团队大作业报告

人工智能实验 2024 年春季学期

小组成员：陈伟宁，梁斯哲，刘予桁，吴非泽，张均逸（字母顺序排列）
小组队名(Optional)：_果果特工_NZHZY

一．数据收集与标注

图片来源：Google Images，这里展示部分图片链接：

https://www.healthline.com/nutrition/10-health-benefits-of-apples

https://www.everydayhealth.com/diet-nutrition/diet/what-you-get-from-banana-plus-answers-other-questions/

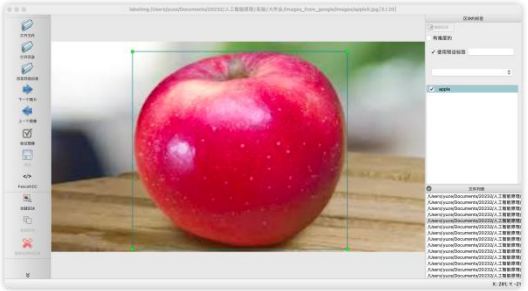
https://www.thespruceeats.com/types-of-oranges-and-tangerines-2216772

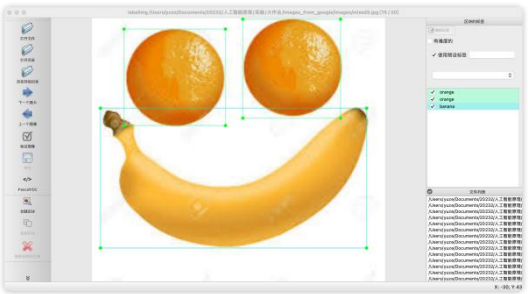
https://www.123rf.com/photo_27887317_oranges-and-banana-fruit.html

图片标注样例

请在此部分显示标注好的 1 张只包含苹果的，1 张只包含香蕉的，1 张只包含橙子的，1 张包含两种或两种以上(mixed)的水果的图片案例，注意，要把对应的水果框出来。

我们使用 labelimg 进行标注。下图展示标注过程。



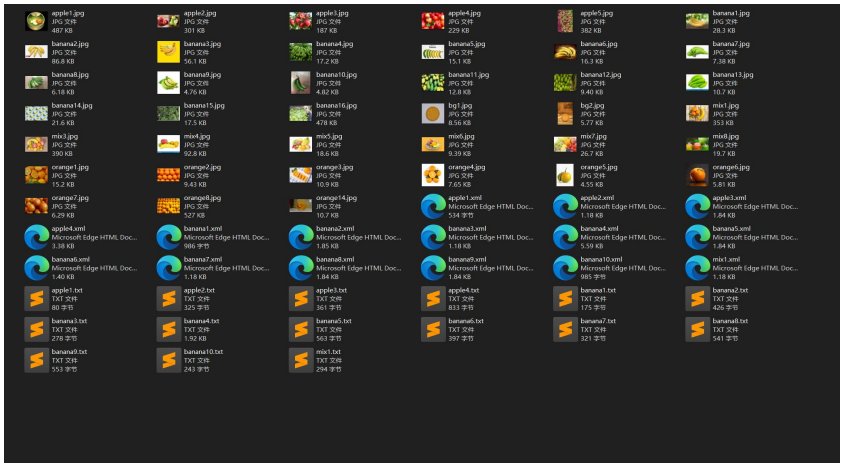
二． 水果检测模型

模型说明：

YOLOv8 是 Ultralytics 推出的最新目标检测算法。该算法采用了单阶段（one-stage）检测方法，融合了 YOLO 系列的多项优势，通过工程实践优化和多尺度模型设计，使其能够高效适应不同应用场景。我们基于在 GitHub 上开源的预训练 YOLOv8 模型进行了微调，以用于水果检测。

训练模型：

数据集 1：Fruit Images for Object Detection
https://www.kaggle.com/datasets/mbkinaci/fruit-images-for-object-detection/data
数据集 2：在数据集 1 基础上进行手动标注的数据集



数据集说明：数据集 1 是根据团队大作业说明文档提供的训练集，数据集 2 则是在使用数据集 1 训练后，针对识别效果较差的情况，手动标注的额外数据训练过程：

训练过程：

我们首先使用数据集 1 进行了预训练模型的初步训练，此过程包括在训练集上进行单次训练周期，总共迭代 60 次。这一阶段的训练旨在让模型能够识别并区分多种水果，克服预训练模型只能使用单一边界框标记所有同类水果的限制。

在初步训练完成后，为了评估模型的泛化能力，我们从互联网上搜集了多样化的水果图像进行预测。对于模型识别精度不佳的样本，我们实施了针对性的数据增强策略：筛选出预测效果较差的图像，并搜集相似的图像样本进行专业的手动标注，从而扩充和丰富原始数据集，形成数据集 2。

随后，我们根据上述数据增强策略对模型进行了进一步的微调训练，累计进行了二十多个训练周期。这一连续的训练过程不仅优化了模型对特定水果样本的识别能力，还提升了模型对新数据的适应性和泛化性。

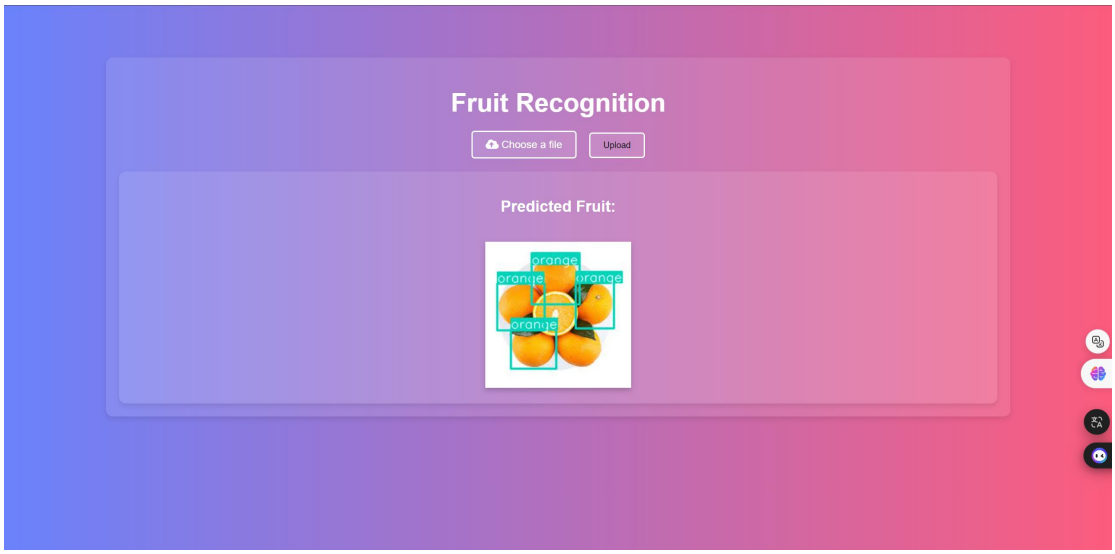
验证模型：

YOLO 自带的验证方法会输出四个参数：Precision，Recall，mAP50，mAP50-95。然而，该方法不提供 IoU（Intersection over Union）的输出，因此我们额外编写了一个计算 IoU 的函数。由于 YOLO 模型的训练函数会自动选出训练过程中表现最佳的模型，我们无需手动

观察损失曲线。

三． 水果检测前端 demo

前端界面展示：



前端界面实现：

前端界面的实现围绕以下几个核心文件展开：index.html、index.css 和 index.js，并通过结合 HTML、CSS 和 JavaScript 来完成。

1． **HTML**：

index.html 文件定义了页面的结构和内容，包括标题、文件上传控件、按钮和用于显示结果的容器。例如，文件上传部分使用了<input type="file" id="upload" accept="image/*">，并用一个自定义标签 <label> 来美化上传按钮。

2． **CSS**：

index.css 文件负责设置页面的样式。使用了各种选择器和 CSS 属性来定义网页的外观和布局风格。例如，body 标签使用了渐变背景和文本居中对齐设置，.container 类设置了页面主要内容的布局和边距，以及按钮和卡片的样式。同时考虑到美观程度，我们设置如.custom-file-upload 和.btn，通过增加 hover 效果来提升用户体验。

3． **JavaScript**：

index.js 文件实现了页面的交互逻辑。主要涉及文件上传、调用 API 处理文件、接收响应并更新页面内容。主函数 uploadFile 会获取文件输入元素的文件并将其上传到服务器，处理服务器的响应并更新页面上的图像。

前端界面的基本逻辑：

1． **页面加载**：

- 加载 HTML 结构，包括文件上传控件、上传按钮和结果展示区域。
- 加载 CSS 文件，应用定义的样式，以确保页面元素按照设计要求展示。
- 加载 JavaScript 文件，确保定义的交互逻辑能够执行。

2. **用户交互**：
    - 用户在文件选择控件中选择一个文件。
    - 用户点击上传按钮后，触发 JavaScript 中的 uploadFile 函数。
    - 在函数中，获取上传的文件，并使用 Fetch API 将文件以表单数据的形式发送到服务器。
3. **数据呈现**：
    - 服务器返回响应，JavaScript 接收并处理。
    - 如果上传成功，获取到的图像数据用 Base64 编码展示在<img id="fruit-image">标签中。
    - 如果上传失败，显示错误提示信息。

四．组员分工

数据收集与标注：吴非泽，张均逸

模型选取与训练：刘予桁，吴非泽

模型测试与代码优化：陈伟宁，刘予桁

系统集成与部署：陈伟宁，梁斯哲

报告整理与撰写：梁斯哲，张均逸

（字母顺序排列）