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）。

我们参赛选择的题号是（从组委会提供的赛题中选择一项填写）：A

我们的参赛编号（请填写完整参赛编号）：202418001003

所属学校（请填写完整的全名）：国防科技大学

参赛队员（打印后签名）：1. 杨一赫

2. 徐心成

3. 牛中颜

指导教师或指导教师组负责人（打印后签名）：张兴龙

日期：2024 年 7 月 12 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：基于智能感知技术的人体运动状态识别研究

摘要：随着智能化感知技术的发展，人体活动状态的智能识别（HAR）系统在运动健康方面发挥着重要的作用，HAR 成为了一个重要的研究领域。本文依据采集的不同年龄、身高、体重的被试人员的加速度计和陀螺仪数据，设计了有效的分类模型和判别模型，对于不同数据的状态进行了预测，同时提出了根据数据输出人员画像的有效算法。

针对问题 1，我们设计了一种基于谱聚类的逆层次树状聚类算法，将 12 类运动状态根据运动特征逐层划分。具体来说，第一层将运动状态划分为急促动作和平缓动作，第二层将急促动作划分为跳跃和向前跑，将平缓动作划分为动态动作和静态动作。第三层将动态动作划分为水平运动和倾斜运动，将静态动作划分为站立状态和休息状态，第四层完成了水平运动、倾斜运动、站立状态、休息状态的细致聚类分析。利用分类模型，我们预测了附件一中三个被试人员的不同运动状态。

对于问题 2，根据附件中提供的有标签的实验数据，本文设计了一种基于深度集成网络的人体姿态识别的方法，具体来说，本文根据对较难分的不同状态较为有效的设计特征将原始数据扩展到高维度，将高维度数据使用线性平滑插值的方法转为多维特征图像，使用了 VGG19、Resnet101、Efficient-b3 深度网络进行训练集成，在实验数据上达到了较高的准确率。同时，我们比较了使用分类模型的结果以及不同类别的分类准确率。

最后，对于问题 3 本文分析了不同活动状态与被试人员的年龄、身高、体重等因素的关系，依据分化思想分别建立了不同因素的深度集成网络，对于预测的多维标签使用了 ANN 近相邻算法快速查找最相似的向量，快速而准确地确定了未知数据的来源人员，实现了根据活动状态数据进行人员画像。

关键词：人体活动识别 谱聚类 集成学习 深度神经网络 ANN

目录

1 问题综述

1.1 问题背景

人类活动识别（HAR）如今已成为一个重要的研究领域，由于其在人类中心研究领域的重要贡献，旨在改善人们的生活质量：环境智能、普适计算和辅助技术[1][2][3] 。这些领域利用 HAR 系统作为提供有关人们行为和动作信息的工具[4] 。通常，这通过收集环境和可穿戴传感器的信号，并通过机器学习算法进行分类处理来实现。目前有许多应用场景使用 HAR 系统，例如连续监测运动障碍患者以进行健康诊断和个性化药物调整[5] ，以及自动监视公共场所以预防犯罪[6] 。在过去的十年中，已经提出和调查了多种 HAR 系统[7][8][9] 。这些系统涵盖了来自不同应用领域的多种活动，包括运动、日常生活活动、交通和体育[10][11] （例如步行、烹饪、驾驶和跑步）。关于活动的持续时间和复杂性，它们通常分为三类主要群组：短事件、基本活动（BAs）和复杂活动。前者是短暂的活动（几秒钟的量级），如姿势转换（例如坐到站立）和身体手势。基本活动则特征在较长时间内进行，可以是动态的或静态的（例如跑步和阅读）[12] 。复杂活动组则由上述简单活动的进展组成，涉及与物体和其他个体的互动（例如进行体育运动、社交活动）[13] 。本文所研究和建模的重点是前两类活动。

对于人体运动状态分类识别的问题人们提出了许多卓有成效的算法，其中包括支持向量机[14] 、随机森林[15] 、决策树[16] 等极具代表性经典机器学习算法的算法。这些早期的算法只能对少数几种运动状态进行有效识别，这些算法基本是基于公开数据集 ADL 设计实现的。这些经典算法在应对分类类别更多的问题显得力不从心，采用无监督进行人体运动状态识别的方法目前还比较少。问题 1 是一个无监督的多分类的问题，本文设计了一种基于谱聚类的逆向层次聚类的方法用于解决该问题，该方法通过将多分类任务分解为多个类别更少的分类子问题。近年来，随着深度学习的兴起，基于深度神经网络的方法也被广泛用于人体运动状态识别[17][18][19] ，但深度学习存在泛化能力、鲁棒性较差的问题。为此本文设计了一种深度集成网络，通过继承多个网络来提升模型的泛化能力和鲁棒性，用于解决问题 2。

1.2 问题提出

通过加速度计和陀螺仪测量数据进行人类活动识别（HAR）任务将涉及多方面的问题，往往由数据分析，数据预处理、分类器建模、后处理等要素构成。本文在满足一下几点假设下，使用加速度计和陀螺仪测量数据设计分类算法对人体动作数据进行分类任务。

从数据分析、数据预处理、分类器建模、后处理等角度考虑，解决下列问题：

(1)问题 1：根据附件 1 中的 3 名试验人员的每人 60 组实验数据，设计无监督分类器，将实验数据分为 12 类。

(2)问题 2：根据附件 2 中的 10 名实验人员的运动数据，根据提供的活动数据提取 12 类人员活动状态的典型特征，建立人员活动状态的判别模型。使用问题 1 中的无监督分类器对附件 2 中数据进行分类，并对比无监督分类器和本问中判别模型的结果，分析采用无监督分类模型对不同活动类型分类时的分类准确度。并使用本问所建立的判别模型对附件 3 中某实验员的 30 组活动数据进行分类并给出结果。

(3)问题 3：根据附件 4 给出的 13 位实验人员的年龄、身高、体重等数据，分析不同人员的同一活动状态是否存在差异，活动状态数据与实验人员的年龄、身高、体重有无关系，讨论能否使用活动传感器数据进行人员画像。进一步，根据附件 5 中给出了

10 位实验人员中的 5 位的某次活动数据，使用你们的模型判断他们分别最可能来源于问题 2 中哪一名实验人员。

2 模型假设、符号说明及数据预处理

2.1 模型基本假设

(1) 假设实验人员 1-13 所采集的运动状态数据时所处环境基本相似。
(2) 假设加速度计和陀螺仪所采集到的数据均认为是人体运动数据，不考虑传感器异常情况。
(3) 假设实验人员与可穿戴式数据采集设备固定牢靠，不会出现测量传感器与人体发生相对运动现象。
(4) 假设将实验人员运动视为刚体运动，对其运动进行分析。
(5) 假设所有实验人员使用的传感器为同一传感器，传感器的零漂、噪声干扰基本相似。

2.2 符号说明

2.2.1 运动状态编码

表1 运动状态编码

活动状态	编号	活动状态	编号
向前走	1	跳跃	7
向左走	2	坐下	8
向右走	3	站立	9
步行上楼	4	躺下	10
步行下楼	5	乘坐电梯向上移动	11
向前跑	6	乘坐电梯向下移动	12

2.2.2 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2 符号说明

符号	含义	单位
a_x	加速度计测量的 X 轴加速度	$\mathrm{m/s^2}$
a_y	加速度计测量的 Y 轴加速度	$\mathrm{m/s^2}$
a_z	加速度计测量的 Z 轴加速度	$\mathrm{m/s^2}$
a	合成加速度	$\mathrm{m/s^2}$
w_x	陀螺仪测量的绕 X 轴转动角速度	dgree/s
w_y	陀螺仪测量的绕 Y 轴转动角速度	dgree/s
w_z	陀螺仪测量的绕 Z 轴转动角速度	dgree/s
f_s	传感器采样频率，固定值为 100	Hz

2.3 附件数据处理

2.3.1 数据滤波

在本文中，使用了两种滤波器对附件中的数据进行处理。首先我们使用了一种自适应的加速度计数据滤波算法[23] [24],数据滤波效果较为明显。自适应加速度计滤波算法具体如下：

在惯性传感器读取数据时，由于外界原因，加速度计总会受到各种外界因素的干扰导致传感器测量数据并不准确，测量值为

$$a_{measured} = a_{real} + a_{error} \tag{1}$$

式中，a_{real}为数据真实值，a_{error}为传感器误差。传感器误差值通常可以表示为零漂和噪声干扰，数学表达为：

$$a_{error} = E + a_{noise} \tag{2}$$

式中，零漂为常数，可以通过标定等方法进行处理。而噪声干扰需要选择滤波器进行平滑滤波处理。本文所使用的自适应加速度计滤波算法主要是通过差分的方式，结合数据分布自适应调整对本次测量值和上次测量值的权重并计算期望，进而对数据进行平滑滤波。我们使用X作为测量值，使用Y作为滤波输出值，基本原理为

$$Y(k) = mX(k) + (1-m)Y(k-1) \tag{3}$$

式中$Y(k)$为本次滤波值，$X(k)$为本次测量值，$Y(k-1)$为上次滤波值，m为该滤波器系数。根据数据分布，可以自适应调节系数m，进而滤波出置信度更高的测量值。滤波系数更新条件如下

$$m\ （k）= \begin{cases} 1-\dfrac{\Delta_a}{\Delta}p_0 & \Delta = Y(k)-Y(k-1) > \Delta_a \\ m(k-1) & else \end{cases} \tag{4}$$

式中Δ为两次滤波输出的差值，Δ_a为条件判断阈值，一般认为是加速度的静态数据的标准差，p_0为滤波器参数，此为滤波器超参数。

使用该自适应加速度计滤波算法前应先计算加速度计静态数据的标准差，观察了附件中的数据分布，取站立、乘坐电梯向上运动、乘坐电梯向下运动、坐下等几个相对静态的动作的稳定值作为该加速度计的静态数据，并计算标准差，静态数据如图 1 所示。

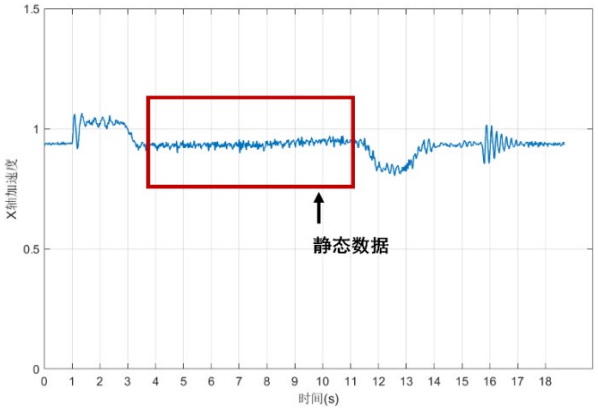
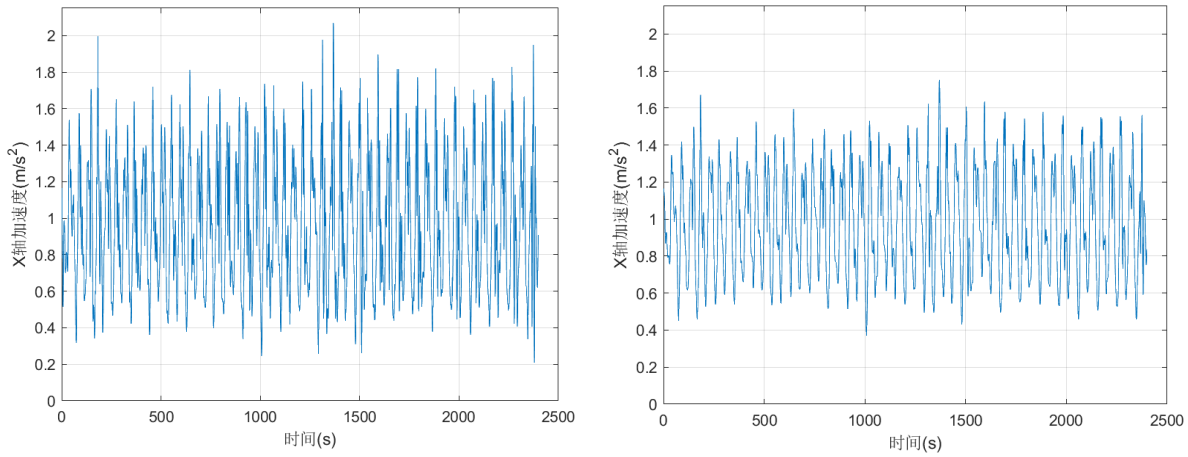


图1 静态数据

计算静态数据标准差后运动滤波器算法，滤波前后对比如图 2 所示。



(a) (b)

图2 自适应滤波算法滤波前后对比, (a)滤波前; (b)滤波后

2.3.2 数据预处理

（1）数据归一化

由于各个特征之间的量纲、变化范围等因素不同，如直接将将特征用于分类难以获得理想的效果。为了消除特征量纲和尺度差异的影响，以达到同等对待不同维度的特征的目的需要对数据进行归一化处理。对任意数据 $X=[x_{ij}]_{n\times m}$ 进行极大-极小归一化的方法为：

$$\overline{x}_{ij}=\frac{x_{ij}-x_{min}}{x_{max}-x_{min}} \tag{5}$$

其中 x_{max},x_{max} 分别代表该矩阵中的极大值和极小值。

（2）数据升维

附件中提供的数据包含六个维度，具体为三个方向的线性加速度数值和三个方向的角加速度数值。然而，这些维度在区分 12 种不同的运动状态时显得不够显著。例如，站立和坐下这两种状态的特征非常相似，难以通过仅有的六个维度进行明确区分。

为了提高数据的区分度，根据第三章中逆向层次聚类的分析结果，我们识别出了一些对特定类别具有较高判别力的特征。这些特征可以作为额外的信息进行提取，从而扩展原有数据的维度。这种方法不仅增加了数据的丰富性，还提高了相似数据之间的差异性，进一步提升模型对不同运动状态的识别能力。

具体而言，逆向层次聚类分析帮助我们识别了各类别之间的细微差异，这些差异在初始数据集中可能未能被充分利用。通过提取这些有效的特征，我们能够构建一个更具辨识力的特征空间，使得模型在面对相似运动状态时，依然能够准确地区分。例如，我们可以引入新的特征，如加速度变化率、角速度变化率以及不同方向加速度和角速度的组合特征等。这些新特征的引入，将显著提高模型的分类性能。通过分析和提取更加细化的特征，我们有效扩展了数据维度，增强了模型在处理复杂、多样化运动状态时的区分能力。

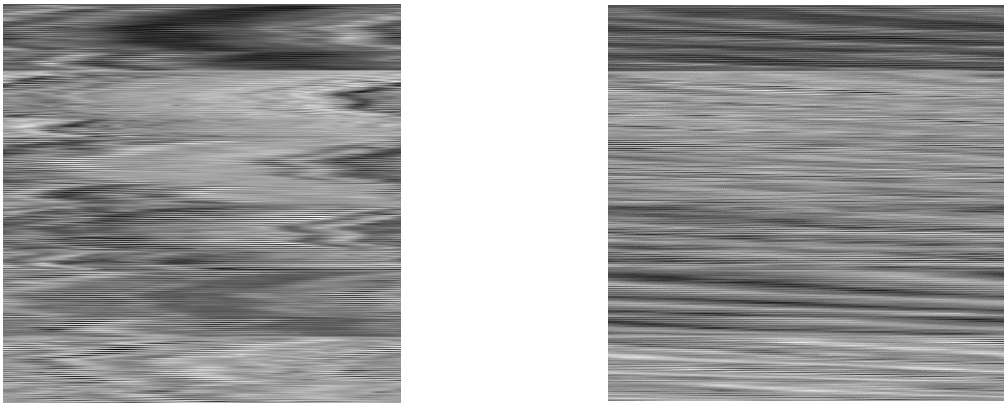
（3）线性插值获得多维特征图像

选择合适的图像尺寸是图像处理和特征提取中的关键一步。适中的图像尺寸不仅有助于优化神经网络的训练过程，还能有效地保留数据中的关键特征，从而提高模型的整体性能和准确性。

我们在设计预期的多维特征图像时，选择了 399×399 的图像尺寸。这一尺寸的选择是经过深思熟虑的，主要基于两个重要原因。首先，过大的图像尺寸不利于后续神经网络的训练，可能导致计算资源的浪费和训练时间的增加。其次，在线性平滑插值的前提下，过大的图像尺寸会导致原本显著的特征被过度平滑处理，从而丧失重要的细节信息。

如图 3 所示，（a）图像的尺寸为 384×384，（b）图像的尺寸为 600×600。尽管这两幅图像使用了相同维度的特征提取方式，但从直观上可以看到，（a）图像的纹理性状更加清晰，而（b）图像中的特征由于尺寸过大而显得过于平滑，不够明显。这一对比清楚地表明，适当的图像尺寸对于保留原始数据的显著特征至关重要。

通过多次实验和对比分析最终确定，图像尺寸在 350 至 500 之间最为适合。在这个范围内，图像既能保持足够的分辨率以展示细节，又不会因为过大而导致特征的模糊和平滑。特别是 399×399 这一尺寸，经过验证可以在保证神经网络高效训练的同时，最大程度地保留数据的显著特征，提供更准确的分类和识别结果。



（a）　　（b）

图3 同一多维特征下不同尺寸的特征图

根据我们选定的 21 维特征，通过应用本节讨论的线性平滑插值方法，我们成功生成了 12 种不同运动状态的多维特征图。这些图像在图 4 中展示，从左到右、从上到下依次排列，分别对应编号 1 至 12 的运动状态。

每幅特征图像都是基于所选特征进行插值处理的结果，展现了各运动状态在多维空间中的特征分布情况。通过这种方式，我们能够将原始数据中的 21 维度特征以更加直观、可视化的形式呈现出来。线性平滑插值方法确保了特征的平滑过渡，使得图像中的细节信息得以保留，同时消除了可能的噪声和不连续性。

这些多维特征图像为后续的深度集成网络训练提供了丰富的数据基础，使模型能够更好地理解和学习不同运动状态之间的差异。通过观察图 4 中的特征图像，我们可以清晰地看到不同运动状态的特征模式，这为模型的训练和分类提供了有力支持。

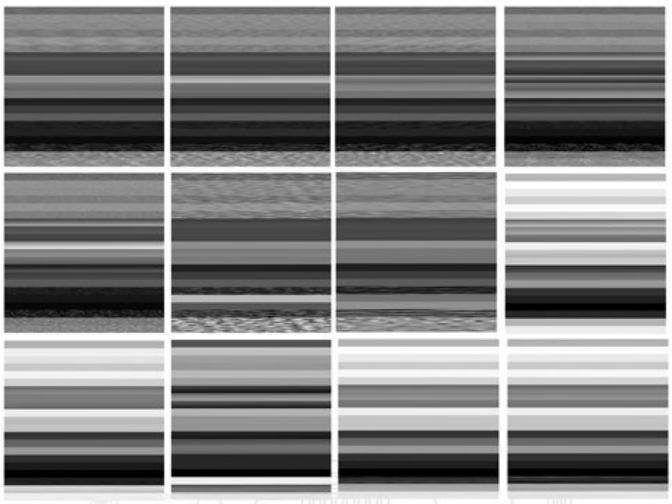


图4 12 种运动状态的多维特征图像

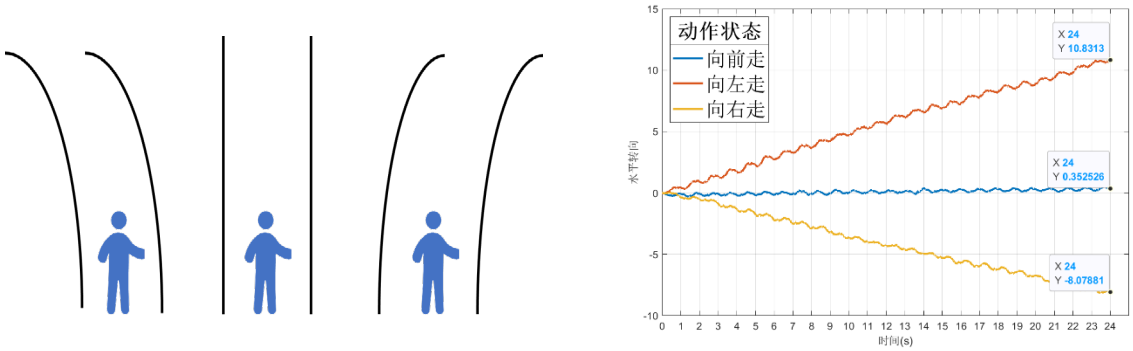
3 基于谱聚类的逆层次树状聚类算法的人体运动状态分类

3.1 问题分析

在该问题中，需要对附件 1 中的 3 名实验员的加速度计和陀螺仪数据进行分类，但并未给出分类标签。在算法设计前，首先查看了附件 2 中实验员 4 的数据，对 12 种运动模态进行理论分析。

(1)向前走:这种运动模态没有特别明显的特征，暂时将该运动模态作为一种标准运动模态，用于与其他部分运动模态进行对比。

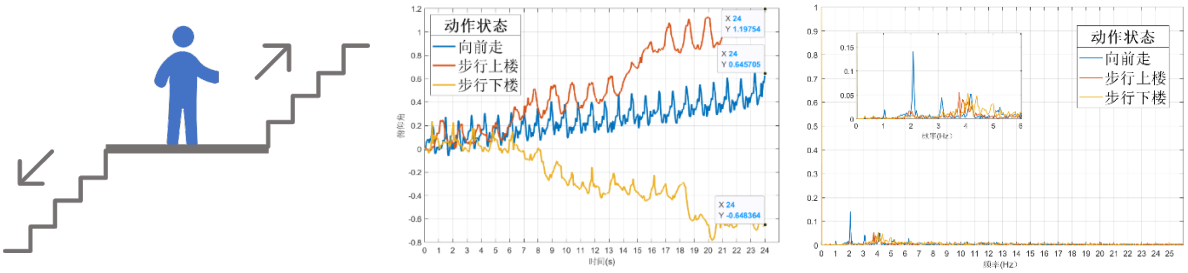
(2)向左走和向右走，在这两种模态下，实验员会向左前方或右前方行走，相较于向前走的运动，该运动状态会有显著的绕 X 轴旋转倾向，因此选择对三种运动状态的 X 轴角速度数据进行积分处理并对比，对比结果如图 1 所示。图 5 显示，模态 2、3 有显然的绕 x 轴的运动转角。



(a) (b)

图5 (a)向前走、向左走、向右走动作示意图; (b)三种动作水平转角图

(3)步行上楼和步行下楼：与向前走相比，该种模态会有显然的俯仰角变化，因此选择对 Z 轴角速度数据进行积分处理，对比结果如图 6 所示。同时，实验人员在进行上楼和下楼动作时，抬腿频率基本保持不变，这是步行上下楼动作的另一个主要特征，

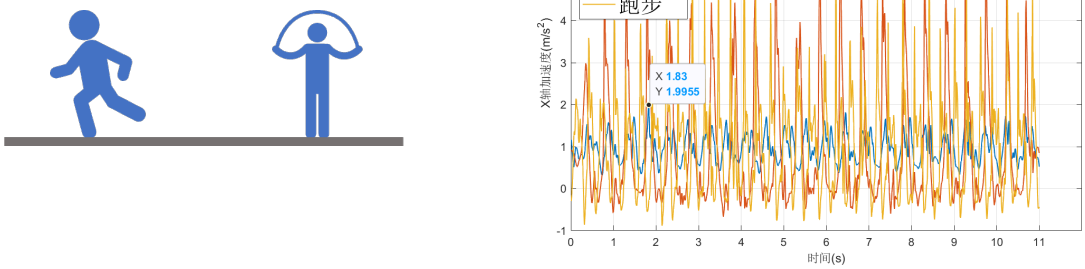


(a) (b) （c）

图6 (a)步行上楼和步行下楼示意图;

(b)三种动作绕 Z 轴角位移对比图; (c)三种动作频谱对比图

(4)跳跃：根据这种运动状态的特点，可以发现在进行跳跃动作时，人体竖直方向，即 x 轴加速度会呈周期性变化，且 y 轴和 z 轴的数据不会有明显的特征。跳跃模态的 x 轴加速度如图 7(b)所示。

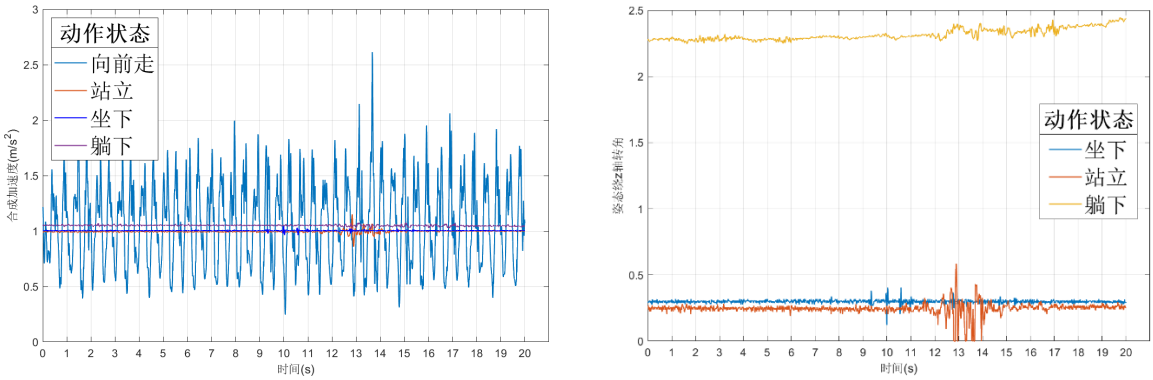


(a) (b)

图7 (a)向前跑和跳跃动作示意图; (b)向前走、向前跑和跳跃 X 轴加速度对比图

(5)向前跑:通过对该运动状态的分解和简化，可以认为该运动状态时前进速度较快的向前走和竖直方向的跳跃运动的组合。因此该运动的特征应同时具备向前走和跳跃两种运动特征，且运动数据的方差较大，向前跑 X 轴加速度如图 7(b)所示。

(6) 坐下、站立、躺下：这三种运动模态的数据相对平稳，可以认为是在几乎静止的的情况下进行的，因此对这三种模态的 XYZ 三轴加速度进行合成应基本等于重力加速度，且合成加速度数据平稳，这是三种运动的共同特点。进一步对比这三种运动状态的特征，可以发现坐下和躺下后的姿态角都相对稳定，陀螺仪测量数据不会有较大的波动，而站立动作可能受到外界物理因素影响，XYZ 三轴角速度测量值波动较大。对于坐下和躺下的区分，躺下时传感器的姿态角与坐下动作存在明显差异，可以根据这一特征进行区分。三种运动的陀螺仪数据和姿态角数据如图 8 所示。

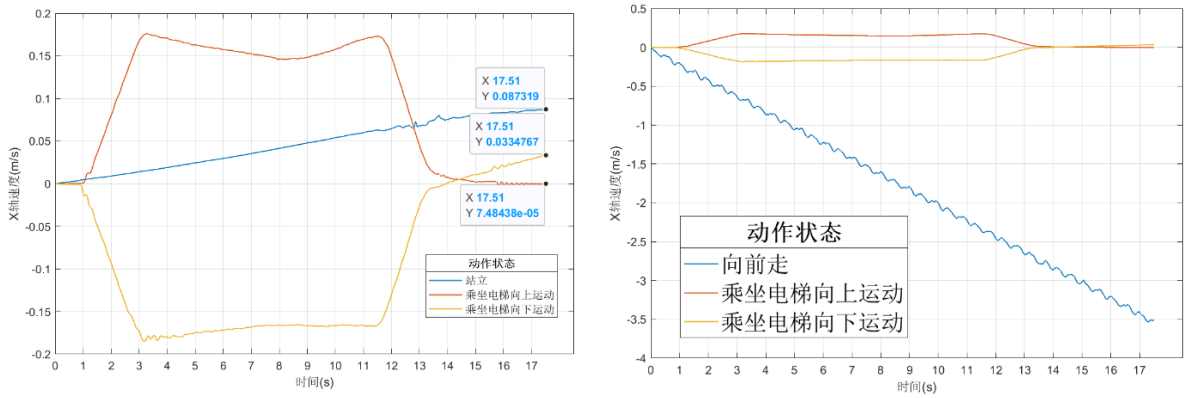


(a) (b)

图8 （a）向前走、站立、坐下、躺下四种动作合成加速度对比图;

（b）坐下、站立、躺下 Z 轴重力倾角对比图

(7)乘坐电梯向上（下）运动：对于坐电梯向上（下）运动的运动特征较为明显，在电梯启动和停止时，加速度计在 X 轴方向上可以检测到较大的加速度数值，而在电梯运行过程中为匀速状态，此时加速度无明显变化，为更进一步鉴别出该运动模态，我们同时考虑对 X 轴加速度进行积分获得速度曲线，由速度曲线得出，电梯运行前后人体速度均为 0，这是一个较明显的判断特征，该运动速度曲线如图 9 所示。



(a) (b)

图9 （a）乘坐电梯与站立速度曲线;（b）乘坐电梯与向前走速度曲线

通过以上分析，我们可以将这 12 类动作状态逐层根据动作特征进行划分，划分结果如图 10 所示。首先我们将 12 类动作状态按照人体在运动时的呼吸状态分为急促动作和平缓动作。急促动作状态中包含了向前跑和跳跃两种剧烈运动动作。平缓动作状态则为其他十种运动状态。进一步，我们对平缓运动状态进行分类，可将其分为动态动作和静态动作，其中动态动作包含向前走、向左走、向右走、步行上楼和步行下楼，静态动作由坐下、站立、躺下、乘坐电梯向上运动和乘坐电梯向下运动组成。

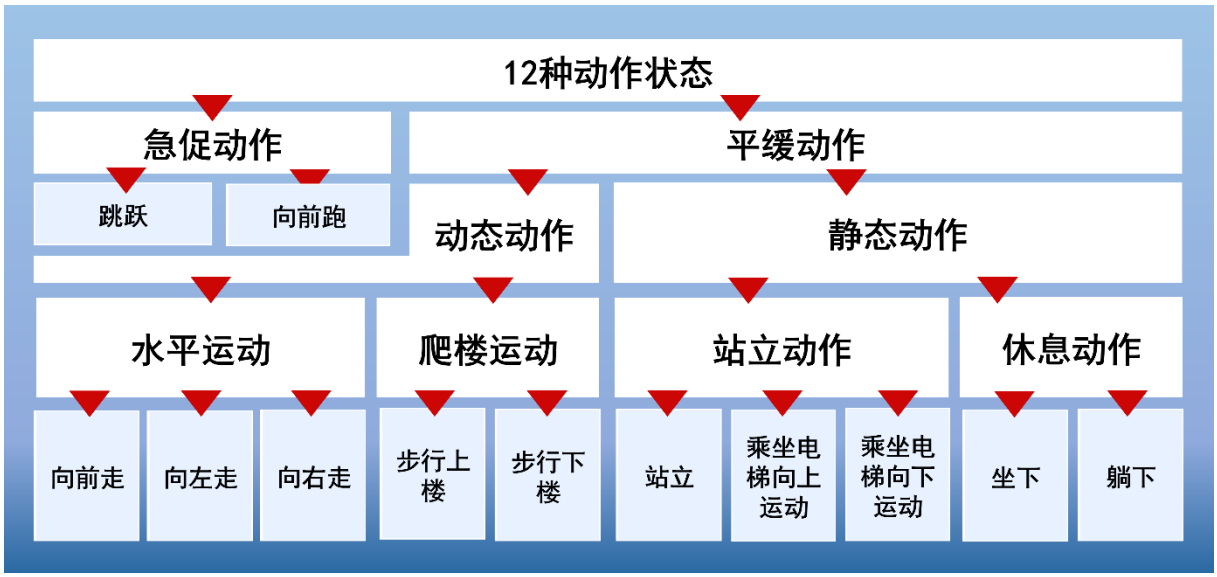


图10 12 类动作状态分类图

3.2 建立无监督分类器模型

对于问题 1 的无监督分类方法，我们提出了一种自上而下的逆层次树状聚类方法，在光谱聚类算法的基础上的基础上结合提取的数据特征对附件 1 的三名实验人员数据进行聚类分析，在每个分支节点通过数据特征进行分层聚类操作。与传统的层次分类不同，该分类结果通过在节点处加入人为提取特征，提高了模型分类的准确率，能更好的把具有相同特征的样本归为同一个簇。

3.2.1 特征提取

根据分类图所示，我们对问题描述的 12 种运动状态进行了五层分类，并根据不同运动状态的特征提取的不同特征进行多次聚类。所提取的特征如下：

(1) 第一层特征

第一层特征提取的主要目的是将 12 种运动状态分成急促动作和平缓动作，此处提取的是加速度计测量数据合成的加速度值，使用合成加速度的均值和方差作为第一层聚类特征，可以较好地分出两大类运动状态，进一步根据两类的聚类中心识别出急促动作和平缓动作的簇，实现第一类分类。

合成加速度的计算公式为：

$$a=\sqrt{a_x^2+a_y^2+a_z^2} \tag{6}$$

附件中给出的是一个长的时间序列求得合成加速度的时间序列仍然直接作为分类特征，故而提取序列的统计特征提取特征为：

$$\overline{a}=\frac{1}{N}\sum_{i=1}^{N}a_i,\sigma_a=\frac{1}{N}\sum_{i}^{N}(a_i-\overline{a})^2 \tag{7}$$

(2) 第二层特征

第二层特征提取的主要目的是为了将急促动作中跳跃和向前跑区分开和将平缓动作中的动态动作和静态动作分类。在急促动作分类中，将绕 Z 轴的转角特征，即为加陀螺仪绕 Z 轴的角位移特征作为显著特征，计算公式为:

$$\triangle\varphi_z=\varphi_z-\varphi_{z0}=\int_0^t w_z dt \;=\; \frac{1}{f_s}\sum_{i=1}^{N} w_{zi} \tag{8}$$

式中$i=1,2,...,N$, 其中N序列长度。

为了将平缓动作中的动态动作和静态动作分类，提取了合成加速度的数值特征，进一步使用合成加速度的均值和方差进行聚类，合成加速度、均值和方差计算公式如式(3)所示。

(3) 第三层特征

在第三层特征提取过程中，主要进行的任务之一是将动态动作中的水平运动和倾斜运动区分开，根据附件 2 中数据分布，发现倾斜运动，即步行上楼和步行下楼，具有较强的周期性变化，而水平运动数据并无明显周期性。根据这一差异，对两者数据进行傅里叶变化，查看两组数据的频谱分布，将频谱分布作为聚类特征，具有较好的聚类性能。

第三层特征提取的另一项任务是将静态动作的站立动作和休息动作区分出来。在第二层和第一层特征提取中，我们使用了合成加速度a的均值和方差等传感器数值信息，由于静态动作时，合成加速度值基本等于重力加速度，可以很好的将静态动作区分出来。在这里我们进一步使用传感器所测量的角度信息作为特征，特征提取具体如下：

本文所使用的三轴加速度计和三轴陀螺仪可作为 IMU，即为惯性测量单元使用，用于测量物体的加速度和角速度。本文中定义 X 轴作为沿重力方向且与重力相反，当加速度计水平放置时，X 轴竖直向上，其读数为 1g，即重力加速度，而 Y 轴和 Z 轴的读数为 0，表示坐标为$(g,0,0)$。由于本层中静态动作可以认为合成加速度与重力加速度基本相等，根据加速度测量的分量即可判断出物体倾角。该倾角与无人机、惯性导航领域所涉及的欧拉角概念不同，这里称为重力倾角，X、Y、Z 轴的重力倾角分别用ϕ,θ,ψ表示，重力倾角如图 11 所示。

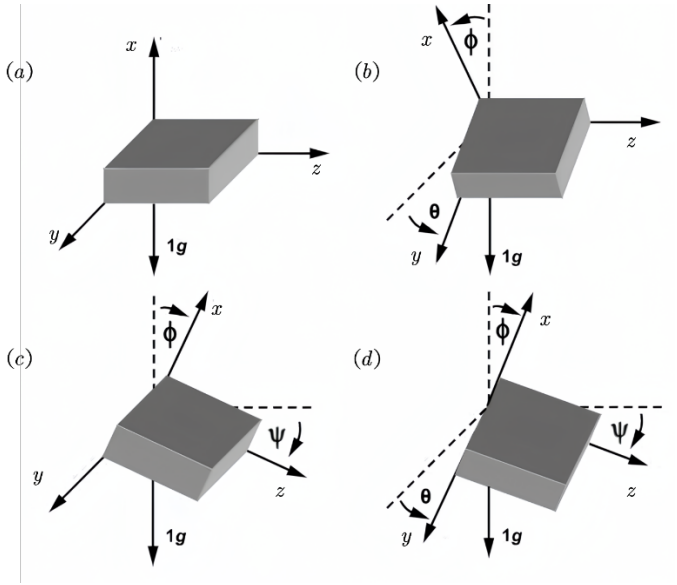


图11 重力倾角示意图

参考市场中的 ADI 加速度计使用说明，根据重力分解原理可反推出重力倾角值：

$$\phi \;=\; \tan^{-1}\left(\frac{\sqrt{a_y^2+a_z^2}}{a_x}\right) \tag{9}$$

$$\theta = \tan^{-1}\left(\frac{a_y}{\sqrt{a_x^2 + a_z^2}}\right) \tag{10}$$

$$\psi = \tan^{-1}\left(\frac{a_z}{\sqrt{a_x^2 + a_y^2}}\right) \tag{11}$$

根据静态动作数据计算各个采样时刻的倾角值的时间序列，统计重力倾角值的均值和方差，作为显著特征。

(4) 第四层特征

第四层特征提取任务分为四个部分，分别完成水平动作、倾斜动作、站立动作和休息动作的最细致聚类分析。在水平动作的特征提取中，考虑到向前走动作和向左走、向右走动作之间存在是否转向区别，因此提取人体运动的水平角位移，即绕 X 轴角速度的积分曲线作为特征进行聚类。

$$\triangle\varphi_x = \int_0^t w_x dt = \frac{1}{f_s}\sum_{i=1}^{N} w_{xi} \tag{12}$$

其中$\triangle\varphi_x$表示水平方向的偏角。

在倾斜动作的特征提取中，考虑步行上楼和步行下楼过程中，人体会有 X 轴的位移和俯仰角度的差异，提取 X 轴加速度的二次积分曲线和 Z 轴角速度的一次积分曲线作为显著特征，特征提取计算如下：

$$\triangle s_x = \int_0^t \int_0^t a_x dt dt \tag{13}$$

$$\triangle\varphi_z = \int_0^t w_z dt = \frac{1}{f_s}\sum_{i=1}^{N} w_{zi} \tag{14}$$

站立动作的聚类显著特征主要为 X 轴速度曲线变化。因为人在乘坐电梯上下移动过程中，电梯启动前和启动后的竖直方向速度均为 0，这是一个可以区分站立和乘坐电梯运动的显著特征。X 轴的速度曲线可以有 X 轴加速度积分求和获得，进一步区分乘坐电梯向上和向下运动时，可以使用速度曲线的符号来进行判断。

休息动作的特征提取较为简单，仍然继续考虑重力倾角的变化，躺下与坐下两种动作的姿态存在显著差异，进而两种动作的重力倾角也存在显著差异，体征提取过程如式(9)(10)(11)所示。

3.2.2 谱聚类原理

谱聚类是一种基于图论的聚类方法，它将数据视作空间中的点，并通过边将这些点连接起来形成加权图。在这个图中，边的权重反映了数据点之间的相似性或距离关系：距离较近的点之间的边权重较高，而距离较远的点之间的边权重较低。谱聚类的核心思想是将整个图切分成若干个子图，以实现聚类的目的。谱聚类利用图的结构和特征向量分析数据点之间的关系，适用于处理非凸形状的聚类问题，其结果能够较好地揭示数据的聚类结构和组织形式。

具体实现时，谱聚类通常通过对图的拉普拉斯矩阵进行特征分解来完成图的切分和子图的提取。拉普拉斯矩阵有多种形式，其中归一化的拉普拉斯矩阵是应用较广泛的一种形式。特征分解后得到的特征向量能够有效地捕捉数据点在低维空间的投影信息，这些特征向量的分布通常反映了数据点的聚类结构。最后，通过对特征向量进行聚类，例

如使用 K-means 算法或直接基于特征向量的谱聚类算法，可以得到最终的聚类结果。由于谱聚类是基于图论的聚类方法，此处简要介绍图论的基本概念。

(1) 无向权重图

对于一个图G而言，一般用点的集合V和边的集合E来描述的，即可表示为$G(V,E)$，其中$V=\{v_1,v_2,...,v_n\}$。对于图中的点而言，每个点之间都是由权重存在的，我们定义v_i和v_j之间的权重为w_{ij}，在无向图中有$w_{ij}=w_{ji}$。当点v_i和v_j有边连接时，$w_{ij}>0$，当v_i和v_j无边连接时，定义$w_{\mathrm{ij}}=0$.进一步可以定义邻接矩阵W:

$$W=[w_{ij}]_{n\times n}\in\mathbb{R}^2 \tag{15}$$

来表示任意两个点之间的权值连接情况。进一步定义每个顶点的度，顶点的度为该顶点与其他顶点连接权值之和，计算公式为

$$d_i=\sum_{j=1}^{N}w_{ij} \tag{16}$$

利用度的定义，可以得到度矩阵D，这是一个对角矩阵，第i个对角元为第i个顶点的度，即

$$D=\mathrm{diag}(d_1,d_2,...,d_N) \tag{17}$$

(2) 邻接矩阵W求解

邻接矩阵是由任意两点的权重值w_{ij}组成的矩阵。但在谱聚类中，通常只给出了数据点的定义，并没有直接给出邻接矩阵，需要通过一些方法进一步计算，如全连接法、K临近法、ϵ－临近法。本文中使用的求解方法为全连接法。

在全连接法中，所有点之间的权重均大于 0，这是全连接法的体现。全连接法可以选择不同的核函数来定义边的权重，其中常用的核函数为多项式核函数，高斯核函数核 sigmoid 核函数等。本文中选择使用的是高斯核函数，此时邻接矩阵和相似矩阵相同为

$$w_{ij}=s_{ij}=e^{-\frac{\|x_i-x_j\|_2^2}{2\sigma^2}} \tag{18}$$

(3) 拉普拉斯矩阵

拉普拉斯矩阵又称为基尔霍夫矩阵，是用来表示图的特征的一种矩阵，给定有 n 个顶点的图，则其拉普拉斯矩阵被定义为

$$L=D-W \tag{19}$$

式中D为图的度矩阵，W为图的邻接矩阵。

拉普拉斯矩阵具有很多优点，性质如下：

- 拉普拉斯矩阵是对称半正定矩阵，这可以由D和W都是对称矩阵而得；
- $L1=01$，即最小特征值是 0，相应的特征向量是 1。
- L有 n 个非负实特征值$0=\lambda_1<\lambda_2<...<\lambda_n$
- 对$\forall f\in\mathbb{R}^2$，均有

$$f^TLf=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}w_{ij}(f_i-f_j)^2$$

(4) 谱聚类算法流程

算法 1 谱聚类

输入： 样本集 $X=(x_1,x_2,...,x_n)$，邻接矩阵生成方式，聚类方法，簇类数目 k_2

输出： 划分簇 $C=(c_1,c_2,...,c_k)$

1：根据数据集 X 构建相似矩阵 S

2：根据相似矩阵 S 构建邻接矩阵 W 和度矩阵 D

3：计算拉普拉斯矩阵 L

4：计算标准化的拉普拉斯矩阵 L_{norm}

5：计算 L_{norm} 的最小的 k_1 个特征值及其对应的特征向量 f

6：构建特征向量矩阵 F，并对每行进行标准化

7：使用 K-means 算法对 F 进行聚类，得到簇划分 C

3.2.3 自上而下的逆层次树状聚类算法流程

由于数据模态和来源不同，简单的对所有数据进行归一化会减小类别之间的差异，导致分类效果较差。基于多核的聚类算法如多核 K 均值聚类能够应对多源数据特征，但使用该类算法的计算成本较高、效率低下，核函数的选择比较依赖经验。基于前文的分析，本文将类别维度高的分类问题分解为多个层的简单分类问题，图 12 所示便是这一算法的基本流程。

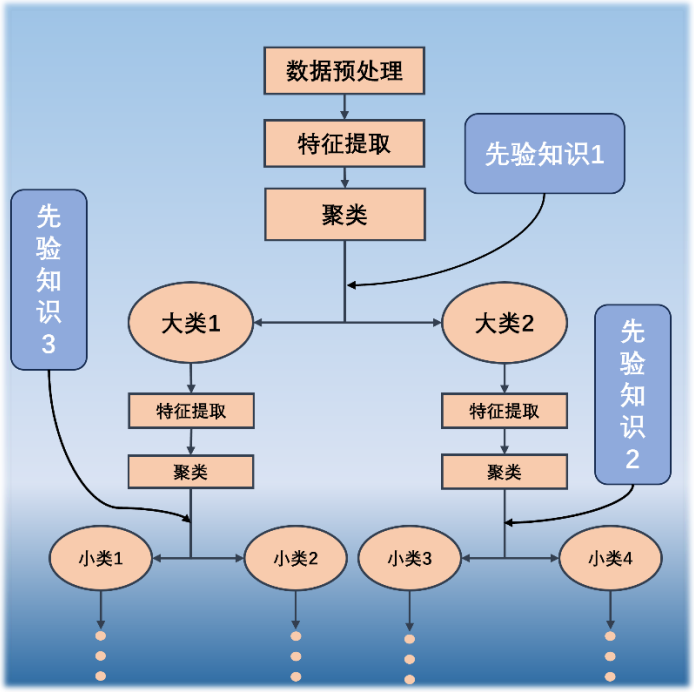
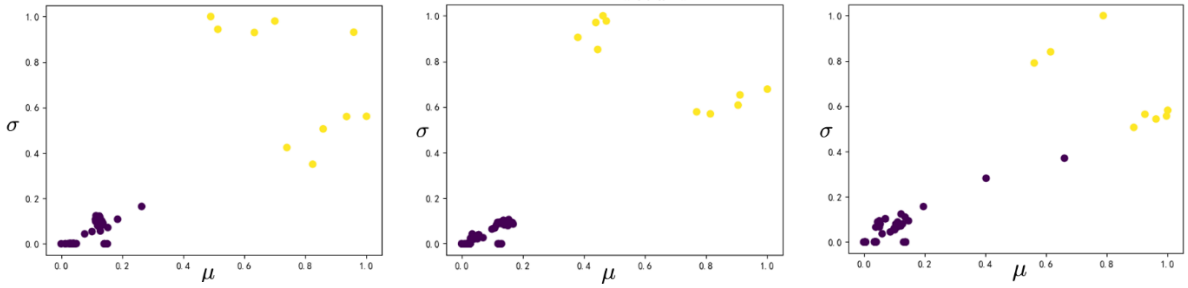


图12 算法流程示意图

(1)第一层分类使用合成加速度用于分类急促动作和平缓动作，从人体运动机理出发可知人在跑步和跳跃这类急促的运动时加速度计检测到的合成加速度回显著大于平缓运动。



(a) (b) (c)

图13 急促与平缓运动聚类结果(a)Person1; (b)Person2; (c)Person3

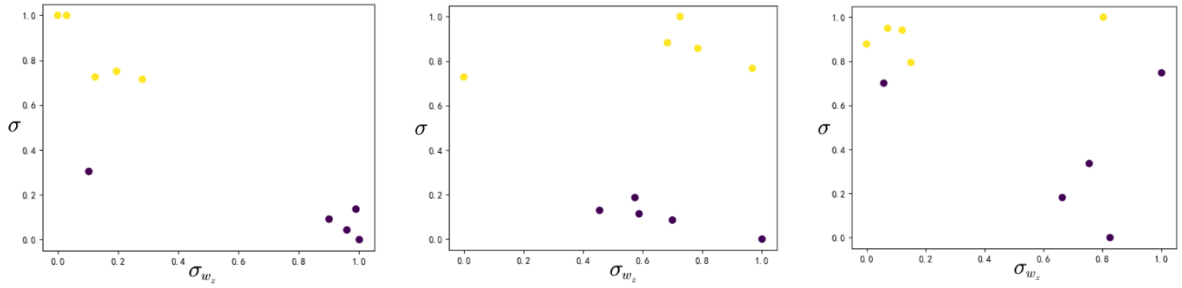
图 13 中横轴是合成加速度序列的均值，纵轴是方差，从上图中可以看出使用合成加速度能够有效区分不同的剧烈程度的运动状态。在本层分类中的先验知识可以描述为合成加速度均值和方差大的代表急促运动，右上角的点代表急促运动。第一层的分类结果如表 3 所示：

表3 急促运动和平缓运动分类结果

实验人员	动作分类	分类结果
Person1	急促运动	SY1，SY15，SY19，SY2，SY23，SY26，SY36，SY43，SY44，SY50
	平缓运动	else
Person2	急促运动	SY1，SY13，SY27，SY28，SY34，SY4，SY42，SY48，SY49，SY50
	平缓运动	else
Person3	急促运动	SY10，SY18，SY22，SY4，SY41，SY5，SY53，SY57，SY58，SY59，
	平缓运动	else

(2)第二层分类时有两个分支，分别时对急促运动和平缓运动再次分类。

i. 急促运动分为跳跃和向前跑，从上一层的分类结果可知根据合成加速度的均值和方差基本能够有效区分跑步和跳跃两种运动状态但还不够准确。



(a) (b) (c)

图14 向前跑和跳跃的聚类结果 (a)Person1; (b)Person2; (c)Person3

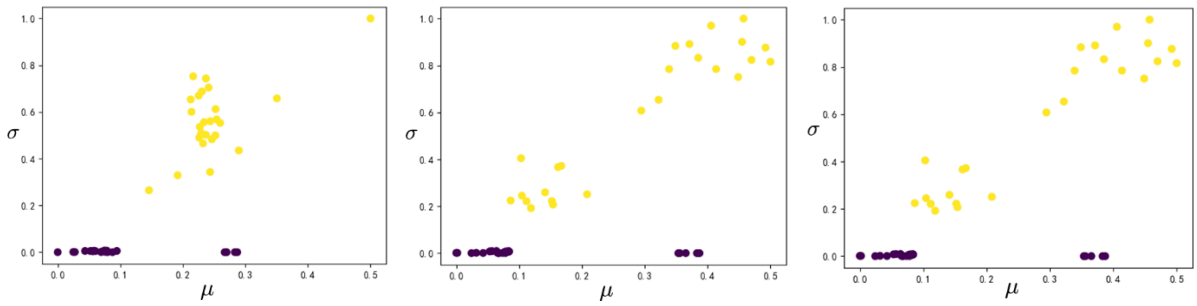
由前文的分析可知跳跃运动状态在 X 方向的加速度变化呈周期性变化，因此将对这些数据进行傅里叶变换可以提取到有效的频域特征。此外，跑步和跳跃时人身体的倾角变化的剧烈程度存在较大的差异，因此可以将绕 Z 轴的角速度的均值和方差作为一个聚类特征。

图 14 中由于用于分类的特征是 5 维的，为了获得更好的可视化结果使用合成加速度和绕 Z 轴角速度的方差作为纵轴和横轴，在这一分支的分类中先验知识可以描述为加速度方差小，沿 X 方向的加速度频率分布较为集中的为跳跃。分类结果如表 4 所示：

表4 向前跑和跳跃动作分类结果

实验人员	动作分类	分类结果
Person1	向前跑	SY19，SY26，SY43，SY44，SY50
	跳跃	SY1，SY15，SY2，SY23，SY36
Person2	向前跑	SY1，SY4，SY48，SY49，SY50
	跳跃	SY13，SY27，SY28，SY34，SY42
Person3	向前跑	SY18，SY22，SY4，SY58，SY59
	跳跃	SY10，SY41，SY5，SY53，SY57

ii. 对于平缓运动的分类又可以分为运动状态和静止状态，聚类示意图如图 15 所示。



(a) (b) (c)

图15 静态动作和动态动作的聚类结果(a)Person1; (b)Person2; (c)Person3

对于静态动作和动态动作同样可以使用合成加速度的均值和方差作为特征进行分类，在第一层分类时急促动作的均值和方差同平缓动作和静态动作的值在做完归一化之后显著大于平缓动作和静态动作的差异。其分类结果如下表 5 所示：

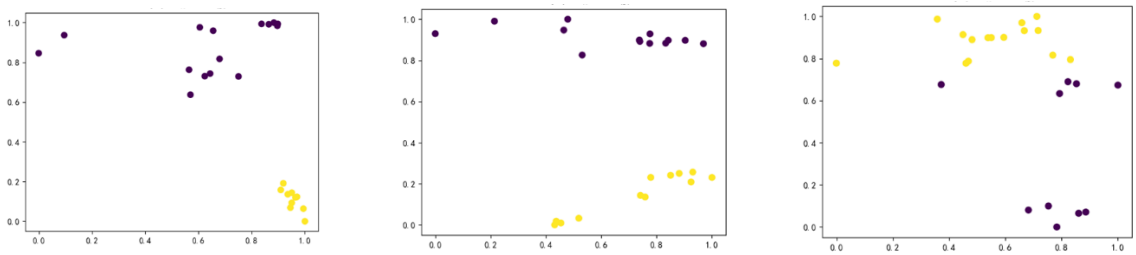
表5 动态动作和静态动作分类结果

实验人员	动作分类	分类结果
Person1	动态动作	SY14，SY17，SY21，SY25，SY27，SY28，SY29，SY31，SY33，SY37，SY38，SY41，SY42，SY46，SY47，SY48，SY49，SY5，SY52，SY53，SY56，SY58，SY59，SY7，SY8
	静态动作	SY10，SY11，SY12，SY13，SY16，SY18，SY20，SY22，SY24，SY3，SY30，SY32，SY34，SY35，SY39，SY4，SY40，SY45，SY51，SY54，SY55，SY57，SY6，SY60，SY9
Person2	动态动作	SY10，SY12，SY16，SY19，SY2，SY20，SY21，SY23，SY24，SY26，SY29，SY3，SY30，SY33，SY37，SY38，SY40，SY43，SY44，SY46，SY47，SY51，SY57，SY60，SY7
	静态动作	SY11，SY14，SY15，SY17，SY18，SY22，SY25，SY31，SY32，SY35，SY36，SY39，SY41，SY45，SY5，SY52，SY53，SY54，SY55，SY56，SY58，SY59，SY6，SY8，SY9

实验人员	动作分类	分类结果
Person3	动态动作	SY11，SY13，SY14，SY19，SY23，SY24，SY25，SY27，SY29，SY30，SY31，SY35，SY36，SY37，SY39，SY42，SY43，SY44，SY49，SY51，SY52，SY56，SY60，SY8，SY9
	静态动作	SY1，SY12，SY15，SY16，SY17，SY2，SY20，SY21，SY26，SY28，SY3，SY32，SY33，SY34，SY38，SY40，SY45，SY46，SY47，SY48，SY50，SY54，SY55，SY6，SY7

(3)第三层中运动状态根据主要有向前走、向右走、上下楼梯五种，大致上又可以区分成两类分在向前和左右走是在水平面上的运动，而上下楼梯是有竖直方向的运动，因而姿态会有倾角；对于静止状态可以分成两类，同样可以根据姿态角进行分类。

i. 对于水平运动和上下楼梯两种运动在加速度解算的姿态角和绕 Z 轴的角速度积分得到的转角可以进行区分，由于楼梯台阶高度通常是固定的因而人在上下楼梯时 X 方向的加速度的频率组成较为集中，聚类结果如图 16。在这个子问题中先验知识可以描述为在 X 方向的姿态角值较大、绕着 Z 轴角速度积分值较大的且频谱分布集中的为爬楼运动否则为水平运动。



(a) (b) (c)

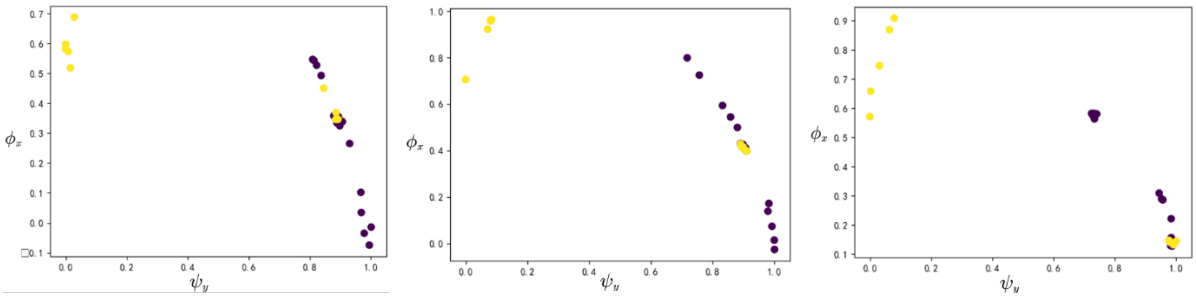
图16 水平运动和爬楼运动的聚类结果 (a)Person1; (b)Person2; (c)Person3

分类结果如表 6 所示：

表6 水平运动和爬楼运动分类结果

实验人员	动作分类	分类结果
Person1	水平运动	SY33,SY38,SY42,SY58,SY7,SY14,SY17,SY28,SY41,SY47,SY46,SY5,SY56,SY59SY8
	爬楼运动	SY21,SY25,SY29,SY48,SY53,SY27,SY31,SY37,SY49,SY52
Person2	水平运动	SY10,SY12,SY2,SY26,SY47,SY16,SY21,SY29,SY40,SY43,SY20,SY3,SY38,SY51,SY60
	爬楼运动	SY24 ,SY33, ,SY44 ,SY57,SY7, SY19,SY23,SY30,SY37,SY46
Person3	水平运动	SY27,SY29,SY51,SY56,SY36,SY14,SY30,SY37,SY43,SY9, SY23,SY39,SY42,SY52,SY8
	爬楼运动	SY13,SY19,SY11,SY60 ,SY49,SY24,SY25,SY31,SY35,SY44

ii. 对于静态状态可以分成两大类休息动作和站立的大类，这一子问题可以根据解算的姿态角的均值和方差进行分类，聚类示意图如图 17 所示:



(a) (b) (c)

图17 休息动作和站立类动作聚类结果(a)Person1; (b)Person2; (c)Person3

在这一子问题中先验知识可以描述为 X 方向姿态角的方差大的便是休息动作。分类结果如表 7 所示：

表7 站立动作和休息动作分类结果

实验人员	动作分类	分类结果
Person1	站立动作	SY18,SY24 ,SY39,SY4,SY57, SY11,SY16,SY3,SY55,SY6, SY45,SY51,SY60,SY30,SY54
	休息动作	SY13,SY10,SY12,SY9,SY20, SY18,SY24 ,SY39,SY4,SY57
Person2	站立动作	SY15,SY25,SY35,SY58,SY6, SY36,SY41,SY55,SY8,SY9, SY14,SY17,SY18,SY45,SY53
	休息动作	SY11,SY22,SY5,SY54,SY56, SY31,SY32,SY39,SY52,SY59
Person3	站立动作	SY12,SY2,SY34,SY47,SY48, SY1,SY26,SY40,SY45,SY7, SY15,SY16,SY20,SY28,SY46
	休息动作	SY11,SY22,SY5,SY54,SY56, SY31,SY32,SY39,SY52,SY59

(4)对于第四层的分类任务都较为简单，对于站立大类的动作对沿 X 方向的加速度进行积分加速度进行积分便可以获得显著的特征，并不需要使用聚类算法。对于左右走的区分只需要对绕 X 轴的角速度进行积分便能够做区分，最终的分类结果如表 8 所示。

3.3 结果展示

对于 180 组数据的聚类分析结果如下：

表8 附件 1 数据分类结果

分类	Person1	Person2	Person3
第 1 类	SY33,SY38,SY42,SY58, SY7,	SY10,SY12,SY2,SY26,SY47	SY27,SY29,SY51,SY56,SY36
第 2 类	SY14,SY17,SY28,SY41, SY47,	SY16,SY21,SY29,SY40, SY43	SY14,SY30,SY37,SY43,SY9
第 3 类	SY46,SY5,SY56,SY59SY8,	SY20,SY3,SY38,SY51,SY60	SY23,SY39,SY42,SY52,SY8
第 4 类	SY21,SY25,SY29,SY48, SY53,	SY24 ,SY33, ,SY44 ,SY57,SY7	SY13,SY19,SY11,SY60 ,SY49

分类	Person1	Person2	Person3
第 5 类	SY27,SY31,SY37,SY49,SY52,	SY19,SY23,SY30,SY37,SY46	SY24,SY25,SY31,SY35,SY44,
第 6 类	SY15,SY19,SY26,SY43,SY44,SY50,	SY1,SY4,SY48,SY49,SY50,	SY10,SY18,SY22,SY41,SY57,SY58,SY59,
第 7 类	SY1,SY2,SY23,SY36,	SY13,SY27,SY28,SY34,SY42	SY4,SY5,SY53
第 8 类	SY13,SY10,SY12,SY9,SY20	SY11,SY22,SY5,SY54,SY56	SY17,SY21,SY33,SY38,SY6
第 9 类	SY18,SY24 ,SY39,SY4,SY57,	SY15,SY25,SY35,SY58,SY6	SY12,SY2,SY34,SY47,SY48
第 10 类	SY22,SY32,SY34,SY35,SY40,	SY31,SY32,SY39,SY52,SY59	SY3,SY32,SY50,SY54,SY55
第 11 类	SY11,SY16,SY3,SY55,SY6	SY36,SY41,SY55,SY8,SY9	SY1,SY26,SY40,SY45,SY7
第 12 类	SY45,SY51,SY60,SY30,SY54	SY14,SY17,SY18,SY45,SY53	SY15,SY16,SY20,SY28,SY46

4 基于深度集成网络的人体运动状态识别

本研究使用附件 2 中记录的 10 名实验人员的活动数据，数据包括每位实验人员在不同活动状态下的 5 组加速度计和陀螺仪数据，每组数据明确标注了对应的活动状态。基于此数据，本章旨在利用附件 2 中的活动数据，提取出 12 种典型的人员活动状态特征，并建立一个有效的人员活动状态判别模型。

4.1 深度集成网络训练

在本次建模任务中，我们将问题转化为一个典型的图像分类任务。深度学习模型，特别是卷积神经网络（Convolutional Neural Networks, CNNs），因其卓越的性能，被广泛应用于图像分类任务中。CNN 能够有效地从图像数据中自动提取和学习有用的特征，通过适当的训练，这些模型可以展现出优良的泛化能力，能够对未见过的图像数据做出准确的预测。

进一步来说，我们还可以利用预训练的深度学习模型进行迁移学习。迁移学习是一种利用在大规模数据集（如 ImageNet 数据集）上预训练的模型，将其应用到特定任务中的技术。这样可以显著提高模型的识别能力和训练效率，同时也能减少所需的训练时间和计算资源。这些预训练模型已经在大量数据上学习到了丰富的特征表示，因此在新的任务中，我们只需进行微调即可使其适应新的数据分布。

基于以上考虑，本节选用深度学习模型来对多维特征图像进行分类。具体地，我们在三个预训练模型上进行训练和调整，这三个模型分别是：VGG19、ResNet-101 和 EfficientNet-b3。VGG19 以其深层网络结构和简单易懂的设计受到广泛关注；ResNet-101 通过引入残差连接，解决了深层网络中的梯度消失问题，显著提高了模型的性能；而

EfficientNet-b3 则在模型尺寸和计算效率之间实现了良好的平衡，是近年来深度学习模型架构设计的一大进步。

通过对这三种预训练模型进行训练和调整，我们期望能够充分利用它们各自的优点，提升模型在多维特征图像分类任务中的表现，并最终实现对未见过数据的高准确度预测。这不仅验证了深度学习方法在图像分类任务中的有效性，也为解决类似问题提供了新的思路和方法。

4.1.1 VGG19 深度网络

VGG（Visual Geometry Group）网络是一种经典的卷积神经网络（Convolutional NeuralNetwork, CNN）架构，由牛津大学的 Visual Geometry Group 提出。如图 18 所示，该网络架构的主要创新在于其采用多个连续的小卷积核（通常为 3x3 大小）来替代传统的较大卷积核。这一设计选择不仅简化了模型的计算复杂度，还使得网络能够更有效地捕捉和学习图像中的细粒度特征。

VGG 网络通过重复堆叠卷积层和池化层来逐步增加网络的深度，从而提升其特征提取能力和表达能力。具体而言，卷积层负责提取输入图像的局部特征，而池化层则用于下采样，减少特征图的尺寸，同时保持重要信息。这样的结构设计能够有效地保留图像的空间信息和语义信息，并逐层提取出更抽象、更具代表性的特征。

VGG 网络的最著名变体是 VGG16 和 VGG19，其中的数字代表网络中包含的权重层（即卷积层和全连接层）的数量。VGG16 包含 13 个卷积层和 3 个全连接层，而 VGG19 则包含 16 个卷积层和 3 个全连接层。这些变体通过增加网络的深度，提高了模型的复杂性和表现力，使其在各类图像分类任务中表现出色。

此外，VGG 网络在设计时还采用了一些优化策略，如在每个卷积层后面加上 ReLU（Rectified Linear Unit）激活函数，以引入非线性，增强模型的表达能力；在池化层前增加多个卷积层，提升特征提取的丰富度。这些策略进一步提升了 VGG 网络的性能和泛化能力，使其成为深度学习领域的一个重要里程碑，并广泛应用于计算机视觉任务中。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

图18 VGG19 网络架构示意图[20]

4.1.2 Resnet101

ResNet（Residual Network）是一种由微软研究院开发的深度卷积神经网络，如图 19 所示，ResNet 的核心创新在于引入了“残差学习”（Residual Learning）的概念，这一创新通过“跳跃连接”（skip connections）使得网络中的信息可以绕过某些层直接传递给后续层。

具体而言，在传统的神经网络中，输入数据依次通过每一层进行处理。然而，当网络层数增加时，梯度消失和梯度爆炸问题变得越来越严重，导致训练过程变得困难。为了解决这一问题，ResNet 在网络结构中加入了残差块（residual blocks）。在每个残差块中，输入数据不仅传递给下一层进行卷积处理，还通过跳跃连接直接加到后续层的输出上。这种设计使得每一层实际上学习的是输入和期望输出之间的残差（即差值），而不是直接学习复杂的映射函数。

跳跃连接的引入具有多方面的优势。首先，它能够有效缓解深层网络中的梯度消失问题，使得梯度可以更有效地传递回前面的层，从而稳定了深层网络的训练过程。其次，它有助于解决梯度爆炸问题，通过直接传递输入信息，避免了梯度在层层传递过程中可能出现的指数级增长。此外，残差学习还能够提升模型的表达能力，使得网络可以在不增加复杂性的前提下，学习到更加丰富和精细的特征。

ResNet 的这一创新使得构建非常深的网络成为可能，例如 ResNet-50、ResNet-101 甚至 ResNet-152，这些网络结构分别包含 50 层、101 层和 152 层权重层。尽管层数显著增加，但通过残差块的设计，这些深度网络仍能高效地训练，并在各种图像识别任务中取得了卓越的性能。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

图19 Resnet101 网络架构示意图[21]

4.1.3 Efficient-b3

EfficientNet 是一种高效的卷积神经网络架构，通过综合优化网络的深度、宽度和输入图像的分辨率来提升模型的性能和效率。EfficientNet 的设计基于一个基础模型（EfficientNet-B0），并通过对这三个维度进行复合缩放（compound scaling）来创建一系列更大的模型（例如，EfficientNet-B1 到 EfficientNet-B7）。

如图 20 所示，是 EfficientNet 的核心创新之一。传统的神经网络通常在单一维度上进行缩放，例如增加深度、宽度或分辨率。而 EfficientNet 提出了一种更加系统化的方法，通过复合缩放同时均衡地调整这三个维度，从而在提升模型准确性的同时保持计算效率。具体而言，复合缩放方法通过固定一组缩放系数，依据这些系数同步调整网络的深度、宽度和分辨率，使得模型在不同规模下都能达到最优的性能表现。

EfficientNet-B0 是基于神经架构搜索（Neural Architecture Search, NAS）自动生成的基础模型，通过大量实验和优化，选择出具有最佳性能和效率的网络结构。在此基础上，通过复合缩放方法，进一步构建了 EfficientNet-B1 到 EfficientNet-B7，这些模型在多个图像分类任务中表现优异，显著优于传统的卷积神经网络。

在我们的研究中，我们选用了 EfficientNet 的一个变体——EfficientNet-B3。EfficientNet-B3 在保持计算效率的同时，进一步提升了模型的表现能力。其网络结构在深度、宽度和分辨率上相较于基础模型 B0 有适度增加，使其能够捕捉更丰富的图像特征，并在图像分类任务中取得更高的准确率。

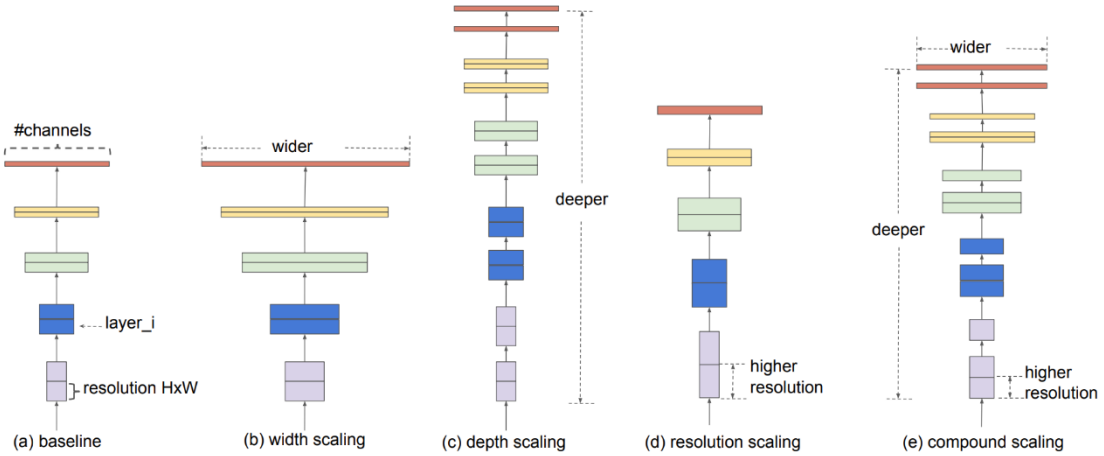


图20 Efficient-bx 架构示意图[22]

4.1.4 集成学习

在完成对四个模型的训练之后，我们将这四个模型分别记为 M1、M2、M3 对于任意一张输入图像 x，我们需要对其进行标签预测。为此，如图 21 所示，我们采用了一种集成学习的方法，以充分利用每个模型的预测能力，从而提高整体预测的准确性和稳健性。

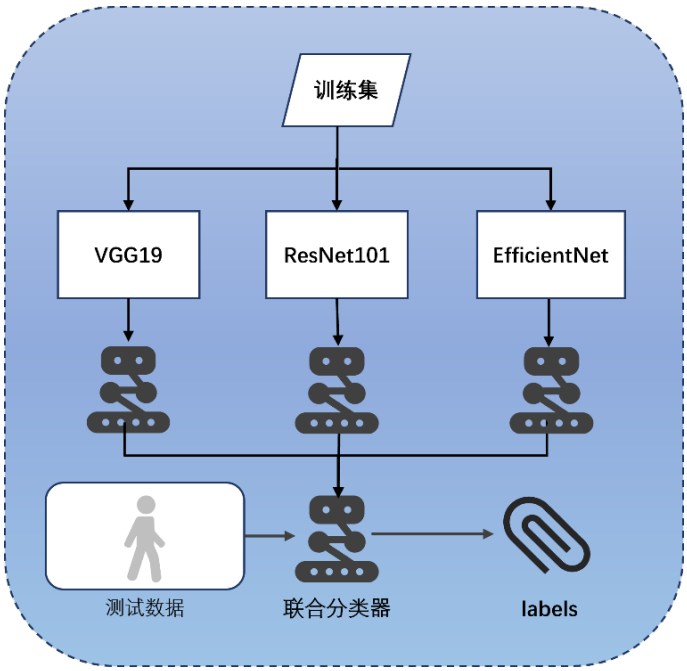


图21 深度集成模型架构

具体地，对于输入图像 x，我们首先将其分别输入到模型 M1、M2、M3 中，以获取每个模型对图像 x 的预测输出。记这些输出分别为 y1、y2、y3，其中 yi 表示模型 Mi 对图像 x 预测的标签。我们使用了堆叠集成的方法，使用了一个额外的模型（称为元学习器）来学习如何组合 M1、M2、M3 的输出，以产生最终的预测结果。元学习器可以基于各个模型的输出特征，进一步提升预测的准确性。使用元学习器具有以下优势：

① 提高准确性：通过综合多个基模型的预测，元学习器可以有效地提升整体模型的准确性和鲁棒性。

② 降低过拟合风险：不同的基模型可能在不同的数据子集上表现良好，元学习器能够平衡各个模型的偏差和方差，从而减少过拟合的风险。

③ 灵活性：元学习器可以使用不同类型的模型，这使得它在处理复杂数据时具备很高的灵活性。

我们选用 SVM 作为元学习器，集成预测得到不同标签：

$$y_{pred} = SVM(M_1(x) + M_2(x) + M_3(x)) \tag{20}$$

4.2 深度网络测试结果

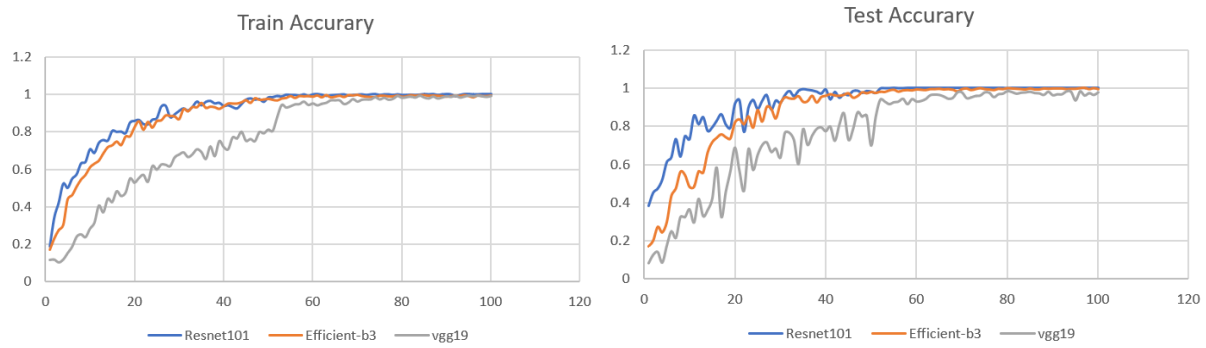
在本次测试中，我们采用模型的准确率作为主要的性能衡量标准，准确率是评价分类模型效果的重要指标，能够反映模型在所有预测样本中正确预测的比例。同时我们给出了测试模型的混淆矩阵作为一个更加直观更加详细地分析依据。

我们对三个所选的模型的训练数据和测试数据进行了记录，如图 22 所示，在训练过程中，我们观察到了 ResNet101、EfficientNet-b3 和 VGG19 这三个深度集成网络模型在训练集和测试集上的表现。首先，ResNet101 在训练集上展现出了快速的准确度提升，特别是在经过大约 60 个 epoch 后，准确度迅速接近 1.0，显示出其强大的拟合能力和学习能力。EfficientNet-b3 模型同样表现出了快速的训练准确度上升，其表现接近甚至超过了 VGG19，在训练末期达到接近 100%的准确度水平。

然而，模型在测试集上的表现展现了一些差异和趋势。ResNet101 在测试集上的初始表现非常出色，但随着训练的进行，准确度出现了轻微波动，并最终稳定在一个略低于训练准确度的水平，这可能暗示了轻微的过拟合现象。相比之下，VGG19 模型在开始阶段的表现较为低迷，但随后迅速提升，并保持了相对稳定的表现。然而，它在测试准确度方面是三个模型中最低的，这表明其泛化能力可能不如其他两个模型。

EfficientNet-b3 在测试集上的表现也开始时较为低下，但随着训练的继续，准确度迅速上升，并在后续步骤中趋于稳定。最终，它的测试准确度接近于训练准确度，这显示出了它优异的泛化能力和稳健性。

总的来说，ResNet101 和 EfficientNet-b3 展现了更好的泛化能力，尽管 ResNet101 可能会稍微过拟合。而 VGG19 在测试集上的表现相对较弱，可能需要更多的调优或者数据增强来改善其泛化能力。这些观察结果为我们进一步优化模型和提升解决问题的效果提供了有价值的见解和指导。

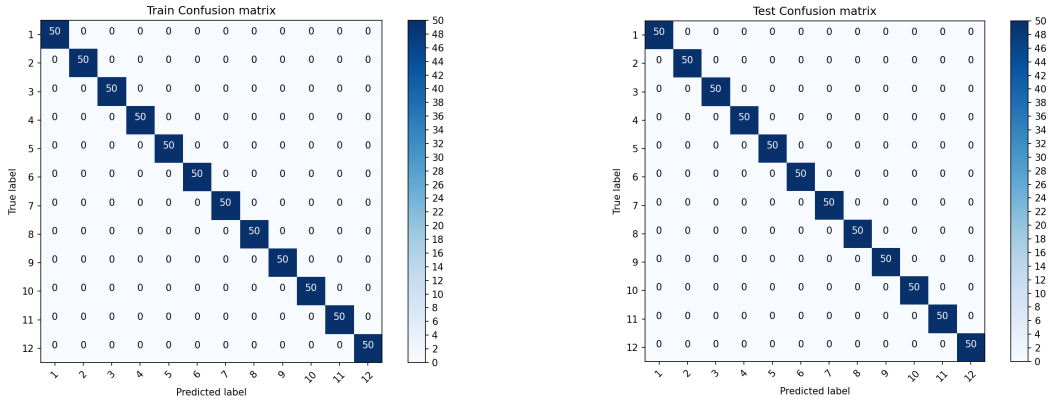


(a) (b)

图22 不同模型的训练和测试可视化

对于三个模型集成后的训练集和测试集，我们进行了混淆矩阵的分析，结果如图 23 所示。观察到的显著现象是，在训练集和测试集上，所有模型的准确率都达到了 100%。

这表明在训练阶段，集成模型能够完美地学习和拟合训练数据，每个模型的预测与实际标签完全匹配。这种结果通常意味着模型在训练数据上没有出现过拟合或者欠拟合的情况，能够充分利用数据的信息进行有效学习。在测试阶段，模型同样表现出了卓越的能力。

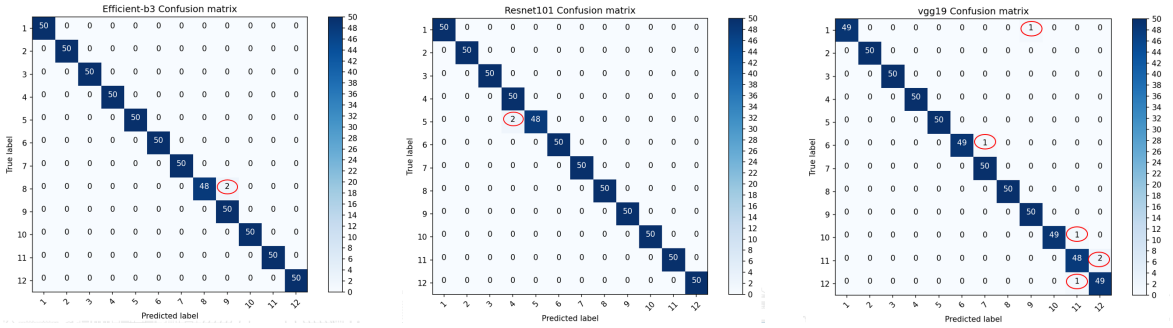


(a) (b)

图23 训练和测试阶段的混淆矩阵

(a) 训练集分类混淆矩阵; (b) 测试集分类混淆矩阵

在我们的研究中，我们采用了集成学习的方法来提升模型的性能。集成学习的核心思想是利用多个学习器的优势，通过各自的预测结果的综合，提高整体预测的稳定性和准确性。在我们的实验中，集成模型展现出了对数据特征和难以捕捉的模式更好的学习能力，从而在各个方面都超过了单个模型的表现。通过消融实验的设计，我们能够明确地分析集成模型与单个模型之间的性能差异，进一步验证了集成学习的有效性和优越性。如图 24 所示，这些实验结果不仅证明了我们方法的有效性，也为未来优化模型和提升解决问题能力提供了重要的指导和启示。



(a) (b) (c)

图24 单模型的混淆矩阵 (a) Efficient 模型混淆矩阵;

(b)Resnet101 模型混淆矩阵; (c) VGG19 模型混淆矩阵

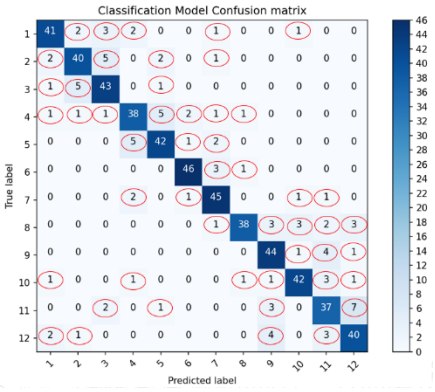


图25 问题 1 所提方法的分类结的混淆矩阵

4.3 深度集成网络模型与基于谱聚类的逆向层次聚类模型结果对比

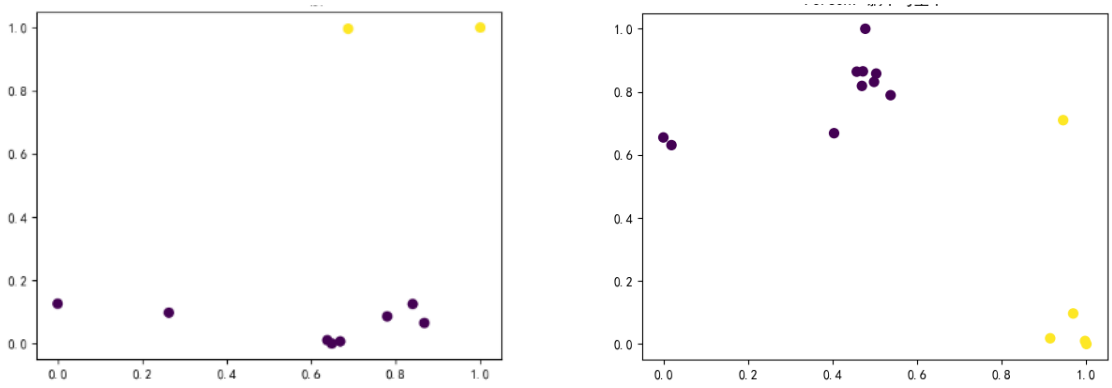
采用基于谱聚类的逆向层次聚类的分类结果的混淆矩阵如图 25 所示，问题 1 和问题 2 所提方法的精度对比如表 9 所示：

表9 两种方法精度对比

状态	逆向层次聚类	深度集成网络模型	状态	逆向层次聚类	深度集成网络模型
1	82%	100%	7	90%	100%
2	80%	100%	8	76%	100%
3	86%	100%	9	88%	100%
4	76%	100%	10	84%	100%
5	84%	100%	11	74%	100%
6	92%	100%	12	80%	100%

基于谱聚类的逆向层次聚类算法的总的识别准确率为 82.7%，而基于深度集成网络的识别准确率高达 100%。基于聚类的方法在识别滴 4 类和第 8 类时的识别准确率最低，两类典型错误分类示意图如图 26 所示，在识别第 4 和第 8 类时都用到了原始数据进一步变换的数据，例如在区分上下楼时会用到了 X 轴方向加速度序列的均值和方差以及对加速度的积分，由于传感器存在一定的误差积分过程回对误差进行累积，并且人在上楼时需要克服重力做功因而加速度的变化剧烈程度同下楼有着较大的区别，这使得分类结果存在较大的偏差，下图所示一个具有代表性聚类出现错误的结果。对于坐下这一动

作使用姿态角和 Z 轴角速度来同躺下进行区分，陀螺仪通常会存在零点漂移等误差积分会使得这些误差不断累积最终导致识别错误。



(a) (b)

图26 两个典型的错误分类结果 (a)步行上楼和步行下楼; (b)躺下和坐下

总体而言人工构造特征使用传统的算法进行分类时鲁棒性不如基于深度学习的算法。传感器天然存在误差，并且误差的组成、来源并没有在构造特征时深入考虑，导致分类结果不理想，而基于深度的方法并不存在这些问题，可以根据数据集进行有效的学习将特征升到足够高的维度中，以实现有效的分类。并且在基于聚类的算法中面对多源数据时并没有很好的将特征融合，而是简单的将数据做一个极大-极小的归一化。

4.4 深度集成网络模型对人体活动状态识别

针对附件三中提供的 30 次实验数据，我们首先依据前文提到的有效数据特征维度进行计算。这一步骤的目的是通过扩展原始数据的特征维度，增强不同数据记录之间的差异性，确保后续分析的准确性和有效性。

在扩展数据维度之后，我们采用线性平滑插值的方法，顺利生成了各实验数据对应的特征图像。这种插值方法通过平滑处理，有效地保留了原始数据的关键特征，同时消除了可能存在的噪声和不连续性，保证了特征图像的清晰度和可靠性。

图 27 展示了通过这种方法得到的特征图像，每幅图像都代表了一个实验数据的特征分布。通过观察这些图像，我们可以直观地看到各次实验数据在不同特征维度上的分布情况，这为我们后续的深度集成网络训练和模型优化提供了重要的可视化支持。

特征图像的生成不仅提高了数据的可解释性，还为模型的训练提供了更加丰富和有价值的信息。通过这些图像，深度集成网络能够更好地理解和学习不同实验数据之间的特征差异，从而提升模型的整体性能和预测准确性。

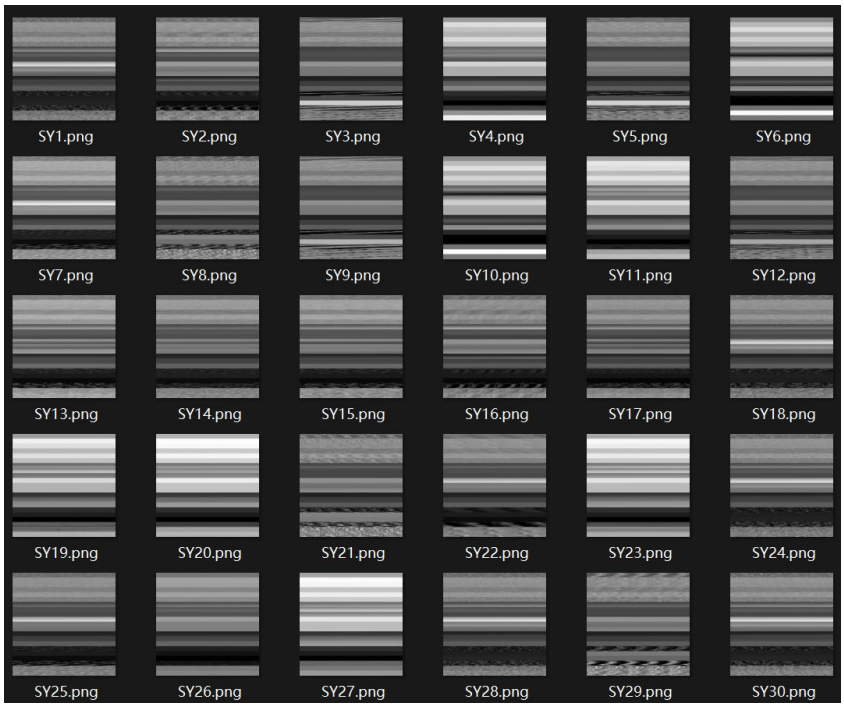


图27 附件 3 活动状态的多维特征图

使用我们训练得到的深度集成网络，我们预测了这 30 次实验数据所对应的活动状态编号，如表 10 所示。

表10 深度集成网络模型预测 30 组数据结果

活动类型	判别状态	活动类型	判别状态
SY1	5	SY16	1
SY2	1	SY17	4
SY3	7	SY18	5
SY4	11	SY19	8
SY5	7	SY20	8
SY6	10	SY21	6
SY7	1	SY22	2
SY8	6	SY23	8
SY9	7	SY24	5
SY10	10	SY25	2
SY11	9	SY26	9
SY12	7	SY27	8
SY13	4	SY28	5
SY14	3	SY29	6
SY15	4	SY30	5

5 人体活动识别与年龄、身高、体重等因素关系探究

在处理附件 4 中的不同人员的年龄、身高、体重等信息时，我们考虑了两种方法来处理这些数据，以便更好地应用于模型训练和预测。第一种方法是将年龄、身高、体重整合成一个元组，然后使用哈希表将其映射到一个无冲突的值，作为当前数据的标签值进行训练。这种方法的优势在于能够通过一次训练得到一个可以直接使用的模型，简化了后续的预测过程。然而，这种方法有一个显著的局限性，即它需要大量的数据支持。如果数据量不足，训练出的模型很可能是过于个性化的，缺乏泛化能力，难以在新数据上表现良好。因此，在数据量较少的情况下，这种方法并不适用于当前问题。

为了克服上述方法的局限性，如图 28 所示，我们提出了第二种处理方式，即分化处理。具体而言，我们为年龄、身高、体重的数据分别建立独立的深度集成网络判别模型。在实际应用中，对于一个测试数据，这三个独立的模型将分别输出预测的年龄、身高和体重。这种方法不仅提高了模型的泛化能力，还能够更准确地进行人员画像，提供更精细的个体特征预测。

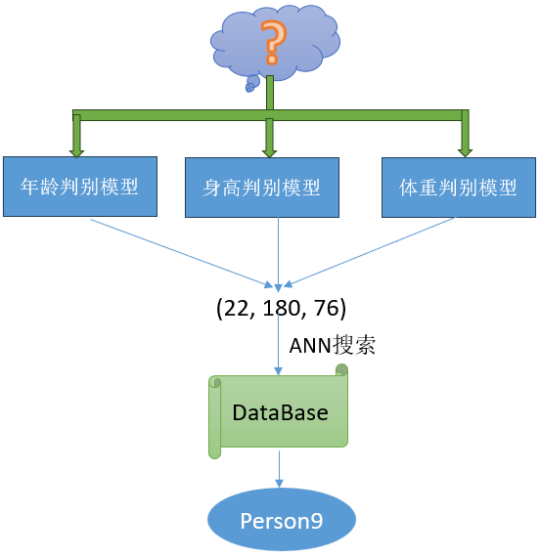


图28 分化处理预测数据来源人员

在进一步的分析中，我们发现，对于当前的数据集，只需要对原始数据中的 6 个维度特征进行模型训练，便能够准确地预测人员的年龄、身高和体重等信息。数据库中不同年龄、身高、体重的 6 维度特征图像如图 29、30、31 所示。这表明，通过合理的特征选择和模型设计，可以在保持数据简洁性的同时，实现高精度的预测。

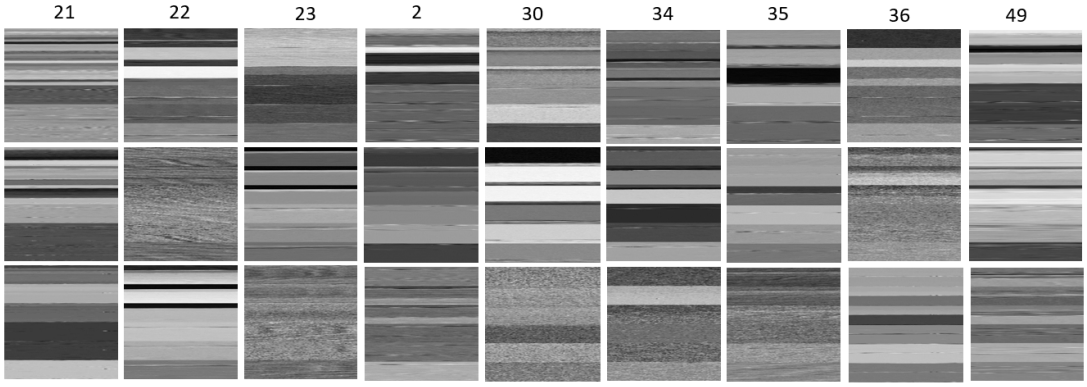


图29 不同年龄的部分 6 维度特征图像

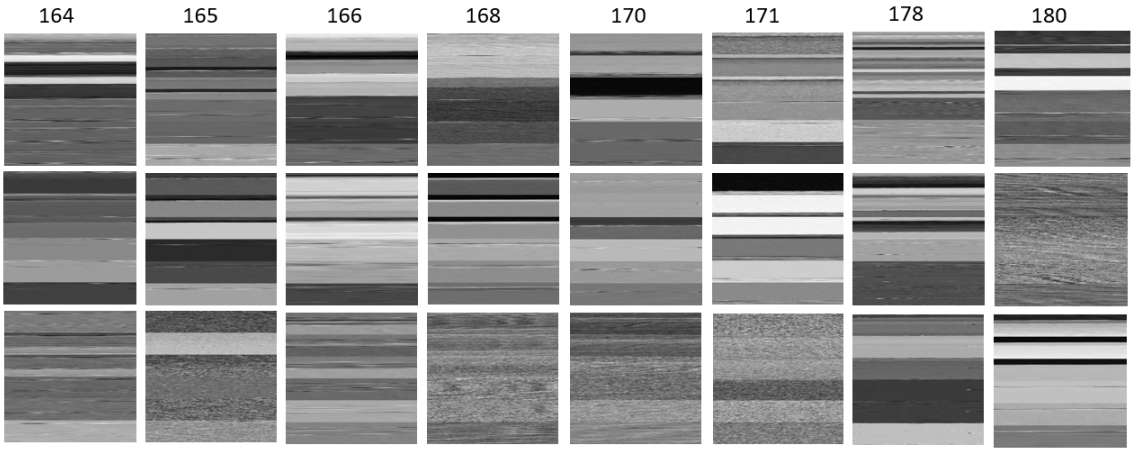


图30 不同身高的部分 6 维度特征图像

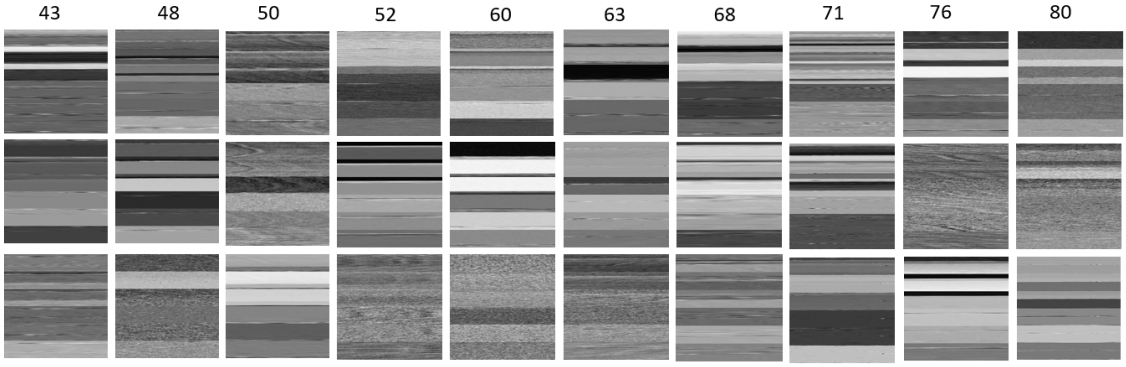


图31 不同体重的部分 6 维度特征图像

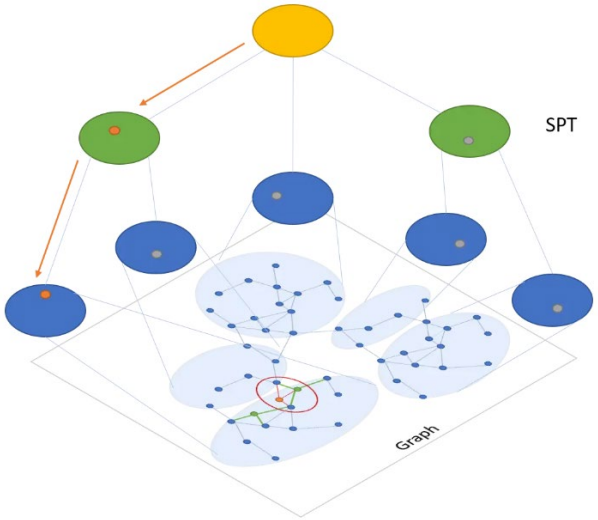


图32 ANN 搜索算法的架构

此外，在解决问题 3 中实验人员来源确定的任务时，我们采用了近似近邻（Approximate Nearest Neighbor，ANN）算法。这种算法通过在已有数据库中查找最相似的向量，快速而高效地确定测试数据的来源人员。ANN 算法的搜索架构如图 32 所示，ANN 算法在处理大规模数据时表现尤为出色，它能够在保证较高准确率的同时显著降低计算复杂度和时间成本。我们的预测结果如表 11 所示。

表11 实验员判断结果

活动类型	预测年龄	预测身高	预测体重	库中最相似特征	判别结果
Unknow1	49	166cm	68kg	(49,166,68)	Person10
Unknow2	35	170cm	63kg	(35,170,63)	Person7
Unknow3	27	164cm	50kg	(27,164,50)	Person6
Unknow4	22	180cm	76kg	(22,180,76)	Person9
Unknow5	36	170cm	80kg	(36,170,80)	Person13

6 模型评价

6.1 模型的优点

(1) 问题 1 中提出的基于谱聚类的逆层次聚类模型根据现实生活中对各种运动状态的感性的认识，同时考虑了加速度计和陀螺仪测量数据的分布情况，分层提取出了各类运动的显著特征，模型的输出基本符合题目要求，能解决实际问题。

(2) 基于谱聚类的逆层次聚类模型结构简单对于类别数目较多的无监督分类任务提供了一种富有吸引力的解决方案，采用该方法可以在一定程度上缓解多源信息融合的带来的麻烦。

(3) 问题 2 中构建的深度集成模型，不仅有深度网络强大的拟合能力，同时集成不同的模型能在一定程度上缓解深度网络泛化能力差的问题，最终实现在具有较高的准确率的同时还能保持相当的泛化能力。

(4) 使用深度集成神经网络能够有效地提高模型的检测准确率，减少样本误分的数目。

(5) 通过分化处理分别建立集成判别模型，充分利用了有限的数据，增强了泛化能力。

6.2 模型的不足

(1) 基于谱聚类的逆层次聚类模型虽然能基本符合题目要求，但在个别动作上的识别仍有不足，如部分实验数据中无法较好识别出上楼动作，模型准确度仍有提升空间。

(2) 基于深度集成网络的人体运动状态识别模型虽然识别准确度高，继承了各网络的优点，但模型训练依赖大量数据训练，所需算力较大，可解释性有待提高。

6.3 模型的改进

对于问题 1 中提出的基于谱聚类的逆向层次聚类方法，我们考虑进一步采用基于简单核的聚类分析算法来替换原有的谱聚类方法。这一改进能够有效地将数据特征映射到更高维的、易于分类的空间中，同时，此方法也适应了多源信息融合的需求，增强了模型在处理复杂数据环境中的适应性和鲁棒性。

对于问题 2 提出的深度集成网络模型，我们考虑进一步使用元组标签哈希映射的方法进行深度集成网络模型的训练。这种方法允许我们以端到端的方式处理大规模数据，避免了为每个预测目标单独建立模型的需要。通过这种整体训练策略，可以显著提高模型的训练效率和预测精度，同时简化模型的维护和更新过程。这些改进将极大地增强模型的实用性和扩展性，为复杂问题提供更加精确和高效的解决方案。

参考文献

[1] Chen L, Hoey J, Nugent C D, et al. Sensor-based activity recognition[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2012, 42(6): 790-808.

[2] Cook D J, Das S K. Pervasive computing at scale: Transforming the state of the art[J]. Pervasive and Mobile Computing, 2012, 8(1): 22-35.

[3] Campbell A, Choudhury T. From smart to cognitive phones[J]. IEEE Pervasive Computing, 2012, 11(3): 7-11.

[4] Clarkson B P. Life patterns: structure from wearable sensors[D]. Massachusetts Institute of Technology, 2002.

[5] Avci A, Bosch S, Marin-Perianu M, et al. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey[C]//23th International conference on architecture of computing systems 2010. VDE, 2010: 1-10.

[6] Lin W, Sun M T, Poovandran R, et al. Human activity recognition for video surveillance[C]//2008 IEEE international symposium on circuits and systems (ISCAS). IEEE, 2008: 2737-2740.

[7] Lara O D, Labrador M A. A survey on human activity recognition using wearable sensors[J]. IEEE communications surveys & tutorials, 2012, 15(3): 1192-1209.

[8] Mannini A, Sabatini A M. Machine learning methods for classifying human physical activity from on-body accelerometers[J]. Sensors, 2010, 10(2): 1154-1175.

[9] Poppe R. A survey on vision-based human action recognition[J]. Image and vision computing, 2010, 28(6): 976-990.

[10] Nham B, Siangliulue K, Yeung S. Predicting mode of transport from iphone accelerometer data[J]. Machine Learning Final Projects, Stanford University, 2008.

[11] Intille E M T S S, Larson K. Activity recognition in the home using simple and ubiquitous sensors[C]//Proceedings of Second International Conference on Pervasive Computing. 2004: 158-175.

[12] Bao L, Intille S S. Activity recognition from user-annotated acceleration data[C]//International conference on pervasive computing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004: 1-17.

[13] Aggarwal J K, Ryoo M S. Human activity analysis: A review[J]. Acm Computing Surveys (Csur), 2011, 43(3): 1-43.

[14] Hassan MM, Uddin MZ, Mohamed A, et al. A robust human activity recognition system using smartphone sensors and deep learning. Future Generation Computer Systems, 2018, 81: 307–313.

[15] Reyes-Ortiz JL, Oneto L, Samá A, et al. Transition-aware Software Technique Algorithm human activity recognition using smartphones. Neurocomputing, 2015, 171: 754–767.

[16] Chernbumroong S, Atkins AS, Yu HN. Activity classification using a single wrist-worn accelerometer. Proceedings of the 5th International Conference on Software, Knowledge Information, Industrial Management and Applications (SKIMA). Benevento, Italy. 2011. 1–6.

[17] Chen Z H, Zhang L, Cao Z G, et al. Distilling the knowledge from handcrafted features for human activity recognition[J]. IEEE Transactions on Industrial Informatics, 2018, 14(10): 4334-4342.

[18] Alejandro B, Yago S, Pedro I. Evolutionary design of convolutional neural networks for human activity recognition in sensor-rich environments[J]. Sensors, 2018, 18(4): 1288-1312.

[19] Lee S M, Yoon S M, Cho H.Human activity recognition from accelerometer data using convolutional neural network[C]//IEEE International Conference on Big Data and Smart Computing.Jeju: IEEE, 2017: 131-134.

[20] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[21] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[22] Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks[C]//International conference on machine learning. PMLR, 2019: 6105-6114.

[23] 孙宇航,周建钦,张学锋.基于加速度传感器的人体运动模式识别[J].计算机系统应用, 2020, 29(6):8.DOI:10.15888/j.cnki.csa.007443.

[24] 赵阳, 娄小平, 刘锋, 等. 自适应 MEMS 加速度计滤波算法[J]. 传感器与微系统, 2016, 35(11): 120-122.

附　录

附录 A：支撑材料列表

支撑材料列表

序号	文件名	材料说明
1	Based_on_SC_InverHericaClusting.zip	基于谱聚类的逆向层次聚类算法源代码
2	Deep_Boost_Neural_Network.zip	深度集成网络的源代码

附录 B：主要程序/关键代码

代码环境	操作系统: Windows11 编程语言: Python 3.11 编辑器: PyCharm 2022

1. 基于谱聚类的逆向层次聚类算法代码：

(1)特征提取函数：

```
1.def FFT(Fs, data):##计算频域特征
2.    """
3.    对输入信号进行 FFT
4.    :param Fs:  采样频率
5.    :param data:待 FFT 的序列
6.    :return:
7.    """
8.    L = len(data)  # 信号长度
9.    N = np.power(2, np.ceil(np.log2(L)))  # 下一个最近二次幂，也即N 个点的FFT
10.       result = np.abs(np.fft.fft(a=data, n=int(N))) / L * 2
11.       axisFreq = np.arange(int(N / 2)) * Fs / N
12.       result = result[range(int(N / 2))]
13.       return axisFreq, result

1.def AccSynthesis(Acc_XYZ,namelist):##计算合成加速度
2.    if len(namelist)==3:
3.        Acc_X=Acc_XYZ[namelist[0]].values
4.        Acc_Y = Acc_XYZ[namelist[1]].values
5.        Acc_Z = Acc_XYZ[namelist[2]].values
6.        Acc_Syn=np.sqrt(np.square(Acc_X)+np.square(Acc_Y)+np.square(Acc_Z))
7.    if len(namelist)==2:
8.        Acc_X=Acc_XYZ[namelist[0]].values
```

```
        Acc_Y = Acc_XYZ[namelist[1]].values
            Acc_Syn=np.sqrt(np.square(Acc_X)+np.square(Acc_Y))
        mu,sigma=GetSdata(Acc_Syn)
        Acc_Syn_F=np.array([mu,sigma])
        return Acc_Syn_F

def Normalizetion(data):##归一化
    Ndata=(data-min(data))/(max(data)-min(data))
    return Ndata

def GetSdata(data):###计算均值和方差
    Mu=np.mean(data)
    Sigma=np.var(data)
    return Mu,Sigma

def butter_lowpass_filter(data, cutoff_freq, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff_freq / nyquist
    b, a = signal.butter(order, normal_cutoff, btype='low', ana-
log=False)
    return signal.filtfilt(b, a, data)

def GetPosture(Acc_XYZ,namelist):##解算姿态角
    Acc_X = Acc_XYZ[namelist[0]].values
    Acc_Y = Acc_XYZ[namelist[1]].values
    Acc_Z = Acc_XYZ[namelist[2]].values
    Pos=np.zeros((np.size(Acc_X),3))
    for i in range(np.size(Acc_X)):
        Pos[i, :] = np.ar-
ray([np.arctan(Acc_X[i] / np.sqrt(Acc_Z[i]**2+Acc_Y[i]**2)),
                                np.arctan(Acc_Y[i] / np.sqrt(Acc_Z[i]**
2+Acc_X[i]**2)),
                                np.arctan( np.sqrt(Acc_Y[i]**2+Acc_X[i]
**2)/ Acc_Z[i])])
    return Pos
```

(2)算法主要函数

```
def getOnePersonFeature(personnum=None,IS_Normalizarion=1):
    Peole='Person'+str(personnum)
    namelist = os.listdir(base_ptah+Peole)
    namelist = os.listdir(base_ptah)
    Rname=[]
    FeatureMap=np.zeros((len(namelist),2))
    i=0
    for name in namelist:
        if name[0:2]!='~$':
            ModePath=base_ptah+Peole+'/'+name
            data=pd.read_excel(ModePath,usecols=columns_to_read)
        acc_syn_F=AccSynthesis(data,columns_to_read[0:3])
```

```
            FeatureMap[i,:]=acc_syn_F
            i = i + 1
        if IS_Normalizarion==1:
            for i in range(2):
                    FeatureMap[:,i]=Normalizetion(FeatureMap[:,i])
        PATH=base_ptah+Peole
        # PATH = base_ptah
        return FeatureMap,namelist,PATH

def DisFastMoveAndElse(People_num=None):
    '''

    :param People_num:
    :return: 返回快速运动文件名
    '''
    PEOPLE='Person'+str(People_num)
    Acc_data,Modelist,Path2Person=getOnePersonFeature(personnum=Peo-
ple_num,IS_Normalizarion=1)
    # print(len(Acc_data))
        print('Feature Extraction Done!')
        y_pred = SpectralClustering(n_clusters=2, ran-
dom_state=0, gamma=5, affinity='rbf').fit_predict(Acc_data)
        plt.figure(1)
        plt.scat-
ter(Acc_data[:, 0], Acc_data[:, 1], c=y_pred, s=50, cmap='viridis')
        plt.title(PEOPLE+' 急促运动与平缓运动')
        center=np.zeros((2,2))
        for i in range(2):
            cu=np.mean(Acc_data[:,0][y_pred==i])
            cs= np.mean(Acc_data[:,1][y_pred == i])
            center[i,:]=np.array([cu,cs])
        for i in range(len(y_pred)):
            print('Real:',Modelist[i],'S-
C:',y_pred[i],'val:',Acc_data[i])
        FAST=y_pred[y_pred==1]
        ##问题1 和2
        if len(FAST)!=10:
            slowD=Acc_data[y_pred==0]
            need=10-len(FAST)
            Sort=Acc_data[:,0].argsort(-1)[::-1]
            F_Idex=Sort[0:10]
        FastMove = []
        SlowMove = []
        #####问题1 和2
        if len(FAST)==10:
            for indx in range(len(y_pred)):
                if y_pred[indx] == 1:
                    FastMove.append(Modelist[indx])
                else:
                    SlowMove.append(Modelist[indx])
        else:
```

```
            for indx in range(len(y_pred)):
                if indx in F_Idex:
                    FastMove.append(Modelist[indx])
                else:
                    SlowMove.append(Modelist[indx])
        print('FastMove:',FastMove, '\n','SlowMoveAndStatic:', SlowMove)
        # print(Acc_data)
        print(center)
        plt.show()
        SM=SlowMove
        Mtion,Static=DisMotionAndStatic(SM,Path2Person)
        # Mtion=[]
        # Static=[]
        return FastMove,Mtion,Static,Path2Person

def DisJumpAndRun(FastMove,path):##Path 使用 PersonX 一级
    JR_Feature=np.zeros((len(FastMove),3))
    i=0
    for name in FastMove:
        if name[0:2]!='~$':
            ModePath=path+'/'+name
            data=pd.read_excel(ModePath,usecols=columns_to_read)
            vol=np.sum(data[columns_to_read[2]].values)/frequency
            mu=np.mean(data[columns_to_read[2]].values)
                sigma=np.var(data[columns_to_read[2]].values)
            JR_Feature[i,:]=np.array([mu,sigma,vol])
            i=i+1
    SPC=SpectralClustering(n_clusters=2, random_state=10, gamma=1, affinity='rbf')
    JR_pred=SPC.fit_predict(JR_Feature)
    for s in range(len(JR_pred)):
        print(FastMove[s],':','Pre:',JR_pred[s],'Val:',JR_Feature[s])
    plt.figure()
    plt.scatter(JR_Feature[:, 0], JR_Feature[:, 1], c=JR_pred, s=50, cmap='viridis')
    center=np.zeros((2,3))
    Run=[]
    Jump=[]
    JRIndx=JR_Feature[:,1].argsort()[::-1]
    JRIndx=JRIndx[0:5]
    R=np.zeros(len(JR_pred))
    for r in range(len(JR_pred)):
        if r in JRIndx:
            Run.append(FastMove[r])
            R[r]=1
        else:
            Jump.append(FastMove[r])
            R[r]=0
    print('Run:',Run,'\n','Jump:',Jump)
```

```
        plt.figure()
        plt.scatter(JR_Feature[:, 0], JR_Fea-
ture[:, 1], c=R, s=50, cmap='viridis')
        T=path[-7:]+' 向前跑和跳跃'
        plt.title(T)
        for i in range(2):
            cu=np.mean(JR_Feature[:,0][JR_pred==i])
            cs= np.mean(JR_Feature[:,1][JR_pred == i])
            cv=np.mean(JR_Feature[:,2][JR_pred==i])
            center[i,:]=np.array([cu,cs,cv])
        print(center)
        plt.show()

1.def DisMotionAndStatic(SlowMove,path):
2.    MS_Feature = np.zeros((len(SlowMove), 2))
3.    i=0
4.    for name in SlowMove:
5.        if name[0:2]!='~$':
6.            ModePath=path+'/'+name
7.            data=pd.read_excel(ModePath,usecols=columns_to_read)
8.        acc_syn_F=AccSynthesis(data,columns_to_read[0:3])
9.        # print(acc_syn_F,i)
10.           MS_Feature[i,:]=acc_syn_F
11.           i = i + 1
12.       for i in range(2):
13.           if i==0:
14.               MS_Feature[:,i]=0.5*Normalizetion(MS_Feature[:,i])
15.           else:
16.               MS_Feature[:, i] = Normalizetion(MS_Feature[:, i])
17.       SPC=SpectralClustering(n_clusters=2, ran-
dom_state=10, gamma=1, affinity='rbf')
18.       MS_pred=SPC.fit_predict(MS_Feature)
19.       for s in range(len(MS_pred)):
20.           print(SlowMove[s],':','Pre:',MS_pred[s],'Val:',MS_Fea-
ture[s])
21.       plt.figure(1)
22.       plt.scatter(MS_Feature[:, 0], MS_Fea-
ture[:, 1], c=MS_pred, s=50, cmap='viridis')
23.       center=np.zeros((2,2))
24.       for i in range(2):
25.           cu=np.mean(MS_Feature[:,0][MS_pred==i])
26.           cs= np.mean(MS_Feature[:,1][MS_pred == i])
27.           center[i,:]=np.array([cu,cs])
28.       print(center)
29.       Sort = MS_Feature[:, 1].argsort(-1)[::-1]
30.       F_Idex = Sort[0:25]
31.       Motion = []
32.       Static = []
33.       MS_pred_Re=np.zeros(len(SlowMove))
34.       for indx in range(len(MS_pred)):
35.           if indx in F_Idex:
```

```
            Motion.append(SlowMove[indx])
            MS_pred_Re[indx] = 1
        else:
            Static.append(SlowMove[indx])
            MS_pred_Re[indx] = 0
    print('Motion:',Motion,'\n','Static:',Static)
    plt.figure(2)
    T=path[-7:]+' 运动状态和静止状态'
    plt.title(T)
    plt.scatter(MS_Feature[:, 0], MS_Fea-
ture[:, 1], c=MS_pred_Re, s=50, cmap='viridis')
    plt.show()
    return Motion,Static
```

```
def DisWalkAndUDstairs(Motion,path):
    WUD_Feature = np.zeros((len(Motion), 2))
    Theta=np.zeros((len(Motion), 2))
    i = 0
    for name in Motion:
        if name[0:2] != '~$':
            ModePath = path + '/' + name
            data = pd.read_excel(ModePath, usecols=columns_to_read)
            # mux=np.mean(data[columns_to_read[0]].values)
            # sigmax=np.var(data[columns_to_read[0]].values)
            # muy=np.mean(data[columns_to_read[1]].values)
            # sigmay=np.var(data[columns_to_read[1]].values)
            # F=AccSynthesis(data,columns_to_read[0:3])
            F=GetPosture(data,columns_to_read[0:3])
            SX=np.convolve(data[columns_to_read[5]].val-
ues[0:1000], np.ones(10) / 10, mode='same')
            SY = np.convolve(data[columns_to_read[4]].val-
ues[0:1000], np.ones(10) / 10, mode='same')
            FreX,PX=FFT(1000,SX)
            FreY, PY = FFT(1000, SY)

            # print(Fre,' ',P)
            iindX=PX.argsort()[::-1][0:2]
            iindY=PY.argsort()[::-1][0:2]
            ffCY=np.sum(FreY[iindY]*PY[iindY])
            ffCX=np.sum(FreX[iindX]*PX[iindX])
            thy_m,thy_s=TimeWindow(data[columns_to_read[4]].val-
ues)
            thz_m, thz_s = TimeWindow(data[col-
umns_to_read[5]].values)

        WUD_Feature[i,:]=np.ar-
ray([np.mean(F[:,0]),np.mean(F[:,1])])
        Theta[i,:]=np.array([np.mean(thy_m),np.mean(thy_s)])
        # Theta[i, :] = np.array([ffCY, ffCX])
        i = i + 1
    WUD_Feature=np.hstack((WUD_Feature,Theta))
```

```
        for i in range(4):
            if i > 1:
                WUD_Feature[:, i] = 1 * Normalizetion(WUD_Feature[:, i])
            else:
                WUD_Feature[:, i] = 1 *Normalizetion(WUD_Feature[:, i])
        SPC = SpectralClustering(n_clusters=2, random_state=0, gamma=1, affinity='rbf')
        WUD_pred = SPC.fit_predict(WUD_Feature)
        for s in range(len(WUD_pred)):
            print(Motion[s], ':', 'Pre:', WUD_pred[s], 'Val:', WUD_Feature[s,0:2])
        plt.figure(1)
        plt.scatter(WUD_Feature[:, 0], WUD_Feature[:, 1], c=WUD_pred, s=50, cmap='viridis')

        center = np.zeros((2, 2))
        for i in range(2):
            cu = np.mean(WUD_Feature[:, 0][WUD_pred == i])
            cs = np.mean(WUD_Feature[:, 1][WUD_pred == i])
            center[i, :] = np.array([cu, cs])
        print(center)
        # Sort = WUD_Feature[:, 1].argsort(-1)[::-1]
        # F_Idex = Sort[0:25]
        # Walk = []
        # UDstair = []
        # WUD_pred_Re = np.zeros(len(slowmove))
        # for indx in range(len(WUD_pred)):
        #     if indx in F_Idex:
        #         Walk.append(Motion[indx])
        #         WUD_pred_Re[indx] = 1
        #     else:
        #         UDstair.append(Motion[indx])
        #         WUD_pred_Re[indx] = 0
        # print('Walk:', Walk, '\n', 'UDstair:', UDstair)
        # plt.figure(2)
        # plt.scatter(WUD_Feature[:, 0], WUD_Feature[:, 1], c=WUD_pred_Re, s=50, cmap='viridis')
        plt.show()
        # return Walk, UDstair


1.def DisUpAndDownStair(UDlist,path):
2.    ###方差大的即下楼
3.    Theta_Feature = np.zeros((len(UDlist), 5))
4.    # Theta = np.zeros((len(UDlist), 3))
5.    i = 0
6.    for name in UDlist:
7.        if name[0:2] != '~$':
8.            ModePath = path + '/' + name
9.            data = pd.read_excel(ModePath, usecols=columns_to_read)
```

```
            thetaX=np.sum(data[columns_to_read[3]].values)/frequency
            thetaY = np.sum(data[columns_to_read[4]].values) / frequency
            thetaZ = np.sum(data[columns_to_read[5]].values) / frequency
            mx=np.mean(data[columns_to_read[0]].values)
            sx = np.var(data[columns_to_read[0]].values)
            # print(UDlist[i],':',thetaY)
        Theta_Feature[i,:]=np.array([mx,sx,thetaX,thetaY,thetaZ])
        i=i+1
    for i in range(5):
        if i==2:
            Theta_Feature[:, i] = Normalizetion(Theta_Feature[:, i])
        else:
            Theta_Feature[:, i] = Normalizetion(Theta_Feature[:, i])
    SPC=SpectralClustering(n_clusters=2, random_state=0, gamma=1, affinity='rbf')
    UD_pred=SPC.fit_predict(Theta_Feature)
    for s in range(len(UD_pred)):
        print(UDlist[s],':','Pre:',UD_pred[s],'Val:',Theta_Feature[s])
    plt.figure()
    plt.scatter(Theta_Feature[:, 0], Theta_Feature[:, 1], c=UD_pred, s=50, cmap='viridis')
    T=path[-7:]+' 上下楼'
    plt.title(T)
    center=np.zeros((2,3))
    for i in range(2):
        cu=np.mean(Theta_Feature[:,0][UD_pred==i])
        cs= np.mean(Theta_Feature[:,1][UD_pred == i])
        cv=np.mean(Theta_Feature[:,2][UD_pred==i])
        center[i,:]=np.array([cu,cs,cv])
    print(center)
    plt.show()

def DisFRLWalk(Walk,path):

    FRL_Feature = np.zeros((len(Walk), 3))
    # Theta = np.zeros((len(UDlist), 3))
    i = 0
    for name in Walk:
        if name[0:2] != '~$':
            ModePath = path + '/' + name
            data = pd.read_excel(ModePath, usecols=columns_to_read)
            thetaX=np.sum(data[columns_to_read[3]].values)/frequency
            # thetaY = np.sum(data[columns_to_read[4]].values) / frequency
```

```
            # thetaZ = np.sum(data[columns_to_read[5]].values) / frequency
            mz=np.mean(data[columns_to_read[2]].values)
            sz = np.var(data[columns_to_read[2]].values)
            # print(UDlist[i],':',thetaY)
            # acc_syn_F = AccSynthesis(data, columns_to_read[1:3])
        FRL_Feature[i,:]=np.array([mz,sz,thetaX])
        # FRL_Feature[i, :] =np.hstack((acc_syn_F,thetaX))
        i=i+1
    for i in range(3):
        if i==2:
            FRL_Feature[:, i] = Normalizetion(FRL_Feature[:, i])
        else:
            FRL_Feature[:, i] = Normalizetion(FRL_Feature[:, i])
    SPC=SpectralClustering(n_clusters=3, random_state=0, gamma=1, affinity='rbf')
    FRL_pred=SPC.fit_predict(FRL_Feature)
    for s in range(len(FRL_pred)):
        print(Walk[s],':','Pre:',FRL_pred[s],'Val:',FRL_Feature[s])
    plt.figure()
    plt.scatter(FRL_Feature[:, 0], FRL_Feature[:, 1], c=FRL_pred, s=50, cmap='viridis')
    center=np.zeros((2,3))
    for i in range(2):
        cu=np.mean(FRL_Feature[:,0][FRL_pred==i])
        cs= np.mean(FRL_Feature[:,1][FRL_pred == i])
        cv=np.mean(FRL_Feature[:,2][FRL_pred==i])
        center[i,:]=np.array([cu,cs,cv])
    print(center)
    plt.show()

def DisSSLsattic(Static,path):
    SSL_Feature = np.zeros((len(Static), 4))
    # Theta = np.zeros((len(UDlist), 3))
    L=np.zeros(len(Static))
    i = 0
    for name in Static:
        if name[0:2] != '~$':
            ModePath = path + '/' + name
            data = pd.read_excel(ModePath, usecols=columns_to_read)
            POS = GetPosture(data, columns_to_read[0:3])
        SSL_Feature[i, :] = np.array([np.mean(POS[:, 0]),np.var(POS[:, 0]),np.mean(POS[:, 1]),np.var(POS[:, 1])])
        i=i+1
    for i in range(2):
        if i<=2:
            SSL_Feature[:, i] = Normalizetion(SSL_Feature[:, i])
            # SSL_Feature[:, i]=SSL_Feature[:, i]
        else:
```

```
                SSL_Feature[:, i] = Normalizetion(SSL_Feature[:, i])
        PX=SSL_Feature[:,0].reshape(-1)
        Idx=PX.argsort()
        Idx=Idx[0:5]
        LD=[]
        SS=[]
        for i in range(np.size(PX)):
            if i in Idx:
                LD.append(Static[i])
                L[i]=1
            else:
                SS.append(Static[i])
        SS_Feature = np.zeros((len(Static) - len(Idx), 2))
        print('Lay:', LD, '\n', 'SS:', SS)
        i = 0
        for name in SS:
            if name[0:2] != '~$':
                ModePath = path + '/' + name
                data = pd.read_excel(ModePath, usecols=col-
umns_to_read)
                # SX = np.sum(data[columns_to_read[0]].values) / fre-
quency
                mz = np.mean(data[columns_to_read[2]].values)
                # my = np.mean(data[columns_to_read[1]].values)
                # sy = np.var(data[columns_to_read[2]].values)
                # acc_syn_F = AccSynthesis(data, columns_to_read[0:3])
                POS = GetPosture(data, columns_to_read[0:3])
            # SS_Feature[i, :] = np.ar-
ray([acc_syn_F[0], acc_syn_F[1], np.mean(POS[:, 0]), np.mean(POS[:, 1])])
            SS_Feature[i, :] = np.array([mz,np.mean(POS[:, 0])])
            ###使用Z轴的加速度的均值和方差效果最佳（目前）
            i = i + 1
        for i in range(2):
            SS_Feature[:, i] = Normalizetion(SS_Feature[:, i])
        PS = SS_Feature[:, 0].reshape(-1)
        SIdx = PS.argsort()[::-1]
        SIdx = SIdx[0:5]
        Sit=[]
        S=[]
        for sit in range(len(SS)):
            if sit in SIdx:
                Sit.append(SS[sit])
                L[sit]=1
            else:
                S.append(SS[sit])
        print('Sit:',Sit,'\n','S_s:',S)
        Stand_Evelator=S
        S_Feature = np.zeros((len(SS) - len(SIdx), 3))
        i = 0
        for name in Stand_Evelator:
            if name[0:2] != '~$':
```

```
                ModePath = path + '/' + name
                data = pd.read_excel(ModePath, usecols=col-
umns_to_read)
                SX = np.sum(data[columns_to_read[0]][0:1300].values-
1) / frequency
                # print('序列长度为: ',len(data[col-
umns_to_read[0]].values))
                mx = np.mean(data[columns_to_read[0]].values)
                # my = np.mean(data[columns_to_read[1]].values)
                sx = np.var(data[columns_to_read[0]].values)
                # acc_syn_F = AccSynthesis(data, columns_to_read[0:3])
                # POS = GetPosture(data, columns_to_read[0:3])
            # SS_Feature[i, :] = np.ar-
ray([acc_syn_F[0], acc_syn_F[1], np.mean(POS[:, 0]), np.mean(POS[:, 1])])
            S_Feature[i, :] = np.array([mx,sx,SX])
            ###使用Z轴的加速度的均值和方差效果最佳（目前）
            i = i + 1
        for i in range(3):
            S_Feature[:, i] = Normalizetion(S_Feature[:, i])
        SPC=SpectralClustering(n_clusters=2, ran-
dom_state=0, gamma=1, affinity='rbf')
        S_pred=SPC.fit_predict(S_Feature)
        Temp=S_Feature[:,1].reshape(-1)
        InvIdx=Temp.argsort()[::-1]
        # for s in range(len(S_pred)):
        # for s in InvIdx:
        #     print(Stand_Evela-
tor[s],':','Pre:',S_pred[s],'Val:',S_Feature[s])
        # plt.figure()
        plt.scatter(S_Feature[:, 0], S_Fea-
ture[:, 1], c=S_pred, s=50, cmap='viridis')
        T=path[-7:]+' 躺下与坐下'
        plt.title(T)
        SIdx=S_Feature[:,1].argsort()[0:5]
        Stand=[]
        Evelator=[]
        for i in range(len(S_pred)):
            if i in SIdx:
                Stand.append(Stand_Evelator[i])
            else:
                Evelator.append(Stand_Evelator[i])
        print('Stand:',Stand,'\n','Evelator:',Evelator)
        E_Feature=np.zeros((len(Evelator),2))
        i = 0
        for name in Evelator:
            if name[0:2] != '~$':
                ModePath = path + '/' + name
                data = pd.read_excel(ModePath, usecols=col-
umns_to_read)
                mx = np.sum(data[columns_to_read[0]].values)
                # sx = np.var(data[columns_to_read[0]].valus)
```

```
            my = np.sum(data[columns_to_read[1]].values)
            # sy = np.var(data[columns_to_read[1]].values)
            # a=AccSynthesis(data,columns_to_read[0:3])
        E_Feature[i,:]=np.array([mx,my])
        i=i+1
    for i in range(2):
        E_Feature[:,i] = Normalizetion(E_Feature[:, i])
    SPC = SpectralClustering(n_clusters=3, ran-
dom_state=10, gamma=1, affinity='rbf')
    E_pred = SPC.fit_predict(E_Feature)
    plt.figure()
    plt.scatter(SSL_Feature[:, 0], SSL_Fea-
ture[:, 2], c=L, s=50, cmap='viridis')
    plt.show()
    # for i in range(len(Evelator)):
    for i in E_Feature[:,0].argsort()[::-1]:
        print(Evelator[i],':',E_pred[i],':',E_Feature[i])

    center=np.zeros((3,2))
    for i in range(3):
        cu=np.mean(S_Feature[:,0][S_pred==i])
        cs= np.mean(S_Feature[:,1][S_pred == i])
        center[i,:]=np.array([cu,cs])
    print(center)
```

2. 深度集成算法代码：

(1) 引入高维数据：

```
# 一次积分
# 添加第七列数据，数据为第一列数据的累积和（一次积分）
df['first-integral-acc_x'] = df.iloc[:, 0].cumsum()
# 添加第八列数据，数据为第二列数据的累积和（一次积分）
df['first-integral-acc_y'] = df.iloc[:, 1].cumsum()
# 添加第九列数据，数据为第三列数据的累计和（一次积分）
df['first-integral-acc_z'] = df.iloc[:, 2].cumsum()
# 添加第十列数据，数据为第四列数据的累计和（一次积分）
df['first-integral-gyro_x'] = df.iloc[:, 3].cumsum()
    # 添加第十一列数据，数据为第五列数据的累计和（一次积分）
    df['first-integral-gyro_y'] = df.iloc[:, 4].cumsum()
    # 添加第十二列数据，数据为第六列数据的累计和（一次积分）
    df['first-integral-gyro_z'] = df.iloc[:, 5].cumsum()

    # 二次积分
    # 添加第十三列数据，数据为第一列数据的二次积分
    df['second-integral-acc_x'] = df['first-integral-
acc_x'].cumsum()
```

```
    # 添加第十四列数据，数据为第二列数据的二次积分
    df['second-integral-acc_y'] = df['first-integral-acc_y'].cumsum()
    # 添加第十五列数据，数据为第三列数据的二次积分
    df['second-integral-acc_z'] = df['first-integral-acc_z'].cumsum()

    # 合成加速度
    # 添加第四列数据，名称为'mmm'，数据为对第一、二、三列数据平方和的平方根
    df['acc_syn'] = np.sqrt(df.iloc[:, 0]**2 + df.iloc[:, 1]**2 + df.iloc[:, 2]**2)
    df['acc_syn_mean'] = df['acc_syn'].mean()
    df['acc_syn_var'] = df['acc_syn'].var()

    # 加速度的acos
    df['acc_x_acos'] = np.arccos(np.clip(df.iloc[:, 0], -1, 1))
    df['acc_y_acos'] = np.arccos(np.clip(df.iloc[:, 1], -1, 1))
    df['acc_z_acos'] = np.arccos(np.clip(df.iloc[:, 2], -1, 1))
```

(2)线性插值转为多维特征灰度图

```
min_data_dict = {1: -3.5, 2: -6.5, 3: -5.5, 4: -770, 5: -776, 6: -810, 7: -3890, 8: -2910,
                 9: -3512, 10: -187970, 11: -63285, 12: -61068, 13: -12282570, 14: -11439510, 15: -24552975, 16: 0.0056,
                 17: 0.94, 18: 0, 19: 0, 20: 0, 21: 0}
max_data_dict = {1: 7, 2: 5, 3: 4.3, 4: 845, 5: 560, 6: 818, 7: 12875, 8: 6918,
                 9: 8703, 10: 147125, 11: 73818, 12: 71395, 13: 87145262, 14: 35574420, 15: 43543388, 16: 8.53,
                 17: 1.82, 18: 3, 19: 3.15, 20: 3.15, 21: 3.15}


# 定义生成每列数据图像的函数
def generate_column_image(data, target_shape=(19, 399), num=-1):  #####  (24, 384)  (19, 399)
    target_size = target_shape[0] * target_shape[1]

    # Normalize data
    data = np.array(data)
    data = (data - min_data_dict[num]) / (max_data_dict[num] - min_data_dict[num]) * 255

    # Interpolation
    x = np.linspace(0, len(data) - 1, num=len(data))
```

```
        f = interp1d(x, data, kind='linear')
        x_new = np.linspace(0, len(data) - 1, num=target_size)
        data_interpolated = f(x_new)

        # Reshape to target shape
        data_padded = data_interpolated.reshape(target_shape)

        return data_padded

    ......

    df = pd.read_excel(file_path)
    num = 1
    # 读取每列数据并生成对应的图像
    images = []
    for col in df.columns:
        column_data = df[col].dropna().values  # 去除空值
        column_image = generate_column_image(column_data, num=num)
        images.append(column_image)
        num += 1
    # 拼接所有列生成最终的 384x384 图像
    final_image = np.vstack(images)
```

(3)深度网络训练

```
def train(data_dir, save_dir, model_name, batch_size, set_epoch):
    filename = ''
    if 'math_model_appendix2_data' in data_dir:
        filename = 'math_model_appendix2_data_' + model_name

    epoches_list = [50, 70, 90]

    epochs = set_epoch
    csv_dir = os.path.join(save_dir, 'csv')
    os.makedirs(csv_dir, exist_ok=True)  # 创建保存路径
    num_class = len(os.listdir(data_dir))  # 读取分类数目

    t1 = time.time()
    data_test = datasets.ImageFolder(data_dir,
                                     transform=transforms.Compose([
                                         transforms.Resize((399, 399)),  # 修改这里，确保图像缩放到模型期望的大小
```

```
    18.
    19.                                                 transforms.ToTensor(),
    20.                                                 # 将图像的像素值标准化为均值为 mean，标准差为 std 的分布。通常，这些均值和标准差是在训练数据上
    21.                                                 # 计算得到的。在这里，它使用的是在 ImageNet 数据集上计算得到的均值和标准差。
    22.                                                 transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    23.                                             ]))
    24.         # data_test = CustomDataset(data=[item[0] for item in data_test], targets=[item[1] for item in data_test]) # 加速测试
    25.         dataloder_test = tud.DataLoader(data_test, batch_size=batch_size, shuffle=True)
    26.         data_train = datasets.ImageFolder(data_dir,
    27.                                           transform=transforms.Compose([
    28.                                               transforms.Resize((399, 399)),  # 修改这里，确保图像缩放到模型期望的大小
    29.
    30.
    31.                                               # 随机旋转图像，旋转角度在 -45 度到 +45 度之间。参数 expand=False 表示不会自动调整图像大小
    32.                                               # 以包含所有旋转后的内容，而 interpolation=Image.BILINEAR 表示使用双线性插值进行图像的旋转。
    33.                                               transforms.RandomRotation(45, expand=False, interpolation=Image.BILINEAR),
    34.                                               # 有 50% 的概率应用锐化操作。RandomAdjustSharpness 用于随机调整图像的锐化程度，
    35.                                               # sharpness_factor=2 表示锐化因子为 2。
    36.                                               transforms.RandomApply([transforms.RandomAdjustSharpness(sharpness_factor=2)], p=0.5),
    37.                                               # 有 50% 的概率应用去锐化操作。这是上一个步骤的反操作，即降低图像的锐化程度。
    38.                                               transforms.RandomApply([transforms.RandomAdjustSharpness(sharpness_factor=0)], p=0.5),
    39.                                               # 随机调整图像的亮度、对比度和饱和度。参数 brightness=0.5 表示亮度可调整的最大程度为 0.5，
    40.                                               # contrast=0.5 表示对比度可调整的最大程度为 0.5，saturation=0.5 表示饱和度可调整的最大程度为 0.5。
    41.                                               transforms.ColorJitter(brightness=0.5, contrast=0.5, saturation=0.5),
    42.                                               transforms.ToTensor(),
    43.                                               transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
```

```
    44.                                         ]))
    45.         print(f'训练样本数量: {len(data_train)}')
    46.         dataloder_train = tud.DataLoader(data_train, batch_size=batch_size, shuffle=True)
    47.         t2 = time.time()
    48.         print(f'Load Data Time:{t2 - t1:.6f}')
    49.
    50.         model = initialize_model(model_name, num_class, use_pretrained=True, update_param=True).to(device)
    51.
    52.         loss_fn = nn.CrossEntropyLoss()
    53.         momentum = 0.5
    54.
    55.         head = ''
    56.         training_accuracy_list = [head]
    57.         test_accuracy_list = [head]
    58.         loss_list = [head]
    59.         t_list = [head]
    60.         test_accuracy = 0.0
    61.         loss_train = 0.0
    62.         for epoch in range(epochs):
    63.             lr = 0.002 if epoch > (0.5 * epochs) else 0.01
    64.             optimizer = optim.SGD(model.parameters(), lr=lr, momentum=momentum)
    65.             # 训练模型
    66.             loss, training_acc, t = train_model(model, dataloder_train, loss_fn, optimizer, epoch + 1)
    67.             t_list.append(t)
    68.             loss_list.append(loss)
    69.             training_accuracy_list.append(training_acc.data.cpu().numpy())
    70.             # 测试模型
    71.             test_acc = test_model(model, dataloder_test, loss_fn)
    72.             test_accuracy_list.append(test_acc.data.cpu().numpy())
    73.             if test_acc.data.cpu().numpy() > test_accuracy:
    74.                 test_accuracy = test_acc.data.cpu().numpy()
    75.                 loss_train = loss
    76.                 torch.save(model.state_dict(), os.path.join(save_dir, filename + '_best.pth'))
    77.             elif test_acc.data.cpu().numpy() == test_accuracy:
    78.                 if loss < loss_train:
    79.                     loss_train = loss
```

```
                torch.save(model.state_dict(), os.path.join(save_dir, filename + '_best.pth'))

            if epoch in epoches_list:
                torch.save(model.state_dict(), os.path.join(save_dir, filename + '_' + str(epoch) + '.pth'))
```

(4)集成学习

```
# 获取基模型在第二部分数据集上的输出
preds_train1 = M1(train_data2)
preds_train2 = M2(train_data2)
preds_train3 = M3(train_data2)

# 将所有基模型的输出堆叠在一起作为元学习器的输入特征
base_predictions_train = np.column_stack([
    preds_train1, preds_train2, preds_train3
])

# 使用支持向量机训练元学习器
meta_learner = SVC(probability=True)
meta_learner.fit(base_predictions_train, train_labels2)

# 获取基模型在新数据集上的输出
preds_test1 = M1(test_data)
preds_test2 = M2(test_data)
preds_test3 = M3(test_data)

# 将所有基模型的输出堆叠在一起作为元学习器的输入特征
base_predictions_test = np.column_stack([
    preds_test1, preds_test2, preds_test3
])

# 使用 SVM 元学习器进行最终预测
final_predictions = meta_learner.predict(base_predictions_test)
```

(5)ANN 近相似搜索

```
# 使用余弦相似度作为距离度量。对于高维数据，通常效果较好。
annoy_index = AnnoyIndex(dimension, 'angular')

# 添加描述符库向量到索引
for idx, vector in enumerate(keys_as_lists):
    annoy_index.add_item(idx, vector)

# 将描述子向量作为键，使用元组(tuple)进行转换
```

```
    9.neighbors = loaded_annoy_index.get_nns_by_vector(descriptor, n_neigh-
bors, include_distances=False)
    10.  matched_opt_key = vocabulary_dict_keys_list[neighbors[0]]
```