目标检测

1 模型介绍

此项目采用YOLO模型实现机动车与行人的二维目标检测。

1.1 模型概述

YOLO（You Only Look Once）是主流的二维目标检测模型，最早由Joseph Redmon及其团队在论文《You Only Look Once: Unified, Real-Time Object Detection》中提出，被后人继续优化发展，至今已到第十代。该模型的基本思想是将目标检测问题转化为回归问题，通过卷积神经网络提取图像的特征向量，并同时预测图像中的边界框坐标和类别标签。YOLO在整个图像上进行全局预测，从而实现快速且实时的目标检测。

在机动车与行人的二维目标检测任务中，输入为摄像头捕捉到的交通流图像，输出为被检测出的各个目标的边界框范围与类别。

1.2 模型架构

YOLO的模型架构在不同版本中略有变化，但核心思想保持一致。

- 基础网络：利用一系列卷积层提取图像特征，从浅到深堆叠，可以得到含有复杂信息的特征表达。
- 输出预测：在得到图像特征向量后，需要基于此进行目标检测边界框范围与置信度的预测，这需要在卷积层后再添加全连接层。
- 损失函数：损失函数包含三个部分，一是边界框的定位误差，二是置信度误差，三是类别预测的交叉熵损失，这与输出是对应的。

1.3 具体实现

- **数据预处理**：将视频序列按照一定的帧率拆分成下标有序的图像序列。
- **选择模型**：根据推理的速度与精度范围的需要，选择训练好的模型并加载。
- **可视化**：用模型在图像上进行推理，并把推理结果预测框可视化到新的图像或视频流中。

2 本地部署

- 设备要求：Windows/Linux下均可运行，预装Anaconda/Miniconda；
- 需要配有Nvidia独立显卡、并安装Cuda、Cudnn、Pytorch。如果是Windows系统，建议在安装Cuda前，先安装Visual Studio。
- 用RTX 4060 Laptop GPU在352x640大小的图像上进行推理，只需要41.1ms的时间，下面给了案例。

2.1 数据集下载和准备

采用的是Youtube上下载的某段Traffic Footage的视频截图，代码中已经给出demo文件夹，不需要额外下载。

2.2 Python虚拟环境配置

```
conda create --name onsite python=3.12
conda activate onsite

# 安装pytorch
pip3 install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu124

pip install -U ultralytics
```

2.3 具体步骤

2.3.1 数据预处理

```
import cv2
video_path = 'demo/demo.mp4'
cap = cv2.VideoCapture(video_path)
```

- video_path：交通流视频文件的存放路径

2.3.2 选择模型

```
from ultralytics import YOLO
model_name = 'yolov10b.pt'
model = YOLO(model_name)
```

- model_name：采用模型的名称，如果本地搜索不到，会联网自动下载，可能需要执行两次

2.3.3 可视化

```
window_name = 'Object Detection'
escape_key = 'q'

while cap.isOpened():
    success, frame = cap.read()

    if not success:
        break

    results = model(frame)

    if len(results)==0:
        cv2.imshow(window_name, frame)
    else:
        annotated_frame = results[0].plot()
        cv2.imshow(window_name, annotated_frame)

    if cv2.waitKey(1) & 0xFF == ord(escape_key):
        break
```

- window_name：可视化窗口名称
- escape_key：退出的按键名称

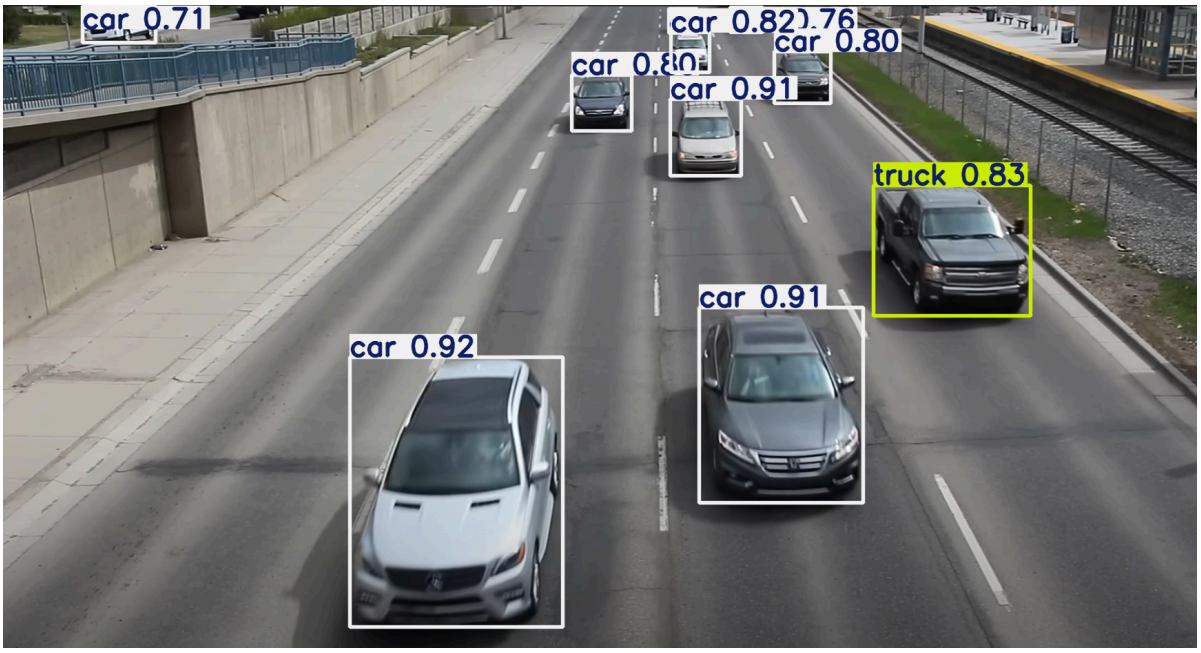
3 完整Demo

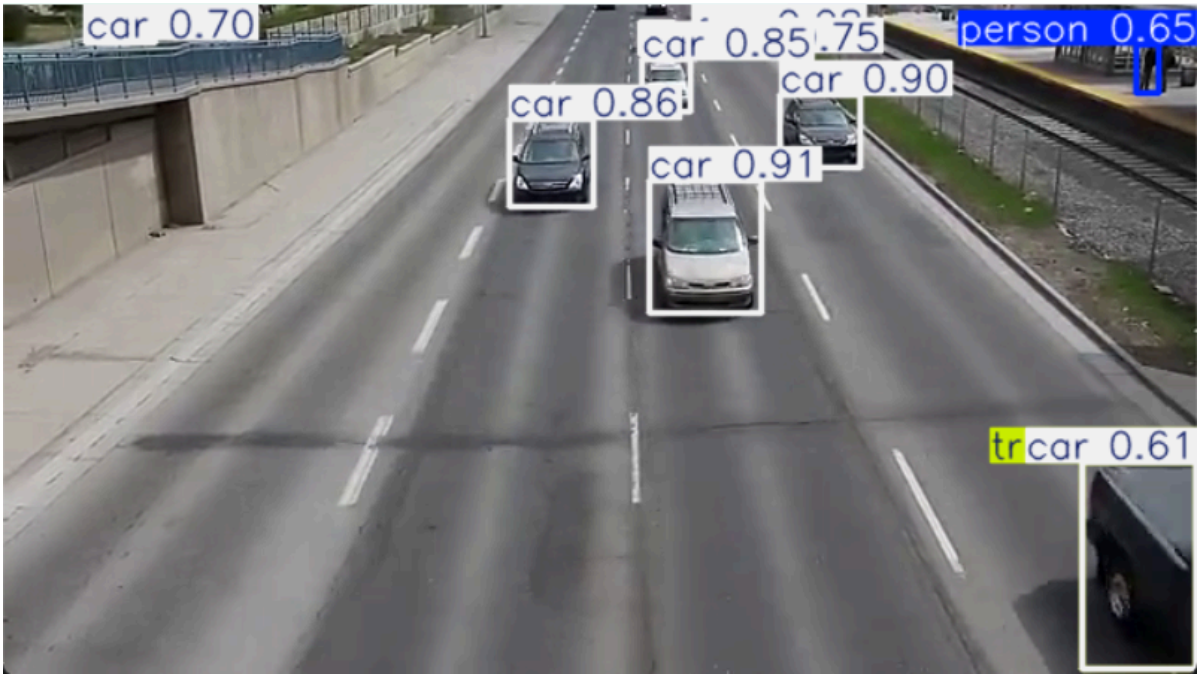
```
from ultralytics import YOLO
import cv2

# 数据预处理
video_path = 'demo/demo.mp4'
cap = cv2.VideoCapture(video_path)

# 选择模型
model_name = 'yolov10b.pt'
model = YOLO(model_name)

# 可视化
window_name = 'Object Detection'
escape_key = 'q'

while cap.isOpened():
    success, frame = cap.read()

    if not success:
        break

    results = model(frame)

    if len(results) == 0:
        cv2.imshow(window_name, frame)
    else:
        annotated_frame = results[0].plot()
        cv2.imshow(window_name, annotated_frame)

    if cv2.waitKey(1) & 0xFF == ord(escape_key):
        break

# 结束后关闭可视化窗口
cap.release()
cv2.destroyAllWindows()
```

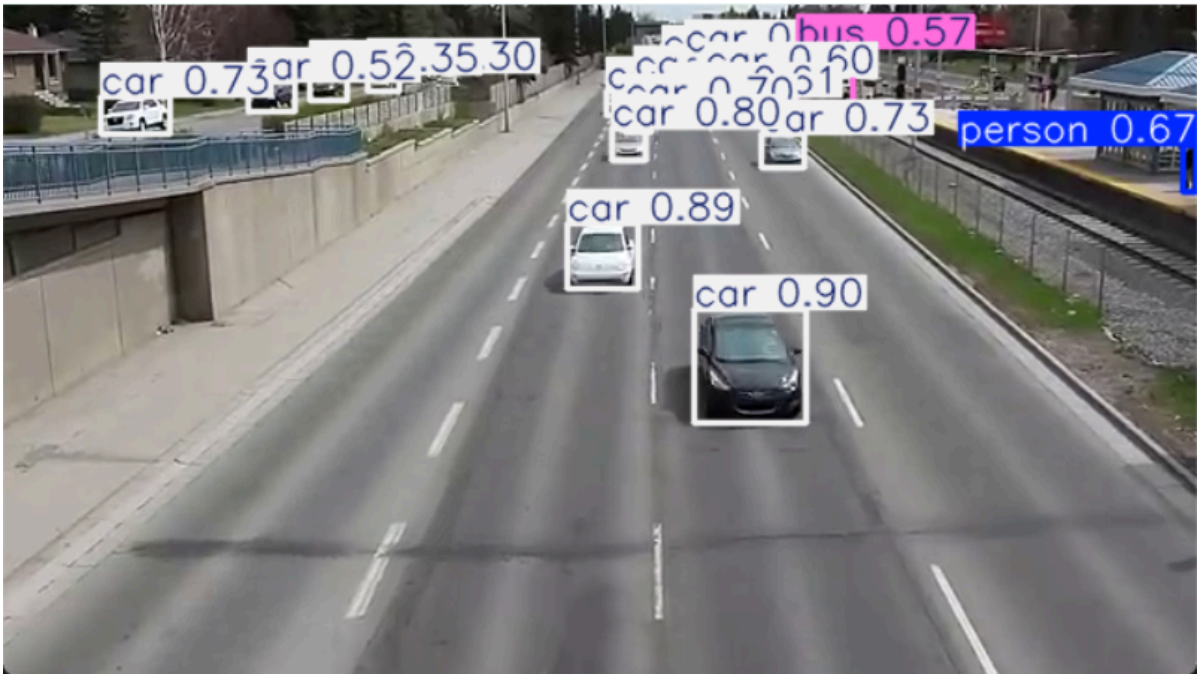
结果输出：

```
0: 384x640 17 cars, 1 train, 9.5ms
Speed: 0.0ms preprocess, 9.5ms inference, 0.0ms postprocess per image at shape
(1, 3, 384, 640)

0: 384x640 16 cars, 1 train, 10.0ms
Speed: 0.0ms preprocess, 10.0ms inference, 0.0ms postprocess per image at shape
(1, 3, 384, 640)

...
```


car 0.61
car 0.76
person 0.61
car 0.65
car 0.54
car 0.93


car 0.71
car 0.80
car 0.91
truck 0.83
car 0.91
car 0.92


car 0.70
car 0.85
person 0.65
car 0.90
car 0.86
car 0.91
car 0.61


car 0.73
car 0.52
bus 0.57
car 0.80
car 0.73
person 0.67
car 0.89
car 0.90