课程设计 Part2 目标识别（Python）

——刘崇轩 2312801 智能科学与技术

一、实验目的

1．掌握 Ultralytics YOLO 的基本使用，包括模型加载、推理、置信度过滤及可视化流程。
2．理解单模型检测与多模型检测的差异，并比较二者在不同类别目标上的检测性能。
3．学会使用自定义数据集对 YOLO 模型进行训练，并输出最优权重 best1.pt 和 best2.pt。
4.通过实验报告汇总两段代码的实现细节，并对检测结果进行可视化分析。

二、实验原理

1.YOLO（You Only Look Once）

YOLO 是一种一阶段目标检测算法，能在图像中快速定位并分类目标。YOLO11 作为新版模型，具备高精度、高速度等优势。

2.模型训练流程

- 使用带标注的图像数据集，划分训练集和验证集。
- 通过命令行运行 yolo task=detect mode=train model=yolo11x.pt data=xxx.yaml epochs=100，训练出 runs/detect/train/weights/best.pt。
- best1.pt 专注于人（People）和车（Bike）类的高精度检测。
- best2.pt 主要提升了灯（Light）和路障（Roadblock）的识别效果。

3.单模型检测

使用原始模型 yolo11x.pt 对 People 类进行检测。

4.多模型融合检测

使用 best1.pt 和 best2.pt 分别检测不同目标类别，然后将检测结果合并输出，实现多类兼顾、效果最优。

5.可视化分析

OpenCV 用于绘制边界框，结合颜色区分与标签标注，提高检测结果直观性和可验证性。

三、实验步骤

（1）S_TOTAL 检测执行流程：

1.读取 S_TOTAL.jpg 图片：cv2.imread()。
2.初始化 YOLO 模型：使用 YOLO("yolo11x.pt")。
3.设置置信度阈值为 0.125（多次测试取得），仅检测 class_id=0（person）。
4.使用模型进行推理：model(image, conf=0.125)。
5.遍历输出框：保留 cls==0 且 conf>=0.125。
6.结果写入 JSON 文件：刘崇轩_predicted_result.json。
7.在图片上用绿色框标记 People，并写置信度文本。
8.保存可视化图像为 vis_S_TOTAL.jpg。

（2）D_TOTAL 检测执行流程：

1.读取 D_TOTAL.jpg 图片：cv2.imread()。
2.加载两个模型：

- YOLO("best1.pt")：检测 People (id=0)和 Bike (id=1)，置信度阈值 0.9。
- YOLO("best2.pt")：检测 Light (原 id=0 ->输出 id=2)和 Roadblock (原 id=2->输出 id=3)，置信度阈值 0.6。

3.对每个模型调用推理：model(image)。
4.遍历预测结果，分别按各自类别和阈值过滤：输出统一映射的 class_id 和标签。
5.不同类别使用不同颜色框绘制，并标注文字。
6.将所有目标合并写入同一 JSON 文件：刘崇轩_predicted_result.json。
7.保存可视化图像为 vis_D_TOTAL.jpg。

四、程序代码

（1）S_TOTAL 检测代码

```
import json
import os
import cv2
from ultralytics import YOLO

# ------------------ 配置区 ------------------
STUDENT_NAME      = "刘崇轩"
SCORE_THRESH      = 0.125       # 置信度阈值
YOLO_MODEL        = "yolo11x.pt" # 模型权重文件
# 只保留以下 COCO 类别：
#   0: person -> "People"
ALLOWED_CLASS_IDS = {
    0: "People",
}
# 可视化时框和文字的样式
BOX_COLOR   = (0, 255, 0)       # 绿色框 (B,G,R)
TEXT_COLOR  = (0, 0, 0)         # 黑色文字
FONT        = cv2.FONT_HERSHEY_SIMPLEX
FONT_SCALE  = 0.6
THICKNESS   = 2
# --------------------------------------------

# 全局加载 YOLO 模型
model = YOLO(YOLO_MODEL)

def add_detection(results, label, xmin, ymin, xmax, ymax):
    """向 results 字典中添加一个检测框"""
    results["objects"].append({
        "label": label,
        "bbox": [int(xmin), int(ymin), int(xmax), int(ymax)]
    })

def detector(image_path):
    """
    对单张图片使用 YOLO11x 检测 People 类，
    保存 JSON 并可视化输出带框的图片。
    """
    image = cv2.imread(image_path)
    if image is None:
        print(f"[WARN] 无法读取图片：{image_path}")
        return
```

```
    h, w = image.shape[:2]
    image_name = os.path.basename(image_path)
    results = {
        "image_name": image_name,
        "objects": []
    }

    # 推理
    preds = model(image, conf=SCORE_THRESH)[0]
    boxes_xyxy  = preds.boxes.xyxy.cpu().numpy()
    confidences = preds.boxes.conf.cpu().numpy()
    class_ids   = preds.boxes.cls.cpu().numpy().astype(int)

    # 可视化用的画布
    vis = image.copy()

    # 遍历所有检测框
    for (xmin, ymin, xmax, ymax), conf, cls_id in zip(boxes_xyxy, confidences,
class_ids):
        conf = float(conf)
        cls_id = int(cls_id)
        if conf < SCORE_THRESH or cls_id not in ALLOWED_CLASS_IDS:
            continue

        label = ALLOWED_CLASS_IDS[cls_id]
        # 添加到 JSON 结果
        add_detection(results, label, xmin, ymin, xmax, ymax)
        # 在 vis 图上画框
        pt1 = (int(xmin), int(ymin))
        pt2 = (int(xmax), int(ymax))
        cv2.rectangle(vis, pt1, pt2, BOX_COLOR, THICKNESS)
        # 在框上方画文字背景
        text = f"{label} {conf:.2f}"
        (text_w, text_h), _ = cv2.getTextSize(text, FONT, FONT_SCALE, 1)
        cv2.rectangle(vis,
                      (pt1[0], pt1[1] - text_h - 8),
                      (pt1[0] + text_w, pt1[1]),
                      BOX_COLOR, -1)
        # 写文字
        cv2.putText(vis, text, (pt1[0], pt1[1] - 4),
                    FONT, FONT_SCALE, TEXT_COLOR, 1, cv2.LINE_AA)

    # 保存 JSON
    json_name = f"{STUDENT_NAME}_predicted_result.json"
    with open(json_name, "w", encoding="utf-8") as f:
        json.dump(results, f, ensure_ascii=False, indent=2)
    print(f"[INFO] 已保存检测结果: {json_name}")
```

```
    # 保存可视化图片
    vis_name = f"vis_{image_name}"
    cv2.imwrite(vis_name, vis)
    print(f"[INFO] 已保存可视化图片: {vis_name}")

if __name__ == "__main__":
    # 对当前文件夹下的 S_TOTAL.jpg 进行检测并可视化
    detector("S_TOTAL.jpg")
```

（2）D_TOTAL 检测代码

```
import json
import os
import cv2
from ultralytics import YOLO

# ------------------ 配置区 ------------------
STUDENT_NAME            = "刘崇轩"
SCORE_THRESH_MODEL1     = 0.9      # 模型 1 置信度阈值
SCORE_THRESH_MODEL2     = 0.6      # 模型 2 置信度阈值
YOLO_MODEL1             = "best1.pt"  # 模型 1 权重文件
YOLO_MODEL2             = "best2.pt"  # 模型 2 权重文件

# 模型 1 只保留以下类别：
ALLOWED_CLASSES_MODEL1 = {
    0: "People",
    1: "Bike",
}
# 模型 2 只保留以下类别：
ALLOWED_CLASSES_MODEL2 = {
    0: "Light",
    2: "Roadblock"
}

# 输出 JSON 时，指定新的 class_id
OUTPUT_ID_MAP = {
    "Light": 2,
    "Roadblock": 3
}

# 每个类别对应的可视化框颜色 (B, G, R)
BOX_COLOR_MAP = {
    "People":    (0, 255, 0),   # 绿色
    "Bike":      (255, 0, 0),   # 蓝色
    "Light":     (0, 0, 255),   # 红色
    "Roadblock": (0, 255, 255), # 黄色
}
# 每个类别对应的文字颜色 (B, G, R)
TEXT_COLOR_MAP = {
    "People":    (0, 0, 0),     # 黑色
```

```
    "Bike":      (255, 255, 255), # 白色
    "Light":     (255, 255, 255), # 白色
    "Roadblock": (0, 0, 0),     # 黑色
}

# 文本样式
FONT            = cv2.FONT_HERSHEY_SIMPLEX
FONT_SCALE      = 0.6
THICKNESS       = 2

# 全局加载两个 YOLO 模型
model1 = YOLO(YOLO_MODEL1)
model2 = YOLO(YOLO_MODEL2)

def add_detection(results, label, output_id, xmin, ymin, xmax, ymax):
    """向 results 字典中添加一个检测框"""
    results["objects"].append({
        "label": label,
        "class_id": int(output_id),
        "bbox": [int(xmin), int(ymin), int(xmax), int(ymax)]
    })

def run_model(model, allowed_classes, thresh, image, vis, results):
    """
    使用指定模型检测并将符合阈值和类别限制的结果添加到 results 中，同时绘制到 vis 图像上。
    """
    preds = model(image)[0]
    boxes = preds.boxes.xyxy.cpu().numpy()
    confs = preds.boxes.conf.cpu().numpy()
    cls_ids = preds.boxes.cls.cpu().numpy().astype(int)

    for (xmin, ymin, xmax, ymax), conf, cls_id in zip(boxes, confs, cls_ids):
        if conf < thresh or cls_id not in allowed_classes:
            continue
        label = allowed_classes[cls_id]
        # 根据 label 映射新的 class_id
        output_id = OUTPUT_ID_MAP.get(label, cls_id)
        add_detection(results, label, output_id, xmin, ymin, xmax, ymax)
        # 框与文字颜色映射
        box_color = BOX_COLOR_MAP.get(label, (0, 255, 0))
        text_color = TEXT_COLOR_MAP.get(label, (0, 0, 0))
        # 画框
        pt1 = (int(xmin), int(ymin))
        pt2 = (int(xmax), int(ymax))
        cv2.rectangle(vis, pt1, pt2, box_color, THICKNESS)
        # 文字背景
        text = f"{label} {conf:.2f}"
        (tw, th), _ = cv2.getTextSize(text, FONT, FONT_SCALE, 1)
        cv2.rectangle(vis,
```

```
                      (pt1[0], pt1[1] - th - 8),
                      (pt1[0] + tw, pt1[1]),
                      box_color, -1)
        # 写文字
        cv2.putText(vis, text, (pt1[0], pt1[1] - 4),
                    FONT, FONT_SCALE, text_color, 1, cv2.LINE_AA)

def detector(image_path):
    """
    对单张图片使用两个模型分别检测指定类别，合并输出 JSON 和可视化图片。
    """
    image = cv2.imread(image_path)
    if image is None:
        print(f"[WARN] 无法读取图片: {image_path}")
        return

    image_name = os.path.basename(image_path)
    results = {"image_name": image_name, "objects": []}
    vis = image.copy()
    # 分别运行两个模型并合并结果
    run_model(model1, ALLOWED_CLASSES_MODEL1, SCORE_THRESH_MODEL1, image, vis, results)
    run_model(model2, ALLOWED_CLASSES_MODEL2, SCORE_THRESH_MODEL2, image, vis, results)
    # 保存 JSON 结果
    json_name = f"{STUDENT_NAME}_predicted_result.json"
    with open(json_name, "w", encoding="utf-8") as f:
        json.dump(results, f, ensure_ascii=False, indent=2)
    print(f"[INFO] 已保存合并检测结果: {json_name}")
    # 保存可视化图片
    vis_name = f"vis_{image_name}"
    cv2.imwrite(vis_name, vis)
    print(f"[INFO] 已保存可视化图片: {vis_name}")

if __name__ == "__main__":
    # 对当前文件夹下的 D_TOTAL.jpg 进行检测并合并输出
    detector("D_TOTAL.jpg")
```

五、实验结果显示

可视化结果如下：

（1）vis_S_TOTAL.JPG

检测出 16 个 People（置信度 0.125）

（2）vis_D_TOTAL.JPG

检测出 5 个 People、7 个 Bike、2 个 Light、3 个 Roadblock（置信度 0.9、0.6）

（刘崇轩_predicted_result.json 文件已存储在程序文件夹中，此处不再显示）

六、实验分析总结

1. 单模型 vs 多模型检测差异：
   - 单模型对 People 检测效果直观，但无法检测其他目标。
   - 多模型融合了不同模型优势，可同时检测多类，灵活性更高。
   - 自动裁剪去除黑边，使结果更紧凑。
2. 阈值设置对比：
   - 单模型低阈值（0.125）提高了召回率；
   - 多模型分别使用高阈值（0.9/0.6）以保证不同类别精度。
3. 训练模型的重要性：

   自训练的 best1.pt 和 best2.pt 在各自关注的类别上显著提升了检测质量，适合复杂多目标场景下的精细控制。

4.可视化意义

不同颜色的框及文本让小目标（如 Light）和大目标（如 People）一目了然，便于快速检查。

5.改进方向

- 融合更多模型或使用软 NMS 进一步提升检测质量；
- 将实验结果量化为准确率、召回率等指标，进行更全面评估。