# 基于 YOLO v8 的智能垃圾分类回收系统

## 摘要

传统的垃圾分类回收主要依赖人工操作，容易受到人力资源不足、环境因素影响及时间成本限制，导致识别效率低下、误分类与漏分类现象频发，进一步引发可回收物资源浪费和环境二次污染。为此，项目提出了基于 YOLO v8 模型的智能垃圾分类回收系统。系统通过高分辨率摄像头实时采集垃圾图像，并将图像发送至 ELF2 开发板，完成图像预处理和推理计算；通过部署轻量化 YOLO v8 检测模型，并结合高性能舵机实现不同垃圾桶的自动化分类，从而有效提升垃圾分类的效率与准确率，实现了资源的高效回收与利用。同时，本系统能够灵活应对实际应用中的多种情况，无论是识别单一垃圾物品时的迅速响应，还是面对多种垃圾混杂投放时的同步分类和分拣，亦或在强烈日照或复杂光照环境下依然保持高水平的识别能力，均能保障垃圾分类工作的顺利开展，满足不同场景下的智能回收需求。

**关键字：**垃圾分类回收系统、YOLO v8、ELF2 开发板、自动化分类

# 第一部分　作品概述

## 1.1 功能与特性

应用程序界面：为了提升用户的交互体验，将 PyQt5 应用程序部署到了高性能的计算板卡 RK3588 上。这款板卡的强大计算能力与 PyQt5 的丰富功能相结合，为用户提供了一个直观、友好且功能全面的主要交互界面。通过这个界面，用户可以轻松地进行各种操作，实现高效的工作流程，如图 1 所示。

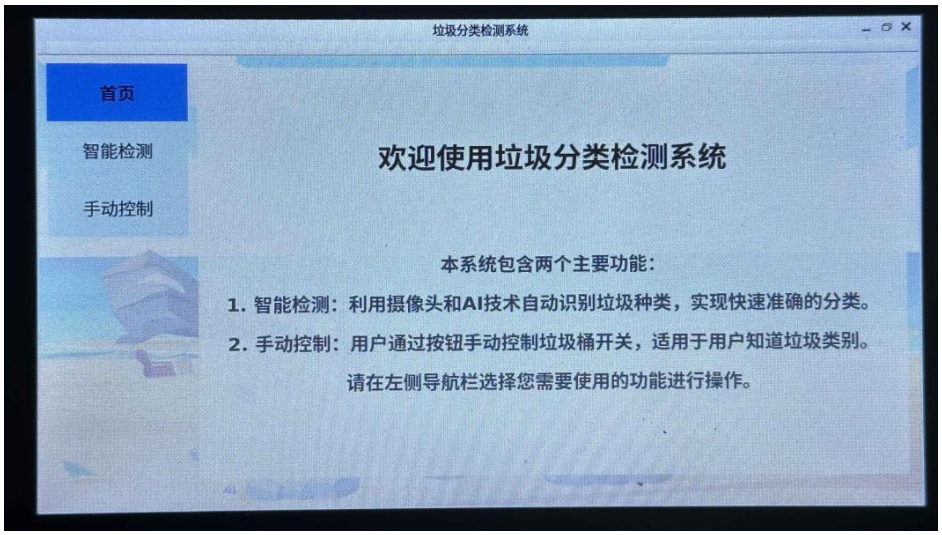


图 1 垃圾分类监测系统用户交互界面

图像实时识别功能：计算板卡 RK3588 可实时、稳定地接收摄像头传输的高清图像[2]。在该板卡上部署改进的 YOLO v8 检测模型，引入先进的轻量化网络结构与动态特征融合算法，大幅提升了模型对不同类型垃圾的识别速度与精准度，如图 2 所示。


图 2 图像实时识别

智能检测模式：

1）单垃圾场景（正常光照环境）：系统在用户投放单个垃圾物品时，能够迅速捕捉并识别该物体的类别。依托于优化后的 YOLO v8 模型，系统在毫秒级内快速响应，判断垃圾类型，并且第一时间自动开启对应的垃圾桶盖，整个过程实现智能联动，无需用户手动操作，大幅提升了投放的便捷性与准确率，如图 3 所示。多垃圾场景（正常光照环境）：当用户一次性识别多种不同类别的垃圾时，系统能够对每一个目标物体进行独立识别和分类。模型具备同时检测多个目标的能力，并能根据识别结果自动联动多个垃圾桶同步开启，实现分门别类的精准投放。这一功能极大地提高了多垃圾混投环境下的分类效率与自动化水平，减少人工干预和误投现象，如图 4 所示。


图 3 单垃圾识别（正常光照环境）


图 4 多垃圾识别（正常光照环境）

2）单垃圾场景（复杂光照环境）：面对阳光直射、强光反射或室内外光线变化等复杂光照条件，系统依然能够稳定接收高清画面，并通过算法优化抵抗光照干扰，准确判断垃圾类别，确保分类结果依旧可靠，如图 5 所示。多垃圾场景（复杂光照环境）：将多种垃圾同时放置在摄像头前，系统能够有效分辨各类目标，即使光线变化剧烈，也不会影响模型的识别能力和分类准确率，如图 6 所示。整个过程中，无论是单一目标还是多个目标，系统都能稳定输出高质量的识别结果，适用于恶劣环境的实际应用需求。


图 5 单垃圾识别（复杂光照环境）


图 6 多垃圾识别（复杂光照环境）

手动控制模式：当用户知道垃圾属于哪一类时，只需要在界面上选择相应的按钮，就可以轻松控制垃圾桶的开合。这个模式让投放过程更加直接方便，不管是投放单个垃圾还是连续投放多个同类垃圾，都能一键完成操作，如图 7 所示。

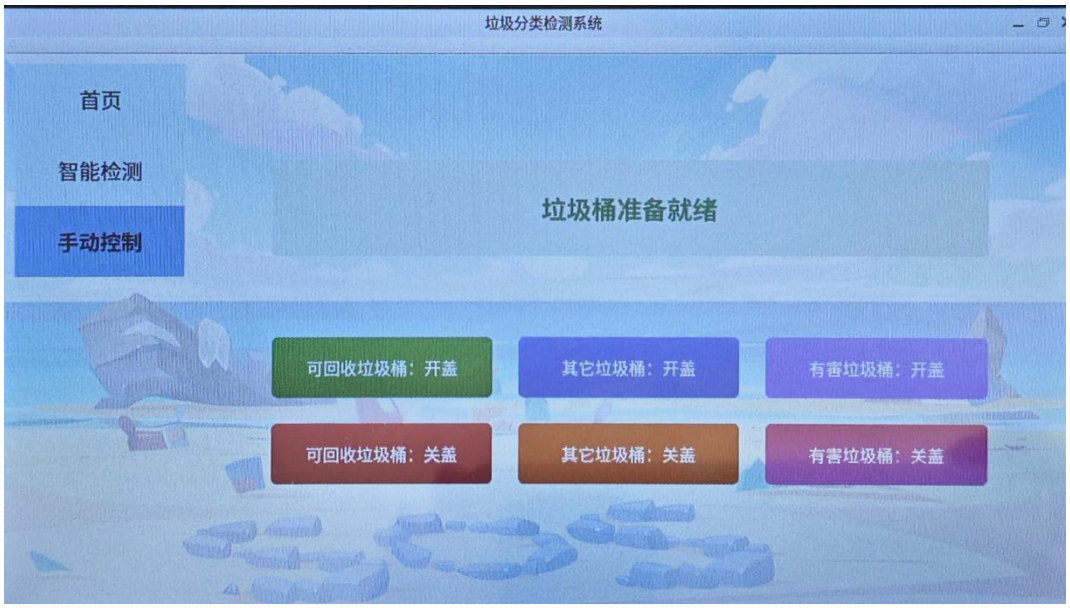


图 7 手动控制模式

## 1.2 主要技术特点

1）ELF2 开发板：具备高算力与低功耗优势，能够高效支持本地 AI 推理，保障垃圾分类系统的实时响应。其硬件接口丰富，方便接入摄像头和舵机等外设，系统集成灵活。稳定的运行性能适用于复杂多变的应用环境，为智能垃圾分类提供坚实的硬件基础。

2）改进的 YOLO v8 模型：空间金字塔池化的升级和 MPDIoU 损失函数的策略，使得模型在小目标检测、遮挡目标处理和复杂场景应用中展现出更高的准确率和更强的适应性。

3）PyQt5 应用程序：在计算板卡 RK3588 上部署应用程序[1]，该程序能够实时显示摄像头获取的画面，自动将识别到的垃圾类别直观地展示在界面上。同时，系统还具备毫秒级的识别速度，能够在极短时间内完成目标识别并自动控制打开对应的垃圾桶盖，大幅提升了智能垃圾分类的便捷性。

## 1.3 主要性能指标

本系统的性能评估采用了目标检测领域常用的多项关键指标，全面反映模型在训练和验证过程中的表现。主要包括损失类指标（如框损失、类别损失和分布式焦点损失），用于衡量模型在目标定位和分类方面的误差；以及精确率、召回率和平均精度（mAP）等评价指标，主要用于衡量模型在垃圾识别任务中的准确

性、召回能力和整体检测效果。这些指标共同构成了对模型检测能力、分类能力和泛化能力的综合评估体系。

表 1 性能指标评估表

| 指标名称 | 功能说明 |
| --- | --- |
| 训练集框损失（train/box_loss） | 衡量模型在训练集上预测边界框与真实边界框的误差 |
| 训练集类别损失（train/cls_loss） | 衡量模型在训练集上对目标类别预测的误差 |
| 训练集 DFL 损失（train/dfl_loss） | 优化边界框预测精度的辅助损失 |
| 精确率（precision） | 表示预测为正样本中真正例的比例，反映误报情况 |
| 召回率（recall） | 表示实际正样本中被正确检出的比例，反映漏检情况 |
| 验证集框损失（val/box_loss） | 衡量模型在验证集上边界框预测的误差 |
| 验证集类别损失（val/cls_loss） | 衡量模型在验证集上类别预测的误差 |
| 验证集 DFL 损失（val/dfl_loss） | 验证集上的边界框回归辅助损失 |
| 验证集 mAP0.5 | 不同类别目标在 MPDIoU 阈值 0.5 下的平均精度均值 |
| 验证集 mAP0.5-0.95 | 多个 MPDIoU 阈值下的平均精度均值，反映综合检测效果 |

### 1.4 主要创新点

本作品的主要创新是在 YOLO v8 模型的基础上进行了针对性优化，显著提升了垃圾分类任务中的识别精度。通过深入分析垃圾分类场景的多样化需求与特性，我们对模型进行了改进，增强了其特征提取与目标检测能力。这些优化不仅大幅提升了模型在复杂环境下的鲁棒性和适应性，还使其能够更高效地识别各类垃圾对象。

在实际应用中，模型可对单个垃圾物品进行快速准确识别，确保高效投放；面对多个垃圾物品同时出现的情况，能够精准区分不同类别，并联动多个对应垃圾桶，实现自动化分拣；即使在强光或阳光直射等复杂光照环境下，模型依然保持出色的识别准确率与稳定性。

此外，系统支持手动与智能两种垃圾桶控制方式。手动模式下，用户可自行点击按钮实现垃圾桶的开启与关闭，便于个性化操作；智能模式下，系统通过摄像头智能识别垃圾类型，自动控制对应垃圾桶的开启与关闭，极大提升了智能化和便捷性。整体性能的提升为智能垃圾分类系统提供了更加可靠的技术支撑，也为推动环保技术的落地应用与持续发展注入了新动力。

### 1.5 代码地址

https://github.com/wu20249/garbage-sorting_001

# 第二部分　系统组成及功能说明

## 2.1 整体介绍

系统整体框架如图 8 所示：

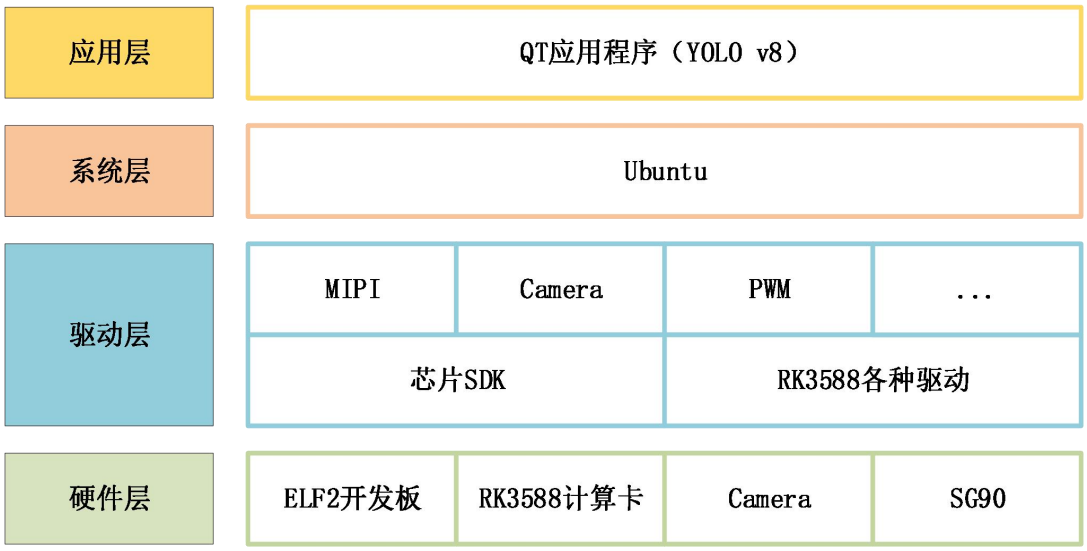


图 8 系统整体框图

基于上述系统整体框图，ELF2 开发板与 RK3588 计算卡共同构成了智能垃圾分类回收系统的核心硬件平台。其中，RK3588 计算板卡搭载了功能强大的 Ubuntu 操作系统，并集成了丰富的硬件资源和接口。系统在 RK3588 上部署了基于 PyQt5 开发的应用程序，应用层通过 QT 界面与用户交互，并负责整体流程的调度。通过将摄像头与 ELF2 开发板连接，系统能够实时采集垃圾图像，并将采集到的数据输入到 YOLO v8 深度学习模型中进行垃圾分类识别，完成智能分拣。

在系统分层架构中，硬件层包括 ELF2 开发板、RK3588 计算卡、摄像头以及舵机，保障系统具备充足的算力和数据采集能力。驱动层通过芯片 SDK 和各类驱动实现对 MIPI、Camera 和 PWM 等外设的控制和管理，为系统层提供底层硬件支持。系统层以 Ubuntu 为核心操作系统，负责统一资源调度与多任务管理，提供稳定可靠的运行环境。应用层则以 QT 应用程序为核心，集成 YOLO v8 目标检测模型，实现对垃圾的自动化、智能化分类识别，并通过在板载 MIPI 上进行显示界面，向用户展示分类结果和系统状态。

整个系统分层清晰、结构合理，各层之间通过标准化接口和协议进行高效协作，有效提升了垃圾分类识别的自动化、智能化水平，也为后续系统扩展、功能升级和维护提供了坚实的架构基础。

## 2.2 硬件系统介绍

### 2.2.1 硬件整体介绍

硬件整体由 ELF2 开发板、MIPI 液晶屏、摄像头和舵机构成。ELF2 开发板作为系统的核心控制单元，负责数据处理和任务调度；液晶屏是系统交互与展示的核心部件，通过高速低功耗的 MIPI 协议与 ELF2 开发板互联，可清晰显示垃圾图像并支持触控交互；摄像头用于实时采集垃圾图像，为分类识别提供数据支持；舵机则用于驱动垃圾桶盖的开合，实现垃圾的自动分拣与投放。这些硬件模块协同工作，为系统的智能化运行提供了坚实的基础。

### 2.2.2 硬件模块介绍

1）ELF2 开发板：飞凌官方开发板，基于瑞芯微 RK3588 处理器设计，如图 9 所示。

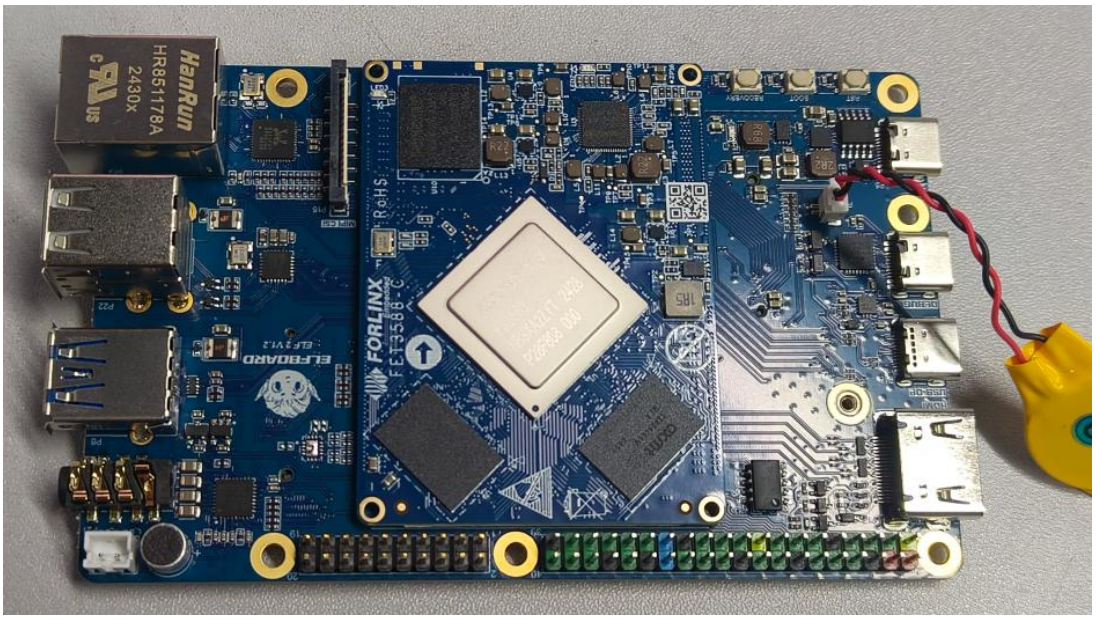


图 9 ELF2 开发板

2）液晶屏：MIPI 液晶屏，通过开发板上的 MIPI-DSI 接口实现，如图 10 所示。

图 10 MIPI 液晶屏

3）摄像头：2K 摄像头，通过 USB 与开发板连接，如图 11 所示。

图 11 USB 摄像头

4）舵机：SG90 舵机，通过 PWM 信号线与开发板相连，如图 12 所示。

图 12 SG90 舵机

## 2.3 软件系统介绍

### 2.3.1 软件整体介绍

软件整体由部署在 RK3588 上的 QT 应用程序构成，集成了用户界面、图像采集、目标检测和舵机控制等功能，实现了垃圾分类的全流程智能管理与交互[4][5]。

### 2.3.2 软件模块介绍

#### 2.3.2.1 QT 应用程序

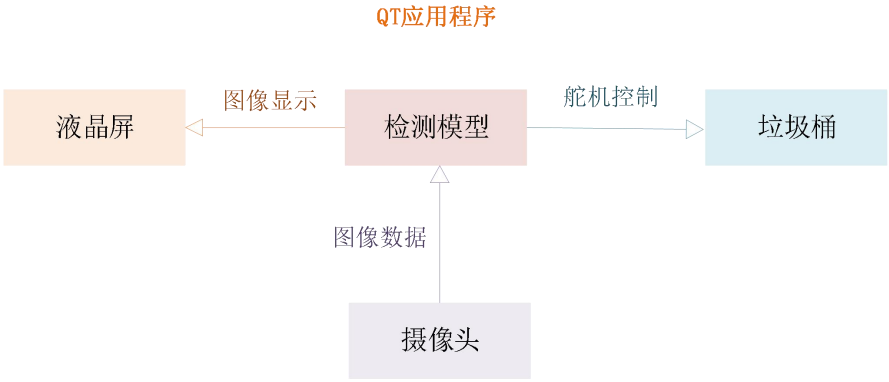


图 13 QT 应用程序框图


图 14 接收图像数据检测模型

系统将 QT 应用程序部署在 RK3588 平台上，作为与用户进行主要交互的界面。检测模型实时接收来自摄像头的图像数据，并将其输入到改进后的 YOLO v8 模型进行垃圾分类识别。识别结果不仅会即时显示在界面上，便于用户查看和操作，同时系统还会根据分类结果自动控制舵机，精准开启对应垃圾桶的桶盖，实现垃圾的智能分类投放。

### 2.3.2.2 YOLO v8 模型改进

#### 1、模型改进

基于 YOLO v8 的改进模型[6]-[12]整体分为 Backbone、Neck 和 Head 三部分。在主干部分，模型依次堆叠了多层卷积（Conv）和 C2f 模块，逐步提取输入图像的深层特征。与原始 YOLO v8 不同，本模型在主干末端引入了 SimSPPF（Simplified Spatial Pyramid Pooling-Fast）模块。SimSPPF 通过更简洁高效的结构在多尺度上聚合空间信息，提升了特征表达能力，并优化了推理速度。Neck 部分采用多层特征融合策略，通过 Concat、UpSample、C2f 等操作实现不同尺度特征的有效整合。Head 部分由多组卷积层组成，分别对应不同尺度的特征输出，实现对各类目标的高精度检测。整体来看，引入 SimSPPF 模块后，模型不仅提升了复杂场景下的检测精度，还兼顾了轻量化和推理效率，适用于多种实际应用场景。新改进的模型结构图如下所示。

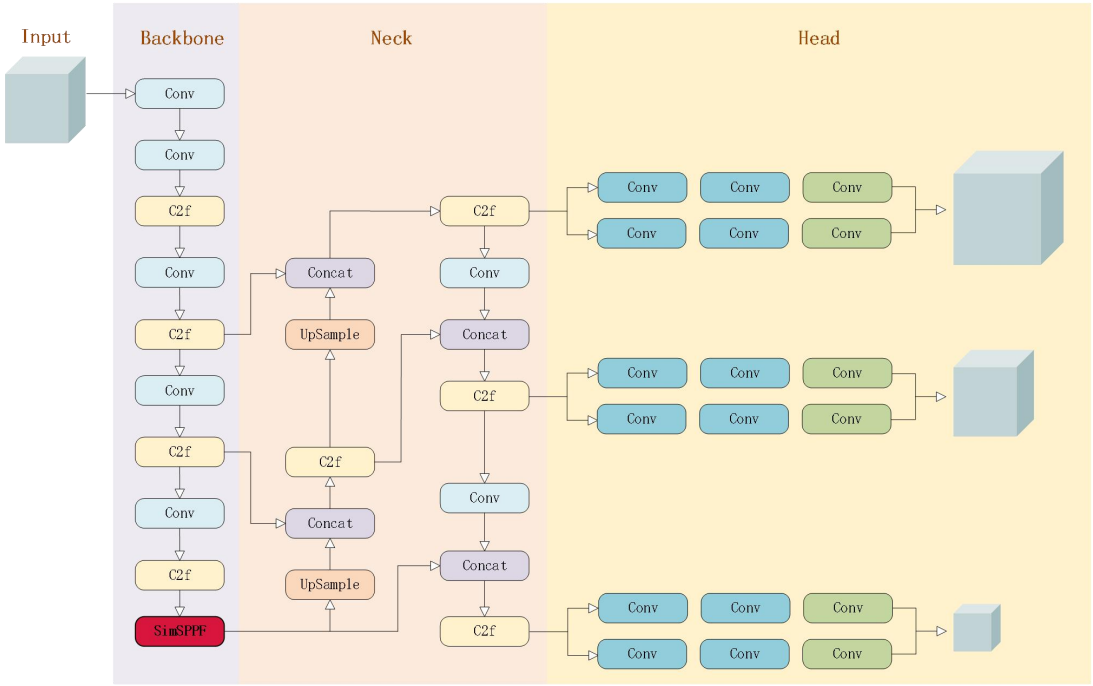


图 15 改进的 YOLO v8 框架图

对于模型的改进主要为以下几点：

1）SimSPPF(分层空间金字塔池化)模块[3][7]：实验发现，在融合多尺度特征时，过度依赖池化操作，容易导致空间细节信息丢失，使得特征图变得粗糙，降低了小目标和细粒度目标的检测准确率。同时，其池化与特征拼接过程频繁分配和管理中间节点，尤其在高分辨率输入下大幅增加了内存和计算资源消耗，降低了模型在嵌入式等场景下的运行效率。此外，池化与卷积混合使用导致结构复杂、存在冗余和无效计算，影响整体卷积效率和推理性能。由于结构层次深、路径长，

还使得梯度在反向传播过程中容易出现消失或爆炸，影响模型训练的收敛速度和稳定性，增加了训练的难度。针对该问题，我们提出了分层空间金字塔池化模块，通过分层节点池复用、结构简化和优化特征传递路径，不仅减少了内存和计算资源的消耗，还提升了特征细节的保留能力和卷积效率，同时改善了梯度传播的稳定性。这些改进使得模型在各种硬件环境下都能保持较高的检测精度和推理效率。SimSPPF 模块结构图如下所示。

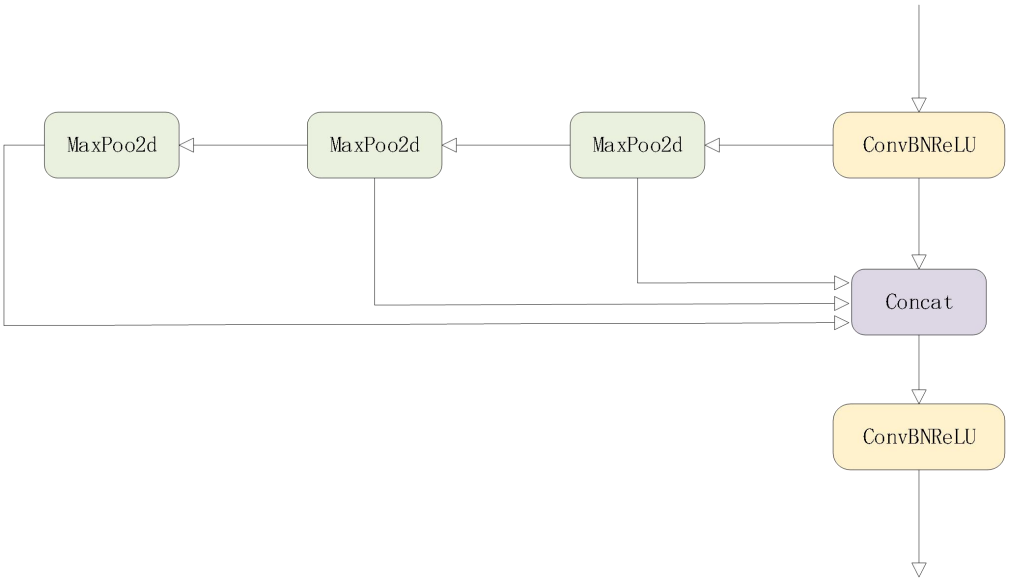


图 16 SimSPPF 框架图

2）引入了 MPDIoU 损失函数[13][14]：MPDIoU 是一种基于最小点距离的新型边界框相似度比较度量标准，直接最小化预测边界框与实际标注边界框之间的左上角和右下角点距离。MPDloU 包含了现有损失函数中考虑的所有相关因素，即重叠或非重叠区域、中心点距离、宽高偏差，同时简化计算过程。

**2、参数设置**

本项目使用了 4 张 12G NVIDIA GeForce RTX 3060 显卡，同时为确保 YOLO v8 模型在垃圾分类检测任务中取得优异的性能，针对具体任务设定合理的训练参数：

1）批次大小（batch size）：根据硬件显存大小设置适当的批次大小，本项目设置为 128，较大的批次大小有助于模型的稳定训练。

2）初始学习率（learning rate）：选择合适的学习率（如 0.01），并采用 Cosine Annealing 策略，使学习率逐渐下降，防止模型在训练后期发生震荡。

3）训练轮数（epochs）：设置 100 轮训练，使模型充分学习不同垃圾类别的特征。

**3、训练过程**

1）预训练模型加载：为了提升模型的收敛速度和性能，加载 yolov8n.pt 的预训练权重，这些权重基于大型数据集训练，具备较强的特征提取能力。

2）模型验证：在每轮训练后，使用验证集评估模型性能，通过计算准确率、平均精度（mAP）等指标监测模型的训练进展。验证结果用于调整模型的学习率和其他超参数，保证训练的稳定性和高效性。

**2.3.2.3 数据处理**

**1、数据来源**

本系统所使用的数据集由多种来源构成，主要包括以下方面：

（1）公共数据集

垃圾分类数据集：AI Studio 包含不同角度、光照条件和背景环境下的实物图像，例如摆放杂乱的可回收纸箱、沾有油渍的厨余果皮等复杂场景，最大程度模拟真实生活环境。数据集支持图像分类、目标检测等多类深度学习任务，适配多种算法研究需求。

（2）自定义数据集

自采集数据：根据具体需求（如城市垃圾、家庭垃圾等）自定义数据集；

社区收集：在社区、学校或公共场所拍摄各种垃圾的照片，确保覆盖不同类型的垃圾。

**2、数据采集方法**

相机拍摄：使用高质量的数码相机或智能手机进行拍摄，选择清晰、明亮的环境，以减少光照干扰。捕捉垃圾的不同角度和细节，以便于后续标注。

环境设置；在特定环境下（如家庭厨房、街道、公园等）拍摄，确保样本多样性。对于不同种类的垃圾，尽量在自然环境中拍摄，以捕捉其在实际场景中的外观。

**3、数据处理**

在完成数据采集后，需对数据进行处理，以提高模型训练的效果。

1）数据清洗：

去重：检查数据集中是否存在重复图像，并去除重复样本，确保数据集的多样性。

质量检测：筛选出模糊、过曝或暗淡的图像，并进行修正或删除，保留高质量的样本。

2）数据增强：

为了增加数据集的多样性并提高模型的鲁棒性，采用数据增强技术，包括以

下方面：

旋转：随机旋转图像（0°、90°、180°、270°）。

缩放：随机调整图像大小，保持长宽比。

裁剪：随机裁剪图像，选取其中的一部分进行训练。

颜色调整：随机改变图像的亮度、对比度和饱和度。

翻转：水平或垂直翻转图像。

MixUp: 将两张图片进行混合，增加模型的鲁棒性。

CutMix: 随机剪切部分区域并将其与另一图像合并，增强特征学习。

3）数据集划分：

将处理后的数据集按比例划分为训练集、验证集和测试集，确保模型的训练和评估效果。

4）图像标准化：

将所有图像进行像素值归一化处理，以适应深度学习模型的输入要求。

**4、数据集展示**

该数据集由 48 个细分类别组成。训练集包括 37528 张图片，为模型参数优化提供丰富样本；测试集与验证集分别包含 4691 张和 9382 张图片。这种标准化的数据划分模式，既保证了模型训练阶段的充分学习空间，又通过独立的测试集与验证集为模型性能评估提供了科学的量化依据，有效规避了数据偏差对算法精度的影响，为后续分类任务的准确性奠定了坚实的数据基础。如图 17 所示。

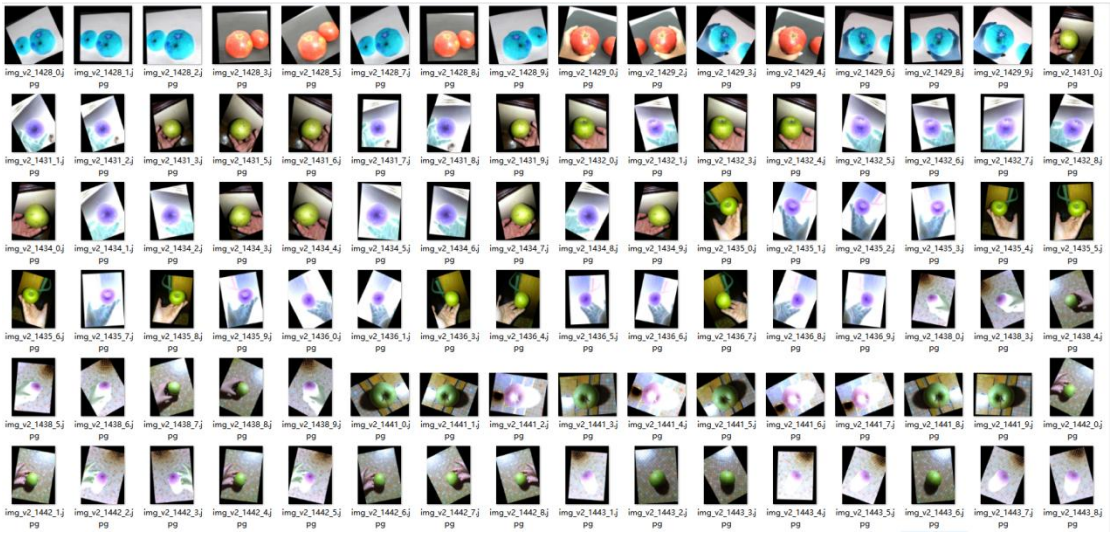

图 17 垃圾分类数据集（部分）

**2.3.2.4 检测模型计算板卡移植**

1）环境配置：本地电脑环境，安装版本为 ubuntu17.5.x 虚拟机、pycharm、python3.8、rknn-toolkit2_1.6.0、以及相应的 python 库函数。板卡环境，python3.8、

rknn-toolkit_lite2_1.6.0、以及相应的 python 库函数。

2）模型转换

将改进的 YOLO v8 模型从 PyTorch 权重（.pt）格式转换为 ONNX 格式，再进一步转换为 RKNN 格式，最后交叉编译生成可执行文件，并最终部署到计算板卡上进行推理的过程中，可以遵循以下流程图步骤：

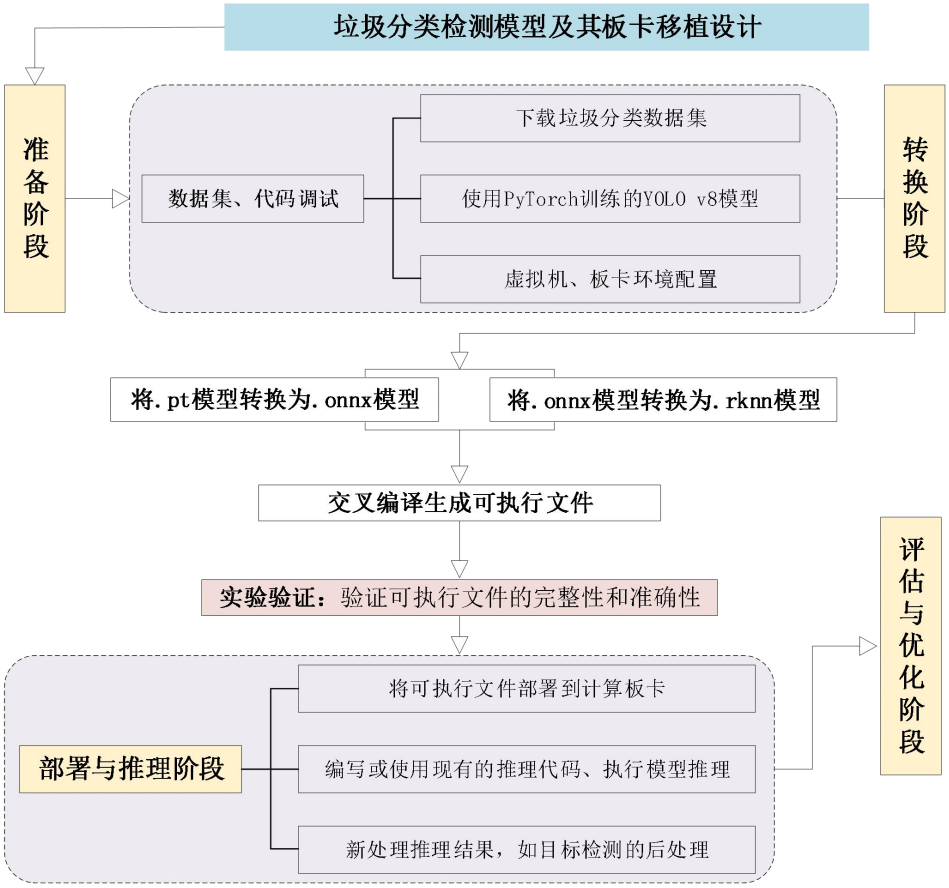


图 18 模型转换流程图

① 准备环境

确保安装了必要的软件包，例如 PyTorch、ONNX、ONNX Runtime、RKNN SDK 等，安装计算板卡所需的开发工具和驱动、及其所需的 python 库函数。

② PyTorch 权重（.pt）格式转换为 ONNX 格式

使用 ultralytics 工具将 PyTorch 模型转换为 ONNX 格式。首先需要在安装有 PyTorch 和 ONNX 库的 Python 环境中，加载训练好的改进 YOLO v8 模型的 PyTorch 权重文件。然后，设定模型的输入规范，如输入图像的尺寸、数据类型等。接着，通过 torch.onnx.export 函数，将 PyTorch 模型导出为 ONNX 格式。在导出过程中，要确保模型中使用的算子是 ONNX 支持的，避免出现转换失败的情况。

③ ONNX 格式转换为 RKNN 格式

该转换步骤在虚拟机上执行，使用瑞芯微官方提供的 RKNN-Toolkit 工具将 ONNX 模型转换为 RKNN 格式。利用该工具，首先初始化 RKNN 环境，配置目标平台为 RK3588，并设置模型的输入尺寸、量化方式等参数。接着，加载第一步生成的 ONNX 模型文件，通过构建操作，将 ONNX 模型转换为 RKNN 格式。量化是这一步的关键，通过 INT8 量化等方式，可在不损失过多精度的情况下，大幅降低模型体积，提升在 RK3588 板卡上的推理速度。

④ 交叉编译生成可执行文件

获得 RKNN 格式的模型后，最后一步是进行交叉编译。首先，基于 RKNN Runtime API，使用 C 或 C++ 语言编写推理应用程序，程序内容涵盖模型加载、图像预处理、推理调用以及后处理等功能。然后，配置 aarch64-linux-gnu 交叉编译工具链，指定编译器路径，并链接 RKNN Runtime 库和其他必要的依赖库。最后，通过编译操作，生成能够在 RK3588 开发板上运行的可执行文件，实现改进 YOLO v8 模型在实际场景中的部署与应用。

⑤ 部署到 RK3588 板卡

将交叉编译生成的可执行文件、RKNN 格式的模型文件，以及可能用到的配置文件和图像预处理脚本，通过 TF 卡方式传输到 RK3588 板卡。同时需要确保板卡的 RKNN-Toolkit 工具和必要的驱动已经正确安装和配置，进行加载模型并进行推理。

⑥ 测试和验证

通过 putty 软件远程连接至 RK3588 计算板卡后，进入可执行文件所在目录启动推理程序，运行时需精准配置模型文件路径、摄像头设备号、图像输入路径等关键参数；若执行出现异常，需依据报错信息快速定位问题，如检查模型加载状态与输入数据格式，并借助板卡日志输出功能记录关键环节信息辅助排障。完成程序运行后，在计算板卡上对模型开展全面测试，通过对比实际推理结果与标准数据验证准确性，并基于测试反馈灵活调整模型超参数、优化处理流程和改进算法结构，持续提升模型性能，确保其能高效精准适应不同分辨率、光照条件及复杂场景下的输入数据，充分发挥 RK3588 板卡的硬件性能。

⑦ 部署和应用

将经过验证的模型部署到该项目中，使用摄像头实现实时数据流的稳定采集，结合板卡强大的 NPU 算力与优化后的算法架构，系统可对输入画面进行毫秒级响应处理，精准识别并分类 48 类目标物体。

# 第三部分　完成情况及性能参数

## 3.1 整体介绍

最终实现的成果符合预期系统设计要求，所有预设功能均已实现。

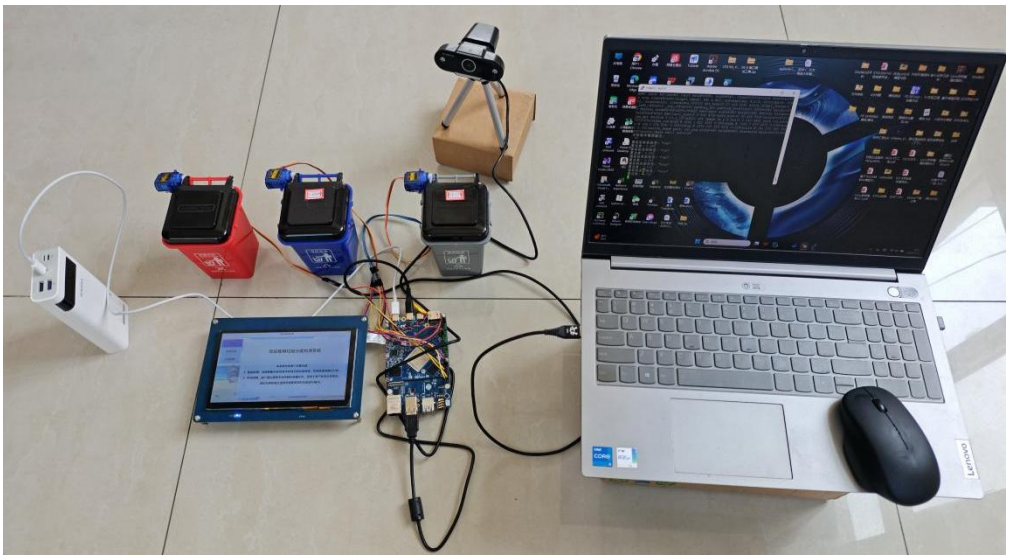

图 19 系统实物图

## 3.2 工程成果

### 3.2.1 硬件成果

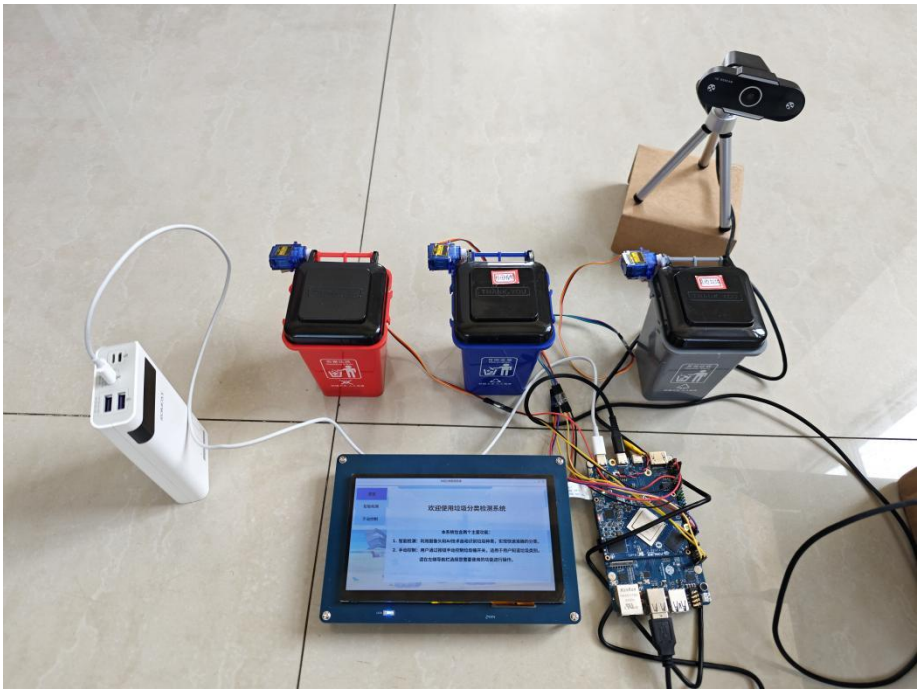

图 20 硬件实物图

### 3.2.2 软件成果

部署在 RK3588 的应用程序界面：系统集成了摄像头实时采集、图像拍摄、智能识别及分类结果展示等多项功能。首页：介绍垃圾识别的两种模式，用户可以根据自己的实际需求在左侧导航栏自行选择模式，如图 21 所示；智能检测界面：左侧为实时摄像头画面预览区，能够动态显示摄像头当前捕捉到的画面；右侧则会展示识别出的垃圾类别。此外，界面下方还会同步显示识别结果，让用户一目了然地知道垃圾属于哪一类，如图 22 所示。手动控制界面：适用于用户已经了解垃圾类别的情况，只需点击界面上对应的垃圾桶按钮，就可以实现垃圾桶的自动开关盖操作，如图 23 所示。

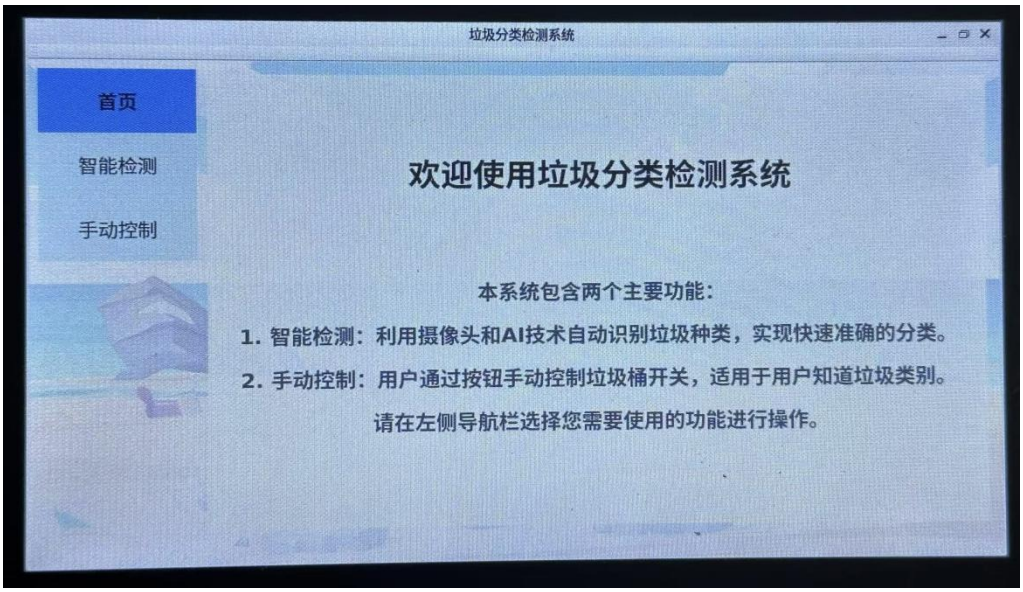


图 21 系统首页


图 22 智能检测界面

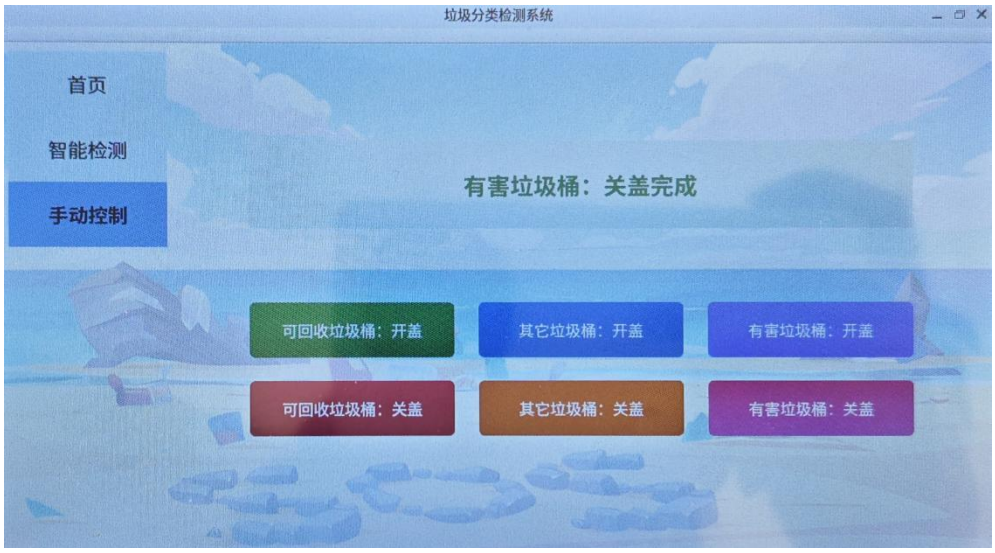


图 23 手动控制界面

### 3.3 特性成果

为验证所提出改进模型在垃圾分类任务中的有效性，本文设计了改进前后模型的对比实验，并选取了典型场景下的检测结果进行展示。

如图 24 所示，改进前后的模型在多个实际样本上的表现存在明显差异。第一列实验结果显示，改进后的模型在饮料罐和金属拉环的检测精度上有明显提升，能够更准确地区分并标注不同类别目标；第二列结果表明，经过改进后，模型的检测框更加精准，目标物体的边界识别更为准确，显著提升了定位效果；第三列结果中，改进前的模型将饮料盒误识别为过期药物，而改进后的模型能够正确识别饮料盒类别，有效降低了误检率；第四列实验结果表明，改进前的模型未能检测出毛巾类别，改进后模型则成功检测到毛巾，进一步验证了模型在小目标检测和复杂场景下的鲁棒性。

综上，改进后的模型在检测精度、目标定位、类别判别以及小目标检测等方面均优于原始模型，能够更好地适应实际垃圾分类场景的需求。

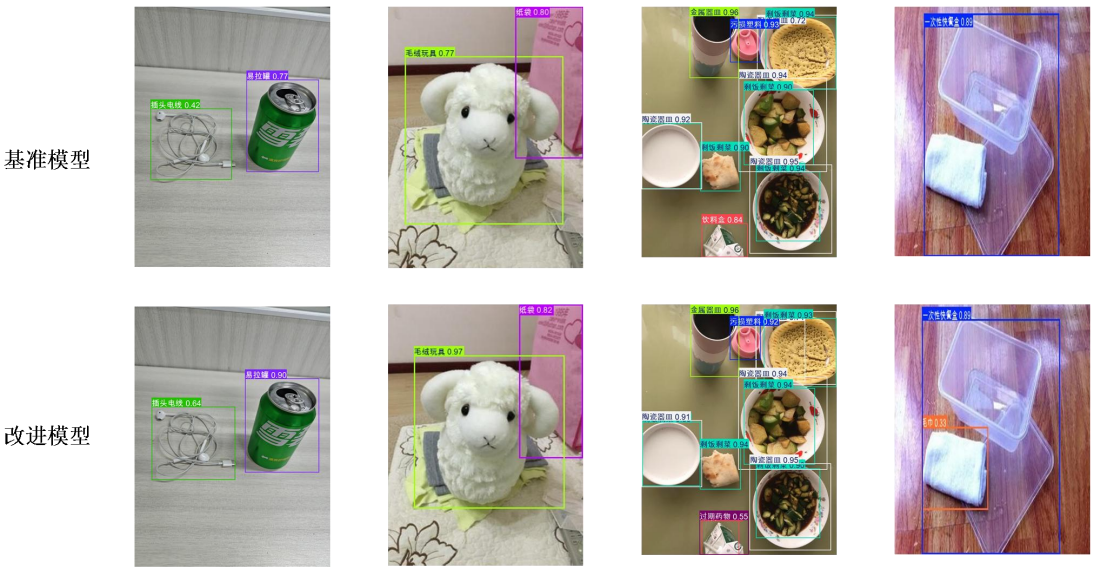


图 24 垃圾分类识别效果对比图

## 3.4 性能评估

本系统在训练与验证过程中表现出优异的性能，各项损失指标持续下降，表明模型的收敛性良好；同时，精确率、召回率及平均精度等评价指标持续提升，最终达到较高水平，体现了模型在垃圾目标检测与分类任务中的准确性和鲁棒性。具体来看，无论是在单个目标还是多目标检测场景中，模型均能实现较低的识别误差和较高的识别准确率。在不同光照条件和复杂场景下，系统依然保持稳定的检测表现，具备良好的泛化能力和实际应用价值。以下为主要性能指标情况：

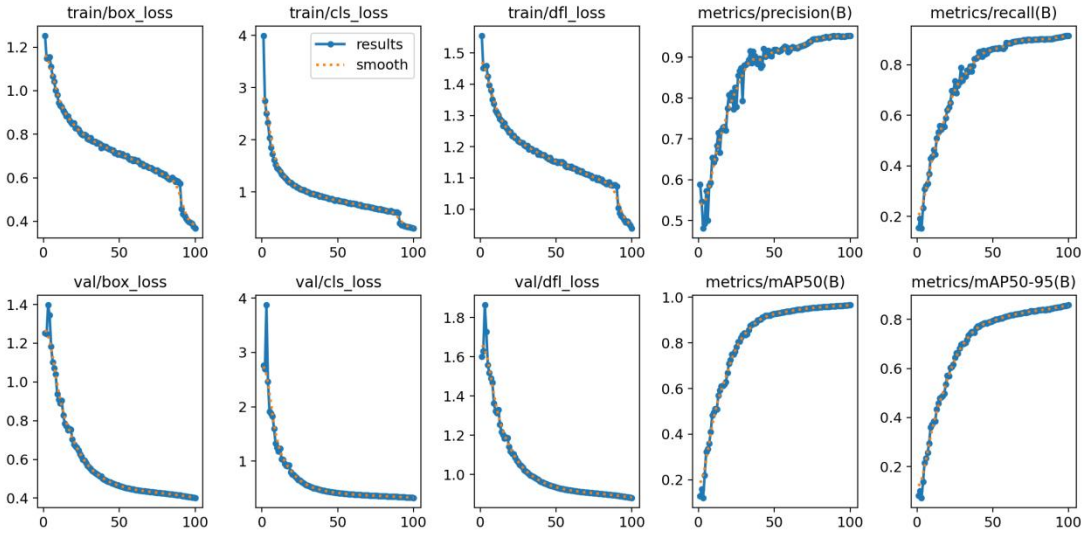


图 25 主要性能指标曲线

表 2 性能指标评估表

| 指标名称 | 最终值 | 趋势说明 |
| --- | --- | --- |
| 训练集框损失（train/box_loss） | 约 0.4 | 持续下降，趋于收敛 |
| 训练集类别损失（train/cls_loss） | 约 0.3 | 持续下降，趋于收敛 |
| 训练集 DFL 损失（train/dfl_loss） | 约 1.0 | 持续下降，趋于收敛 |
| 精确率（precision） | 约 0.98 | 持续提升，趋于稳定 |
| 召回率（recall） | 约 0.90 | 持续提升，趋于稳定 |
| 验证集框损失（val/box_loss） | 约 0.38 | 持续下降，趋于收敛 |
| 验证集类别损失（val/cls_loss） | 约 0.3 | 持续下降，趋于收敛 |
| 验证集 DFL 损失（val/dfl_loss） | 约 0.9 | 持续下降，趋于收敛 |
| mAP@0.5 | 约 0.98 | 持续提升，趋于稳定 |
| mAP@0.5:0.95 | 约 0.85 | 持续提升，趋于稳定 |

# 第四部分　总结

## 4.1 可扩展之处

1）当前系统尚未接入云服务平台，无法实现垃圾桶数据的远程上传和集中管理。管理人员无法在电脑或移动端实时查看各个垃圾桶的运行状态、投放情况以及是否存在满溢问题，也无法进行历史数据的分析和综合调度，导致日常运维和管理还主要依赖人工现场巡查，效率和智能化程度有限。

2）目前系统没有集成温度传感器，无法对垃圾桶内部环境的温度变化进行实时监测。因此，无法针对垃圾桶内可能发生的高温、发酵、气体释放等化学反应进行预测和及时干预，缺乏对特殊垃圾反应风险的感知与早期预警能力，可能增加安全隐患。

3）本系统暂未具备短信、微信或 APP 推送等多渠道的报警和通知功能，当垃圾桶出现满溢、温度异常或设备故障等异常情况时，无法将警报信息第一时间通知到相关管理人员。这会影响问题的及时发现与处理，存在管理盲区，不利于提升垃圾分类系统的管理水平和服务体验。

## 4.2 心得体会

在系统开发过程中，我们面临了两个主要的难点。第一个难点是从头开始学习和掌握 ELF2 嵌入式平台。ELF2 作为一款专为人工智能应用设计的高性能嵌入式计算平台，其硬件架构、资源调度方式以及 AI 模型部署流程与传统的单片机或开发板有着较大差异。为了充分发挥 ELF2 平台的算力优势和丰富的外设接口，我们需要系统性地了解其处理器架构、内存管理机制以及与摄像头、舵机等外设的通信方式。同时，还要熟练掌握平台所支持的推理引擎、模型转换工具链等相关开发流程。学习和实践过程中，我们不仅遇到了硬件兼容性、资源分配等问题，还需要不断优化推理速度和系统稳定性。为此，我们查阅了大量技术文档和案例资料，通过持续的实验与调试，逐步掌握了 ELF2 平台的开发方法和调优技巧，为项目的顺利实施打下了坚实的基础。

第二个难点是将基于 PyTorch 框架的改进 YOLO v8 模型成功移植到 RK3588 平台上。由于 RK3588 平台在硬件结构、计算能力及接口兼容性等方面与常规 PC 端有较大不同，模型的优化和平台适配过程极具挑战性。整个移植过程涉及模型量化、算子兼容性调整、硬件加速支持等多个技术环节。我们需要根据 RK3588 的算力与存储资源，进行模型裁剪与参数优化，确保 YOLO v8 能够高效部署、稳定运行并实现实时目标检测。通过不断调优，最终我们顺利实现了模型在 RK3588 平台上的高效部署，为项目的实际应用提供了坚实的技术支撑。

# 第五部分　参考文献


[1] 杜俊.基于 YOLO v8 和迁移学习的垃圾分类方法[J].智能计算机与应用，2024，14(09):63-69.DOI:10.20169/j.issn.2095-2163.240909.

[2] 花严红，樊辉娜.基于 STM32 技术的语音和图像双重识别智能垃圾分类产品设计[J].南方农机，2025，56(09):143-146.

[3] 唐梦雨，曾志林，文勇.基于改进 YOLO v8 算法的垃圾分类识别方法研究[J].环境工程，2025，43(04):110-120.DOI:10.13205/j.hjgc.202504011.

[4] 林厚健，黄凯升，王小增，等.基于 STM32 的智能垃圾桶[J].物联网技术，2025，15(07):110-113+118.DOI:10.16667/j.issn.2095-1302.2025.07.023.

[5] Zhang Jiaqi， Yu Haixia， Liu Yonghui， Jia Junjie， Yuan Wenya， Fu Yifan. Intelligent garbage sorting terminal controlled by STM32 microcontroller[J]. Internet of Things Technology， 2023(01).

[6] Lu Y， Yu J， Zhu X， et al. YOLO v8-Rice: a rice leaf disease detection model based on YOLO v8[J]. Paddy and Water Environment， 2024: 1-16.

[7] He K ， Zhang X ， Ren S ，et al.Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition[M]. 2014.

[8] Wang Bo， Wang Jinmei， Sun Yihao， Qiao Junchao. Design of automatic watering system based on STM32 microcontroller[J]. Modern Information Technology， 2023(12).

[9] Wang P ， Fan X ， Yang Q ，et al.Object detection of mural images based on improved YOLO v8: Object detection of mural images based. : P. Wang et al[J].Multimedia Systems，2025， 31(1).DOI:10.1007/s00530-025-01687-8.

[10] LiJie，XieShuhua，ZhouXinyi，et al.Real-time detection of coal mine safety helmet based on improved YOLO v8[J].Journal of Real-Time Image Processing， 2024.

[11] Wang D ， Song H ， Wang B .YO-AFD: an improved YOLO v8-based deep learning approach for rapid and accurate apple flower detection[J].Frontiers in Plant Science，2025.DOI:10.3389/fpls.2025.1541266.

[12] Zhang R ， Ou H ， Shi P ，et al.A lightweight dense pedestrian detection and tracking algorithm based on improved YOLO v8 and deep sort[J].Signal， Image and Video Processing， 2025， 19(6).DOI:10.1007/s11760-025-04101-y.

[13] Wang H ， Guo X ， Zhang S ，et al.Detection and recognition of foreign objects in Pu-erh Sun-dried green tea using an improved YOLO v8 based on deep learning[J].PLOS ONE，2025， 20(1).DOI:10.1371/journal.pone.0312112.

[14] Wang Z ， Liu S ， Chong M .An efficient YOLO v8-based model with hierarchical feature fusion for enabling real-time detection of miner unsafe behaviors[J].IOP Publishing Ltd，2025.DOI:10.1088/1361-6501/ade280.

[15] Ding Y， Jiang C， Song L， et al. RVDR-YOLO v8: A Weed Target Detection Model Based on Improved YOLO v8[J]. Electronics， 2024， 13(11): 2182.