

**人工智能 课程设计**

**设计题目 基于YOLO等模型的应用系统开发**

**学院名称 计算机与网络安全学院**

**专业名称 数字媒体技术专业**

**学生姓名 张凯舒、李颖**

**目 录**

目录

[1.实验目的和背景实验环境描述 2](#_Toc186053675)

[2.系统设计和算法描述 4](#_Toc186053676)

[3.实验步骤和结果展示 7](#_Toc186053677)

[4.实验分析和讨论 11](#_Toc186053678)

[5.结论和展望 12](#_Toc186053679)

### 1.实验目的和背景实验环境描述

##### 1.1实验目的

本实验旨在通过设计并实现一个基于YOLO等模型的人工智能应用系统，掌握现有人工智能模型的基本原理和技术，熟悉这些模型在实际应用场景的部署，并通过实践加深对人工智能模型原理的理解和应用场景使用的基本技能。

利用 YOLO（You Only Look Once）模型进行视频流中的物体实时检测系统。通过此实验，我们希望达到以下目标：

1. 理解 YOLO 模型的基础原理和工作机制。

2. 掌握如何在 Python 环境下使用预训练的 YOLO 模型进行实时视频处理。

3. 学会调试和优化模型以应对不同的环境条件，提高检测效率和准确性。

随着深度学习技术的发展，计算机视觉任务，如目标检测和识别，得到了显著提升。YOLO 系列模型作为其中的佼佼者，以其高速和高效的特点被广泛应用于各种实时场景。这些应用包括但不限于自动驾驶、监控安防、无人机巡查等。因此，掌握 YOLO 模型的使用具有重要的实际意义。

##### 1.2实验环境描述

硬件环境

CPU: Intel Core i7 9700K

GPU: NVIDIA GTX 1060 6GB

内存: 16 GB DDR4

软件环境

操作系统: Windows 10 64-bit

Python 版本: 3.9

开发IDE: PyCharm Community Edition

依赖库:

OpenCV: 用于视频处理的图像库

PyTorch: 深度学习框架，用于加载和推理 YOLOv5 模型

### 2.系统设计和算法描述

##### 系统设计

**图形用户界面（GUI）**：基于 PyQt5，主要用于与用户交互，包括显示检测结果、控制模型运行、设置检测参数等。

 **目标检测线程（DetThread）**：负责进行目标检测，读取输入（视频或摄像头流），处理图像并输出检测结果。

 **模型管理与推理（YOLOv5）**：使用 YOLOv5 模型进行目标检测，包含模型加载、推理、非最大抑制（NMS）等。

 **配置管理**：通过 JSON 文件读取和保存用户设置（例如阈值、帧率、是否保存结果等）。

**主要模块解析**

**2.1 DetThread 线程**

DetThread 是一个继承自 QThread 的线程类，负责在后台进行视频流或图像的目标检测。该线程包括如下核心功能：

* **模型加载**：使用 YOLOv5 模型进行目标检测，模型权重文件通过 self.weights 属性传递。模型权重可以通过用户界面选择更新。
* **推理过程**：对输入的视频帧或图像进行推理，生成预测结果（包括目标边界框、置信度等）。
* **非最大抑制（NMS）**：对检测结果进行 NMS 处理，去除重复检测框，仅保留最佳检测框。
* **图像处理与显示**：对检测框进行绘制，并将结果通过 send\_img 信号传递到 UI 界面显示。
* **进度更新与统计**：通过信号将检测进度、FPS 和每类目标的统计数据传递到 UI 界面。
* **支持暂停与停止**：通过 is\_continue 和 jump\_out 控制检测的暂停与停止。

pred = model(img, augment=augment)[0]

pred = non\_max\_suppression(pred, self.conf\_thres, self.iou\_thres, classes, agnostic\_nms, max\_det=max\_det)

**2.2 YOLOv5 模型推理**

YOLOv5 模型是通过 attempt\_load 函数加载的，并且在推理时会进行一些预处理和后处理步骤：

* **预处理**：将输入图像转换为 PyTorch 张量，并进行归一化（0-255 -> 0-1）。
* **推理**：调用 YOLOv5 模型进行推理，输出包含目标位置、类别和置信度的预测。
* **后处理**：应用非最大抑制（NMS）去除重复的检测框。

img = torch.from\_numpy(img).to(device)

img /= 255.0 # 归一化

pred = model(img, augment=augment)[0] # 推理

pred = non\_max\_suppression(pred, self.conf\_thres, self.iou\_thres) # 应用NMS

**2.3 GUI 主窗口（MainWindow）**

MainWindow 负责用户界面的显示与交互：

* **窗口管理**：支持窗口的最大化、最小化、拖动和大小调整。
* **输入选择**：支持从文件选择视频、摄像头流或 RTSP 流作为输入源。
* **检测控制**：支持开始、暂停、停止检测，选择不同的模型，设置阈值、帧率等参数。
* **显示结果**：通过 QLabel 显示原始图像与检测结果，更新统计数据和进度条。

self.det\_thread.send\_img.connect(lambda x: self.show\_image(x, self.out\_video))

self.det\_thread.send\_statistic.connect(self.show\_statistic)

self.det\_thread.send\_percent.connect(lambda x: self.progressBar.setValue(x))

**2.4 配置管理**

配置管理通过 config/setting.json 文件保存和加载用户的设置，包括检测阈值（iou、conf）、帧率等。这些设置在启动时加载，并且可以在用户更改时实时保存。

config\_file = 'config/setting.json'

config = json.load(open(config\_file, 'r', encoding='utf-8'))

self.confSpinBox.setValue(config['conf'])

##### 关键算法流程

**模型加载与推理流程**

1. **加载模型**：从指定路径加载 YOLOv5 模型文件。
2. **图像预处理**：将图像从 OpenCV 格式转换为 PyTorch 张量，并进行归一化处理。
3. **模型推理**：对图像进行目标检测，输出预测结果（框、类别、置信度）。
4. **非最大抑制（NMS）**：对多个重叠的检测框进行过滤，仅保留最优框。
5. **结果处理**：根据阈值过滤掉低置信度的框，并将检测框绘制到图像上。

img = torch.from\_numpy(img).to(device) # 转换为张量并送入设备

img = img.half() if half else img.float() # 转为FP16（如果支持）

pred = model(img, augment=augment)[0] # 推理

pred = non\_max\_suppression(pred, self.conf\_thres, self.iou\_thres) # NMS

**进度和统计更新**

* 在每一帧检测完成后，通过信号将检测进度、FPS 和统计数据发送到 UI。
* send\_percent 用于更新进度条，send\_fps 用于显示 FPS，send\_statistic 用于显示每类目标的统计信息。

self.send\_percent.emit(percent)

self.send\_fps.emit(fps)

self.send\_statistic.emit(statistic\_dic)

**控制与交互**

* 用户可以通过按钮控制开始、暂停或停止检测。
* 可以通过配置框选择模型、设置阈值、帧率等。
* 通过文件选择器加载视频或图像文件。

self.runButton.clicked.connect(self.run\_or\_continue)

self.stopButton.clicked.connect(self.stop)

### 3.实验步骤和结果展示

##### 实验步骤

**1. 环境配置**

* **安装依赖**：首先需要确保你的开发环境安装了以下依赖：
  + Python 3.x
  + PyQt5：用于图形界面开发。
  + PyTorch：用于深度学习模型推理。
  + OpenCV：用于图像和视频处理。
  + YOLOv5：物体检测模型库。
  + 其他依赖：numpy, json, os 等。

可以使用以下命令安装依赖：

pip install pyqt5 opencv-python torch torchvision yolov5

* **准备权重文件**：下载YOLOv5的预训练权重文件（如 yolov5s.pt）并放置在 ./pt/ 文件夹下。

**2. 代码初始化**

* **启动 GUI 应用**：运行主程序 (MainWindow 类) 来启动应用程序：

python main.py

* **界面加载**：程序启动时，会显示一个窗口，界面包括：
  + 文件选择按钮，用于加载视频或图像。
  + 摄像头选择按钮，允许用户选择通过摄像头进行视频输入。
  + RTSP 流选择按钮，用于加载网络视频流。
  + 模型选择下拉框，用户可以选择不同的 YOLOv5 模型进行物体检测。
  + 控制按钮，如“开始检测”、“暂停检测”和“停止检测”。

**3. 加载和选择模型**

* 在界面中选择合适的 YOLOv5 模型。系统会自动列出 ./pt/ 目录下的所有 .pt 文件（YOLOv5 模型权重文件）。
* 选择一个模型后，系统会更新当前加载的模型权重路径，并准备进行物体检测。

**4. 选择输入源**

* **文件输入**：点击“打开文件”按钮，选择一个本地的图像或视频文件进行检测。支持的格式包括 .jpg, .png, .mp4, .avi 等。
* **摄像头输入**：点击“选择摄像头”按钮，通过连接的摄像头获取实时视频流进行物体检测。
* **RTSP 流输入**：点击“选择 RTSP 流”按钮，选择一个 RTSP 网络视频流作为输入源。

**5. 设置参数**

* **置信度和IoU阈值**：通过界面中的滑块或输入框设置 YOLOv5 的检测置信度阈值 (conf\_thres) 和 IOU 阈值 (iou\_thres)。这些参数控制物体检测的灵敏度和准确性。
* **推理帧率**：设置推理帧率 (rate) 来调整每秒处理的图像帧数，影响性能和实时性。

**6. 启动检测**

* **运行检测**：点击“开始检测”按钮，系统将启动一个后台线程 (DetThread)，开始处理输入源中的图像或视频。
  + 每一帧图像会通过 YOLOv5 模型进行推理，检测出物体的位置、类别及置信度。
  + 检测结果（例如物体的边界框、类别标签、置信度）将实时显示在界面上。
  + 检测过程中会显示实时帧率（FPS）和进度条。
* **暂停和继续**：点击“暂停”按钮可以暂停检测，点击“继续”按钮则可以恢复检测。

**7. 查看和保存结果**

* **实时显示**：检测过程中，每一帧的处理结果都会显示在图像框中。用户可以看到检测到的物体及其边界框。
* **统计信息**：显示每种物体被检测到的次数，统计信息会在界面中的一个列表框中更新。
* **保存结果**：点击“保存检测结果”复选框，系统会将每一帧的检测结果保存到指定文件夹。支持保存图像或视频文件。

**8. 停止检测**

* 点击“停止检测”按钮，可以停止检测进程，并释放资源。进度条会回到初始状态。

**9. 关闭程序**

* 在程序关闭时，应用会保存当前的配置（如置信度、IoU 阈值等）到配置文件中，以便下次启动时能够恢复之前的设置。

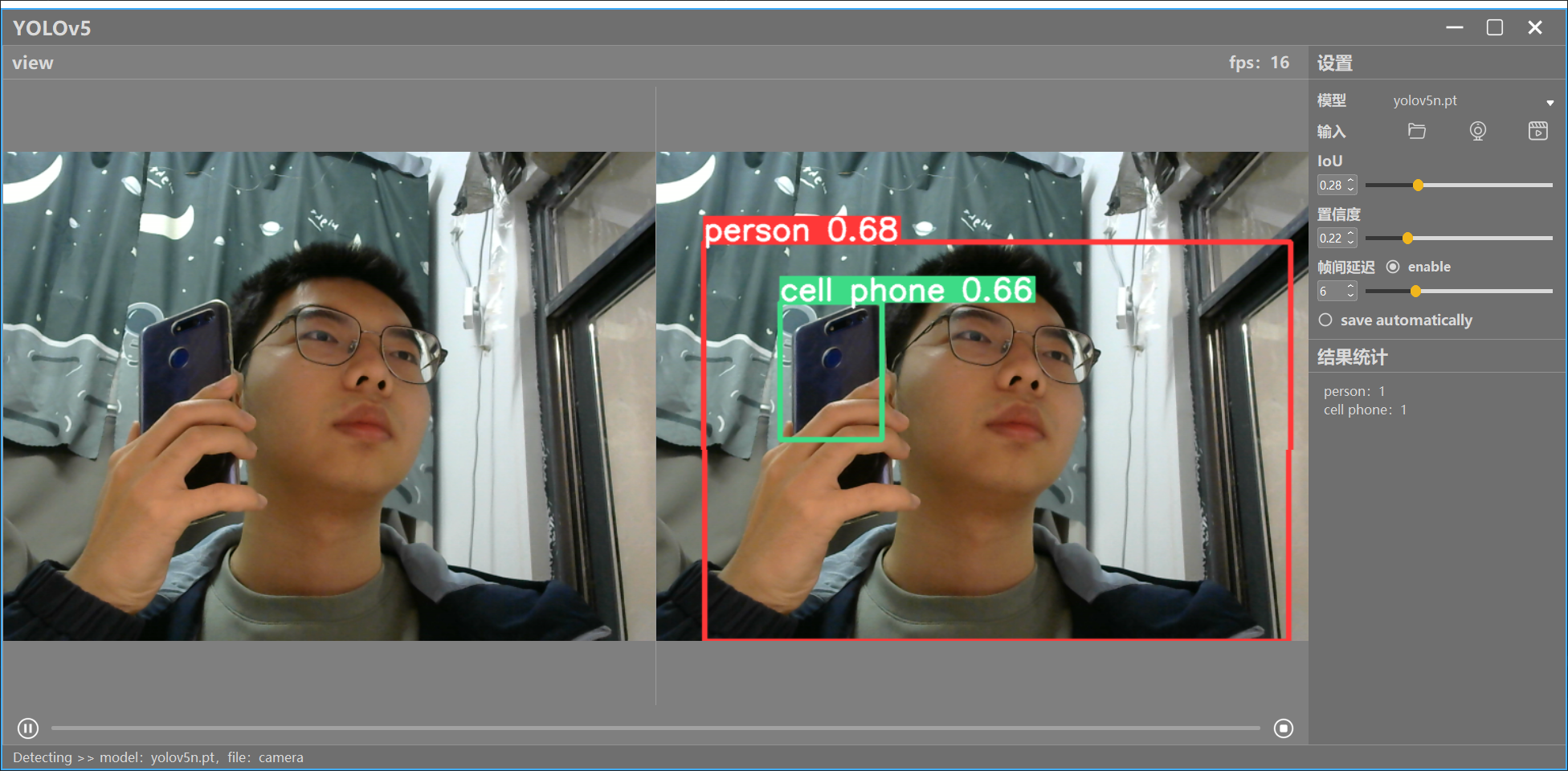
**代码细节**

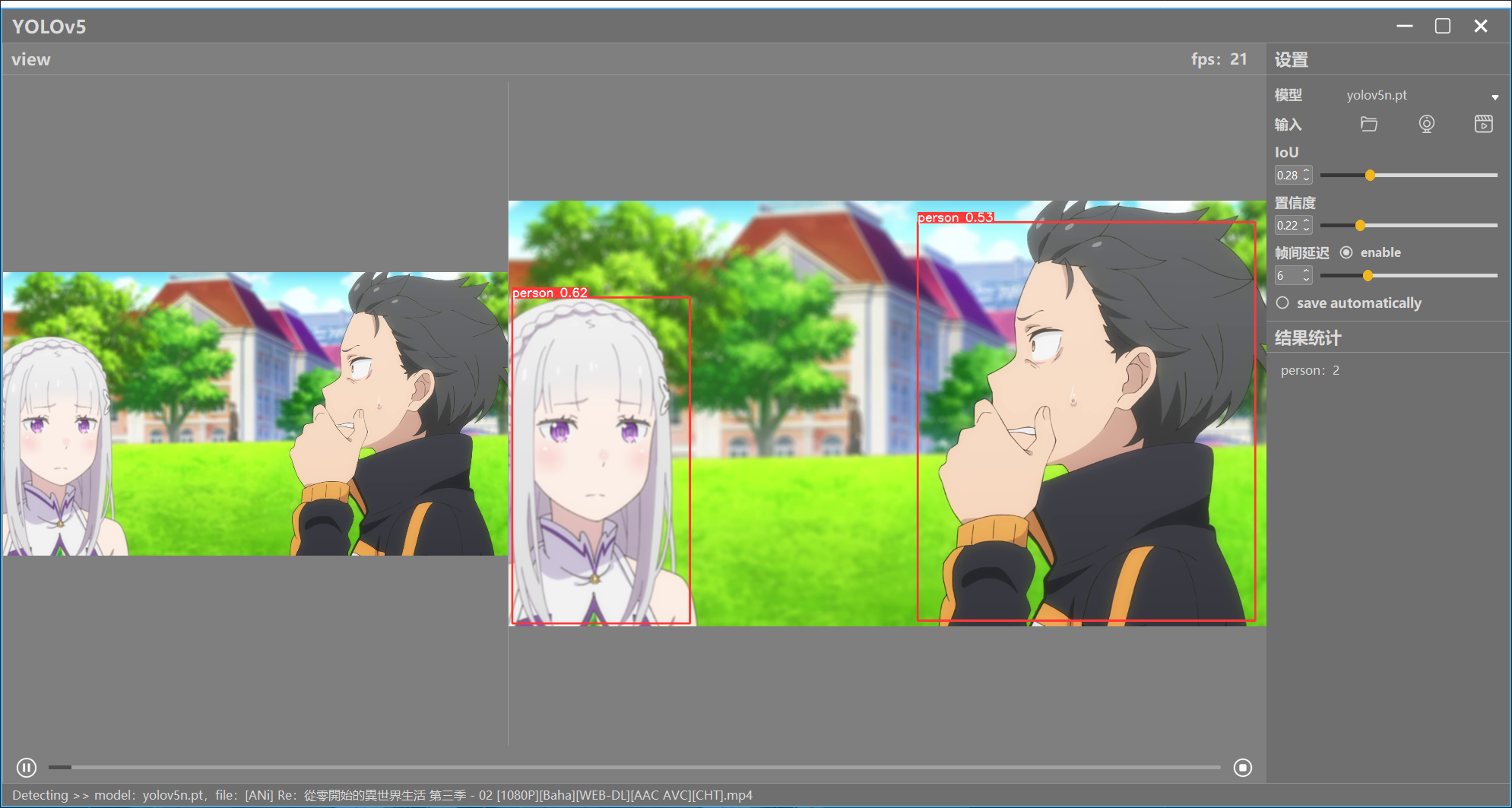
1. **YOLOv5模型推理**：
   * 在 DetThread 类的 run 方法中，加载指定的 YOLOv5 模型，并使用 OpenCV 或摄像头读取图像或视频流。
   * 每一帧图像会被预处理并输入到 YOLOv5 模型进行推理。
   * 使用非极大值抑制（NMS）算法来去除重叠的检测框，并返回最终的检测结果。
2. **图像处理**：
   * 检测框和标签会绘制在图像上，并显示在 GUI 界面中。检测框的颜色和标签内容会根据设置的显示选项进行调整。
3. **多线程**：
   * 图像处理和推理任务在 DetThread 类中运行于单独的线程，以确保界面响应流畅，不会因为长时间的推理任务而卡顿。

**预期结果**

1. **物体检测**：系统能够成功地从视频或图像中检测出物体，并显示边界框、类别标签和置信度。
2. **统计信息**：统计每个物体类别出现的次数，并实时更新界面。
3. **性能**：根据设备性能和设置的推理帧率，系统能够实时进行检测，帧率通常在 10-30 FPS 之间。
4. **结果保存**：可以选择是否将检测结果保存为图像或视频文件。

##### 结果展示：





### 4.实验分析和讨论

**遇到的问题**

1. **检测延迟**：初次实现时，检测存在约 0.5 秒的延迟，影响系统的实时性。
2. **环境光照影响**：在光线较暗或强烈反光的情况下，检测精度有所下降。

**解决方案**

1. **优化代码性能**：
   * 调整视频流的分辨率，降低图片尺寸以加快处理速度。
   * 优化数据预处理及模型推理步骤，减少不必要的计算。
2. **改进模型参数**：
   * 调整信心阈值（如 conf\_thres=0.25）和 NMS 阈值（如 iou\_thres=0.45），以平衡检测精度与速度。

**改进方向**

* **使用更高性能的模型**： 尝试使用 YOLOv5 的大型版本（例如 yolov5m.pt 或 yolov5x.pt）以提高检测精度。
* **引入多线程**： 通过多线程处理视频流和推理过程，进一步降低延迟。
* **增强模型鲁棒性**： 考虑进行更多的数据增强操作，从而提升模型在不同光照和天气条件下的鲁棒性。

### 5.结论和展望

结论

本实验成功实现了利用 YOLO 模型对实时视频流中的人进行检测，验证了 YOLO 在实时应用中的有效性。通过合理调整参数和进行代码优化，我们实现了流畅的检测效果。

展望

未来的工作可以集中在以下几个方面：

1. 扩展应用领域：

- 将技术应用于无人机监控、智能交通等领域。

2. 提升系统智能化：

- 结合其他机器学习技术（如姿态估计）提升系统的综合能力。

3. 适应复杂场景：

- 研究如何在夜间、阴影等复杂场景中保持高效检测。

|  |  |
| --- | --- |
| **学生学习心得** | 通过这次YOLOv5物体检测实验，我深入了解了物体检测算法的原理及其实现过程。学习了如何使用PyTorch加载和推理模型，掌握了图像预处理、实时视频流处理以及结果显示和保存的技术。实验还让我了解了GUI设计与多线程应用，提高了我的编程和问题解决能力。通过调参优化，我体会到了精度与速度之间的平衡，提升了我在计算机视觉领域的实践能力。  学生（签名）：此处必须手写签字  年 月 日 |
| **诚信承诺** | 本人郑重声明所呈交的课程报告是本人在指导教师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注的地方外，论文中不包含其他人已经发表或撰写过的研究成果。与我一同工作的同学对本文研究所做的贡献均已在报告中作了明确的说明并表示谢意。  学生（签名）：此处必须手写签字 |
| **任课**  **教师**  **评语** | 成绩评定：  任课教师（签名）：  年 月 日 |