瑜

2. 汪洋昊

3. 李莹

指导教师或指导教师组负责人（打印后签名）：

日期：2024年 7月 3 日

（请勿改动此页内容和格式。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。）

第九届湖南省研究生数学建模竞赛

题　目: 基于深度学习的人体行为识别

摘要:

本文基于智能手机内置运动传感器测量的人体活动数据对人体活动模式进行判别.首先,对无标签的运动数据根据阈值法建立了决策树形式的判别模型.然后,对带有标签的数据,通过机器学习方法建立了对活动类型分类的判别模型.最后分析研究了年龄、体重、身高对活动类型的影响.

针对问题 1:首先,本文对数据进行预处理,通过巴特沃斯滤波器对高频噪声进行去除,然后进行特征值提取.本文中使用了峰值、均值、标准差等时域特征值以及傅里叶变换后的频域特征信息.根据不同动作的特征值差异,设计基于阈值法的决策树分类模型.该模型结构简单、分类速度快,对某些活动类型能做出准确判断.对三位实验人员的活动状态的数据进行分类,并将分类结果填入论文表格中.

针对问题 2:首先,本文通过滑动窗口、中值滤波和数据归一化方法对数据进行预处理,将长时序数据划分为短时序数据.进一步,本文训练了基于卷积神经网络(CNN)和长短期记忆网络(LSTM)的深度判别模型 CNN-LSTM 用于捕获加速度数据和角速度数据的动态时序特征,以判别特征数据所对应的活动类型.最后,针对第(1)问,本文使用判别模型对附件 2 中的 10 位实验人员的活动状态进行了判断并且与问题一中提出的分类模型结果进行了比较;针对第(2)问,本文使用判别模型对某位实验人员的 30 次活动状态数据进行判别,并将判断的分类结果填入论文表格中.

针对问题 3:首先,本文通过加速度特征与时序之间的关系图证明了活动状态特征和实验人员个人特征之间存在直接联系.进一步,本文使用了问题二中的 CNN-LSTM 模型捕获了动作时序特征,并且分别训练了用于判别年龄、身高和体重数值区间的分类模型用于对具体人物画像.最后,本文使用训练好的判别模型分别判断了五位研究人员的年龄、身高和体重特征,并且通过三种特征的联合判断区分出了他们在附件 2 数据中对应的真实身份.

关键词:运动传感器;特征提取;模式识别;卷积神经网络

目录

1 问题综述

1.1 问题背景

在移动互联网技术的迅猛推动下,智能手机已成为个人健康管理的重要工具,其内置的多种传感器,特别是加速度计和陀螺仪,能够持续捕获用户的运动模式,为运动科学、健康监测等领域的研究提供了丰富的数据资源[1] .随着市场上智能手机健康管理应用的激增,如何高效且准确地利用这些数据资源,以实现对用户人体活动状态的精准识别,成为了当前研究的重要课题.

当前,人体活动识别主要依赖于两大方法:阈值法与机器学习法.阈值法通过预设固定的阈值,与从传感器数据中提取的特征变量进行比较,以此判断用户处于何种活动状态.机器学习算法通过分析传感器信号的时间序列特征,建立模型以识别运动模式.随着深度学习与强化学习技术的兴起,机器学习在复杂数据处理与动态环境决策方面展现出巨大优势.深度学习擅长从原始数据中自动提取高层次特征,而强化学习则通过智能体与环境的交互学习,优化决策过程,为活动识别带来了新的可能性.

尽管如此,现有研究在利用智能手机传感器数据进行活动状态识别时,仍面临诸多挑战与不足.一方面,由于人体生理特征的差异性,固定阈值难以适应所有用户,导致识别准确率受限;另一方面,大多数算法忽略了用户个人特征(如年龄、身高、体重等)对活动状态识别的影响,导致识别结果缺乏个性化.因此,为提升识别精度与效率,需要进一步优化算法设计,特别是在特征选择与模型构建方面.

1.2 问题提出

人体活动状态分类是一个复杂且多面向的问题,它主要依赖于加速度计和陀螺仪的传感器数据来捕捉人体在运动过程中的动态变化.本文需要从数据处理、特征提取、模式识别和机器学习的角度考虑,解决以下三个问题:

(1) 问题 1: 对未标记活动状态的实验人员数据进行分类,根据数据的相似性和差异性将附件 1 中的 60 组数据(每人)分配到 12 种活动状态中.
(2) 问题 2: 基于附件 2 中有标记的活动数据,提取 12 类活动状态的典型特征,建立人体活动状态的判别模型.然后将问题 1 的分类模型与此判别模型进行比较,分析不同模型的分类效果.最后用判别模型对附件 3 中的未知活动数据进行分类.
(3) 问题 3:基于附件 4 分析不同实验人员的同一活动状态是否存在差异,以及活动数据与实验人员的年龄、身高、体重这种个人特征的关系.然后利用模型判断附件 5 中人员身份.

2 模型假设与符号说明

2.1 模型基本假设

(1) 假设同一人在进行相同活动时,其运动模式(如步频、步长等)相对稳定.
(2) 假设实验人员在进行各种活动时,所处的环境(如地面材质、坡度等)相对一致,以减少环境因素对数据的影响.

2.2 符号说明

本文定义了如下 7 个使用次数较多的符号, 其余符号在使用时注明.

表1 符号说明

符号	含义
$\bar{P}$	峰值平均值
σ	标准差
SK	偏度
r_{xy}	相关系数
Accuracy	准确率
Precision	精确率
Recall	召回率

3 问题一的分析与求解

3.1 问题分析

问题一的核心在于对缺乏明确活动状态标签的数据集进行分类处理.具体而言,该任务涉及对附件 1 中提供的 3 名实验对象,在各自五种活动状态下采集的加速度计与陀螺仪数据进行解析与归类.这些传感器数据,包括沿 X 轴(重力方向)、Y 轴(前进方向)、Z 轴(垂直于 XY 平面指向身体一侧)的加速度及角速度变化,共同记录了人体在运动过程中的位姿动态变化.

在理论框架下,人体被视为一个动态变化的刚体系统,其运动状态的变化直接反映在加速度计与陀螺仪所捕获的位姿变化数据中.这些数据中,部分展现出周期性特征,如行走与跑步等连续运动,而另一部分则因涉及静态保持或活动间转换(如乘电梯、起身等)而呈现非周期性.此外,不同运动类型下,加速度信号的频率与振幅特性各异,特别是在跑步与跳跃等高强度活动中,重力方向上的加速度尤为显著.因此,本文采用通过提取信号中的关键特征(如峰值平均值、标准差、偏度、相关系数等)作为构建活动状态分类基础.

在分类方法的选择上,鉴于决策树在逻辑分类领域的广泛应用与高效性,本文采用决策树模型作为分类工具.该模型通过构建从根节点至叶节点的决策路径,形成了直观的分类规则体系.本文进一步将这一模型应用于人体活动状态识别的具体场景中,通过 IF-THEN 的分类逻辑形式,详细阐述了从数据预处理到特征提取,再到最终分类判别的完整流程,旨在实现高效的活动状态分类.

3.2 数据预处理

在数据采集的复杂过程中,不可避免地会遭遇多种性质的噪声干扰.为减轻这些噪声对采样数据质量的负面影响,学者们广泛采用滤波算法作为解决方案.其中,滑动均值滤波、中值滤波以及巴特沃斯低通滤波等算法因其有效性而备受青睐.特别地,巴特沃斯低通滤波器以其独特的性能特点脱颖而出:它在通频带内展现出极为平坦的频率响应曲线,确保信号在该区域内无纹波失真;而在阻频带内,则实现了振幅的连续且逐渐衰减至零,有效隔离了高频噪声.

鉴于低通巴特沃斯滤波器在通频带内的平坦性、阻频带的良好衰减特性,以及其设计上的简洁性,该滤波器在信号处理领域获得了广泛应用.本文基于上述优势,选择低通巴特沃斯滤波器作为工具,旨在有效抑制数据采集过程中噪声对数据的干扰.滤波器数学模型表示为

$$\left|H\left(\omega\right)\right|^2=\frac{1}{1+\left(\frac{\omega}{\omega_c}\right)^{2n}}=\frac{1}{1+\varepsilon^2\left(\frac{\omega}{\omega_p}\right)^{2n}} \tag{1}$$

其中,n 为滤波器阶数,ω_c 为截止频率,ω_p 为通带边缘频率.

美国 MathWorks 公司出品的著名商业数学软件 MATLAB 中提供了巴特沃斯滤波器的工具箱,极大降低了巴特沃斯滤波器设计难度,减少了调试时间.其中主要功能函数包括“buttord 函数”,“butter 函数”等,函数的详细使用可参考 MATLAB 技术文档.如图 1 所示,通过对比滤波前后的传感器信号,可以清晰地观察到,采用低通巴特沃斯滤波后,信号中的噪声得到了显著抑制,数据质量得到了显著提升.

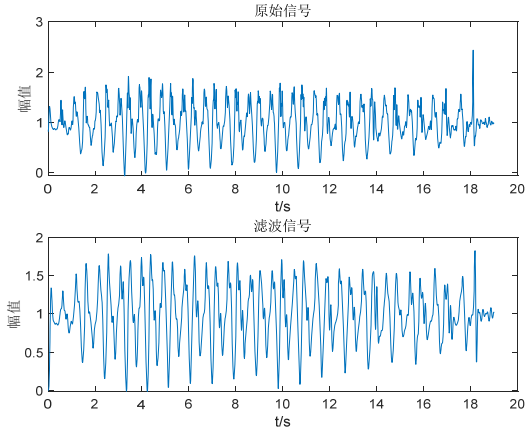


图1 某次测量信息滤波前后的传感器信号变化情况

3.3 分类模型的建立

3.3.1 特征提取

在构建分类模型的过程中,特征提取环节是至关重要的.它涉及将原始的、未处理的数据集转化为一系列精心设计的属性集合,旨在通过操作这些数据生成具有高度描述性、信息丰富且非冗余的特征表示.具体而言,以下是几种关键特征的提取与计算方法:

(1) 峰值,加速度信号变化达到的最大瞬间值.峰值平均值即加速度信号变化达到的最大瞬间值进行求和平均,峰值均值公式为

$$\overline{P}=\frac{1}{n}\left(\sum_{i=i}^{n}P_i\right) \tag{2}$$

其中,P_i 为第 i 个峰值,n 为一组数据中所有峰值总数.

(2) 标准差,方差的算术平方根,用以反映数据的离散程度.计算公式为

$$\sigma=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(X_i-\overline{X}\right)^2} \tag{3}$$

其中 N 为数据总数,$\overline{X}$ 为数据的均值.

(3) 均值,数据求和后除以数据总数,能反映数据中集的数值.计算公式为

$$\overline{X}=\frac{1}{N}\sum_{i=1}^{N}X_i \tag{4}$$

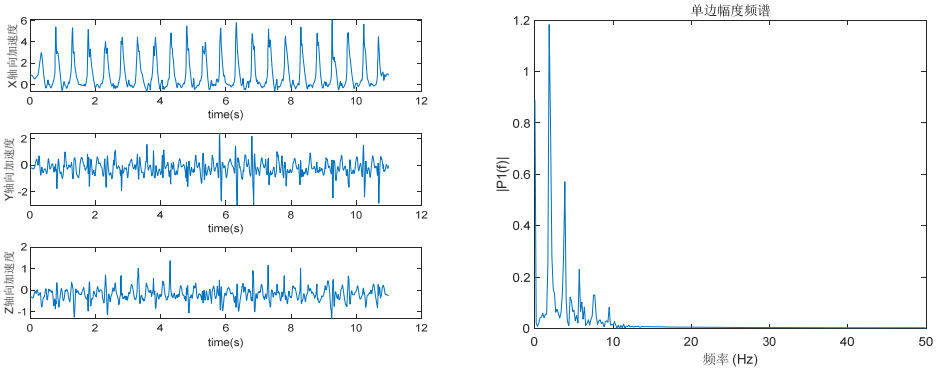
(4) 偏度,反映数据的分布偏斜情况的统计特征,计算公式为

$$SK=\frac{\frac{1}{N}\sum_{i=1}^{N}\left(X_i-\bar{X}\right)^3}{(N-1)(N-2)\sigma^3} \tag{5}$$

(5) 相关系数,反映数据之间线性相关程度的统计特征,计算公式为

$$r_{xy}=\frac{\sum_{i=1}^{N}\left(X_i-\bar{X}\right)\left(Y_i-\bar{Y}\right)}{\sqrt{\sum_{i=1}^{N}\left(X_i-\bar{X}\right)^2}\sqrt{\sum_{i=1}^{N}\left(Y_i-\bar{Y}\right)^2}} \tag{6}$$

(6)频域特征,通过傅里叶变换,可以将时域信号转化到频域上进行表示.MATLAB 中提供了频域分析的工具箱,极大降低了频域分析的设计难度.在 MATLAB 中,使用“fft”函数生成的单边频谱的幅度图如图 2 所示.其中图 2(a)表示某次活动加速度计的时域变化曲线,图 2(b)表示其 X 轴向的加速度时间序列的频谱图.



(a) 某次活动加速度计的时域变化曲线 (b) 表示其 X 轴向的加速度时间序列的频谱图

图2 单边频谱的幅度图

3.3.2 分类原理

在重力方向(即 X 轴)上,跑步与跳跃动作展现出显著的振幅变化,其平均加速度值显著高于其他类型的动作,而躺下动作则因需维持身体平衡而表现出较小的幅度变化,其平均加速度值则明显低于其他动作.因此,本文采用 X 轴加速度数据的平均值与标准差作为关键分类参数,以区分跑步/跳跃类高振幅动作与躺下类低振幅动作.具体而言,标准差较大者归类为跑步或跳跃动作,而平均值较小者则识别为躺下动作.

进一步地,动态与静态动作在加速度数据特征上存在显著差异,这一区别可通过偏度统计量有效反映.静态活动,如站立或乘坐电梯时,人体姿态相对稳定,无需频繁调整,因而加速度数据的偏度较小.相反,在动态活动中,如行走、爬楼梯等,身体姿态的连续变化导致加速度数据的偏度显著增加.

此外,不同活动模式下的加速度变化幅度各具特点.以持续运动为例,下楼梯时 X 轴方向上的瞬时加速度受重力加速影响显著增大,其峰值远高于其他动作.为减少极端值(奇异点)对峰值分析的干扰,本文引入峰值中值作为动作分类的新标准,以提高分类的鲁棒性和准确性.

最后,鉴于人体在空间中的活动是多维度的,且不同动作涉及的维度及参与程度各异,本文计划通过计算各维度加速度数据间的相关系数来量化这些差异.相关系数的应用有助于深入理解不同动作模式在多维空间中的表现特征,为更精细化的动作识别与分类提供理论支撑.

3.3.3 决策流程图

图 3 详尽展示了整个分类流程的逻辑框架,即分类决策流程图.该图作为可视化工具,直观呈现了分类过程中各个阶段的逻辑关系和决策路径.图中的判别参数是基于广泛且深入的文献回顾[4],以确保参数的科学性与合理性.

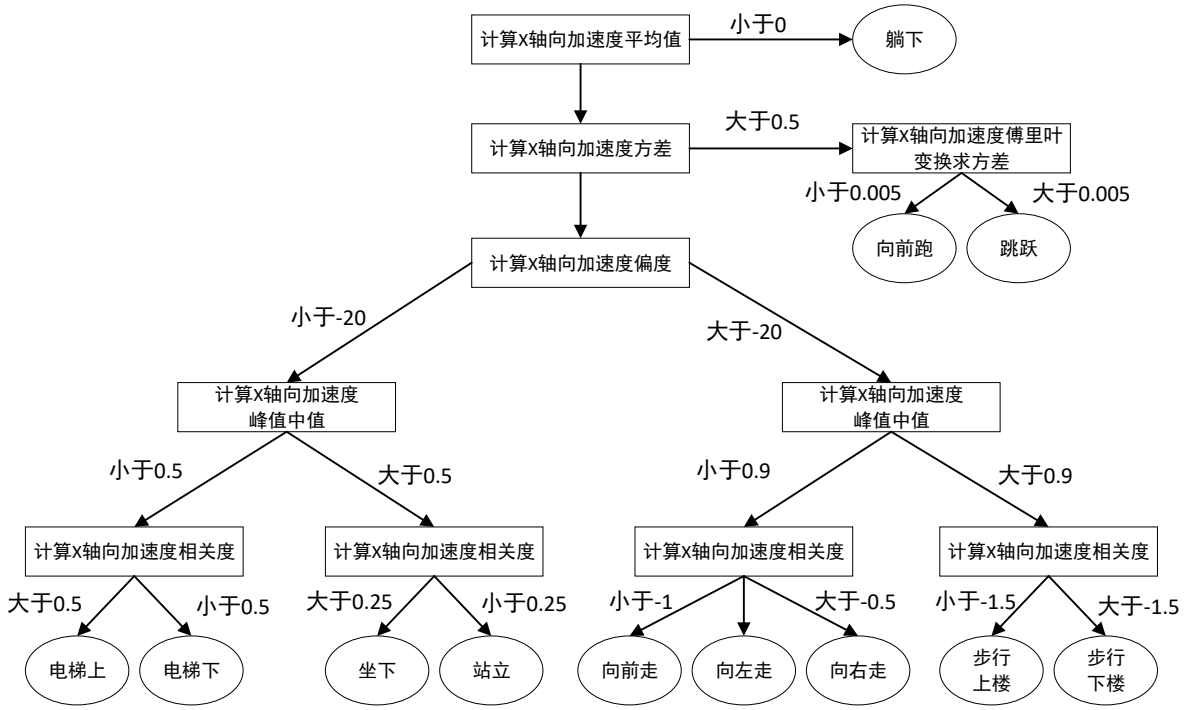


图3 分类的决策流程图

3.4 分类结果与分析

3.4.1 分类结果

基于第 3.3 节所构建的分类模型,本文对附件 1 中详尽记录的活动数据进行了系统性的分类处理.分类结果以表格形式详尽呈现于表 2 中,以便于后续的数据分析与性能评估.

表2 问题 1 分类结果

分类	Person1	Person2	Person3
第 1 类		SY23 SY24 SY57	
第 2 类			
第 3 类			SY19 SY60
第 4 类	SY25 SY27 SY28 SY29 SY31 SY33 SY37 SY38	SY10 SY12 SY16 SY19 SY2 SY20 SY21 SY26	SY11 SY13 SY14 SY23 SY24 SY25 SY27 SY29

	SY41 SY42 SY46 SY47 SY48 SY49 SY5 SY52 SY53 SY56 SY58 SY59 SY7 SY8	SY29 SY3 SY30 SY33 SY37 SY38 SY40 SY43 SY44 SY46 SY47 SY51 SY60 SY7	SY30 SY31 SY35 SY36 SY37 SY39 SY42 SY43 SY44 SY49 SY51 SY52 SY56 SY8 SY9
第 5 类			
第 6 类	SY19 SY26 SY43 SY44 SY50	SY1 SY4 SY48 SY49 SY50	SY10 SY41
第 7 类	SY1 SY15 SY2 SY23 SY36	SY13 SY27 SY28 SY34 SY42	SY18 SY22 SY4 SY5 SY53 SY57 SY58 SY59
第 8 类	SY11 SY13 SY16 SY18 SY3 SY45 SY55 SY57 SY9	SY11 SY14 SY17 SY18 SY22 SY25 SY36 SY41 SY45 SY5 SY53 SY54 SY55 SY8 SY9	SY1 SY15 SY16 SY2 SY20 SY21 SY26 SY28 SY33 SY34 SY38 SY40 SY45 SY46 SY6 SY7

第 9 类	SY10 SY12 SY20 SY24 SY30 SY39 SY4 SY51 SY54 SY6 SY60	SY15 SY35 SY56 SY58 SY6	SY12 SY17 SY47 SY48
第 10 类	SY22 SY32 SY34 SY35 SY40	SY31 SY32 SY39 SY52 SY59	SY3 SY32 SY50 SY54 SY55
第 11 类			
第 12 类			

3.4.2 模型性能分析

针对多样化的活动类型,分类模型展现出了差异化的分类效能,此现象可归因于数据本身的复杂性、特征提取的局限性,或者是实验员之间的差异性.具体分析如下:

(1)跑、跳、躺活动类型特征明显,分类结果合理

在此场景下,活动类型的特征集呈现出高度的区分性和适宜的数量,既未因冗余而复杂化模型,也未因不足而损失关键信息.分类任务相对简化,分类模型能够准确识别并归类.

(2)乘电梯、上下楼梯、向前走类型分类困难

当活动类型间的界限趋于模糊或特征间存在显著重叠时,分类任务的难度显著增加.此现象可能源于模型参数配置的非最优化,导致模型未能有效捕获区分各类别的关键特征;也可能是特征提取策略的限制,即所采用的方法未能充分提取出能够表征类别间差异的有效特征;此外,活动类型本身的高度相似性,以及不同个体在执行相同动作时的差异性,也是造成分类困难的重要因素.

(3)左走、右走、坐下、站立等动作,分类结果差

对于这一系列动作,分类结果表现出显著的误差,即动作常被错误归类至其他类型.此问题可能根源于特征提取过程中的技术缺陷,如噪声处理不当或特征选择失误;亦可能是模型泛化能力不足,对细微差异或噪声过于敏感,从而错误地将非关键信息视为区分类别的依据.

综上,根据已有数据和公开资料设计的分类模型能对部分活动类型进行分类,但仍存在局限性,可通过带标签的数据集进一步分析和改进.

4 问题二的分析与求解

4.1 问题分析

问题二要求建立并验证活动状态判别模型.此过程首要任务是设计并实现一个能够准确区分不同人体活动状态的模型框架.随后,需将这一判别模型与先前问题一中构建的

分类模型进行系统性比较,旨在深入分析两者在分类效能上的差异与优劣.最终,利用构建的活动状态判别模型对附件 3 中未标注的活动数据进行分类预测,以验证其实际应用效果.为此,本文采用了一种结合双通道输入、自注意力机制的卷积神经网络(CNN)与长短期记忆网络(LSTM)的混合网络模型 CNN-LSTM[8],旨在通过空间与时间特征的深度融合,提升人体活动行为的识别精度.

4.2 模型准备

4.2.1 数据预处理

为了确保数据的有效性和模型训练的准确性,必须对数据进行一系列预处理步骤,主要包括窗口划分、滤波和数据归一化等处理[9] .

(1) 窗口划分

在进行人体活动识别时,由于数据通常以长时间序列的形式存在,因此需要通过窗口划分技术将数据切分为多个更短的时间段(即窗口).时间窗口长度的选择很关键:过短的窗口可能无法完整捕捉一个活动周期(如行走的周期约为 0.48-0.63 秒),而过长的窗口则可能包含多种活动,影响识别的准确性.现有研究中,常用的时间窗口长度有 1 秒到 3 秒的短时间窗口和 10 秒及以上的长时间窗口.通过实验验证,窗口长度在 3 秒左右时通常能取得较好的识别效果.综合考虑本文数据集的采样频率(如 UCI 数据集的 50Hz),我们选择 2.56 秒作为滑动窗口的长度,并设置 50%的重叠率,即每个窗口包含 128 个采样点,且相邻窗口间有 64 个采样点重叠.

(2) 滤波

数据采集过程中,由于各种因素的影响,数据中不可避免地存在噪声.为了降低噪声对识别结果的影响,需要对数据进行滤波处理.常用的滤波方法包括滑动均值滤波、中值滤波和巴特沃斯低通滤波等.本文采用中值滤波方法,滤波窗口设置为 8,以有效抑制噪声,提高信号质量.

(3) 数据归一化

数据归一化是将数据特征转换到同一尺度上的过程,有助于提升模型的训练效率和预测性能.经过窗口划分、滤波等预处理步骤后,数据将被划分为训练集和测试集(比例为 0.8:0.2).本文采用最大最小值归一化方法,基于训练集数据的最大值和最小值进行计算,将数据缩放到[0,1]区间内.并将这一标准应用于训练集和测试集,以确保数据的一致性.具体公式如下:

$$\hat{x} = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{7}$$

其中, $\max(x)$ 和 $\min(x)$ 分别代表特征 x 在所有样本上的最大值和最小值.

4.2.2 评价标准

在人体活动识别系统的设计中,确立科学合理的算法性能评价指标是至关重要的.这些指标不仅直接关联到技术手段的选择,还深刻影响着识别任务的成效与优化方向.因此,本研究基于模式识别领域的标准实践,选取了准确率(Accuracy)、精确率(Precision)、召回率(Recall)以及 F1 值(F1-score)作为核心评价指标.

首先定义四个基本分类情况,它们是计算上述指标的基础:TP(True Positive):真正例,指实际为正类的样本被模型正确预测为正类.TN(True Negative):真负例,指实际为负类的

样本被模型正确预测为负类.FP(False Positive):假正例,指实际为负类的样本被模型错误地预测为正类.FN(False Negative):假负例,指实际为正类的样本被模型错误地预测为负类.

准确率(Accuracy)是衡量模型整体分类性能的最直观指标,它计算了所有样本中被正确分类的比例.然而,当数据集存在类别不平衡时,准确率可能无法全面反映模型性能,因此需要结合其他指标进行综合评价.公式如下:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

精确率(Precision)关注于模型对正类样本的识别能力,特别是在避免误报(FP)方面的表现.它计算了被模型预测为正类的样本中,真正为正类的比例.公式如下:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

召回率(Recall)表示模型能够正确识别出的正类样本占所有正类样本的比例,它关注于不漏报(即将正类误判为负类)的能力.公式如下:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

鉴于精确率与召回率之间往往存在权衡关系,F1 值作为两者的调和平均数,提供了一个更为综合且平衡的性能评估视角.F1 值越高,表明模型在精确率和召回率之间取得了更好的平衡,分类效果更佳.公式如下:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{11}$$

4.3 判别模型的构建

本文采用 CNN-LSTM 框架,一种深度神经网络(DNN)架构,来优化人体活动状态的判别.该框架融合了卷积神经网络(CNN)在特征提取方面的优势与长短期记忆网络(LSTM)在时间序列建模上的长处,从而实现对复杂时间序列数据的高效处理与精准分类,其网络架构如图 4 所示.

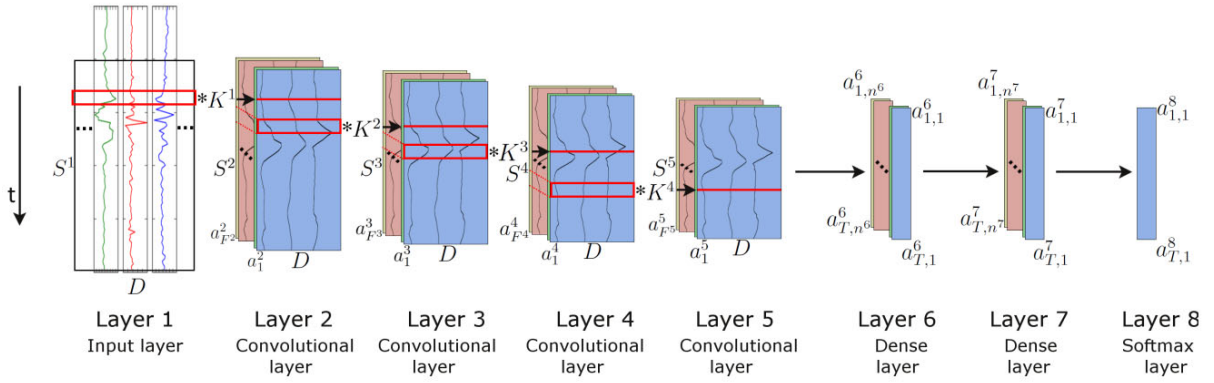


图4 CNN-LSTM 框架的体系结构

CNN-LSTM 模型遵循高度结构化的设计原则,主要包括四个卷积层、两个 LSTM 层以及一个 softmax 层.这四个卷积层负责从原始传感器数据中提取关键特征,形成特征图;两个 LSTM 层则进一步整合这些特征,捕捉时间维度上的动态变化与长期依赖关系;最后,softmax 层输出每个时间步的类别概率分布,实现分类.

4.3.1 CNN 组件层

在深度学习方法中,无论是使用循环神经网络还是前馈神经网络作为模型,原始传感器信号中会包含更丰富的信息[10] .有研究利用专家知识对原始的传感器信号进行特征提取导致了网络对于特征空间的系统探索被限制[11] .卷积神经网络(CNN)被提出来用于解决这一问题,其中单层神经单元通过滤波器对原始的传感器信号进行卷积运算,从输入信号中充分提取特征.在 CNN 中,激活神经元输出被用于表示滤波器与输入信号卷积计算的结果.不管是何种类型的数据模式,通过卷积计算同一输入的不同区域上激活神经元输出,CNN 都可以检测滤波器捕获的数据模式.在 CNN 的训练中,滤波器作为监督训练过程的一部分被优化,以每种类型滤波器的激活水平.特征映射是一组共享相同的参量化权重矢量和偏差的神经元,它们的激活产生了滤波器在整个输入数据中进行卷积计算的结果.

卷积算子的应用依赖于输入维数.对于 2D 图像(例如视频)的时间序列,通常在 2D 空间卷积中使用 2D 内核.对于一维时间序列(例如,传感器信号) ,通常在时间卷积中使用一维内核.在一维域中,内核可以被视为一个过滤器,能够去除离群值、过滤数据或者充当特征检测器,可以在内核的时间跨度内对特定的时间序列捕获原始传感信号的特征.形式上,使用一维卷积运算提取特征的方法是:

$$a_j^{(l+1)}(\tau) = \sigma\left(b_j^l + \sum_{f=1}^{F^l}\left[\sum_{p=1}^{P^l} K_{jf}^l(p)\, a_f^l(\tau - p)\right]\right) \tag{12}$$

其中l层的 $a_j^{(l)}(\tau)$ 表示特征映射j,σ是非线性函数,F^l 是l层的特征映射个数,K_{jf}^l是在l层的特征映射 f上卷积的核,在l+1 层上生成特征映射 j,P^l是l层的核长度,b^l 是偏移向量.在处理传感器数据时,该计算分别应用于输入端的每个传感器通道,如图 5 所示.

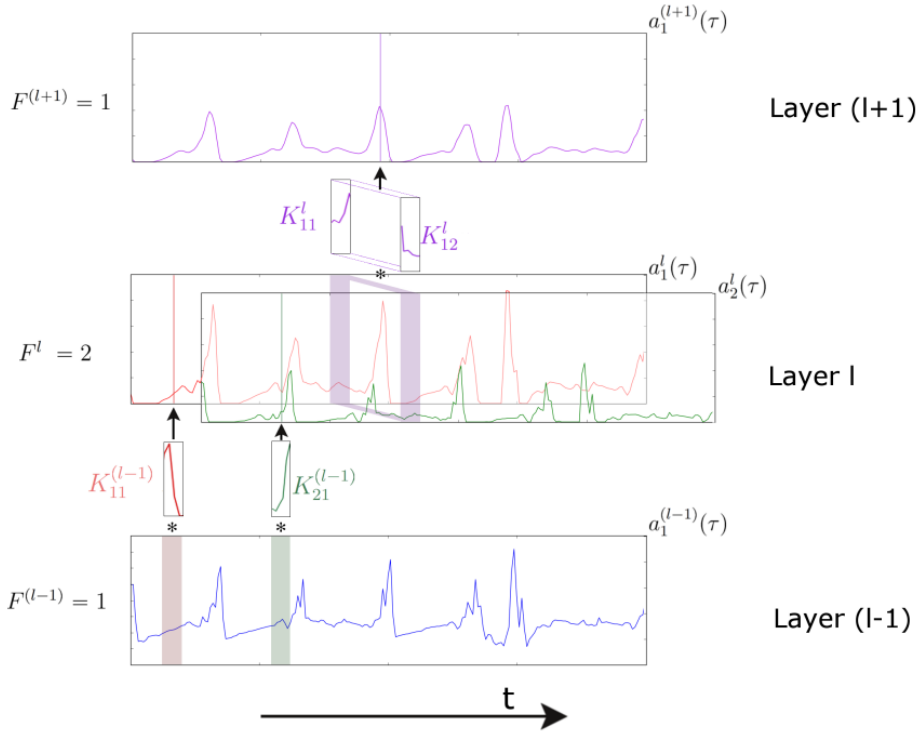


图5 CNN 层表示

滤波器的权重将被作为一个特征检测器来捕获一类数据的特定模式.一个具有多个卷积层的模型,在一个叠加的配置中,层 $l-1$的输出是下一层 l的输入,这样做的目的是使网络能够学习数据的层次化表示.在这个层次中,更深的层逐渐以更抽象的方式表示输

入.深层次的 CNN 已经在诸如内容推荐、语音识别和计算机视觉等领域产生了重大影响,在这些领域,它们已经成为行业标准.

CNN 组件层作为模型的空间特征提取核心,其设计旨在从加速度和角速度数据中高效提取关键的空间信息,其结构如图 6 所示.通过采用 1D-CNN 层、BN 层及 Dropout 层的组合配置,前两个 CNN 组件层有效减少了过拟合风险并加速了训练过程.而第三个 CNN 组件层则专注于特征融合,仅包含 1D-CNN 层与 BN 层,确保了提取的空间特征能够无缝传递给后续的时间特征提取模块.此外,通过层标准化对特征值进行归一化处理,提高了网络的敏感度和训练速度,同时减少了网络对初始化的依赖.Dropout 层的应用则进一步减少了冗余特征,增强了模型的泛化能力.

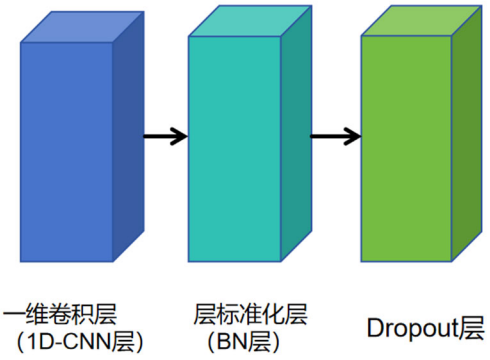


图6 CNN 组件层

4.3.2 LSTM 组件层

LSTM 是深度学习中循环神经网络的一种,专门用于捕获复杂的时序特征以针对时间序列相关数据进行建模.传统的循环神经网络通过一个记忆单元和一个延时单元来反馈梯度给神经元,这为其提供了来自过去的记忆信息以学习动态的时序逻辑.相比于传统的循环神经网络,LSTM 通过记忆单元来存储和输出信息,简化了对于长时间尺度的时序关系学习问题.基于门控机制,LSTM 通过组件式乘法定义了每个独立神经元的行为,并且通过门的激活来更新神经元状态.LSTM 通过将数据送到输入门、输出门和重置门中来控制记忆单元的具体操作.LSTM 记忆单元的输出向量h_t在时间t被更新时,其所有单元的矢量更新如下式:

$$\boldsymbol{f_t} = \sigma_f\left(W_{af}\boldsymbol{a_t} + W_{hf}\boldsymbol{h_{t-1}} + W_{cf}\boldsymbol{c_{t-1}} + \boldsymbol{b_f}\right) \tag{13}$$

$$\boldsymbol{c_t} = \boldsymbol{f_t}\boldsymbol{c_{t-1}} + \boldsymbol{i_t}\sigma_c(W_{ac}\boldsymbol{a_t} + W_{hc}\boldsymbol{h_{t-1}} + \boldsymbol{b_c}) \tag{14}$$

$$o_t = \sigma_o(W_{ao}a_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{15}$$

$$\mathrm{h_t} = o_t\sigma_\mathrm{h}(\mathrm{c_t}) \tag{16}$$

其中$\boldsymbol{i}$、$\boldsymbol{f}$、$\boldsymbol{o}$和$\boldsymbol{c}$分别是输入门、重置门、输出门和记忆激活单元,它们的大小与定义隐藏值的向量 $\boldsymbol{h}$ 的大小相同.函数 σ 表示非线性函数.$\boldsymbol{a_t}$ 矢量是时间 t 时刻记忆单元的输入,$W_{hi}, W_{ci}, W_{af}, W_{hf}, W_{cf}, W_{ac}, W_{hc}, W_{ao}, W_{ho}$ 和 W_{co} 为权重矩阵,下标表示权重矩阵与具体矢量之间的关系.$\boldsymbol{b_i}, \boldsymbol{b_f}, \boldsymbol{b_c}$ 和 $\boldsymbol{b_o}$ 是偏移向量.由于人体活动识别问题要基于大量的时序逻辑数据挖掘其中的动态特征,LSTM 方法在处理这类问题上具备优势.LSTM 的结构如图 7 所示.

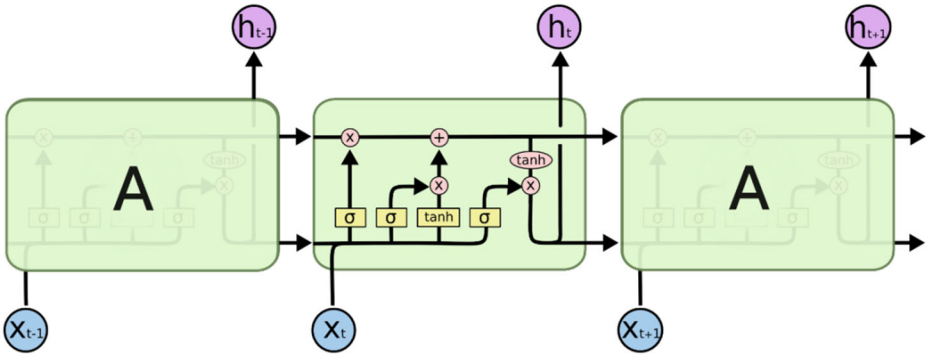


图7 LSTM 的结构示意图

4.3.3 CNN-LSTM 混合网络模型

CNN-LSTM 混合网络模型以加速度计和陀螺仪数据为输入,通过多层 CNN 结构逐层提取空间特征,随后利用 LSTM 捕捉时间依赖性,并引入自注意力机制强化关键时间点的特征表示,最终通过 Softmax 函数输出活动分类结果.

模型首先采用多层卷积神经网络(CNN)架构,通过逐层卷积操作,依据预先定义的数学框架(如公式(12)所示),对输入数据进行空间特征的深度提取.此卷积过程聚焦于时间维度,旨在识别并强化时间序列中的关键特征元素.为确保数据的连贯性与完整性,网络设计严格维持了各层间传感器通道数与特征图的一致对应关系.

紧接着,经过卷积层处理的数据被进一步传递至由长短期记忆网络(LSTM)层构成的递归密集层(具体为第 6、7 层).此设计策略基于处理具有显著连续性和长期依赖性数据的需要,通过至少两层深度的 LSTM 递归层,显著增强了模型对复杂时间序列动态变化的建模能力.在每个时间步,LSTM 层更新其内部状态,有效捕捉并整合时间序列中的长期依赖信息.特别地,第 6 层的输入集成了来自第 5 层在当前时间步及其前一时间步的特征图信息,这一设计进一步丰富了模型对时间序列上下文的理解.

LSTM 单元的激活函数采用双曲正切函数,其非线性特性有助于模型更好地适应和模拟时间序列中的复杂动态变化.最终,模型的输出由一个应用 softmax 激活函数的密集层生成,该层负责在每个时间步输出活动的类别概率分布,从而实现对人体活动数据的精确分类.

4.3.4 模型的训练与优化

本文所构建的神经网络基于 Python 框架并利用 Keras 库实现.Keras 是一个由 Python 编写的开源人工智能神经网络库,可作为 Tensorflow、Microsoft-CNTK 和 Theano 的高级应用程序接口,被用于进行深度学习模型的设计、调试、评估、应用和可视化.模型训练采用全监督学习范式,其中梯度通过 softmax 层反向传播至卷积层,以优化网络参数.优化过程具体采用了自适应运动优化估计算法(Adaptive Moment Estimation, Adam)对网络进行更新.Adam 结合动量和根平方传播的思想通过自适应地调整每个参数的学习率来最小化预测基于交叉熵的损失.DNN 中的参数数量高度依赖于其架构的复杂性,特别是卷积层(CNN)与长短期记忆层(LSTM)的组合,这些参数直接提升了训练过程的计算资源需求与耗时.本文所采用的 CNN-LSTM 网络架构的具体参数配置详见表 3,其中详细列出了各层的参数规模与层维度,特别指出分类任务的类别数对最终层设计的影响.本文使用的 CNN-LSTM 网络参数如表所示,其中 n_c 表示分类任务中的类别数量.

表3 CNN-LSTM 网络参数

层序号	CNN-LSTM	
	参数说明	参数数量
2	K:64×5 b:64	384
3-5	K:$64\times64\times5$ b:64	20544
6	$W_{ai},W_{af},W_{ac},W_{ao}$:$7232\times128$ $W_{hi},W_{hf},W_{hc},W_{ho}$:$128\times128$ b_i,b_f,b_c,b_o:128 W_{ci},W_{cf},W_{co}:128 c:128 h:128	942592
7	$W_{ai},W_{af},W_{ac},W_{ao}$:$128\times128$ $W_{hi},W_{hf},W_{hc},W_{ho}$:$128\times128$ b_i,b_f,b_c,b_o:128 W_{ci},W_{cf},W_{co}:128 c:128 h:128	33280
8	W:$128\times n_c$ b:n_c	$(128\times n_c)+n_c$
总计		$996800+(128\times n_c)+n_c$

为提高训练效率与泛化能力,训练数据集被划分为包含 100 个样本的迷你批次进行迭代处理.每个批次结束后,即计算并累积该批次内所有样本对参数的梯度贡献.训练过程中,学习率设置为$10e^{-3}$,衰减系数设置为$\rho=0.9$,以控制学习进程,同时权重采用随机正交初始化方法以促进训练的稳定性.

此外,为防止过拟合并提升模型泛化能力,在所有密集层之前引入了 drop-out 正则化策略,该策略以 0.5 的概率随机丢弃部分神经元的激活输出,从而有效减少模型复杂度并提高其在未见数据上的预测准确性.

4.4 基于 CNN-LSTM 模型的判别算法

CNN-LSTM 是一种基于卷积神经网络和循环神经网络的深度学习判别模型,可通过捕获加速度计和陀螺仪数据中的时序逻辑特征快速实现人体活动的识别.首先,该算法要依托于大量的数据进行判别模型的训练,使模型能够充分学习数据中的动作时序特征.进一步的,判别模型通过 5 折交叉验证的方式进行评估.最后,评估后的判别模型可被用于测试数据的识别.

算法输入包括:用于训练的加速度计和陀螺仪数据特征矩阵A,训练数据的人体活动标签Y, 总的迭代轮次为 L,用于测试的加速度计和陀螺仪数据特征矩阵T,具体流程如下:

Step 1 将迭代次数初始化$l = 0$; 将用于训练的加速度计和陀螺仪数据特征矩阵A通过数据预处理流程归一化为A_0;

Step 2 迭代次数$l = l + 1$;

Step 3 将A_0输入到构建好的 CNN-LSTM 模型中充分提取特征,并输出预测标签$\hat{Y}$;

Step 4 通过损失$-\sum_{i=1}^{|Y|} y_i \cdot \log \hat{y_i}$来更新模型参数,若$l < L$,返回 Step2,否则结束循环;最终将评估效果最好的模型 G 作为最终的判别模型;

Step 5 将用于测试的加速度计和陀螺仪数据特征矩阵T通过数据预处理流程归一化为T_0;

Step 6 将T_0输入到评估训练好的模型 G 中,并输出预测标签$\hat{p} = G(T_0)$;

附件二的数据被用于模型训练中,其中 80%数据用于划分 5 折交叉验证的训练集和验证集,20%的数据用作测试集.其 5 次训练模型的结果如图 8-13 所示.

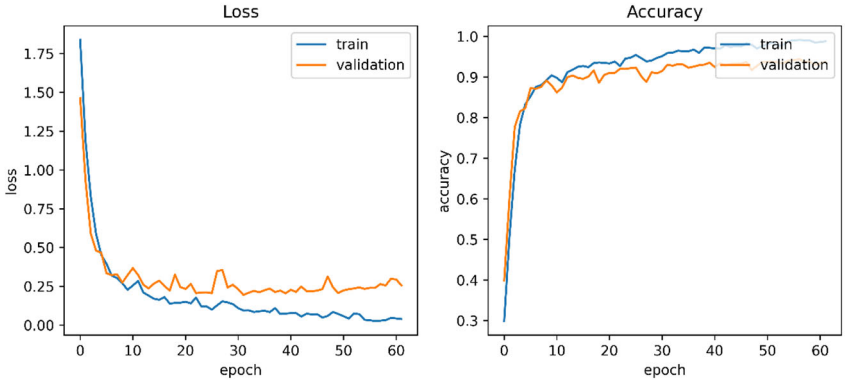


图8 第一次训练结果

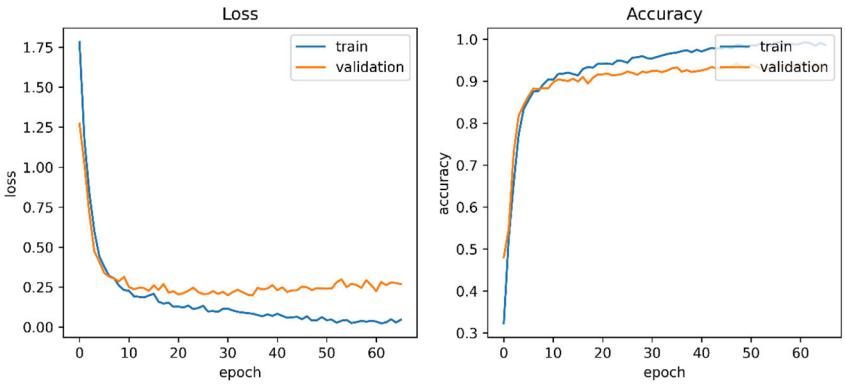


图9 第二次训练结果

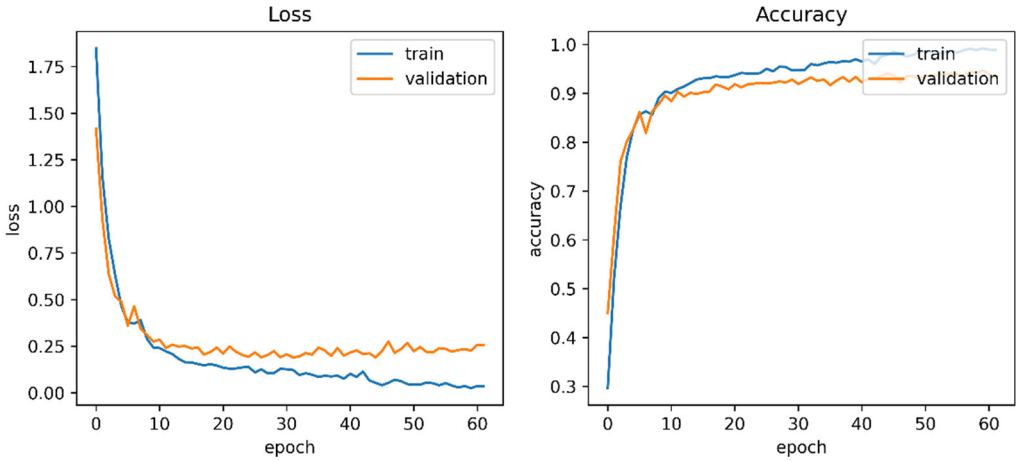


图10 第三次训练结果

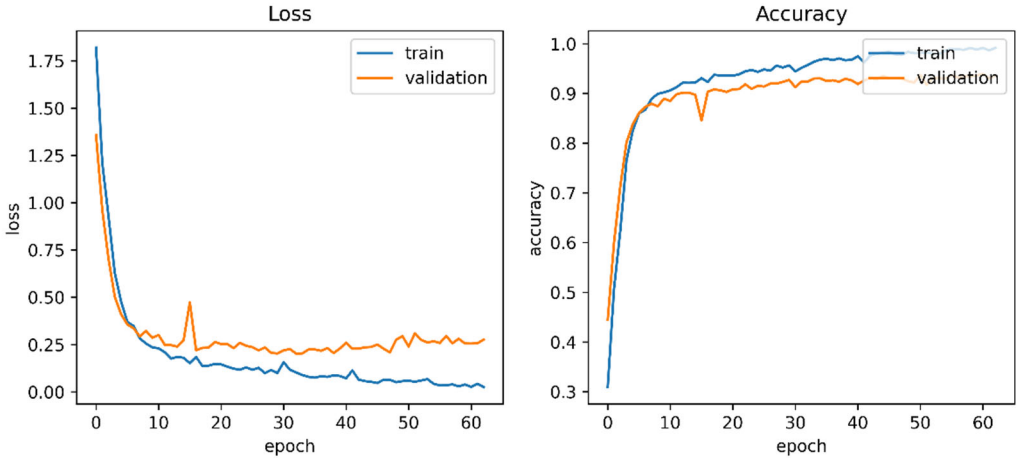


图11 第四次训练结果

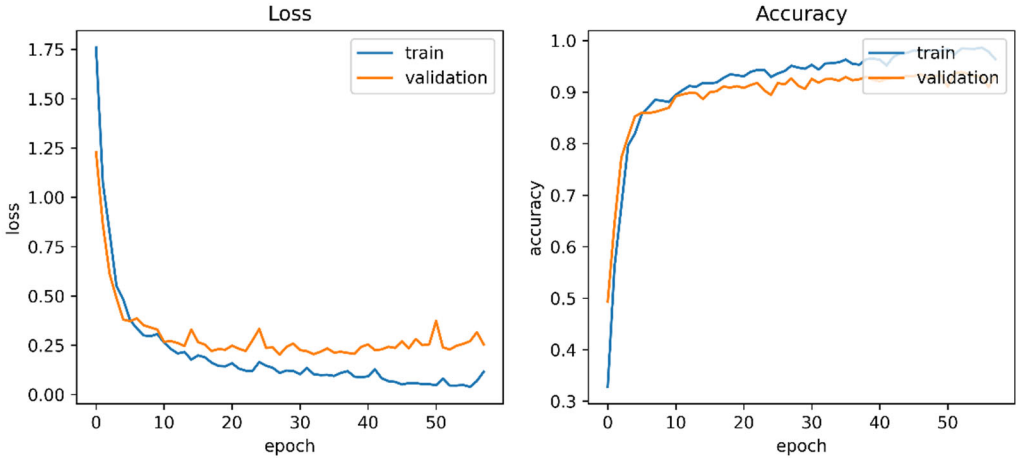


图12 第五次训练结果

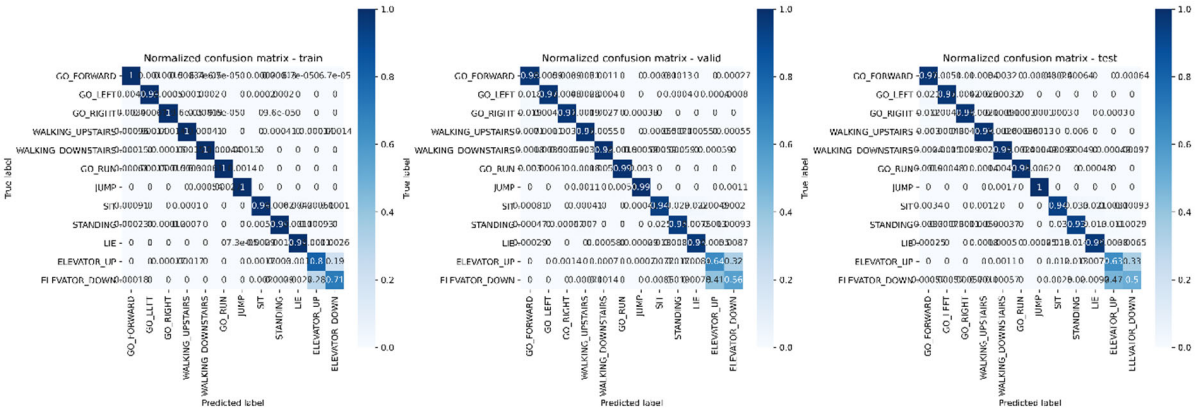


图13 五次训练的模型在训练、验证、测试集上的均值分类混淆矩阵

表4 5 次交叉验证结果的均值表

评估指标	训练集	验证集	测试集
Accuracy	0.9682	0.9279	0.9236
Precision	0.9567	0.9096	0.9032
Recall	0.9550	0.9088	0.9030
F1	0.9546	0.9080	0.9017

通过表 4,CNN-LSTM 模型在 5 次交叉验证中,验证集和测试集上的平均预测率都达到了 92%,平均 Recall 和 F1 分数都达到了 90%,说明了模型具备良好的预测效果.然而,从混淆矩阵中可以看出,模型在电梯向上和电梯向下两个类别中的测试集和验证集上的预测效果不佳,两种类别容易混淆.

4.5 判别模型的应用

针对问题二的第(1)题:进一步运用问题 1 的分类模型对该 10 名实验人员数据进行分类.本文中定义的正确率为预测该类正确的占该类总的比重,准确度为正确预测为正的占全部预测为正的比例.正确率是一个直观的评价指标,能反映出数据中预测正确的比例.然而,由于数据分布不均衡,错分其他类别情况仅通过正确率无法反映,准确度可以反映出对某类辨别时的准确情况.分类详细结果参见支撑材料,分类统计结果如图 14、图 15 所示.

相比于问题 2 中的判别模型,分类模型的整体表现明显弱于判别模型,不同活动类型的分类结果差异大,判别模型针对多种动作类型都具备较高的准确度.具体分析结果如下:

(1)分类模型针对动作 1、动作 2、动作 3、动作 4、动作 12,分类模型未能做出判断;

(2)分类模型对动作 4 的分类正确率高,但准确率低,说明很多动作类型被误划分到动作 4;

(3)分类模型对动作 6、动作 7 的分类正确率高,同时准确率也高,说明分类模型可以对动作 6 与动作 7 进行准确分类;

(4)分类模型对动作 10 的分类正确率较高,同时准确率也较高,说明分类模型可以对动作 10 进行分类,分类结果对判断是否为动作 10 有参考价值;

(5)分类模型对动作 8、动作 9、动作 11、动作 12 的分类正确率低,同时准确率也低,说明分类模型可以对动作 8、动作 9、动作 11、动作 12 无法做出正确分类,分类结果对判断动作类型无参考价值.

(6)判别模型对动作 1—动作 10 都具有很强的分类能力,能够以较高的分类准确度完成分类任务.然而,对于动作 11 和动作 12,无法做出较为准确的判断.

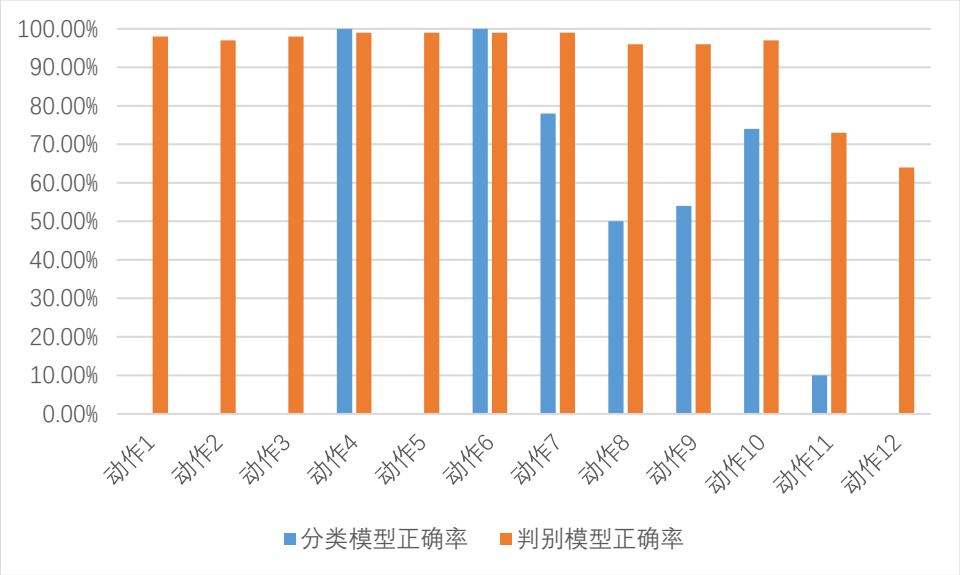


图14 分类模型与判别分类结果正确率

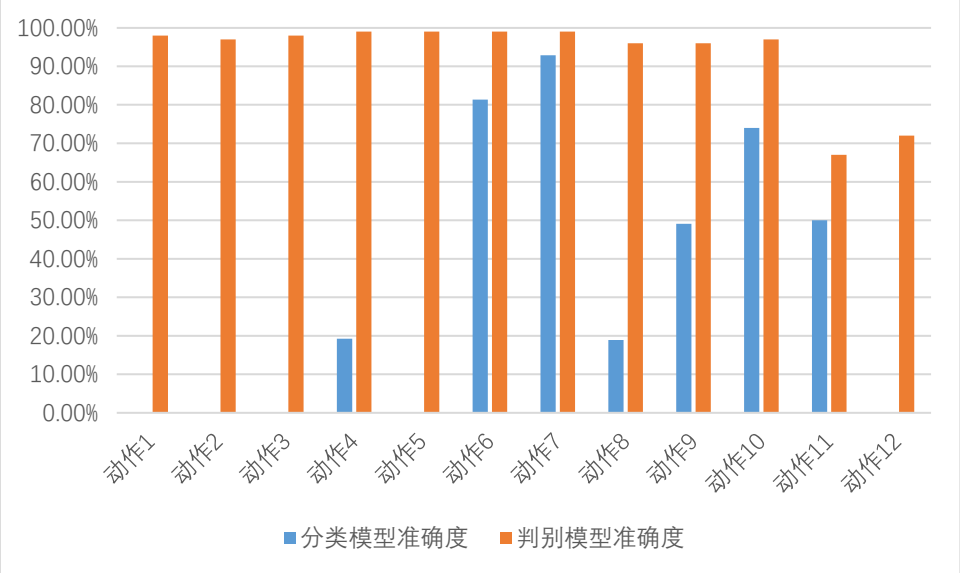


图15 分类模型与判别模型分类结果准确度

针对问题二的第(2)题:附件 3 中的状态数据被放入训练好的判别模型中,结果如表 5 所示:

表5 附件 3 中状态数据的判别结果

活动类型	判别状态
SY1	第 5 类步行下楼
SY2	第 1 类向前走
SY3	第 7 类跳跃
SY4	第 11 类乘坐电梯向上移动
SY5	第 7 类跳跃
SY6	第 10 类躺下
SY7	第 2 类向左走
SY8	第 6 类向前跑
SY9	第 7 类跳跃
SY10	第 10 类躺下
SY11	第 9 类站立
SY12	第 7 类跳跃
SY13	第 4 类步行上楼
SY14	第 3 类向右走
SY15	第 4 类步行上楼
SY16	第 1 类向前走
SY17	第 4 类步行上楼
SY18	第 5 类步行下楼
SY19	第 8 类坐下
SY20	第 8 类坐下
SY21	第 6 类向前跑
SY22	第 2 类向左走
SY23	第 8 类坐下
SY24	第 5 类步行下楼
SY25	第 2 类向左走
SY26	第 9 类站立
SY27	第 8 类坐下
SY28	第 5 类步行下楼
SY29	第 6 类向前跑
SY30	第 5 类步行下楼

5 问题三的分析与求解

5.1 问题分析

问题三要求分析活动状态与实验人员年龄、身高、体重等数据的关系,并通过活动传感器数据对人员特征进行画像.此过程首要任务是通过大量数据的特征判断活动状态数据与实验人员特征有无关系.随后,需要通过人员的年龄、身高、体重构建人员特征的判别模型进行训练,以得到从传感器数据到实验人员特征的映射函数.最终,利用构建的人员特征判别模型对附件 5 中实验人员进行画像,以判断其具体身份.为此,本研究基于问题二的 CNN-LSTM 网络构建了用于判断实验人员身份的深度网络模型.

5.2 数据集与数据处理

本研究仍然需要通过 CNN-LSTM 网络的训练以得到能够用于判断传感器数据与实验人员年龄、身高、体重特征关系的网络模型.为了更好的捕获数据中动态动作的时序特征,判别模型仍然通过与问题二中一样的预处理方式进行预处理.

由于年龄、身高、体重都属于典型的连续变量,它们可以取到实数范围内的任意一个值.在判别模型的构建中,我们不能对所有的具体值都构建一个标签,这将会使深度学习网络面临维度灾难问题.因此,我们对年龄、身高、体重标签进行分段处理的操作,将训练数据中的最大值与最小值作为上下界,取其中每 5 个值为一个区间,给每一个区间定义其对应的标签,具体处理过程如图 16 所示.

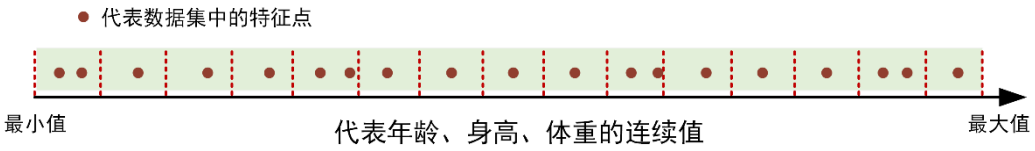


图16 对连续变量进行分段处理

5.3 联合判别模型建立

首先,为了构建合理的判别模型,本研究需要从大量的动态时序数据中提取特征,沿用了问题二中的 CNN-LSTM 网络进行判别模型的训练.然而,不同的人员特征与时序数据的关联是不同的,因此本研究分别训练三种基于不同特征标签的判别模型.进一步的,对年龄、身高、体重三种特征的判别模型进行联合判断,只需要判断出每种类型特征的大致数值区间即可推测出实验人员的具体身份.具体流程如下:

算法输入包括:用于训练的加速度计和陀螺仪数据特征矩阵A,训练数据所使用的分别表示年龄、身高、体重的人体特征标签Y_1、Y_2、Y_3,总的迭代轮次为 L,需要画像的加速度计和陀螺仪数据特征矩阵T,具体流程如下:

Step 1 将迭代次数初始化$l = 0$;将用于训练的加速度计和陀螺仪数据特征矩阵A通过数据预处理流程归一化为A_0;

Step 2 将A_0输入到分别构建好的三个判别模型中充分提取特征,并分别输出预测标签$\hat{Y}_1$、$\hat{Y}_2$、$\hat{Y}_3$;

Step 3 通过损失$-\sum_{i=1}^{|Y|} y_i \cdot \log \hat{y}_i$来分别更新三个模型的训练参数,若$l < L$,返回 Step 2,否则结束循环;最终将评估效果最好的模型G_1、G_2、G_3分别作为最终年龄、身高、体重的判别模型;

Step 4 将需要画像的加速度计和陀螺仪数据特征矩阵T通过数据预处理流程归一化为T_0;

Step 5 将T_0输入到评估训练好的模型G_1、G_2、G_3中,并输出预测标签$\hat{p}_1 = G_{1(T_0)}$、$\hat{p}_2 = G_{2(T_0)}$、$\hat{p}_3 = G_{3(T_0)}$;

Step 6 从$\hat{p}_1$中推断出画像人员的年龄区间,从$\hat{p}_2$中推断出画像人员的身高区间,从$\hat{p}_3$中推断出画像人员的体重区间;根据信息综合判断并得出结论;

具体框架图如图 17 所示.

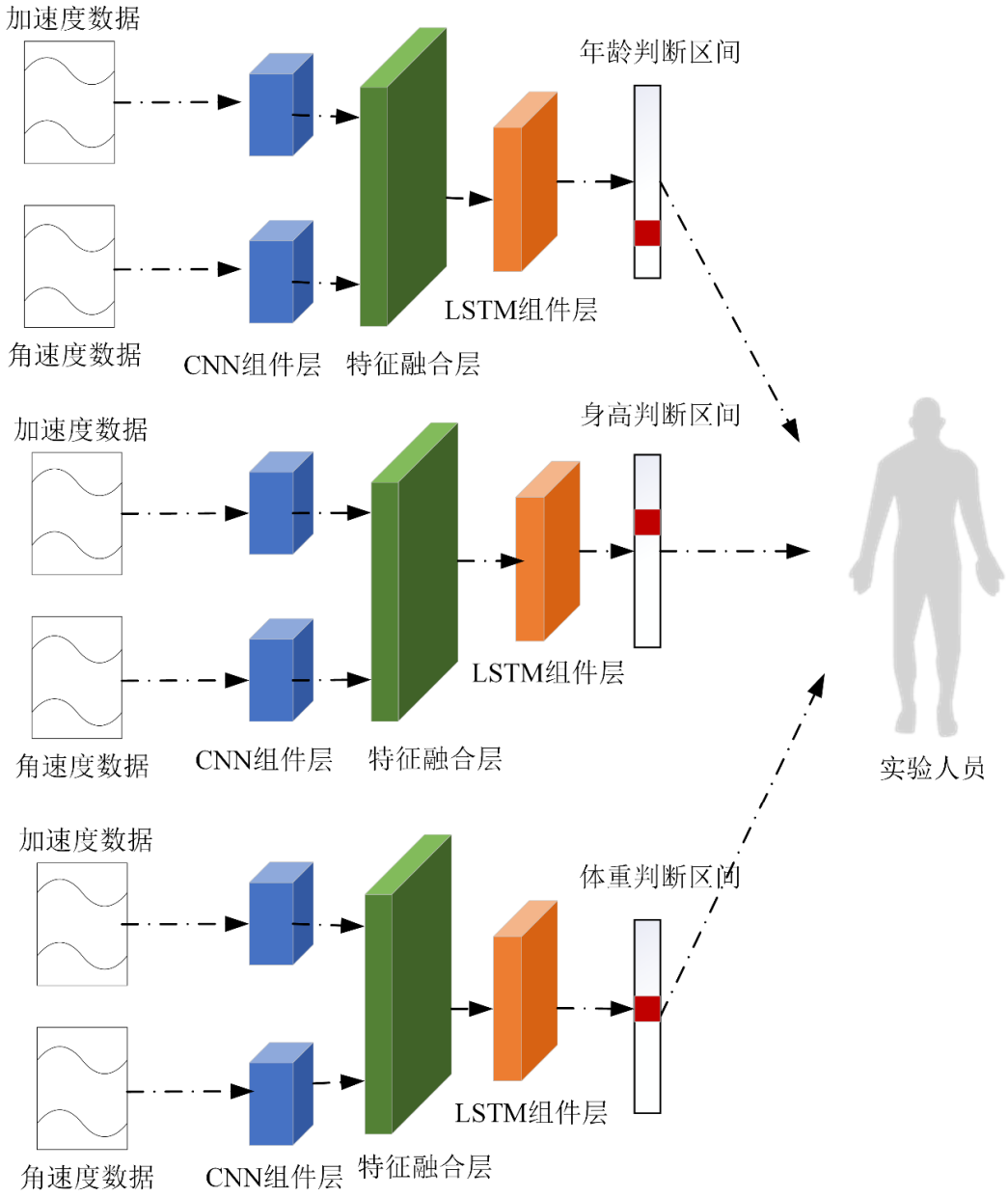


图17 联合判别模型架构

本研究针对三种判别模型统一通过问题二提供的训练方式进行训练.其中,年龄数据被划分为“[20,25)”、“[25,30)”、“[30,35)”、“[35,40)”、“[40,45)”和“[45,50)”6 个标签.身高数据被划分为“[160,165)”、“[165,170)”、“[170,175)”、“[175,180)”、“[180,185)”和“[185,190)”6 个标签.体重数据被划分为“[40,45)”、“[45,50)”、“[50,55)”、“[55,60)”、“[60,65)”、“[65,70)”、“[70,75)”、“[75,80)”和“[80,85)”9 个标签.训练结果如图 18-20 所示.

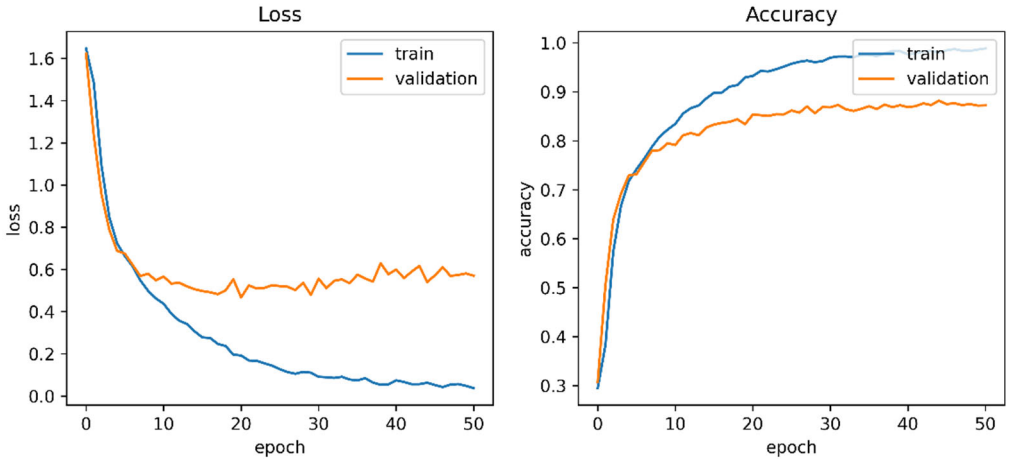


图18 针对年龄特征的训练结果

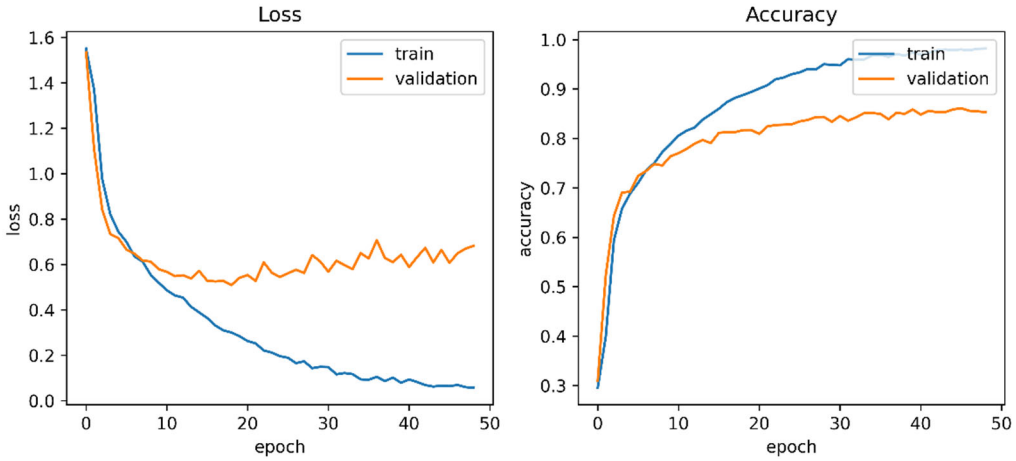


图19 针对身高特征的训练结果

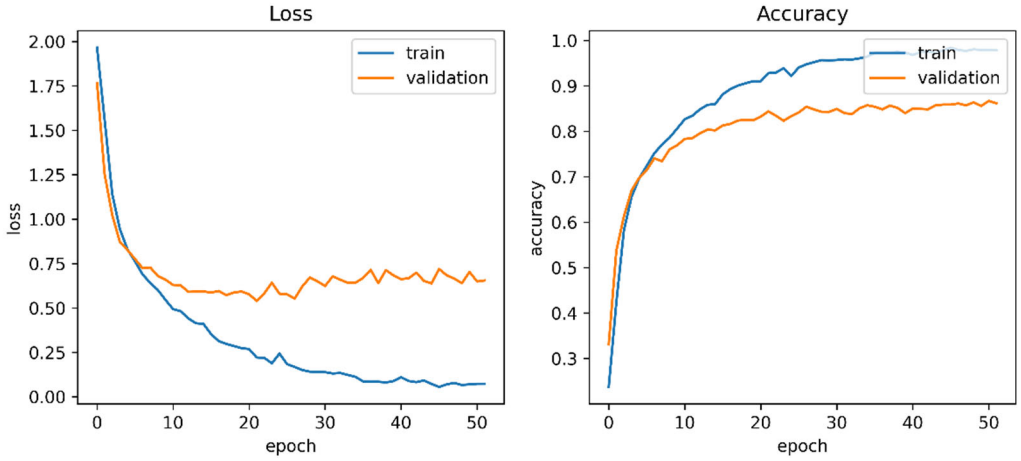


图20 针对体重特征的训练结果

可以看到,CNN-LSTM 模型在以年龄、身高、体重特征为标签训练判别模型时,验证集上的预测率都达到了 80%以上,说明了模型具备较好的判别特征能力,可以通过三种特征的判断识别人的身份.

5.4 联合判别模型应用

针对问题三第一问:问题三要求分析活动状态与个人特征的关系,并进行人员画像和识别.附件 4 给出了问题 1 和问题 2 中参与实验的 13 位实验人员的年龄、身高、体重等数据,根据附件数据,实验人员 7 与实验人员 8 身高体重相仿,年龄相差 5 岁.观察其在跳跃动作上的加速度 X 轴向表现,信号时序图如图 21 所示;实验员 4 和实验员 6 的年龄、身高一致,体重相差 7kg,观察其在跳跃动作上的加速度 X 轴向表现,信号时序图如图 22 所示;实验员 4 和实验员 5 的年龄、体重 x 相仿,身高相差 4cm,观察其在跳跃动作上的加速度 X 轴向表现,信号时序图如图 23 所示.通过整体观察可知,实验人员的在跳跃动作上频率与峰值都有所不同,因此不同人员的同一活动状态存在差异.

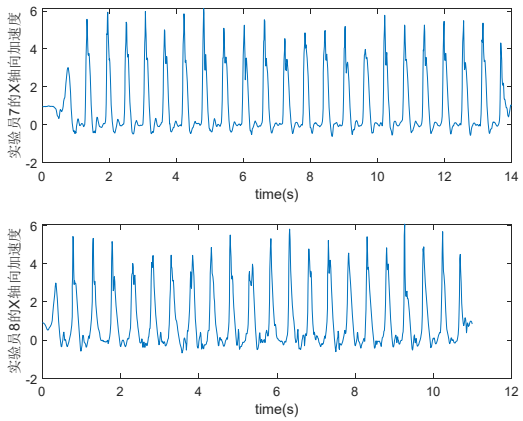


图21 实验员 7 与实验员 8 的跳跃动作时序图

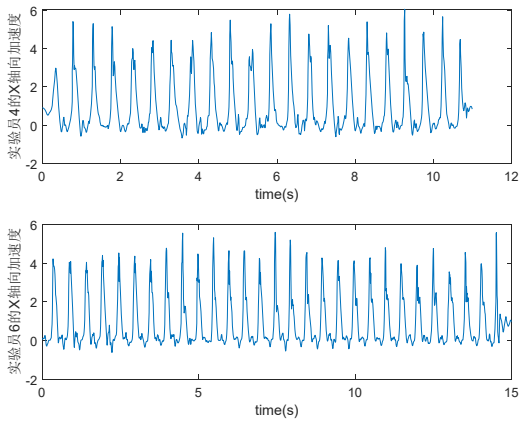


图22 实验员 4 与实验员 6 的跳跃动作时序图

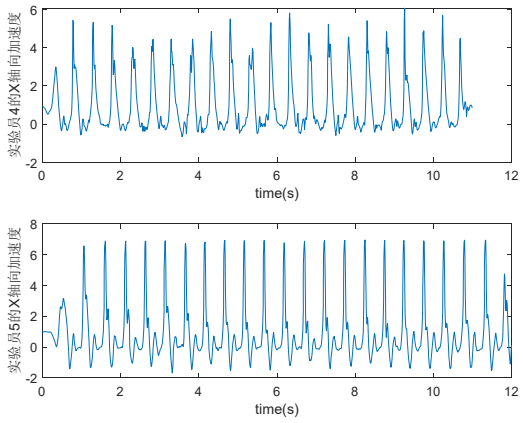


图23 实验员 4 与实验员 5 的跳跃动作时序图

针对问题三第二问:本研究通过训练出的三种判别模型对附件 5 中的五位活动人员的实验数据识别特征并判断其源于问题 2 中的哪五位人员,判别结果如表 6 所示.

表6 附件 5 中状态数据的判别结果

实验人员	判断年龄区间	判断身高区间	判断体重区间	判别结果
Unknow 1	[45,50)	[165,170)	[65,70)	Person 10
Unknow 2	[35,40)	[170,175)	[60,65)	Person 7
Unknow 3	[25,30)	[160,165)	[50,55)	Person 6
Unknow 4	[20,25)	[180,185)	[75,80)	Person 9
Unknow 5	[35,40)	[170,175)	[80,85)	Person 13

6 模型评价

6.1 模型的优点

(1) 基于阈值法的分类模型具有轻量化、分类速度快优点.决策树形式的分类模型采用的 IF-THEN 的分类逻辑,可以对数据进行快速决断,得出分类结果.
(2) 基于深度学习的模型具有强大的拟合能力,可以逼近几乎所有复杂的函数,无论这种函数是线性还是非线性的,这使其在处理复杂问题时表现出色.
(3) 深度学习模型具有自动学习特征的能力,可以在原始数据中提取有用特征,而不需要人工进行特征工程,在处理复杂数据时更加灵活和有效.

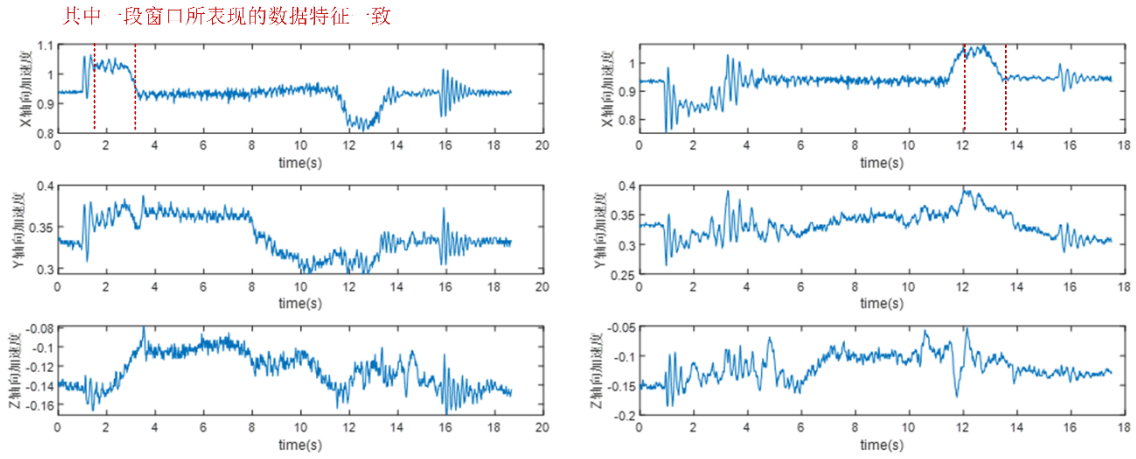
6.2 模型的不足

(1) 基于阈值法的分类模型参数设计困难,需要大量经验,无法在短时间确定一个较为合适的参数,并且无法保证模型参数的通用性.
(2) 基于阈值法的分类模型结构简单,对复杂特征的区分能力较弱,只能区分特征明显的活动类型.
(3) 深度学习模型的性能高度依赖于训练数据的质量和数量.如果训练数据存在偏差、噪声或不足等问题,那么模型的预测结果也可能受到影响.
(4) 在样本量相对较少的情境下,深度学习模型可能无法充分捕捉并学习到足以区分各类别的关键特征,难以有效区分不同类别间细微差异.

6.3 模型的推广

针对问题二中训练的判别模型,本研究发现算法始终难以区分“乘坐电梯向上移动”和“乘坐电梯向下移动”两种类别.从图 13 的混淆矩阵中可以看出, 其他类别与这两类相比较为好区分,本研究将从数据预处理中滑动窗口的角度分析这一问题.

由于加速度和角速度数据通常以长时序的形式存在,因此需要通过窗口划分技术将数据切分为多个更短的时间段(即窗口),本研究是选择 2.56 秒作为滑动窗口的长度,并设置 50%的重叠率,即每个窗口包含 128 个采样点,且相邻窗口间有 64 个采样点重叠.然而,在“乘坐电梯向上移动”和“乘坐电梯向下移动”两种类别中,2.56 秒的时隙间隔可能无法区分两者之间的差别.



(a) “电梯向上”动作中时序与加速度之间的关系　　(b) “电梯向下”动作中时序与加速度之间的关系

图24 “乘坐电梯向上移动”和“乘坐电梯向下移动”两种活动类别中角速度与时序的关系

从图 24 可以看出,通过 2.56 秒的时间窗口对数据进行划分时,两种活动类型所对应的 X 轴加速度时序特征几乎完全一致,模型可能无法依据这一特征对两种类别的数据进行区分.因此,在数据预处理的滑动窗口方面,算法可以适当提高滑动窗口的长度,从而解决这两类活动类型数据难以区分的问题.

参考文献

[1] 徐川龙, 顾勤龙, 姚明海. 一种基于三维加速度传感器的人体行为识别方法[J]. 计算机系统应用, 2013, 22(6): 132-135.

[2] 余杰. 基于加速度传感器的人体行为识别研究[D]. 天津工业大学, 2017.

[3] 苟涛. 基于可穿戴式三轴加速度传感器的人体行为识别[J]. 自动化应用, 2015(12):61-62 .

[4] 孙成开, 梅先明, 徐佩佩, 左保齐. 基于可穿戴加速度传感器的人体运动状态识别研究[J]. 现代丝绸科学与技术, 2019, 34(3): 10-14.

[5] 胡春生, 赵汇东. 基于腕部 IMU 数据采集的人体运动模式识别[J]. 宁夏大学学报(自然科学版), 2021, 42(1):45-57.

[6] 宋贺良, 郑毅, 王克强. 可穿戴式人体姿态识别系统研究进展[J]. 激光与红外, 2011, 51(9):1123-1128.

[7] 张含, 包祖超, 朱文馨, 等. 面向人体运动模式的 SVM 识别方法[J]. 物联网技术, 2024, (5):9-13.

[8] Ordóñez F J, Roggen D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition[J]. Sensors, 2016, 16(1): 115.

[9] 王晓玲, 基于机器学习的人体活动识别[D]. 广西科技大学, 2023.

[10] Palaz D, Collobert R. Analysis of CNN-based speech recognition system using raw speech as input[J]. 2015.

[11] Figo D, Diniz P C, Ferreira D R, et al. Preprocessing techniques for context recognition from accelerometer data[J]. Personal and Ubiquitous Computing, 2010, 14: 645-662.

附　录

附录 I: 主要程序/关键代码

代码环境	操作系统: Windows 10 编程语言: MATLAB,Python 编辑器: 代码详见:

代码清单 1 基于阈值方法的决策树形式分类模型

```
%决策树方法
clc
clear
close all
%% 读入数据
%列向量数据含义分别为加速度xyz,姿态xyz
excel_path1=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件1\Person1\';
  %文件夹路径
excel_path2=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件1\Person2\';
  %文件夹路径
excel_path3=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件1\Person3\';
  %文件夹路径
path_list1 = dir(strcat(excel_path1,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
path_list2 = dir(strcat(excel_path2,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
path_list3 = dir(strcat(excel_path3,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
list_num = 60;
person1date = cell(list_num,1);
person2date = cell(list_num,1);
person3date = cell(list_num,1);
for i=1:list_num1
    person1date{i} = xlsread([excel_path1,path_list1(i).name]);
end
for i=1:list_num2
    person2date{i} = xlsread([excel_path2,path_list2(i).name]);
end
for i=1:list_num3
    person3date{i} = xlsread([excel_path3,path_list3(i).name]);
end

excel_path4=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person4\';
  %文件夹路径
path_list4 = dir(strcat(excel_path4,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
list_num = 60;
person4date = cell(list_num,1);
for i=1:list_num
    person4date{i} = xlsread([excel_path4,path_list4(i).name]);
end

excel_path5=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person5\';
  %文件夹路径
excel_path6=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person6\';
  %文件夹路径
excel_path7=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person7\';
  %文件夹路径
path_list5 = dir(strcat(excel_path5,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
path_list6 = dir(strcat(excel_path6,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
path_list7 = dir(strcat(excel_path7,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
list_num = 60;
person5date = cell(list_num,1);
person6date = cell(list_num,1);
```

```
person7date = cell(list_num,1);
for i=1:list_num
    person5date{i} = xlsread([excel_path5,path_list5(i).name]);
end
for i=1:list_num
    person6date{i} = xlsread([excel_path6,path_list6(i).name]);
end
for i=1:list_num
    person7date{i} = xlsread([excel_path7,path_list7(i).name]);
end

excel_path8=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person4\';
  %文件夹路径
excel_path9=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person5\';
  %文件夹路径
excel_path10=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person6\
';    %文件夹路径
path_list8 = dir(strcat(excel_path8,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
path_list9 = dir(strcat(excel_path9,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
path_list10 = dir(strcat(excel_path10,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
list_num = 60;
person8date = cell(list_num,1);
person9date = cell(list_num,1);
person10date = cell(list_num,1);
for i=1:list_num
    person8date{i} = xlsread([excel_path8,path_list8(i).name]);
end
for i=1:list_num
    person9date{i} = xlsread([excel_path9,path_list9(i).name]);
end
for i=1:list_num
    person10date{i} = xlsread([excel_path10,path_list10(i).name]);
end
excel_path11=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person11
\';    %文件夹路径
excel_path12=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person12
\';    %文件夹路径
excel_path13=  '#####\2024湖南省省赛\2024年湖南省研究生数学建模竞赛赛题\A题\附件2\Person13
\';    %文件夹路径
path_list11 = dir(strcat(excel_path11,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
path_list12 = dir(strcat(excel_path12,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
path_list13 = dir(strcat(excel_path13,'*.xlsx')); %dir 函数 列出当前目录下所有子文件夹和文件
list_num = 60;
person11date = cell(list_num,1);
person12date = cell(list_num,1);
person13date = cell(list_num,1);
for i=1:list_num
    person11date{i} = xlsread([excel_path11,path_list11(i).name]);
end
for i=1:list_num
    person12date{i} = xlsread([excel_path12,path_list12(i).name]);
end
for i=1:list_num
    person13date{i} = xlsread([excel_path13,path_list13(i).name]);
end

%%

class_out = cell(60,1);
path_list = path_list13;
for i = 1:list_num %对于一个人的60组测试数据进行分类
     flag = 0;
     persondate = person13date{i};%对第i个数据进行分类处理
     filer_out = IIRfilter(persondate(:,1));%滤波处理
     size_temp = size(filer_out);
     mean_x = mean(filer_out);%求平均值
     var_x = var(filer_out);%求方差

     if mean_x < 0 %平均值小于0的就是躺
```

```
        class out{i,1} = path list(i).name;%
        class out{i,2} = 10;
        flag = 1;
    elseif var_x >0.5 %方差大于0.5的就是跑或者跳
        outfft = var(plot fft(filer out));
        if outfft<0.005
            class out{i,1} = path list(i).name;%
            class out{i,2} = 6;
            flag = 1;
        else
            class out{i,1} = path list(i).name;%
            class out{i,2} = 7;
            flag = 1;
        end
    end

    if flag == 1
        continue;
    end

    %计算偏度
    sk = 0;
    N = length(filer out);
    for  j = 1:N
        sk = sk + (filer out(j) - mean(filer out))^3;
    end
    SK = (N*sk)/((N-1)*(N-2)*var(filer out)^2);

    %计算峰值的中值
        n = 1;
        for j = 2:N-1
            if filer out(j)>filer out(j-1) && filer out(j)<filer out(j+1)
            peak(n) = filer out(j);
            n = n +1;
            end
        end
        peak x = median(peak);
    %计算相关度
    filer_outx = IIRfilter(persondate(:,1));%滤波处理
    filer_outy = IIRfilter(persondate(:,2));%滤波处理
    rxy1 = 0;
    rxy2 = 0;
    rxy3 = 0;
    for  j = 1:N
        rxy1 = rxy1 + (filer outx(j) - mean(filer outx))*(filer outy(j) - mean(filer outy));
        rxy2 = rxy2 + (filer outx(j) - mean(filer outx))*(filer outx(j) - mean(filer outx));
        rxy3 = rxy3 + (filer outy(j) - mean(filer outy))*(filer outy(j) - mean(filer outy));
    end
    RXY = rxy1/sqrt(rxy2*sqrt(rxy3));
    if SK<-20
        if peak x<0.5
            if RXY>0.5
                class out{i,1} = path list(i).name;%
                class out{i,2} = 11;
            else
                class out{i,1} = path list(i).name;%
                class out{i,2} = 12;
            end
        else
            if RXY>0.25
                class out{i,1} = path list(i).name;%
                class out{i,2} = 8;
            else
                class out{i,1} = path list(i).name;%
                class out{i,2} = 9;
            end
        end
    else
        if peak x<0.89
            if RXY<-0.5
                class out{i,1} = path list(i).name;%
```

```
                class out{i,2} = 1;
            elseif RXY<-0.5
                class out{i,1} = path list(i).name;%
                class out{i,2} = 2;
            else
                class out{i,1} = path list(i).name;%
                class out{i,2} = 3;
            end
        else
            if RXY<1.25
                class out{i,1} = path list(i).name;%
                class out{i,2} = 4;
            else
                class out{i,1} = path list(i).name;%
                class out{i,2} = 5;
            end
        end

    end
end
```

代码清单 2 滤波器函数模块

```
function [output] = IIRfilter(input)
%UNTITLED8 巴特沃斯滤波器
%    此处显示详细说明

fs = 100; % 采样频率
x = input;
t = 0:1:size(input)-1;
t = t/fs;
fc_low = 10; % 低截止频率
fc_high = 7; % 高截止频率

N = 7; % 滤波器阶数

[b, a] = butter(N, fc_low/(fs/2), 'low'); % 计算低通滤波器系数
% [b, a] = butter(N, [fc_low/(fs/2), fc_high/(fs/2)], 'bandpass'); % 计算中通滤波器系数
% [b, a] = butter(N, fc_high/(fs/2), 'high'); % 计算中高通滤波器系数


% 使用中通滤波器对信号进行滤波
y = filter(b, a, x);

% 绘制原始信号和滤波后的信号
% figure;
% subplot(2,1,1);
% plot(t, x);
% title('原始信号');
% xlabel('t/s');
% ylabel('幅值');
%
```

```
% subplot(2,1,2);
% plot(t, y);
% title('滤波信号');
% xlabel('t/s');
% ylabel('幅值');
output = y;
end
```

代码清单 3 傅里叶变换函数模块

```
function [output] = plot_fft(input)
Fs = 100; % 采样频率(Hz)
length(input)
t = 0:1:length(input)-1; % 时间向量
t = t*0.01;



% 计算FFT
N = length(input); % 信号长度
X = fft(input); % 快速傅里叶变换

% 计算双边频谱的幅度
P2 = abs(X/N);

% 计算单边频谱的幅度
P1 = P2(1:N/2+1);
P1(2:end-1) = 2*P1(2:end-1);

% 定义频率轴
f = Fs*(0:(N/2))/N;

%绘制单边幅度频谱
% figure
% plot(f, P1)
% title('单边幅度频谱')
% xlabel('频率 (Hz)')
% ylabel('|P1(f)|')
output = P1;
end
```

代码清单 4 CNN-LSTM 模型结构

```
    def build_model(
    input_shape: Tuple[int, int, int, int] = (128, 6, 1), output_dim: int = 12, lr: float = 0.001
) -> Model:
    model = Sequential()
    model.add(Conv2D(64, kernel_size=(5, 1), input_shape=input_shape))
    model.add(Activation("relu"))
    model.add(Conv2D(64, kernel_size=(5, 1)))
    model.add(Activation("relu"))
    model.add(Conv2D(64, kernel_size=(5, 1)))
    model.add(Activation("relu"))
    model.add(Conv2D(64, kernel_size=(5, 1)))
    model.add(Activation("relu"))
    model.add(Reshape((112, 6 * 64)))
    model.add(LSTM(128, activation="tanh", return_sequences=True))
    model.add(Dropout(0.5, seed=0))
    model.add(LSTM(128, activation="tanh"))
    model.add(Dropout(0.5, seed=1))
    model.add(Dense(output_dim))
    model.add(Activation("softmax"))
    model.compile(
        loss="categorical_crossentropy", optimizer=optimizers.Adam(lr=lr),
metrics=["accuracy"]
    )
    return model
```

代码清单 5 数据读取与预处理

```
    def preprocess_raw_data_for_new(scaler):
    X_all = np.array([])
    y_all = []
    data_num = 10
    data_num_for_one = 12
    data_num_for_k = 5
    start = 4
    for i in range(data_num):
        for j in range(1,data_num_for_one+1):
            for k in range(1,data_num_for_k+1):
                str_for_xls = "a"+str(j)+"t"+str(k)+".xlsx"
                data_raw = os.path.join(DATA_DIR,
"Person"+str(start)+"/"+str_for_xls)
                df = pd.read_excel(data_raw, header=0,
names=["acc_x(g)","acc_y(g)","acc_z(g)","gyro_x(dps)","gyro_y(dps)","gyro
_z(dps)"])
                # cout = 3000 - df.shape[0]
                # if cout > 0:
                #     for zz in range(cout):
```

```
                        #                 df = df._append(pd.DataFrame([[0,0,0,0,0,0]],
columns=df.columns))
                        acc_raw = df.iloc[:,:3]
                        gyro_raw = df.iloc[:,3:6]
                        acc_raw = scale(acc_raw, scaler=scaler)
                        gyro_raw = scale(gyro_raw, scaler=scaler)
                        tAccXYZ = preprocess_signal(acc_raw)

                        tBodyGyroXYZ = preprocess_signal(gyro_raw)
                        features = np.zeros((len(tAccXYZ), 128, 6))

                        for z in range(len(tAccXYZ)):
                            feature = pd.DataFrame(
                                np.concatenate((tAccXYZ[z],
tBodyGyroXYZ[z]), 1),
                                columns=["AccX", "AccY", "AccZ", "GyroX",
"GyroY", "GyroZ"],
                            )
                            features[z] = feature
                            y_all.append(j-1)
                        # print(features.shape)
                        if len(X_all) == 0:
                            X_all = features
                        else:
                            X_all = np.vstack((X_all, features))
                print(start)

                start += 1
        y_all = np.array(y_all)

        X_train, X_test, y_train, y_test = train_test_split(X_all, y_all,
test_size=0.2, random_state=42)
        y_train = np.array(y_train)
        y_test = np.array(y_test)
        X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 6, 1)
        X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 6, 1)
        print("训练集  X:", X_train.shape, "标签  y:", y_train.shape)
        print("测试集  X:", X_test.shape, "标签  y:", y_test.shape)

        return X_train, X_test, y_train, y_test
```

代码清单 5 模型训练与验证

```
        CUR_DIR = os.path.dirname(os.path.abspath(__file__))    # Path to
current directory

# Logging settings
EXEC_TIME = "deep-conv-lstm-" +
```

```
datetime.now().strftime("%Y%m%d-%H%M%S")
LOG_DIR = os.path.join(CUR_DIR, f"logs/{EXEC_TIME}")
os.makedirs(LOG_DIR, exist_ok=True)    # Create log directory

formatter = "%(levelname)s: %(asctime)s: %(filename)s: %(func-
Name)s: %(message)s"
basicConfig(filename=f"{LOG_DIR}/{EXEC_TIME}.log", level=DEBUG,
format=formatter)
mpl_logger = getLogger("matplotlib")    # Suppress matplotlib logging
mpl_logger.setLevel(WARNING)
# Handle logging to both logging and stdout.
getLogger().addHandler(StreamHandler(sys.stdout))

logger = getLogger(__name__)
logger.setLevel(DEBUG)
logger.debug(f"{LOG_DIR}/{EXEC_TIME}.log")

X_train, X_test, y_train, y_test, label2act, act2label = load_raw_data()

logger.debug(f"{X_train.shape=} {X_test.shape=}")
logger.debug(f"{y_train.shape=} {y_test.shape=}")
print(y_train.flatten())
check_class_balance(y_train.flatten(), y_test.flatten(), label2act=label2act)

# Split data by preserving the percentage of samples for each class.
n_splits = 5
cv = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=71)
valid_preds = np.zeros((X_train.shape[0], 12))
test_preds = np.zeros((n_splits, X_test.shape[0], 12))
models = []
scores: Dict[str, Dict[str, List[Any]]] = {
    "logloss": {"train": [], "valid": [], "test": []},
    "accuracy": {"train": [], "valid": [], "test": []},
    "precision": {"train": [], "valid": [], "test": []},
    "recall": {"train": [], "valid": [], "test": []},
    "f1": {"train": [], "valid": [], "test": []},
    "cm": {"train": [], "valid": [], "test": []},
    "per_class_f1": {"train": [], "valid": [], "test": []},
}
# Load hyper-parameters
with open(os.path.join(CUR_DIR, "configs/default.json"), "r") as f:
    dcl_params = json.load(f)["deep_conv_lstm_params"]
    logger.debug(f"{dcl_params=}")

y_test = keras.utils.to_categorical(y_test, 12)

for fold_id, (train_index, valid_index) in enumerate(cv.split(X_train, y_train)):
    X_tr = X_train[train_index, :]
    X_val = X_train[valid_index, :]
    y_tr = y_train[train_index]
```

```
        y_val = y_train[valid_index]

        y_tr = keras.utils.to_categorical(y_tr, 12)
        y_val = keras.utils.to_categorical(y_val, 12)

        logger.debug(f"{X_tr.shape=} {X_val.shape=} {X_test.shape=}")
        logger.debug(f"{y_tr.shape=} {y_val.shape=} {y_test.shape=}")

        pred_tr, pred_val, pred_test, model = train_and_predict(
            LOG_DIR, fold_id, X_tr, X_val, X_test, y_tr, y_val, dcl_params
        )
        models.append(model)

        valid_preds[valid_index] = pred_val
        test_preds[fold_id] = pred_test

        for pred, X, y, mode in zip(
            [pred_tr, pred_val, pred_test], [X_tr, X_val, X_test], [y_tr, y_val,
y_test], ["train", "valid", "test"]
        ):
            loss, acc = model.evaluate(X, y, verbose=0)
            pred = pred.argmax(axis=1)
            y = y.argmax(axis=1)
            scores["logloss"][mode].append(loss)
            scores["accuracy"][mode].append(acc)
            scores["precision"][mode].append(precision_score(y, pred, aver-
age="macro"))
            scores["recall"][mode].append(recall_score(y, pred, aver-
age="macro"))
            scores["f1"][mode].append(f1_score(y, pred, average="macro"))
            scores["cm"][mode].append(confusion_matrix(y, pred, normal-
ize="true"))
            scores["per_class_f1"][mode].append(f1_score(y, pred, aver-
age=None))

# Output Cross Validation Scores
logger.debug("---Cross Validation Scores---")
for mode in ["train", "valid", "test"]:
        logger.debug(f"---{mode}---")
        for metric in ["logloss", "accuracy", "precision", "recall", "f1"]:
            logger.debug(f"{metric}={round(np.mean(scores[met-
ric][mode]))}")

        class_f1_mat = scores["per_class_f1"][mode]
        class_f1_result = {}
        for class_id in range(12):
            mean_class_f1 = np.mean([class_f1_mat[i][class_id] for i in
range(n_splits)])
            class_f1_result[label2act[class_id]] = mean_class_f1
        logger.debug(f"per-class f1={round(class_f1_result)}")
```

```
# Output Final Scores Averaged over Folds
logger.debug("---Final Test Scores Averaged over Folds---")
test_pred = np.mean(test_preds, axis=0).argmax(axis=1)    # average over
folds
y_test = y_test.argmax(axis=1)
logger.debug(f"accuracy={accuracy_score(y_test, test_pred)}")
logger.debug(f"precision={precision_score(y_test, test_pred, aver-
age='macro')}")
logger.debug(f"recall={recall_score(y_test, test_pred, average='macro')}")
logger.debug(f"f1={f1_score(y_test, test_pred, average='macro')}")
logger.debug(f"per-class f1={f1_score(y_test, test_pred, average=None)}")

# Plot comfusion matrix
plot_confusion_matrix(
        cms=scores["cm"],
        labels=[
                "GO_FORWARD","GO_LEFT","GO_RIGHT","WALKING_UP-
STAIRS","WALKING_DOWN-
STAIRS","GO_RUN","JUMP","SIT","STANDING","LIE","ELEVA-
TOR_UP","ELEVATOR_DOWN"],
        path=f"{LOG_DIR}/comfusion_matrix.png",
)

np.save(f"{LOG_DIR}/valid_oof.npy", valid_preds)
np.save(f"{LOG_DIR}/test_oof.npy", np.mean(test_preds, axis=0))    # Aver-
aging
```