东 莞 理 工 学 院

本 科 毕 业 论 文

|  |  |  |
| --- | --- | --- |
| 毕业设计题目： | 基于深度学习的人体动作规范化检测系统 | |
| 学生姓名： | 李添骏 | |
| 学 号： | 202041417620 | |
| 院 系： | 计算机科学与技术学院 | |
| 专业班级： | 计算机科学与技术4班 | |
| 指导教师姓名及职称： | | 任子良 副教授 |
| 起止时间： 2023年 12月--2024年 5月 | | |

**摘要**

随着网络设备和计算机技术的不断发展，在深度学习及计算机视觉领域，关于人体动作识别和规范化检测的研究不断增多。其广泛应用于视频安全监控、视频检索、健康医疗等领域，人体动作的规范化检测可以通过穿戴惯性传感器等方法实现，但穿戴设备成本高昂且不方便，尤其是在大规模体育考试等场景中无法实现。因此，亟需一种无扰式、方便数据采集、造价低的人体动作检测与规范化评价系统，以便于准确及时的提醒用户动作是否规范。其中在深度学习当中最常用的即为yolo系列的目标检测算法，本文则基于yolov8目标检测算法及状态机算法设计了一个运动检测和评估系统，制作了一个包含9个类别，8493张图片的数据集并训练，最终在测试集上本系统实现了平均mAP值约0.9的检测精度以及约15ms的单帧处理速度。

本文的主要工作如下：

1. 对输入的视频数据进行逐帧处理，将输出的图片逐个输入到网络当中。
2. 对用户所做出的各种工作，设计不同的类，对不同的动作采用各自不同的网络模型进行处理。
3. 收集本网络动作相关的数据集，并对数据集进行标签、加工，输入到YOLOv8目标检测模型当中进行训练。
4. 使用YOLOv8-pose的姿态估计算法对人体骨架进行提取，并通过对人体骨架之间的角度计算等来统计动作个数，并使用PyQt5搭建可视化界面。

**关键词：**计算机视觉，动作检测，深度学习，YOLOv8算法

**Abstract**

With the continuous development of network equipment and computer technology, there are more and more researches on human motion recognition and standardized detection in the field of deep learning and computer vision. It is widely used in video security monitoring, video retrieval, health care and other fields,The standardized detection of human movements can be achieved by wearing inertial sensors and other methods, but wearable devices are costly and inconvenient, especially in large-scale sports exams and other scenarios. Therefore, there is an urgent need for a non-disturbing, convenient data collection, low cost human movement detection and standardized evaluation system, in order to accurately and timely remind users whether the movement is standardized. Among them, YOLO series of object detection algorithms are the most commonly used in deep learning. This paper designs a motion detection and evaluation system based on YOLOv8 object detection algorithm and state machine algorithm. A data set containing 9 categories and 8493 images was made and trained. Finally, on the test set, the system achieved an average mAP value of about 0.9 detection accuracy and a single frame processing speed of about 15ms.

The main work of this paper is as follows：

1. The input video data is processed frame by frame, and the output pictures are input to the network one by one.
2. Different classes are designed for all kinds of work done by users, and different network models are used to deal with different actions.
3. The data set related to the network action is collected, and the data set is labeled and processed, and input into the yolov8 target detection model for training.
4. The pose estimation algorithm of YOLOv8-Pose was used to extract the human skeleton, and the number of movements was counted by calculating the angles between the human skeleton, And use PyQt5 to build a visual interface.

**Key words:** computer vision, motion detection, deep learning, YOLOv8 algorithm.

**目录**

[第1章 引言 1](#_Toc164630450)

[1.1背景介绍 1](#_Toc164630451)

[1.2问题陈述 1](#_Toc164630452)

[1.3研究目标和意义 2](#_Toc164630453)

[1.4论文的主要工作 3](#_Toc164630454)

[1.5章节安排 3](#_Toc164630455)

[第2章 相关技术和理论模型 5](#_Toc164630456)

[2.1深度学习相关理论 5](#_Toc164630457)

[2.2姿态提取模块 12](#_Toc164630458)

[2.3相关开发环境 13](#_Toc164630459)

[2.4本章小结 15](#_Toc164630460)

[第3章 系统设计与实现 16](#_Toc164630461)

[3.1系统设计整体框架 16](#_Toc164630462)

[3.2可行性分析 17](#_Toc164630463)

[3.3系统设计的重难点分析 18](#_Toc164630464)

[3.4目标检测模型设计 18](#_Toc164630465)

[3.4.1动作状态转移分析 18](#_Toc164630466)

[3.4.2 数据集的制作与训练 20](#_Toc164630467)

[3.5姿态估计模块设计 22](#_Toc164630468)

[3.5.1 姿态估计程序设计 22](#_Toc164630469)

[3.5.2 姿态估计算法设计 24](#_Toc164630470)

[3.6本章小结 30](#_Toc164630471)

[第4章 系统测试与评估 31](#_Toc164630472)

[4.1实验结果展示 31](#_Toc164630473)

[4.2模型准确度评估 33](#_Toc164630474)

[4.3系统存在的缺陷 38](#_Toc164630475)

[4.4本章小结 39](#_Toc164630476)

[第五章 总结与展望 40](#_Toc164630477)

[参考文献 41](#_Toc164630478)

[致谢 44](#_Toc164630479)

# 第1章 引言

## 1.1背景介绍

随着人工智能技术的快速发展，关于深度学习的目标检测算法研究逐渐在整个计算机视觉领域占据了重要的地位，也是计算机视觉当中的一项基本任务，目标检测的首要目标就是能够在图像或者视频当中识别和定位出某个目的对象。而在目标检测运用的诸多领域当中，体育运动动作的规范化检测也称为了所需检测的重要研究对象之一。

目前，随着全民运动健身的风潮兴起，越来越多的人加入到了各种各样的体育活动当中，其中熟如常见的健身动作如高抬腿、俯卧撑及仰望起坐等动作也为运动初学者的基本动作。但是要想达到锻炼的效果，动作的规范性也成为许多运动者在日常锻炼中需要注意的地方，而一般来说运动者往往难以在自己的动作当中辨别自己的动手是否符合标准，甚至动作执行的不到位也有可能对身体产生一定的伤害。

然而，在传统的体育运动规范检测方面往往需要专业的运动专家或教练来进行目测判断，并且不同的专家对同一个动作的标准化判断的标准也会有所不同，这就导致了对于动作规范度的判断存在主观性较强、且效率低下的问题。

## 1.2问题陈述

虽然现阶段计算机视觉应用已经开始开展的一定数量的研究，且在市面上也流行有许多对用户运动动作进行识别的模型，如下图1-1。但是在其应用在姿态检测以及运动评估之中的案例十分稀少，以及对应的人体数据集也十分稀缺，在网络上也有不少关于运动姿态教学的视频，但是却缺乏一个完善的评估体系对用户的运动姿态进行评估，缺乏与用户的交互性。

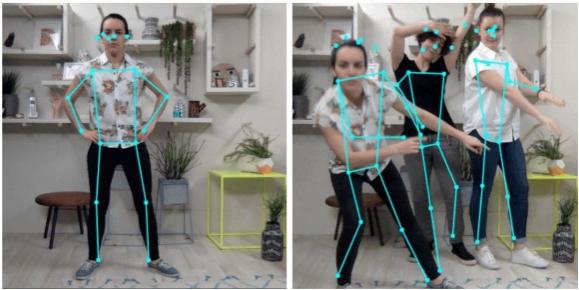


图1-1 市面上常见的人体运动姿态识别场景

因此为了解决上述问题，帮助正在进行体育锻炼的人发现动作上的错误，从而及时纠正，达到锻炼身体的效果，本文设计了一种运动目标姿态检测及评估算法，可以识别一些基本的用户运动动作，并对其进行数量统计和姿态评估，从而帮助用户进行姿态矫正，达到锻炼的效果。

本文所需要解决的主要问题有：

1. 选择一种合适的算法对用户进行检测、识别。
2. 描绘出用户运动的人体骨架结构，根据骨架结构的状态来判断用户动作的状态，从而进行姿态判断。
3. 对用户所做出的不同动作来设计不同的动作类，并将对应的动作放入对应的模型中进行识别
4. 根据用户的动作状态对用户的动作进行计数和评估，当前动作不规范时不予计数，并在可视化界面上显示结果。

## 1.3研究目标和意义

随着我国人民生活水平的提高，为了追求更高更好的生活质量，越来越多的群众通过运动健身的方式来促进身心健康，延年益寿。而能否通过运动来达到强身健体的效果很大一部分也取决于动作是否规范到位，研发基于深度学习和计算机视觉的人体动作规范化检测系统有利于用户在自己的动作当中及时的发现问题并给予纠正，同时通过所统计的动作个数，用户可以清晰直观的看到自己的运动水平，将用户的运动情况反映出来。

研究本动作规范化检测系统具有以下重要意义

1. 可促进人机交互和运动分析，通过识别人体姿态，我们可以实现与计算机的自然互动，尤其在当今已经被广泛应用的虚拟现实(VR)和增强现实(AR)等领域。
2. 人体动作规范检测对于医学界康复患者姿态监督、骨骼恢复以及在体育界运动员的姿态训练和辅助训练具有重大意义。
3. 对于自动驾驶系统设计而言，设计人体运动姿态检测系统来判断及预测行人可能出现的行为，从而及时的采取安全措施来避免交通事故。
4. 将人体动作规范识别应用于身份识别与智能监控上，可以帮助识别人体走姿和面部表情等，有效监控异常行为并及时报警。

## 1.4论文的主要工作

本论文的主要工作是设计一种基于深度学习的人体动作规范化检测系统，该系统主要分为运动状态检测、动作计数以及运动状态评估三个模块。

在图像识别和目标检测模块，采用YOLOv8目标检测算法，并针对3种常见的运动动作进行训练。首先将采集到的视频流进行抽帧处理，并进行人工标注，数据训练则采用Ultralytics官方提供的3个预训练模型同时进行训练。动作计数模块则采用YOLOv8-Pose算法来对识别到的人体动作进行提取，计算骨架之间的角度来判断动作姿态从而判断动作状态是否转移，当达到完成一个动作所需的3个状态后则对该动作进行计数。在运动评估模块则对运动的频率进行计算，根据其与标准频率的差距来对用户的运动状态进行评估，最后完成可视化界面的搭建，采用PyQt5对整个程序进行可视化处理，使得用户能够更加方便的使用该程序。

## 1.5章节安排

本文的章节安排如下：

引言：描述关于深度学习和目标检测算法的主要应用方向和目标，接着提出了在运动方面目标检测和姿态估计应用的应用前景，并且对比了一些市面上传统的运动检测算法，论述了当下运动检测应用上所出现的问题以及存在的不足，在此基础上也概述了本文的研究旨在设计一种基于深度学习的人体动作规范化检测系统，主要实现了运动状态检测、动作计数以及运动状态评估的功能。

相关技术和理论模型：本章节主要对本项目所使用到的理论知识进行一个介绍，具体介绍每个技术模块的原理及主要应用场景，包括YOLOv8目标检测算法的发展历史即大致实现原理以及YOLOv8-pose姿态提取算法的应用，以及实验所需的相关开发环境包括PyCharm、Anaconda和所需软件包如OpenCV、Ultralytics等。

系统设计与实现：本章节介绍了程序的实现流程和程序的整体框架，并对程序实现的可行性以及程序实现的重难点做出了分析，然后对程序流程图当中的每一个模块给出的具体的实现及原理讲解。首先介绍了目标检测算法的实现，包括数据集的采集和模型的训练；然后介绍了姿态估计算法的设计与实现，并给出了动作计算的公式推导。

系统测试与评估：本章节对前面章节设计的程序进行了一个整合，并采用PyQt5搭建出了程序的可视化界面，对几个实验场景进行了测试。最后本文以模型的mAP值作为评价指标来评价模型的准确度，并介绍了mAP的相关概念包括交叉比、准确率和召回率等，最后作出了实验模型的mAP值曲线图。

总结与展望：本章节对整个系统做出了一个总结，并指出了其仍然存在的缺陷，包括模型的准确度、系统的架构、以及可视化界面展示上的问题，并提出了一些优化方案如数据增强、多任务处理等，并就该系统的应用前景做出了展望，提出了更多可行的应用场景。

# 第2章 相关技术和理论模型

## 2.1深度学习相关理论

2001年，P.Viola和M.Jones首次实现了实时的人脸检测，提出了Viola-Jones检测器，其任务为找出图像当中所有感兴趣的物体，其中包括了物体定位和物体分类两个子任务，即同时确定物体的类别和位置，其算法的基本流程如下：

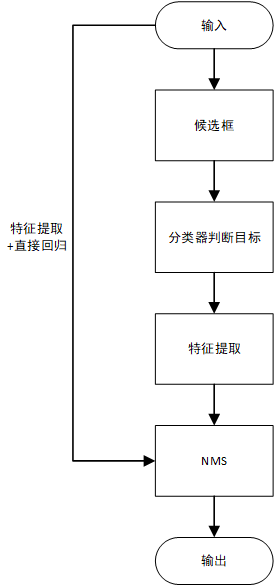


图2-1 Viola-Jones检测器工作流程图

**候选框：**通常采用滑动窗口的方法进行。

**分类器：**主要有支持向量机SVM，AdaBoost等，其根据图片或者视频当中所提取到的特征对目标进行分类。

**NMS：**非极大值抑制，用于计算置信度，进行候选框的合并、过滤，寻找最佳的物体检测位置。

VJ框架在常规700 MHz英特尔奔腾III上，对384×288像素的图像的人脸检测速度达到了每秒15帧。它采用一个单一的灰度图像就实现了高帧速率，较之前的方法有了2个数量级的提高，并且同时保持了很好的精度。

之后，Rainer Lienhart 和 Jochen Maydt 将这个检测器用对角特征进行了扩展《An Extended Set of Haar-like Features for Rapid Object Detection》，在深度学习技术出现之前，VJ一直是人脸检测算法的主流框架。

但是，传统的目标检测算法也存在如下缺点：

1. 特征难以估计。
2. 特征缺乏鲁棒性。效率存在瓶颈。
3. 滑动窗口提取的策略非常复杂。

在2014年，Ross B. Girshick(RBG)等人在CNN的基础上设计了R-CNN模型，R-CNN(Region-based Convolution Neural Networks, 或Regions with CNN features, R-CNN)摒弃了传统的滑动窗口和人工选取特征的方法如HOG和SIFT，将候选区域算法Selective Search和卷积神经网络相结合，使得检测速度和精度明显提升。

R-CNN的网络结构如下图3所示，其采用8层卷积神经网络对图像进行特征提取，其中前5层为卷积层，第一、二、五层往后跟有最大池化层，后面3层为全连接层。在输入网络之前需要把图像归一化为227 \* 277的固定大小，输入层为候选区域边框缩放扩充得到，输出为该候选区域边框的分类结果。

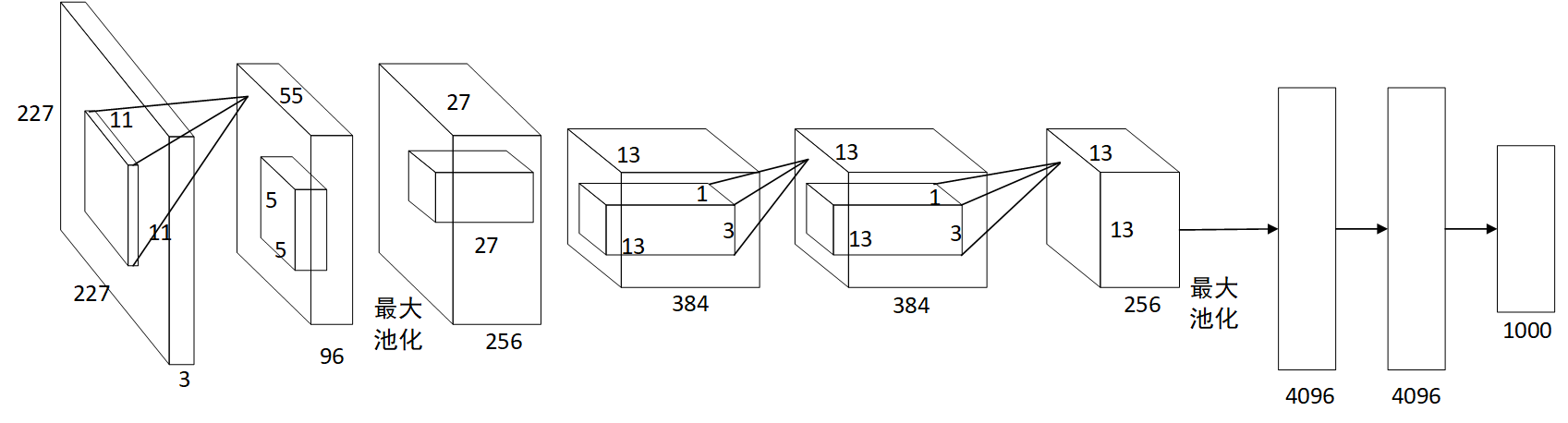


图2-2 R-CNN网络模型结构图

可以看出，卷积层中卷积核的数量在逐层增加，目的是为了减少池化层对特性信息的丢失。池化层虽然在减少计算量和增加平移不变性上有较大的贡献，但其采用的降采样操作会导致图像特征的损失，通过增加卷积核的数量能较好的解决该问题。

继R-CNN之后，又出现了Faster R-CNN算法，其同样使用 VGG16作为网络的backbone，推理速度在GPU上达到5fps(包括候选区域的生成)，准确率也有进一步的提升。

以上两种算法都是基于Region Proposal的two-stage算法，即需要先使用启发式方法或者CNN网络产生Region Proposal，在其基础上做分类和回归。而本文主要使用的则是yolo(you look only once)系列的目标检测算法，即只需要“看”一次即可识别的算法，它是属于单阶段设计(one-stage)一类的算法，即只需要使用一个CNN网络直接预测不同目标的类别与位置。

从最早 Joseph Redmon 等人发表在 CVPR 2016的 YOLO 模型以来, YOLOv1 已经经历过了十数个版本. 从开始YOLOv1便采用one-stage设计, 基于回归同时产生最终的检测框以及类别, 使其在检测的精确度以及速度上取得了良好的平衡. 而相对于先前经典的两阶段(two-stage)模型,如 Fast R CNN, 这类模型将任务分为两步: 检测候选框区域再对候选区域进行分类. 同时YOLO框架后续的版本抛弃了预定义锚框的设计，减少了耗时和算力。如此, YOLO 框架自面世以来已经在机器人, 无人驾驶汽车和视频监控应用等实时物体检测系统中取得了广泛的应用。

本文使用的是由Ultralytics公司最新推出的yolov8目标检测算法，其可用于图像分类、物体检测和实例分割等任务YOLOv8是YOLO算法的最新版本，通过引入空间注意力、特征融合和上下文聚合模块等各种修改，其性能优于之前的版本。其中的YOLOv8x在MS COCO数据集 test-dev 上进行评估, 在图像大小为 640 大小的情况下实现了 53.9% 的 AP (相同输入尺寸下, YOLOv5 的AP为50.7%), 在NVIDIA A100和TensorRT框架下的运行速度为280 FPS。

YOLOv8在精度和速度方面都具有尖端性能，其在之前YOLO 版本的基础上，引入了新的功能和优化，成为了广泛应用中各种物体检测任务的理想选择。

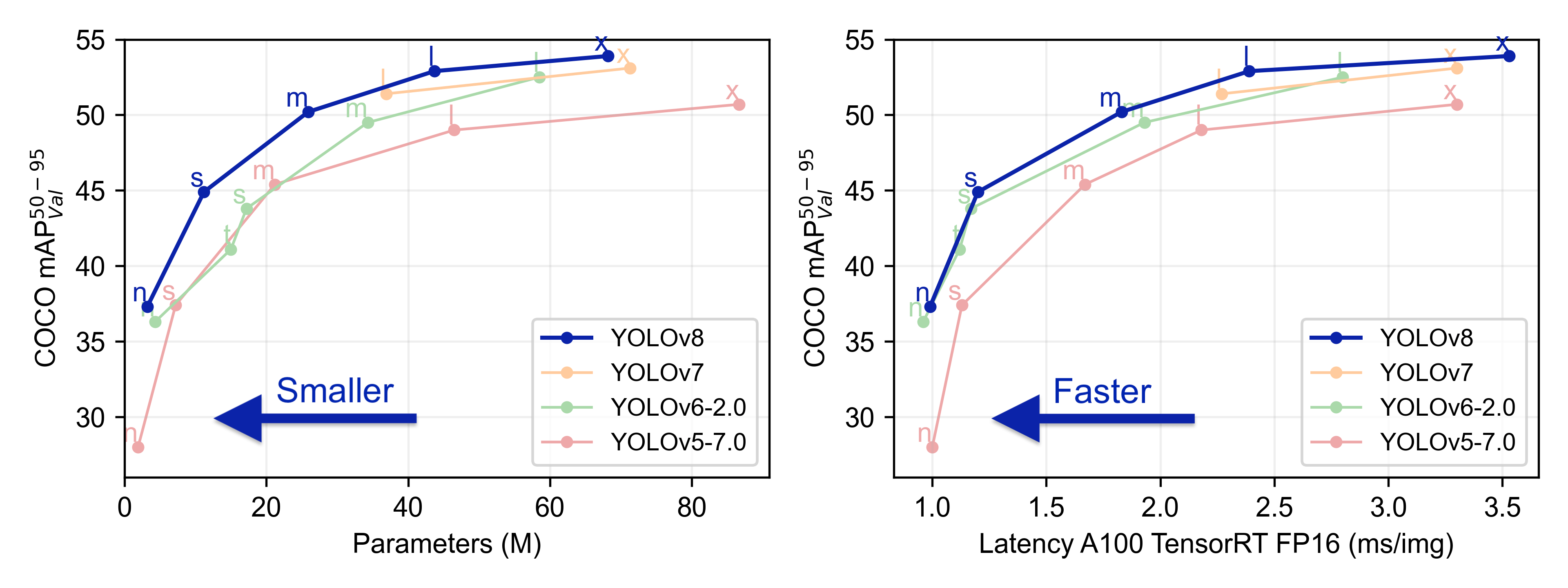


图2-3 Yolov8与先前版本的对比

YOLO框架的结构使用 Backbone、Neck 和Head 三部分进行描述, 它们分别用于特征提取, 特征细化以及利用特征进行预测，在下图中可以看见这三个部分具体的网络结构，更细节的网络组成位于 Details 区域中。

YOLOv8网络结构如下:



图2-4 YOLOv8网络模型结构

（1）Backbone（骨干网络）

其主要负责对输入的图像进行特征提取，它的主要作用就是讲原始的输入图像转化为多层特征图，以便于后续的目标检测任务使用。在YOLOv8的骨干当中采用了一种修改过的CSPDarknet53架构，其为一个由53个卷积层组成的骨干网络，Cross-Stage Partial Connection（跨级部分连接）是CSPDarknet53的关键特性。它通过将不同层的特征图进行拆分，然后将其中一部分与其他层级的特征图进行连接（如图5所示），通过CSP连接，特征图的信息可以在不同层之间自由传递，从而更好地捕获不同尺度的特征。这对于目标检测任务非常重要，因为目标可能在不同尺度上具有不同的外观从而促进了信息的流动，这也有助于提高模型的准确性和性能。

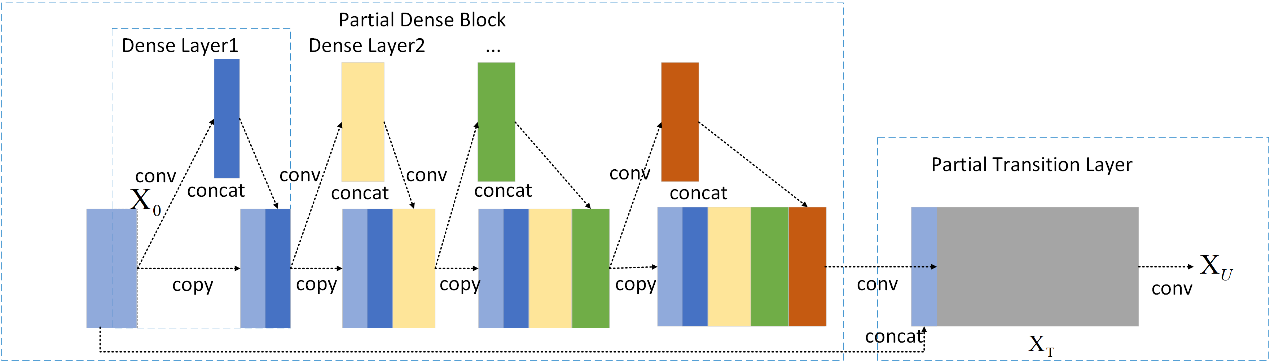


图2-5 Cross-Stage Partial Connection特征拆分及拼接

（2）Neck

[YOLOv8中的Neck部分是连接主干网络（backbone）和头部网络（head）的关键组件。它负责特征融合和处理，以提高目标检测的准确性和效率](https://zhuanlan.zhihu.com/p/677145875" \t "_blank)。

Slim-Neck是YOLOv8当中的一项改进机制，Slim-neck模块为一种用于目标检测的神经网络结构，Slim-neck模块中的模块首先使用轻量级卷积GSConv来替代传统卷积Conv[6]，在这个架构当中，不同尺度的特征图通过GSConv处理，之后通过upsample（上采样）和Concat（拼接）与其他尺度的特征图进行融合。处理过后的特征图再次通过GSConv模块，最后使用VoV-GSCSP模块来进一步提取和融合特征，以准备最终的检测头（head-1, head-2, head-3）进行目标检测。

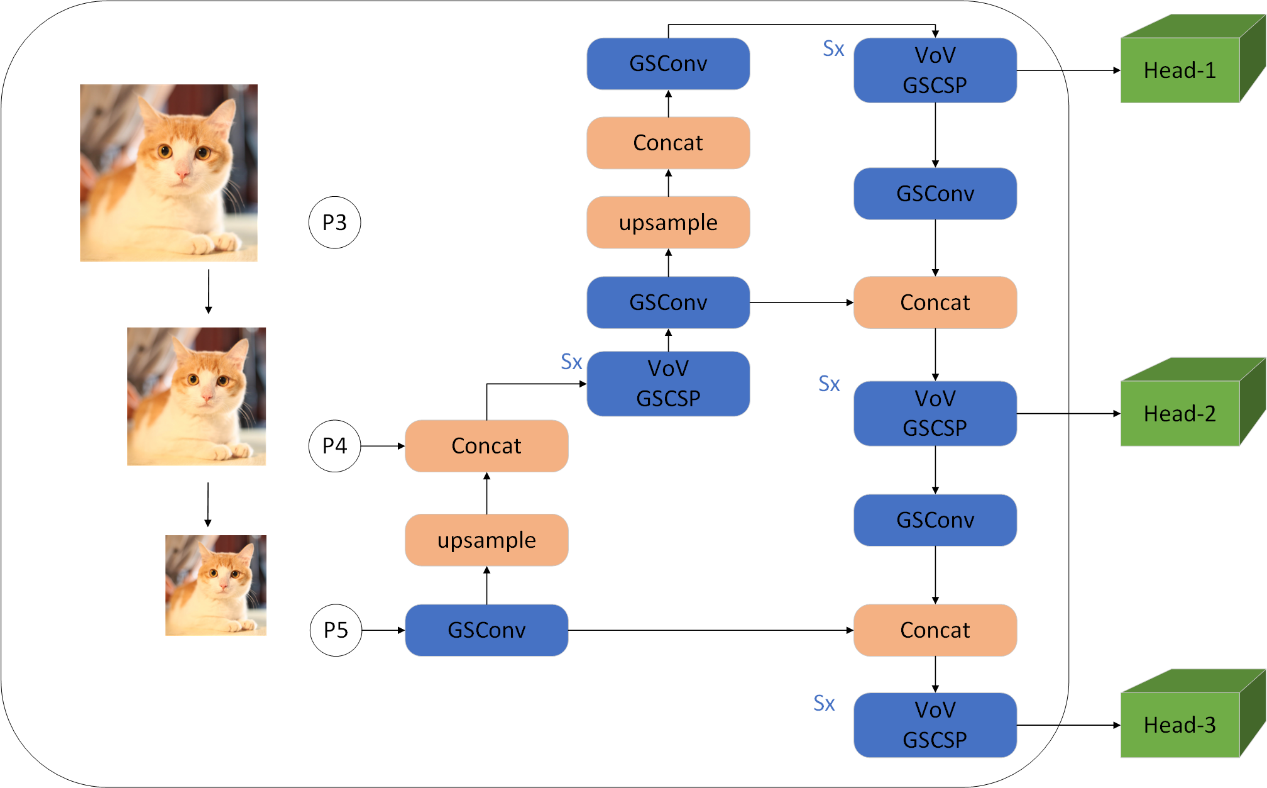


图2-6 Slim-neck架构

（3）检测头（Detection Head）检测头是YOLOv8的核心部分，负责生成目标检测的预测框。它包括多个卷积

层和全连接层，用于预测目标的边界框、类别和置信度分数。具体来说，YOLOv8的检测头包括以下几个关键部分：

1）卷积层：检测头的初始部分包含多个卷积层，用于处理特征图并提取更高级别的特征。

2）全连接层：在卷积层之后，有一系列全连接层。这些层负责将卷积层的输出映射到目标检测的相关参数，例如边界框坐标、目标性分数和类别概率

3）预测层：最后的全连接层输出被送入预测层。预测层根据模型的配置，生成目标的预测框。每个预测框包含目标的位置、类别标签和置信度分数。

YOLOv8的执行过程分为以下的五个步骤，该处理过程是一个端到端的过程，其在速度和准确性之间取得了良好的平衡，主要分为以下几个步骤。

1） 特征提取： 首先，输入图像通过骨干网络（如CSPDarknet53、CSPDarknetTiny或EfficientNet）进行特征提取。这些骨干网络会生成一系列特征图，其中包含不同尺度的特征信息。2） 颈部处理： 颈部连接到骨干网络，进一步处理特征图。它有助于融合不同尺度的特征，以便更好地捕获目标的上下文信息。3） 检测头预测： 检测头是YOLOv8的核心部分。它由卷积层和全连接层组成，用于预测目标的位置、类别和置信度分数。 对于每个特征图，检测头会生成一组预测框。每个预测框包含目标的边界框坐标、类别标签和置信度分数。4） 非极大值抑制（NMS）： 为了筛选出最终的检测结果，我们使用非极大值抑制（NMS）算法。NMS会删除高度重叠的预测框，只保留置信度最高的框，从而得到最终的目标检测结果。5） 输出： 最终的目标检测结果包括目标的位置、类别和置信度分数。这些结果可以用于实时物体检测、实例分割、姿态估计等各种计算机视觉任务。

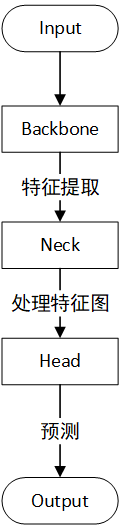


图2-7 YOLOv8的执行过程

## 2.2姿态提取模块

在对用户状态进行检测以后，需要对用户的姿态进行提取来进行计算从而判断用户的姿势是否标准，对于姿态提取，本文采用了YOLOv8的姿态提取模型YOLOv8-pose，YOLOv8-pose是Ultralytics公司最新的YOLOv8系列实时物体检测器的一个变种，专注于姿势估计任务，是一种基于深度学习的姿势估计技术，它结合了YOLOv8 目标检测算法和 Pose Estimation 姿势估计算法，实现了对图像或视频中的人体姿态进行高精度估计的功能。

姿态估计是一项识别图像当中的关键点位置的任务，关键点即物体的各个部位，如人体关键、地图上的地标等其他的显著特征，关键点通常使用一组二维[x, y]或者3D[x, y, z]坐标来表示。

YOLOv8-pose提供了多种经过预训练的姿态模型，其中Detect、Segment 和 Pose 模型是在[COCO](https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/datasets/coco.yaml)数据集上预先训练的，而 Classify 模型则是在[ImageNet](https://github.com/ultralytics/ultralytics/blob/main/ultralytics/cfg/datasets/ImageNet.yaml)数据集上预先训练的，以下是Ultralytics官方提供的模型一些性能参考：

表2-1 YOLOv8-pose模型性能参考

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 模型 | 分辨率 | mAP@0.5 | 速度 | 参数 | FLOPs |
| YOLOv8n-pose | 640 | 50.4 | 80.1ms | 1.18M | 3.3B |
| YOLOv8s-pose | 640 | 60.0 | 86.2ms | 1.42M | 11.6B |
| YOLOv8m-pose | 640 | 65.0 | 88.8ms | 2.00M | 26.4B |
| YOLOv8l-pose | 640 | 67.6 | 90.0ms | 2.59M | 44.4B |
| YOLOv8x-pose | 640 | 69.2 | 90.2ms | 3.73M | 69.4B |

综合精准度和速度考虑，本文采用了YOLOv8s-pose模型进行对人体部位关键点的提取，提取的人体关键点分布如图2-8、图2-9。

通过对人体关键点的提取，可以获取到用户身体部位的具体位置，通过关键点之间的计算可以获取肢体之间的角度信息，然后根据角度与标准角度之间的差别来判定用户姿态的标准程度。

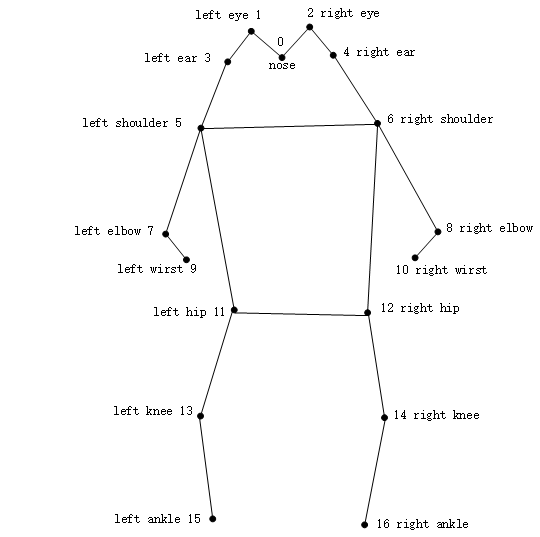


图2-8 YOLOv8-pose人体部位关键点分布图

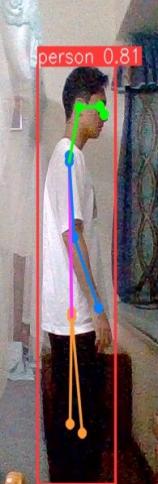
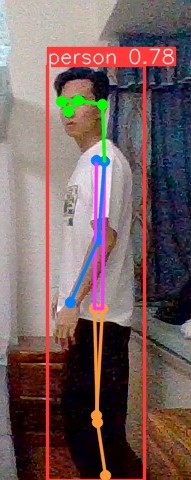
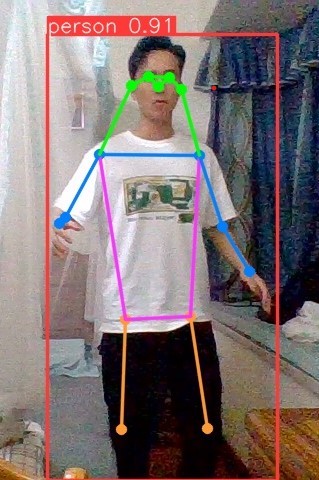
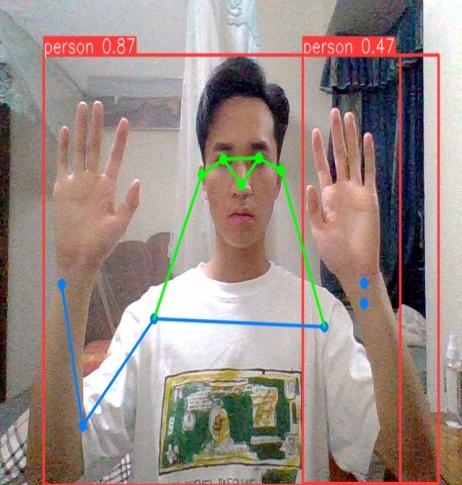


图2-9 YOLOv8-pose关键点提取展示

## 2.3相关开发环境

（1）硬件开发环境

本文实验环境如下表所示：

表2-2 实验环境及配置

|  |  |
| --- | --- |
| 类别 | 版本号 |
| Operating system | Ubuntu 22.04.3 |
| CPU | Intel(R) Xeon(R) Gold 5118 CPU @ 2.30GHz |
| GPU | Tesla V100-PCIE-16GB |
| CUDA | 12.2 |
| cuDNN | 8.9.2 |
| PyTorch | 2.1.1 |

（2）集成开发环境

1）Pycharm：[PyCharm是一款由JetBrains公司开发的强大的集成开发环境（IDE），主要用于Python语言开发](https://zh.wikipedia.org/wiki/PyCharm" \t "_blank)。它具有丰富的功能和工具，帮助开发者提高开发效率、改善代码质量，并支持各种 Python 相关的技术和框架，其主要功能包括代码导航和搜索、项目管理和版本控制、调试和测试以及改善代码质量和重构等。

2）Anaconda：Anaconda是一个安装和管理Python包的软件，其也自带Python、Jupyter Notebook和Spyder以及管理包的conda工具，其主要的特点有开源、安装过程简单、支持高性能使用Python和免费的社区支持等。

（3）软件包和相关库

1）PyTorch：PyTorch 是一个优化的张量库，用于使用 GPU 和 CPU 进行深度学习。其可以动态的计算图，这使得其在处理复杂模型时更加具有灵活性，并且对于研究人员来说更易于理解和调试。PyTorch的API设计较为直观，其与Python深度集成，易于使用和调试，其也提供了广泛的预训练模型，比如Resnet、VGG、Inception、SqueezeNet、EfficientNet等，在硬件利用方面其可以高效的利用NVIDIA的CUDA库进行GPU计算，实现高效的利用GPU，同时其也支持分布式计算，可以在多个GPU或服务器上训练模型。

2）OpenCV：OpenCV是一个跨平台的计算机视觉库，可用于开发实时的[图像处理](https://zh.wikipedia.org/wiki/%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86" \o "图像处理)、[计算机视觉](https://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%A7%86%E8%A7%89" \o "计算机视觉)以及[模式识别](https://zh.wikipedia.org/wiki/%E6%A8%A1%E5%BC%8F%E8%AF%86%E5%88%AB" \o "模式识别)程序。其可用于完成如增强现实、人脸识别、手势识别、人机交互、动作和物体识别、图像分割和视频处理等任务。其也支持多种编程语言如C++、Python、Java等，OpenCV可以在Windows, Android,Maemo, FreeBSD, OpenBSD, iOS, Linux和Mac OS等平台上运行。

3）Ultralytics：其开发的是Yolov8一款前沿的目标检测和图像分割模型。它基于深度学习和计算机视觉领域的尖端技术，在速度和准确性方面具有无与伦比的性能。YOLO (You Only Look Once) 是一种流行的物体检测和图像分割模型， YOLOv8 是 YOLO 的最新版本，由 Ultralytics 提供。其引入了许多创新技术，如可编程梯度信息 (PGI) 和通用高效层聚合网络 (GELAN)。本文调用了Ultralytics的YOLO包来对模型进行训练以及实现目标检测。

4）PyQt5：PyQt5是Qt框架的Python语言实现，由Riverbank Computing 开发，是最强大的 GUI 库之一。它提供了一个设计良好的窗口控件集合，如QtDesigner，用户可以自由的选定界面组件进行可视化界面的搭建，通过PyUIC将可视化界面的相关代码导出，方便用户对每个组件功能进行设计，本文采用PyQt5来为程序搭建一个可视化操作界面。

## 2.4本章小结

本章主要介绍了程序实现所需要的理论基础，以及功能实现对应的算法概述，包括算法当中的模块介绍以及各个模块的实现原理和流程，并对一些专业术语进行了介绍，最后介绍了程序所需的开发环境以及所需的工具和软件包等，这些也为后面的程序实现提供了基础。

# 第3章 系统设计与实现

## 3.1系统设计整体框架

整体系统设计大致思路如下，将视频源输入处理程序中逐帧处理。程序首先使用训练好的YOLOv8模型先进行图像检测与分类，判别目标当前状态，之后进入状态机处理，在检测到某个动作完成之后使用 YOLOv8-pose姿态估计算法对人体骨架进行提取，并使用设计好的评估算法对人体动作规范度进行统计与评估（例如躯体与地面形成的夹角，或者每分钟动作完成的个数），最终产生程序输出，具体流程如下图。

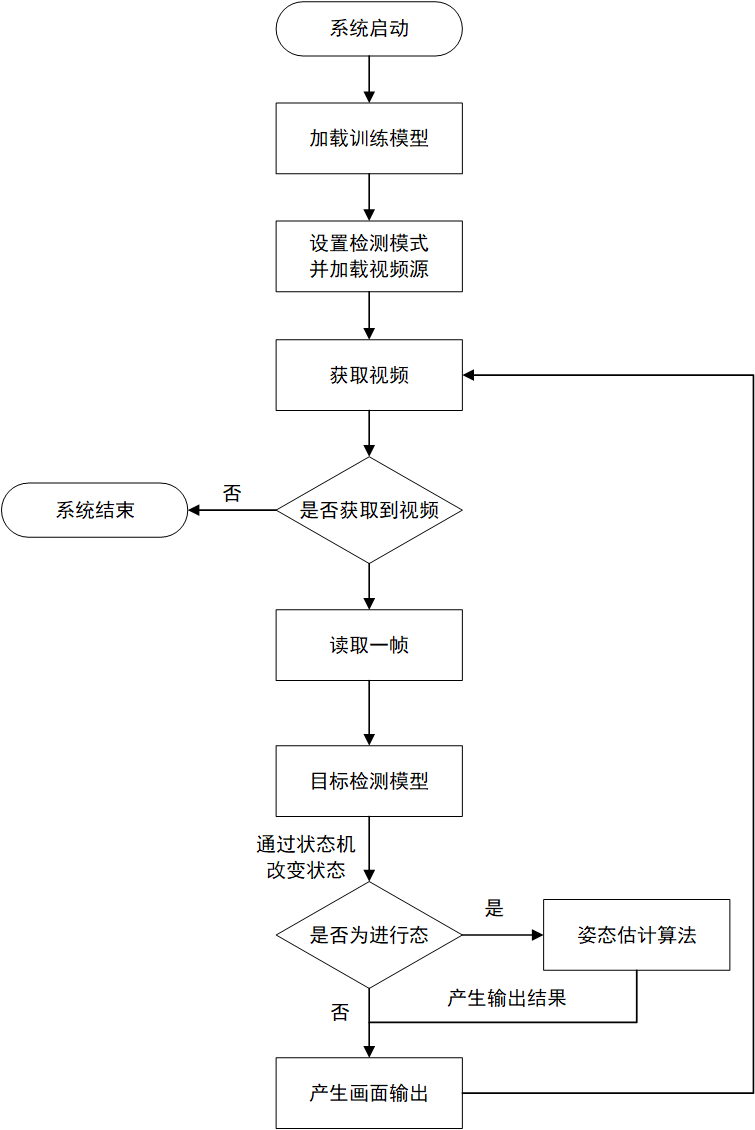


图3-1 系统设计整体流程图

本系统一共制作了三个动作的识别与姿态估计，其中视频检测模式即三个动作类：situp（仰卧起坐）、pushup（俯卧撑）、highknee（高抬腿），选择视频检测模式即选择自己需要检测的动作。

视频调用则通过OpenCV的cv2.VideoCapture()函数来获取视频源，其中输入参数为0时表示调用摄像头视频，输入为视频路径则从指定的路径当中调取视频。

首先需要通过目标检测模型来检测出目标的状态，即检测用户当前是否处于正在运动的状态，然后才能进行下一步的姿态估计过程。

姿态估计采用YOLOv8-pose模型将用户身体部位的关键点提取出来，然后计算所检测动作关键点之间的角度来判断动作的标准程度。

最终检测的画面输出包括：用户所完成的动作数量、用户的肢体角度、身体的平直程度（俯卧撑）等其他姿态评估结果。

## 3.2可行性分析

本文采用了Python语言作为主要的程序设计语言，YOLOv8为主要的目标检测算法和姿态提取算法，使用PyCharm作为程序开发工作，使用Anaconda创建Python解释器。

其中目标检测算法用于识别用户的运动状态，该过程需要获取并训练对应的用户状态数据集以训练模型，姿态提取即在识别出用户状态为进行态后，对人体姿态关键点进行提取，然后进行计算操作，最后输出结果，在理论上是可行的。

在完成了主体程序的设计之后，还需要搭建可视化操作界面以供用户使用，使用PyQt5来进行程序的可视化，使用UI构建工具QtDesigner来设计图形操作界面，再使用PyUIC来导出QtDesigner设计出的可视化界面对应的代码，再对导出的代码进行编写，来实现各个组件的功能，最后可将程序运行的结果展示在可视化界面上。

虽然相比YOLOv5，Yolov8推出的时间更晚且在应用上YOLOv5仍然更加流行，但是YOLOv8的模型部署比YOLOv5更为简单，精确度也更高，识别速度更快，目前在网络上也发布了许多YOLOv8的相关论文和开源项目，本文采用编译器运行方式，直接在PyCharm上运行程序即可显示出可视化操作界面，在运行上也是可行的。

## 3.3系统设计的重难点分析

（1）数据集预处理：对于训练模型所使用的数据集，有时会出现缺失、混乱、不全面的情况，数据预处理的目的是为了提高数据的质量，确保数据的可靠性和有效性，常用的数据预处理方法为数据清洗、数据集成、数据变换和数据缩减。

（2）模型训练时间及硬件环境：在训练模型时，若数据集的数量非常庞大，其对显存的要求也会更高，会导致模型的训练需要花费许多时间，为了能够有更好的训练环境，本文在Autodl服务器租贷平台租借了更好的模型训练环境，以便更加快速的完成模型训练。

（3）数据集的种类和数量：虽然网络上关于运动姿态的视频和图像数据十分繁多，但是大多数数据的质量却远远无法达到能够训练的水平，因此本文所使用的数据集皆为自行采集和拍摄，但由于人员和场地有限，数据集的种类和数量也注定会有所限制。

（4）姿态提取及肢体角度计算：在进行姿态提取的过程中，需要获取对应人体部位的关键点坐标，通过计算点与点之间的夹角来对人体姿态进行评估，计算角度则需要采用反三角函数进行计算，然而反三角函数的计算会出现正负问题以及角度的变换问题。

（5）可视化界面的搭建问题：在使用PyQt5来对可视化操作界面进行搭建时，在设计每个按钮以及输入框等模块的具体实现功能时，首先需要使得功能实现模块与可视化界面展示模块的代码互相独立，这就需要采用UI界面的动态加载功能。

## 3.4目标检测模型设计

### 3.4.1动作状态转移分析

对于不同类别的一系列动作，系统都将他们划分为通用的几种状态Init，Ready以及Ongoing，具体状态示意如下：







图3-2 动作状态示意图

仅当程序检测到目标完成预定的序列时，才会进行计数以及规范化分析。相对于无状态的图像检测处理系统，使用状态转移记录中间过程能保证对不真实的视频输入进行过滤：初始状态永远不可能直接跳转就绪状态到动作进行状态，例如在本系统中，一个“仰卧起坐”动作的完成会包含以下几个过程：从初始状态开始, 目标平躺处于就绪状态， 之后目标开始执行动作处于进行中状态，在这个状态下系统会对动作不断评估， 当触发动作完成条件时进行对上述评估结果进行统计并保存，再回到初始状态，注意其间任何其他输入都会将目标状态置为初始状态。

可以简单的将上述的状态转移过程用一个状态机图来描述，具体如下图3-3所示：

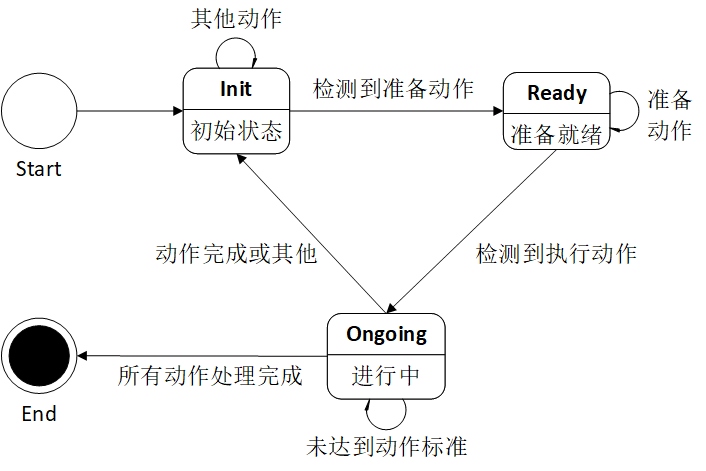


图3-3 程序执行状态机示意图

### 3.4.2 数据集的制作与训练

表3-1 数据集类别及张数

|  |  |  |
| --- | --- | --- |
| 编号 | 类别 | 张数 |
| 0 | SITUP-UP | 1035 |
| 1 | SITUP-ING | 1851 |
| 2 | SITUP-DOWN | 905 |
| 3 | HIGHKNEE-LEFT | 590 |
| 4 | HIGHKNEE-RIGHT | 580 |
| 5 | HIGHKNEE-ING | 1054 |
| 6 | PUSHUP-UP | 897 |
| 7 | PUSHUP-ING | 1054 |
| 8 | PUSHUP-DOWN | 527 |

本次实验采集并标注了一份数据集用于训练，为此特意建立了一个小组，小组成员于12个场景，对10位测试人员的三种侧身运动过程进行拍摄，每个视频中会有将产生长度为3小时30fps的视频序列，然后进行抽帧采样，采用标注工具labelImg对数据集进行标签，最终制作为实验数据集。数据集包含8493帧尺寸为 1920 x 1080的图片（见下图3-4和表3-1），涵盖9个类别, 总共大小约为7 GiB。数据集标签按照COCO 格式组织，每一个样本的标签都包含一个边界框位置以及类别信息标注的数据，以 4：1的比例被划分为训练集以及测试集。



图3-4 实验所用数据集部分展示

接着在训练过程中, 分别采用 ultralytics 官方提供的3个预训练模型同时进行训练，在四张NVIDIA V100上进行了训练，每一个训练时长为6 h，数据集配置文件(.yaml)及模型训练代码如下。

数据集配置文件(sports.yaml)：

path: train: train/images # 训练集文件位置val: val/images # 验证集文件位置names: 0: 'situp-up' 1: 'situp-ing' 2: 'situp-down' 3: 'highknee-left' 4: 'highknee-right' 5: 'highknee-ing' 6: 'pushup-up' 7: 'pushup-ing'

8: 'pushup-down'

模型训练代码：

from ultralytics import YOLO

if \_\_name\_\_ == "\_\_main\_\_": model = YOLO("yolov8m.pt") model.train( data='./sports.yaml', imgsz=640, # 输入图像大小为640\*640 epochs=5, # 训练轮次为五轮 device=[0] # 指定训练设备cuda:0 )

最后，训练出的模型可以识别出以下的九种状态。

situp-down situp-ing situp-up

pushup-up pushup-ing pushup-down

highknee-right highknee-ing highknee-left

图3-5 模型识别效果演示

## 3.5姿态估计模块设计

### 3.5.1 姿态估计程序设计

根据图3-1的程序流程图，在识别出用户运动状态后，当运动状态处于进行态时，即处于Ongoing状态时对用户运动姿态进行计算和估计。

对于不同的动作，需要设计不同的姿态估计算法来对每种姿势进行评估，具体程序设计思路即通过前面的目标检测算法识别出当前动作的状态，判断是否处于进行态，如果是处于进行态就对姿态进行提取并根据先前输入的动作模式将提取的姿态关键点结果送入对应的姿态估计算法当中进行计算并输出结果。

具体的代码实现，首先采用OpenCV的VideoCapture函数提取视频源，然后使用read函数提起每一帧视频，使用模型进行目标检测，然后还需要检测视频是否检测到动作以及当前获取到的动作状态等，当且仅当动作处于进行态时才使用姿态提取和评估算法，关键的代码实现及程序流程图如下：

# 检测是否获取到视频

success, frame = self.cap.read()

if not success:

self.isvideo = False

self.frame\_cnt += 1

# 检测用户的动作状态detection\_results = self.model.predict(frame)# 检查是否在检测\_class\_id\_vec = detection\_results[0].boxes.cls\_class\_id = -1object\_detected = Trueif \_class\_id\_vec.shape[0] == 0: object\_detected = Falseif object\_detected: \_class\_id = (int)(\_class\_id\_vec[0].item())# 获取当前识别到的运动状态，并进行姿态评估 id\_enum\_type = self.sport\_classes\_dict.get(\_class\_id) if id\_enum\_type is not None: pose\_results = self.pose\_model.predict(frame) # 提取关键点

self.fsm.transfer(id\_enum\_type, pose\_results) # 进行姿态估计 frame = pose\_results[0].plot()

# 计算评估结果if self.refresh\_frames\_count % self.score\_refresh\_frames == 0:self.displayed\_angle = self.fsm.angle\_if self.mode == HighkneeFSM:mean\_angle\_fmt = format(self.fsm.mean\_angle\_, '.2f')draw\_text(frame, f'mean angle: {mean\_angle\_fmt}' + ' deg', (50, 400))elif self.mode == PushupFSM:flatness\_fmt = format(self.fsm.flatness\_, '.2f')draw\_text(frame, f'flatness: {flatness\_fmt}', (50, 450))self.refresh\_frames\_count += 1

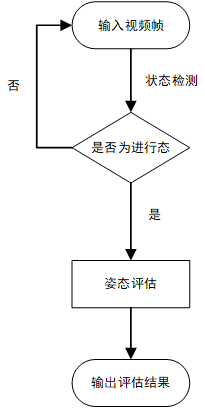


图3-6 姿态估计算法流程

### 3.5.2 姿态估计算法设计

对于姿态估计的算法实现，其核心思想即将姿态提取算法提取到的关键点（参考示意图2-8）进行角度的计算，根据角度的大小来评估姿态的标准程度。

其中提取到关键点，本文使用来表示一个关键点的横坐标和纵坐标，对于角度计算，则采用反正切函数进行计算，如计算关键点a与关键点b与水平面的夹角，则表示为来表示，具体表示法如下：

（1）俯卧撑

对于俯卧撑的计算，我们计算肩膀（keypoint\_16）到脚尖（keypoint\_6）的角度，从而来判断用户的身体到地面的距离，以及平直程度（Flatness），这反映了用户身体的弯曲程度，其由肩膀到臀部（keypoint\_12）与水平面的角度和臀部到脚尖与水平面的角度做差得到。对于计数，当躯干与水平面的角度小于一定程度时，且角度由减小转到增加时，计一次数，具体计算图示以及公式和实现代码如下。

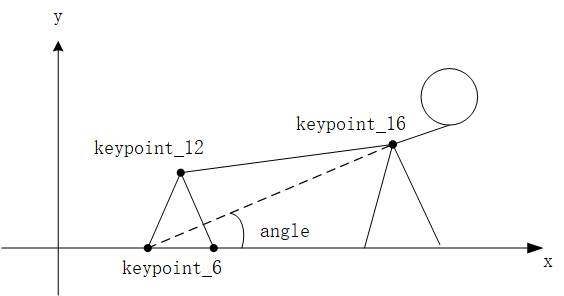
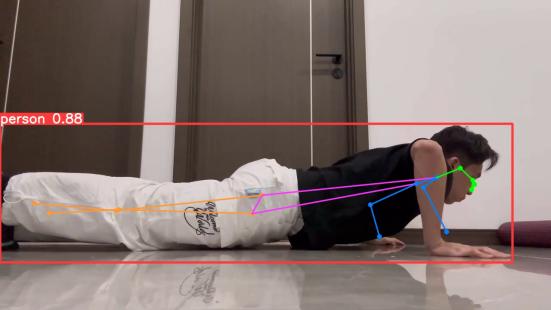


图3-7 俯卧撑角度计算

角度计算公式：

平直程度计算：

关键实现代码：

角度计算：

def evaluate(self, pose\_results):# flatnesskeypoints = pose\_results[0].keypoints.xyn[0] # 获取关键点

y\_leg\_delta = abs(keypoints[16][1] - keypoints[12][1]) # 计算关键点16、12在y轴方向的距离x\_leg\_delta = abs(keypoints[16][0] - keypoints[12][0]) # 计算关键点16、12在x轴方向的距离

leg\_angle = torch.arctan(y\_leg\_delta / x\_leg\_delta) # 计算关键点16，12与水平面的角度y\_body\_delta = abs(keypoints[12][1] - keypoints[6][1]) # 计算关键点12、6在y轴方向的距离x\_body\_delta = abs(keypoints[12][0] - keypoints[6][0]) # 计算关键点12、6在x轴方向的距离body\_angle = torch.arctan(y\_body\_delta / x\_body\_delta) # 计算关键点12，6与水平面的角度self.flatness\_ = abs(leg\_angle - body\_angle) # 计算平直程度# 计算肩膀到脚尖与水平面的角度y\_delta = keypoints[16][1] - keypoints[6][1]x\_delta = abs(keypoints[16][0] - keypoints[6][0])self.angle\_ = torch.arctan(y\_delta / x\_delta) \* 180 / PI # 弧度角度化

动作计数：

# 在进行状态计数

if SportState.ONGOING == self.state\_:if PushupFSM.TypeInput.PUSHUP\_DOWN == \_input:

# 在进行状态计数

if SportState.ONGOING == self.state\_:if PushupFSM.TypeInput.PUSHUP\_DOWN == \_input: last\_angle = self.angle\_ self.evaluate(pose\_results) if last\_angle <= self.angle\_: # 角度由递减到递增时，计一次数 self.count\_ += 1 self.state\_ = SportState.INIT else: self.state\_ = SportState.ONGOINGelif PushupFSM.TypeInput.PUSHUP\_ING != \_input: self.state\_ = SportState.INIT

（2）仰卧起坐

对于仰卧起坐，则计算躯干后背与水平面的角度，当肩膀越过臀部关键点时，角度的计算方式则需要变化，分为一下两种情况 第一种情况，即后背与水平面的角度小于90度时，此时直接使用反正切函数即可计算出角度，后背所连的两个关键点为keypoint\_5和keypoint\_11，当角度小于90度时，此时keypoint\_5不在keypoint\_11和keypoint\_15的一侧，可以以此作为条件来判断角度的范围；第二种情况当角度大于90度时，此时keypoint\_5在keypoint\_11和keypoint\_15的一侧，此时使用反正切函数求出的角度为补角，此时需要进行转换，具体的图解和对应公式如下：

1）角度小于90度的情况：

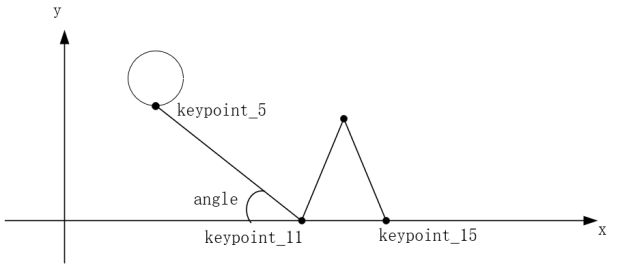


图3-8 仰卧起坐角度计算

角度计算公式：

1. 角度大于90度的情况

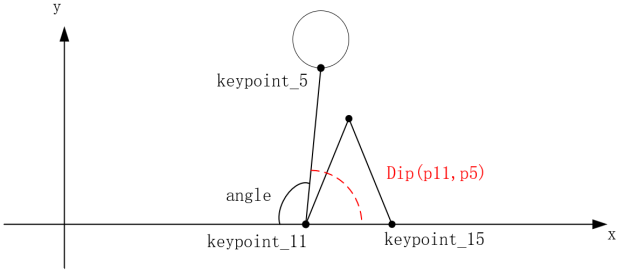


图3-9 仰卧起坐角度计算

角度计算公式：

将上述的两种情况合并为一个分段函数，即分为角度小于和大于90度的情况：

对于计数，当后背与水平面的角度大于最大角度时，且角度由递增转为递减时，计一次数，角度计算及计数的代码示例如下：

角度计算：

def evaluate(self, pose\_results):keypoints = pose\_results[0].keypoints.xyn[0] # 提取关键点x\_delta = keypoints[11][0] - keypoints[5][0] y\_delta = keypoints[11][1] - keypoints[5][1]

angle\_rad = torch.arctan(y\_delta / abs(x\_delta)) # 计算角度if abs(keypoints[11][0] - keypoints[5][0]) + abs(keypoints[11][0] - keypoints[15][0])\== abs(keypoints[5][0] - keypoints[15][0]): # 判断当前角度是否小于或大于90度 self.angle\_ = PI - angle\_radelse: self.angle\_ = angle\_rad self.angle\_ = self.angle\_ \* 180 / PI

动作计数：

if SportState.ONGOING == self.state\_: if SitupFSM.TypeInput.SITUP\_UP == \_input: last\_angle = self.angle\_ # 保存上一次的角度 self.evaluate(pose\_results) if last\_angle >= self.angle\_: # 角度递减时计一次数 self.count\_ += 1 self.state\_ = SportState.INIT else: self.state\_ = SportState.ONGOING

elif SitupFSM.TypeInput.SITUP\_ING != \_input: self.state\_ = SportState.INIT

1. 高抬腿

对于高抬腿，这里选择计算大腿（keypoint\_13与keypoint\_11相连的直线）与y轴之间的角度，而此时的角度计算又分为以下两种情况：

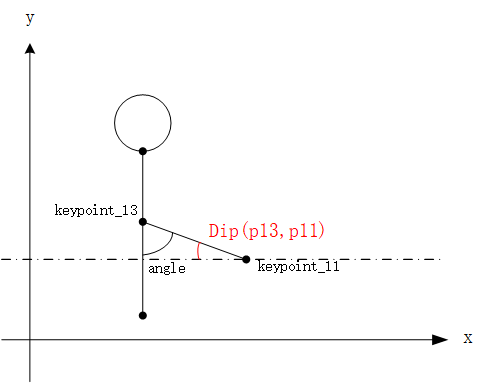
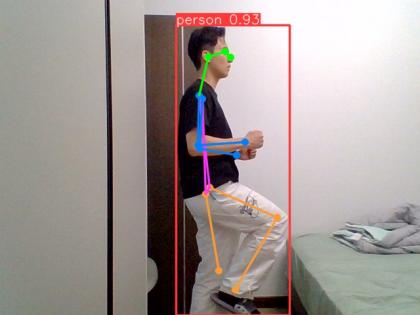


图3-10 高抬腿角度计算

1）如图3-9-1所示，当大腿与y轴形成的角度小于90度时，关键点13在关键点11的上方，此时先使用反正切函数计算出其与水平面的夹角，再求其余角即可，计算公式如下：

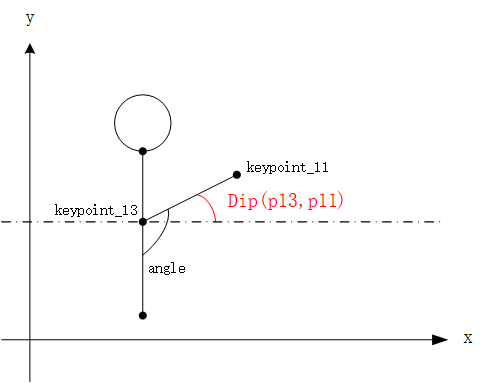
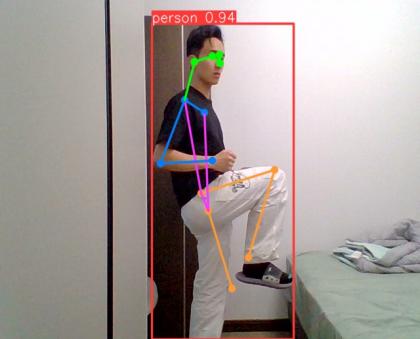


图3-11 高抬腿角度计算

2）如图3-9-2所示，大腿与y轴形成的角度大于90度时，关键点13在关键点11的下方，此时先使用反正切函数计算出其与水平面的夹角，再加上下面的90度即可，计算公式如下：

将上述两种情况合并为一个分段函数，具体如下：

对于计数，当大腿与y轴之间的角度开始递减时，计一次数，相关代码实现如下：

角度计算：

def evaluate(self, pose\_results):keypoints = pose\_results[0].keypoints.xyn[0] # 提取关键点y\_delta = 0.0x\_delta = 0.0if keypoints[13][1] < keypoints[14][1]: # 判断角度大于和小于90度的情况 y\_delta = keypoints[13][1] - keypoints[11][1] x\_delta = abs(keypoints[13][0] - keypoints[11][0])else: y\_delta = keypoints[14][1] - keypoints[12][1] x\_delta = abs(keypoints[14][0] - keypoints[12][0])angle\_rad = torch.arctan(x\_delta / y\_delta).item() # 这里为了简便计算，将x放在上面if angle\_rad < 0: angle\_rad += PI - angle\_radself.angle\_ = angle\_rad \* 180 / PI

动作计数：

def evaluate(self, pose\_results):keypoints = pose\_results[0].keypoints.xyn[0] # 提取关键点y\_delta = 0.0x\_delta = 0.0if keypoints[13][1] < keypoints[14][1]: # 判断角度大于和小于90度的情况 y\_delta = keypoints[13][1] - keypoints[11][1] x\_delta = abs(keypoints[13][0] - keypoints[11][0])else: y\_delta = keypoints[14][1] - keypoints[12][1] x\_delta = abs(keypoints[14][0] - keypoints[12][0])angle\_rad = torch.arctan(x\_delta / y\_delta).item() # 这里为了简便计算，将x放在上面if angle\_rad < 0: angle\_rad += PI - angle\_radself.angle\_ = angle\_rad \* 180 / PI

## 3.6本章小结

本章节介绍了本系统的实现流程以及各个功能模块的实现原理及推导过程，并对系统研究的可行性做出的分析，列举出了一些实现的重难点并提出了解决方法，然后详细介绍了目标检测模型的功能以及训练过程，之后介绍了姿态估计算法的推导过程和及其可能出现的各种情况，后附有关键代码实现。

# 第4章 系统测试与评估

本章节将会对先前所介绍的系统设计的各个模块进行一个整合，完成整体系统的设计，并搭建一个可视化界面，用于展示最终的系统，最后对系统进行一系列的测试与评估。

## 4.1实验结果展示

本实验采用PyQt5的QtDesigner工具为系统搭建了一个可视化操作界面，具体如下：



图4-1 可视化界面

在使用该系统时，首先需要选择视频路径或者调用摄像头，然后选择需要进行评估的动作，动作选项共有高抬腿、仰卧起坐、俯卧撑三种；本实验一共训练出了三个模型，可以调用三种不同的模型进行目标检测，帧间隔即视频输入的速率。

在选择完上述的几个选项之后，点击运行按钮即可开始运行程序，屏幕在默认状态下为黑屏，点击运行之后即可显示，当点击结束按钮时结束视频播放。

其中界面当中会展示评估结果，包括当前的动作角度，动作频率，躯干的平直程度（俯卧撑）、平均角度、已经完成的动作数量等。



图4-2 系统效果展示（仰卧起坐）



图4-3 系统效果展示（俯卧撑）

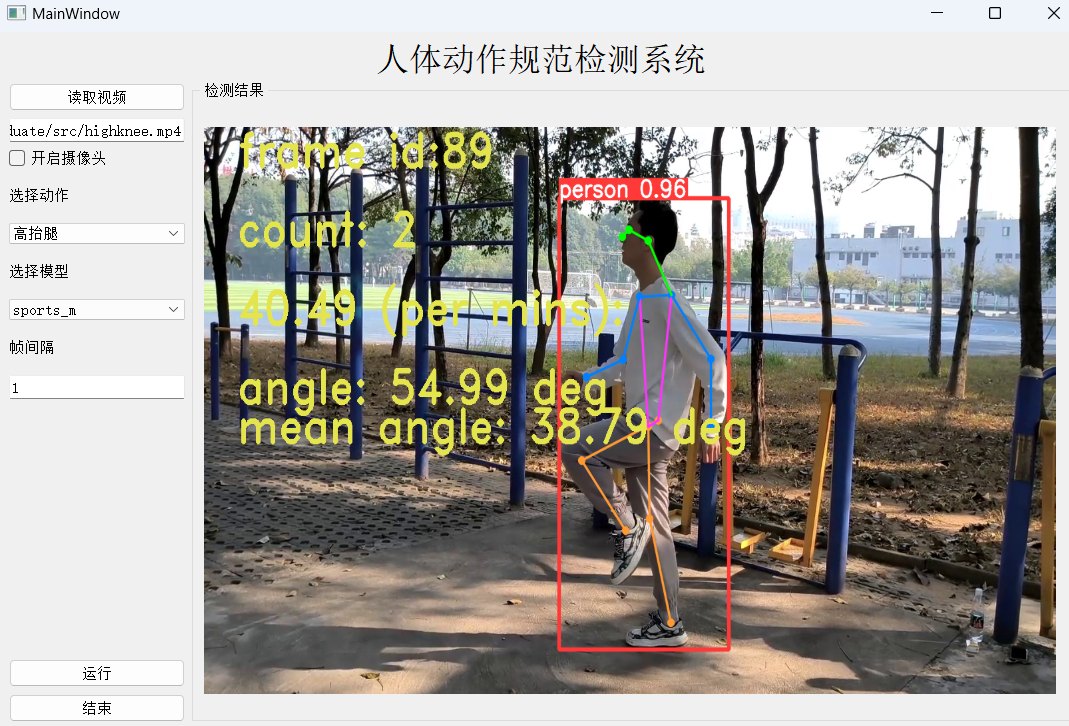


图4-4 系统效果展示（高抬腿）

## 4.2模型准确度评估

本实验采用了微软的 COCO (common Object in Context)的 mAP (mean Average Precision) 作为模型评估指标，以下是一些相关概念的介绍：

1. 交叉比（IoU）：定义为两矩形框交集面积与并集面积之比，是矩形框重合程度的衡量指标，计算公式如下所示：

（2）准确率（precision）：

其衡量的是模型预测的正确率，即预测正确的样本占总样本数的百分比。

（3）召回率（recall）：

其衡量的是模型对正确样本的查全率，其表示实际为整样本当中被预测为正确样本的比例。

准确率和召回率的计算公式如下：

* (TP)（True Positive）：实际为正样本，且被预测为正样本（预测正确）。
* (TN)（True Negative）：实际为负样本，且被预测为负样本（预测正确）。
* (FP)（False Positive）：实际为负样本，但被预测为正样本（预测错误）。
* (FN)（False Negative）：实际为正样本，但被预测为负样本（预测错误）。

准确率：

召回率：

（4）AP值和mAP值：

在一般情况下，准确率和召回率不可同时兼顾，通过改变阈值（IoU大于一定的阈值时被判定为正样本），可以得到不同的准确率和召回率。以得到的召回率为横坐标，准确率为纵坐标画得的图为PR图，AP是指所有图片内的具体某一类的PR曲线下的面积，mAP就是所有图片内的所有类别的AP的平均值。

本实验得到的模型（sports\_m、sports\_n、sports\_x）的PR曲线图如下：

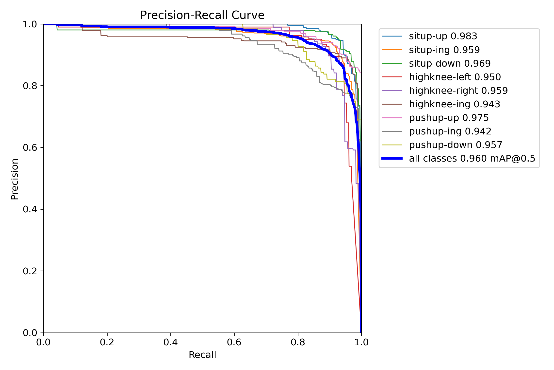


图4-5目标检测模型sports\_m的PR曲线图

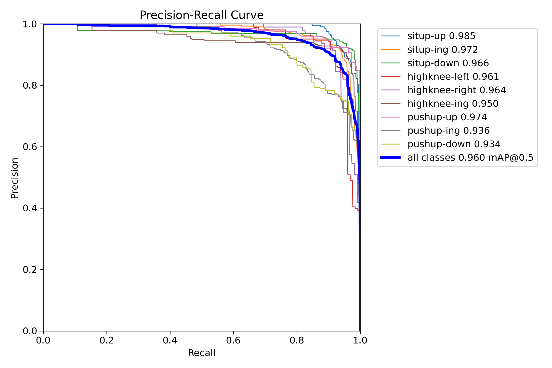


图4-6目标检测模型sports\_n的PR曲线图

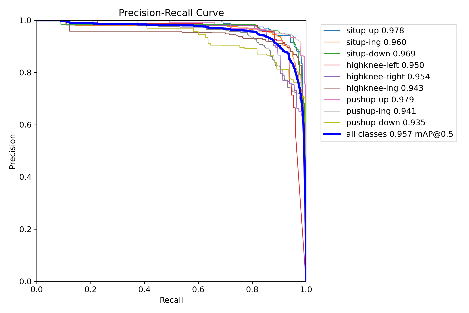


图4-7目标检测模型sports\_x的PR曲线图

不同预训练模型在训练集上取得了不同精度与推理速度，在三个不同规模的预训练模型上，分别使用2张V100进行训练与并在一张 3060Ti 上进行推理测试，图像大小为640 x 640, 得到以下数据。注意推理时间未包括图像处理以及系统其他开销的时间(这些时间约为10ms)。如下表格所示，使用n模型，将推理时间加上其他系统开销，单帧处理速度在15ms左右，即保证检测的实时性（以66 FPS的速度进行检测），各模型得到的各项指标如下表所示：

表4-1 目标检测模型各项指标

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 训练时间(h) | 模型大小(MB) | mAP@50-95 | mAP@50 | 推理时间 ( ms ) |
| sports-m | 2.108 | 6.2 | 0.898 | 0.96 | 5 |
| sports\_n | 3.213 | 52 | 0.899 | 0.96 | 9 |
| sports\_x | 5.937 | 136.7 | 0.9 | 0.957 | 19 |

最后得到训练轮数与平均mAP值关系，从左到右依次为模型sports\_n,m,x。

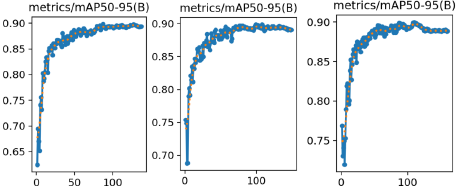


图4-8 mAP值与训练轮数变换曲线图

模型训练的损失函数变换过程及训练效果（节选）展示如下：

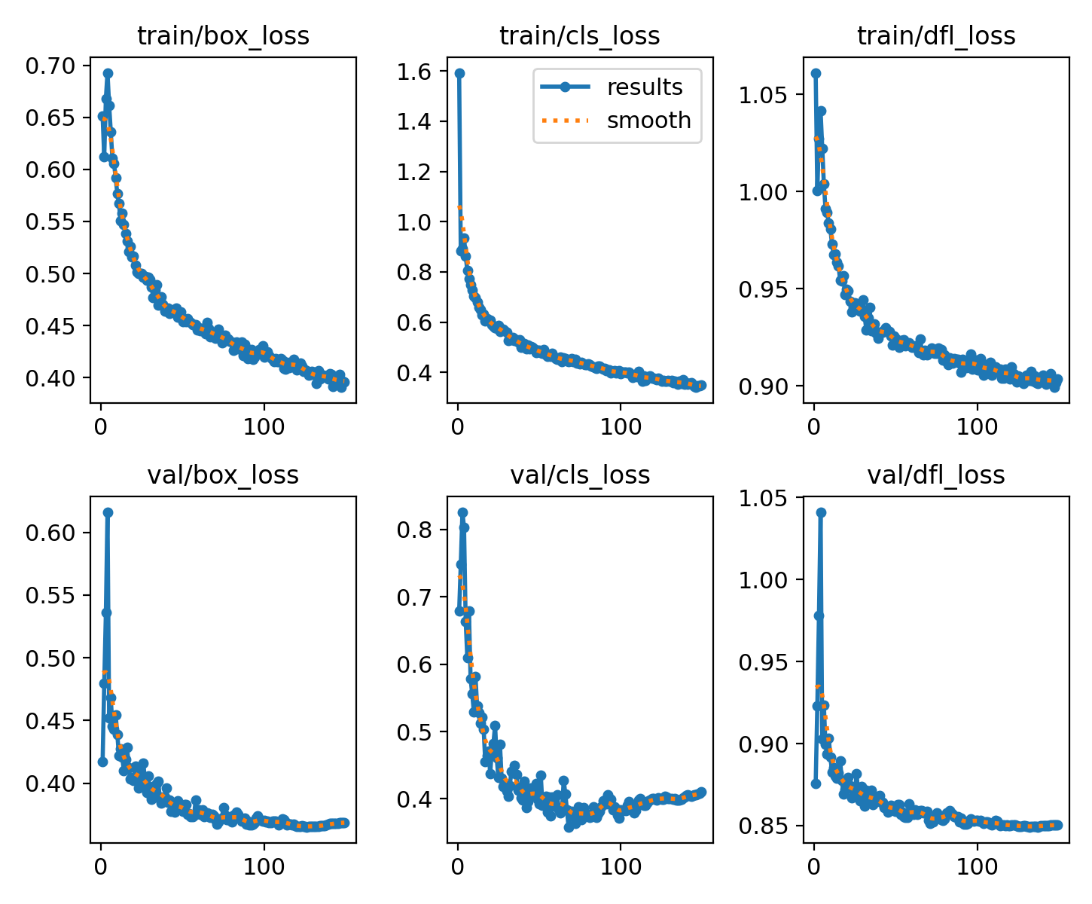


图4-9损失函数变换过程sports\_m

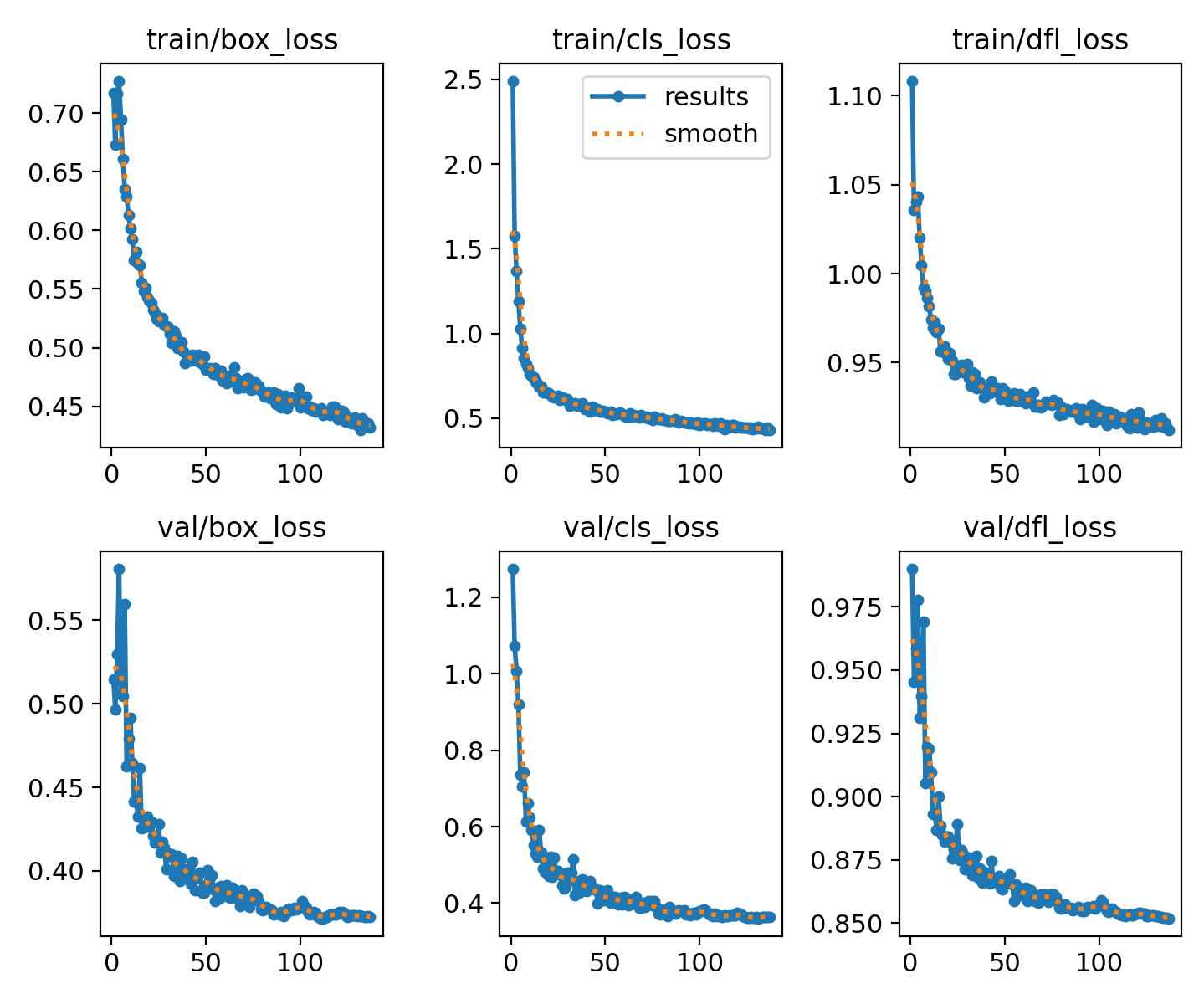


图4-10损失函数变换过程sports\_n

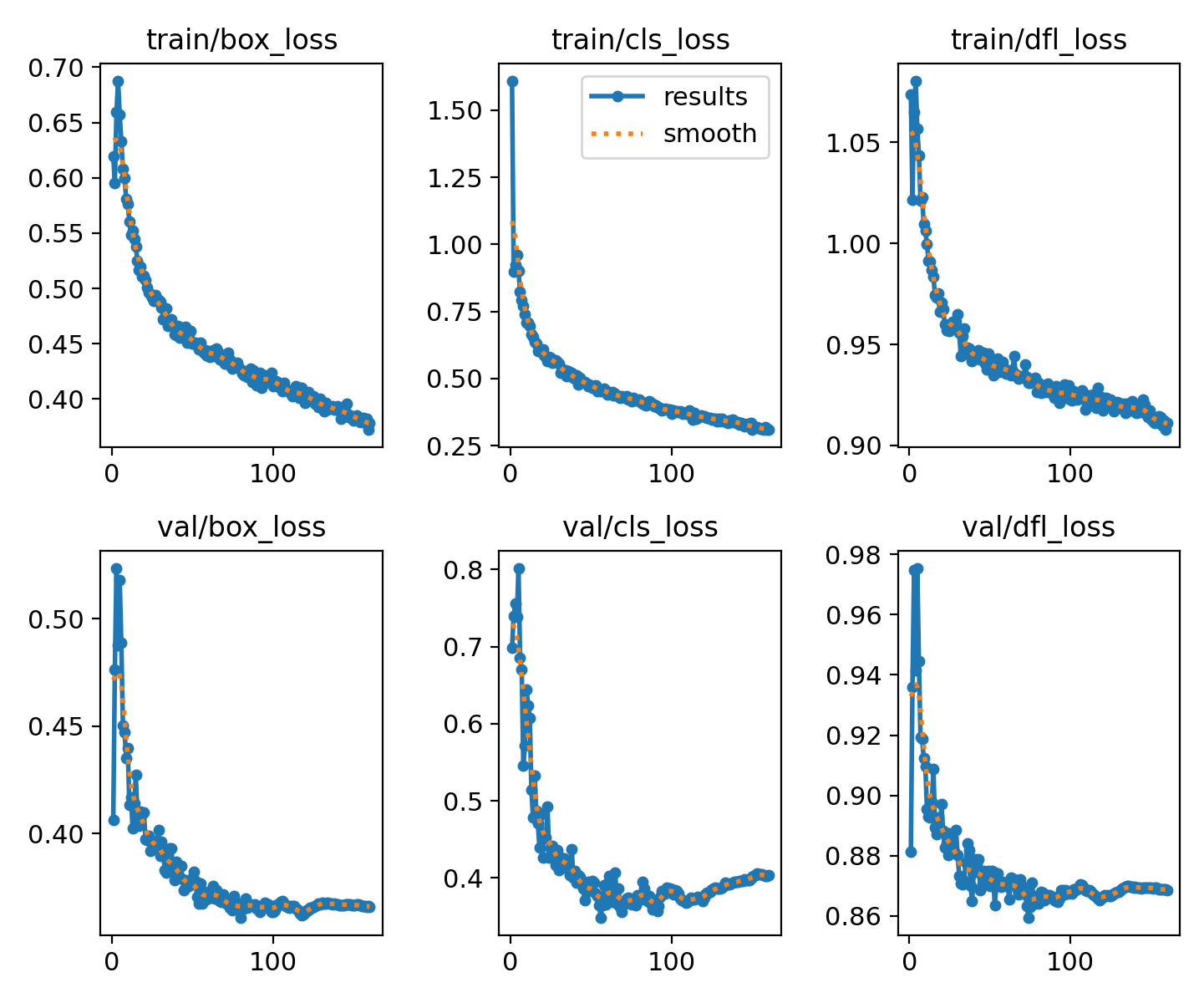


图4-11损失函数变换过程sports\_x

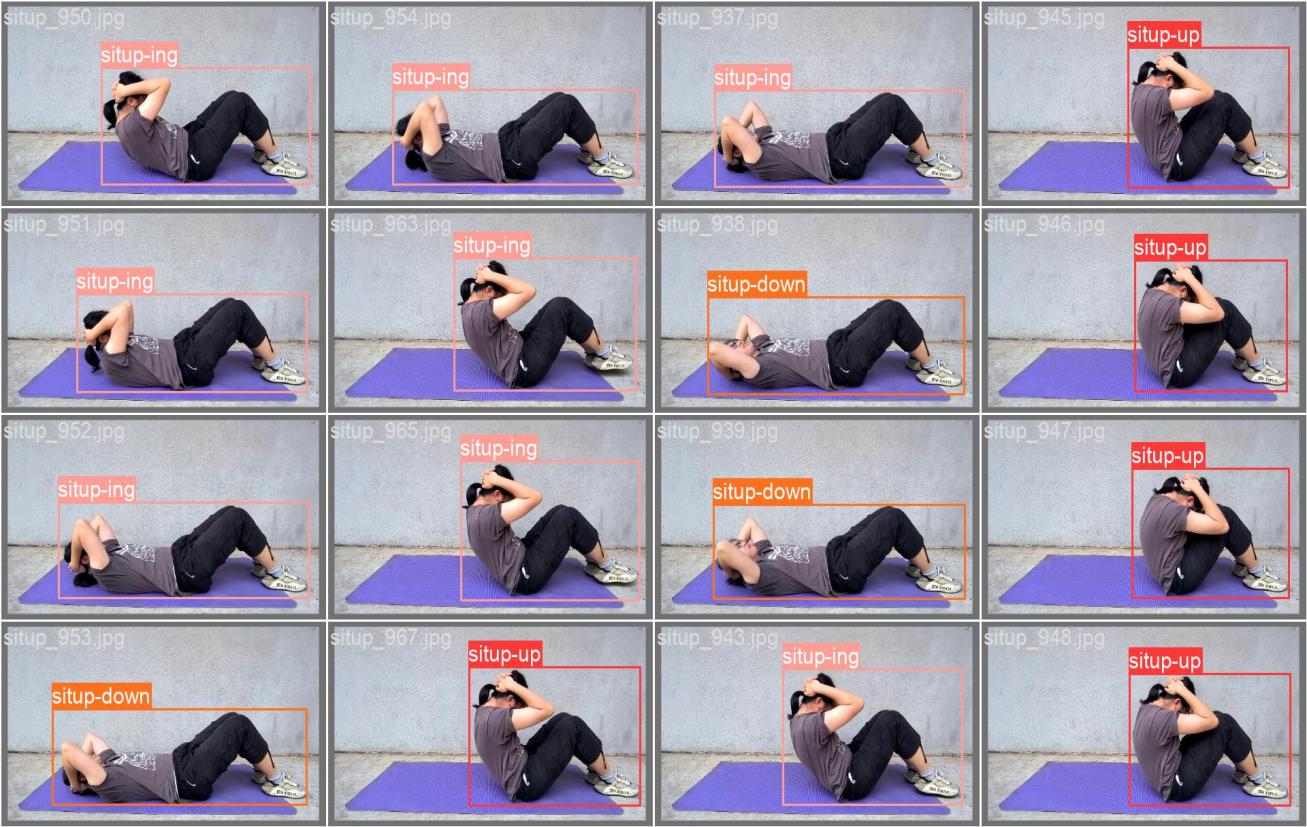


图4-12模型训练效果图

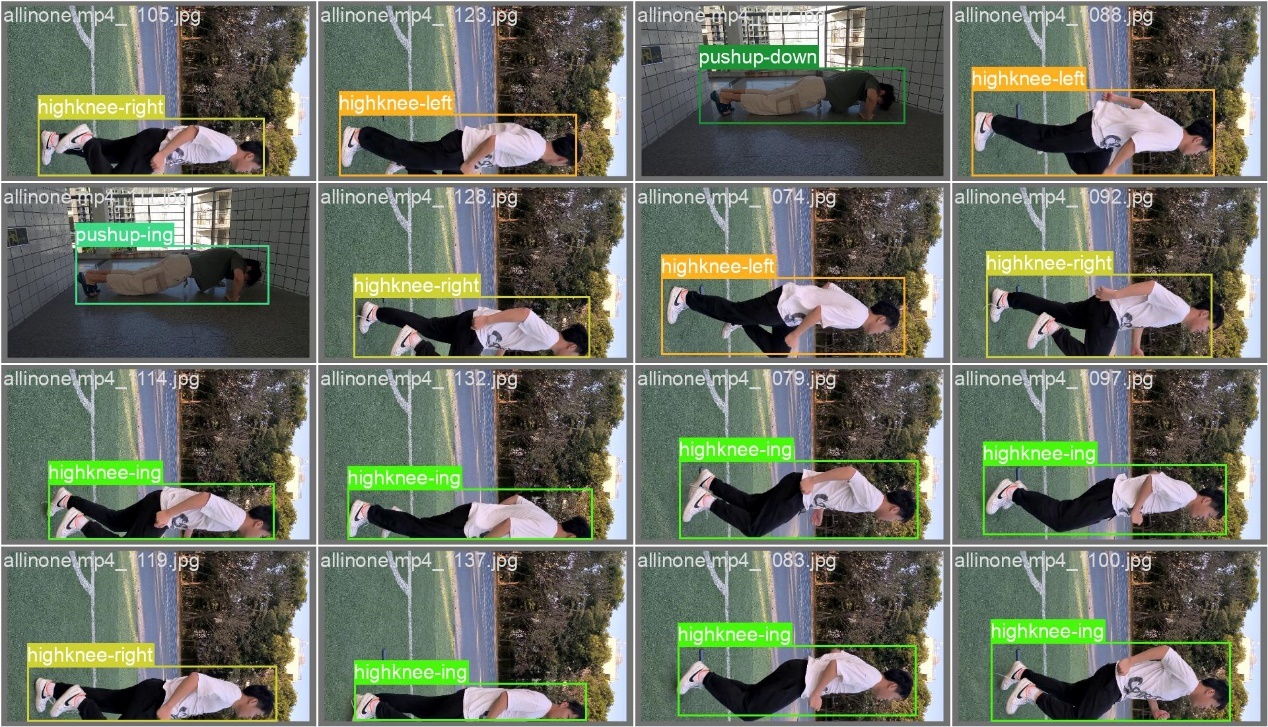


图4-13模型训练效果图

## 4.3系统存在的缺陷

（1）经过实验，发现当有其他外界因素干扰时，模型的精确度会受到影响，从而降低最后的评估结果。该模型在其他的场合，以及对于不同体型差异的用户，精确度也不同，导致模型的泛化能力不是很高（具体演示如图4-6所示，可见尤其是在多人场景当中，识别效果并不理想）。

（2）训练轮次不足，以及数据集较为单一，无法让模型适应更多的场合，模型出现了一些过拟合现象。

（3）存在延迟卡顿问题，该模型与可视化界面结合运行，目前尚未做到多任务处理，导致出现画面的延迟卡顿，且对电脑环境的要求较高。

图4-14 在多人的场景进行识别

## 4.4本章小结

本章主要对本文设计的系统进行了整合，使用PyQt5做出了系统的可视化界面，并采用mAP值、损失函数等指标对系统的准确性进行了评估，以及对上述指标涉及的相关概念进行介绍，并做出曲线图来表示，最后对该系统存在的一些缺陷进行了分析。

# 第五章 总结与展望

本文设计了一种基于YOLOv8和状态机算法的运动评估系统，该系统能够在无需穿戴设备的情况下，对人体动作进行规范化检测和评价。其制作了一个包含9个类别，8493张图片的数据集，并对其进行了训练。在测试集上，本系统实现了平均mAP值约0.9的检测精度，约15ms的单帧处理速度，表明了其在实时性和准确性上的优势。此外，本系统通过对动作的连续性进行考虑，能够更准确地评估动作的规范性，从而提供更准确的反馈。

此外对于整个系统，还存在一下几种可以优化的方案：

1. 数据增强：适当的给模型训练的数据集添加噪音，增加数据的多样性（如换多人进行数据集拍摄及在多人场景下进行识别），减少过拟合，并提高模型对不同角度、尺度和光照等变化的鲁棒性。
2. 网络架构优化：优化 YOLOv8 的网络架构也可以提高算法的准确性，例如，可以添加更多的卷积层或改变卷积层的大小等。这样可以改进模型的特征提取能力。对于多人场景，若提取到了多个提取框，则选择最大的提取框进行识别，从而提高准确性。
3. 更好的特征：改进数据标注，让模型学到更本质的信息，提高泛化效果
4. 减少延迟：对于视频流在qt界面当中的显示，采用多线程技术，实现qt页码的展示与视频处理的同步，或者采用更高配置的GPU终端。

总的来说，该系统提供了一种有效、便捷、低成本的解决方案，对于视频安全监控、视频检索、健康医疗等领域具有广泛的应用前景。未来也将进一步优化该系统，以提高其性能，并扩展其应用范围。

# 参考文献

1. 宋震, 张宇姝, 杨刚. 人体动作识别与评价综述[J]. [中国传媒大学学报(自然科学版), 2021, XX(6): 61-65](https://journal.cuc.edu.cn/frmPdfShow.aspx?id=2042" \t "_blank)
2. 施克权,李祺,隋皓等.IEMAyoloViT：基于改进YOLOv8的水下目标检测算法[J/OL].电讯技术,1-10[2024-02-23].
3. 曹雨淇,徐慧英,朱信忠等.基于YOLOv8改进的打架斗殴行为识别算法：EFD-YOLO[J/OL].计算机工程与科学,1-14[2024-02-23].
4. 朱强军,胡斌,汪慧兰等.基于轻量化YOLOv8s交通标志的检测[J/OL].图学学报,1-10[2024-02-23].
5. 韩强. 面向小目标检测的改进YOLOv8算法研究[D].吉林大学,2024.DOI:10.27162/d.cnki.gjlin.2023.001647.
6. 刘莫尘,褚镇源,崔明诗等.基于改进YOLO v8-Pose的红熟期草莓识别和果柄检测[J].农业机械学报,2023,54(S2):244-251.
7. 丁晓慧,周磊.Wi-Fi动作识别在室内入侵检测中的应用[J].科技创新与应用,2023,13(24):64-67.
8. 黄倩,崔静雯,李畅.基于骨骼的人体行为识别方法研究综述[J/OL].计算机辅助设计与图形学学报,1-22[2024-02-23].
9. 王茹,刘大明,张健.Wear-YOLO：变电站电力人员安全装备检测方法研究[J/OL].计算机工程与应用,1-13[2024-02-23].
10. 陈俊颖,郭士杰,and 陈玲玲.基于解耦注意力与幻影卷积的轻量级人体姿态估计.计算机应用 1-13.
11. 周珍玉,秦学,蔡芳等.基于人物交互的学生课堂行为识别研究[J].现代教育技术,2024,34(02):53-61.
12. 韩康,李敬兆,陶荣颖.基于改进YOLOv7和ByteTrack的煤矿关键岗位人员不安全行为识别[J].工矿自动化,2024,50(03):82-91.DOI:10.13272/j.issn.1671-251x.2024030015.
13. 汪洋继鸿,张路,于越,等.一种轻量化三维人体姿态估计算法[J].通信与信息技术,2024,(02):32-35+41.
14. 赵小虎,王轲鑫,孟献峰,等.基于人体姿态估计的羽毛球动作评估方法研究[J/OL].华中科技大学学报(自然科学版),1-9[2024-04-21].https://doi.org/10.13245/j.hust.240261.
15. 王燕妮, et al.多尺度和多层级特征融合的人体姿态估计.计算机工程与应用 1-12.
16. Karácsony T ,Jeni A L ,Torre L D F , et al.Deep learning methods for single camera based clinical in-bed movement action recognition[J].Image and Vision Computing,2024,143104928-.
17. Weiguang Jiang,and Lieyun Ding.Unsafe hoisting behavior recognition for tower crane based on transfer learning. Automation in Construction 160.(2024) :105299-.
18. Xinpeng Yin, et al.Spatiotemporal Progressive Inward-Outward Aggregation Network for skeleton-based action recognition.Pattern Recognition 150.(2024):110262-.
19. Wang C ,Wang J ,Xu W .Double branch synergies with modal reinforcement for weakly supervised temporal action detection[J].Journal of Visual Communication and Image Representation,2024,99104090-.
20. Solimani F ,Cardellicchio A ,Dimauro G , et al.Optimizing tomato plant phenotyping detection: Boosting YOLOv8 architecture to tackle data complexity[J].Computers and Electronics in Agriculture,2024,218108728-.
21. Wang Y .Abnormal behavior identification of enterprise cloud platform financial system based on artificial neural network[J].Computers and Electrical Engineering,2024,115109110-.
22. Liu Q ,Liu Y ,Lin D .Revolutionizing Target Detection in Intelligent Traffic Systems: YOLOv8-SnakeVision[J].Electronics,2023,12(24):
23. Zheng W ,Yan L ,Siyuan D , et al.An efficient detection of non-standard
24. Shizhong Y ,Wei W ,Sheng G , et al.Strawberry ripeness detection based on YOLOv8 algorithm fused with LW-Swin Transformer[J].Computers and Electronics in Agriculture,2023,215
25. Hang Yu, et al.Research on an Intelligent Identification Method for Wind Turbine Blade Damage Based on CBAM-BiFPN-YOLOV8.Processes 12.1(2024):
26. Tomenotti F F ,Noceti N ,Odone F .Head pose estimation with uncertainty and an application to dyadic interaction detection[J].Computer Vision and Image Understanding,2024,243103999-.
27. Zhou L ,Chen Y ,Wang J .SlowFastFormer for 3D human pose estimation[J].Computer Vision and Image Understanding,2024,243103992-.
28. Xu X ,Liu L ,Yan S .SMPLer: Taming Transformers for Monocular 3D Human Shape and Pose Estimation.[J].IEEE transactions on pattern analysis and machine intelligence,2024,46(5):3275-3289.
29. Yang J G ,Kim H J ,Lee W S .Geometry-driven self-supervision for 3D human pose estimation[J].Neural Networks,2024,174106237-.
30. Chengang D ,Guodong D .An enhanced real-time human pose estimation method based on modified YOLOv8 framework[J].Scientific Reports,2024,14(1):8012-8012.

# 致谢

到此，整篇论文的内容就结束了，这也意味着本科生活就即将画上句号，我在完成本次毕业设计的过程当中也收获颇丰，在这里我也十分感谢指导我完成整篇毕业论文的任子良老师。

回想起这大学四年的学习生涯，我也不禁十分感慨，在这四年里，我觉得自己活的忙碌且充实。在同学、老师的帮助下，我不光学习了许多的专业知识，也学会了许多处理问题的解决方法，在完成毕业设计的时候我也遭遇过许多的困境，但最终还是在自己的不断探索下解决了大部分的问题，在这里我不光要感谢自己的辛勤付出，感谢四年以来同学们的陪伴，也要感谢老师、学校对我的栽培，正是在你们的支持下，我才能成长为一名合格的毕业生。

至此，我也谨以本文，来作为我大学本科四年生活的终点，亦为我人生旅途的一个新的起点。