任务2.2

训练

训练参数

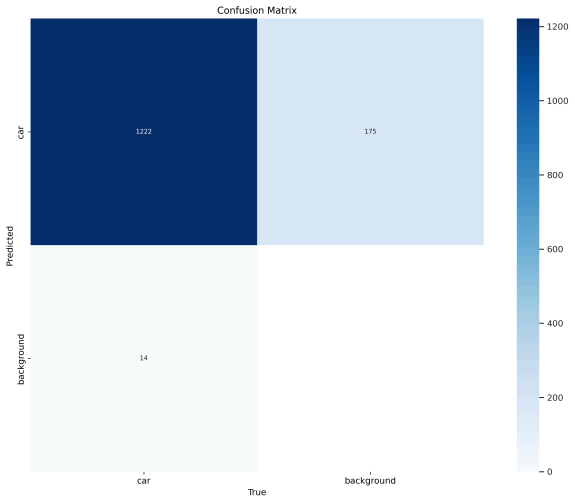
数据集划分：训练集70%，测试集15%，验证集15%
epochs：20，batch：16
模型：yolov8m.pt

训练代码

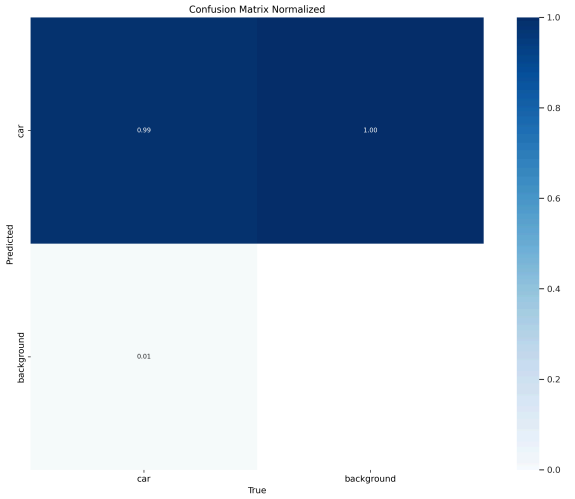
```
# -*- coding: utf-8 -*-
from ultralytics import YOLO
import torch

if __name__ == '__main__':
    # 加载预训练模型，在该模型基础上，训练目标检测的模型
    model = YOLO('./yolov8n.pt')

    # 训练自定义数据集，数据配置保存在 data.yaml 中
    model.train(data='./datasets/phoneandmouse/data.yaml', epochs=1, batch=16)

    # 使用验证集验证效果
    model.val()
```

训练结果

混淆矩阵

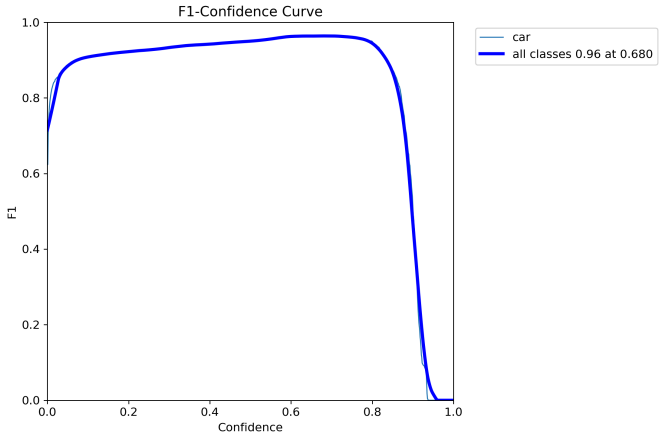


归一化后：

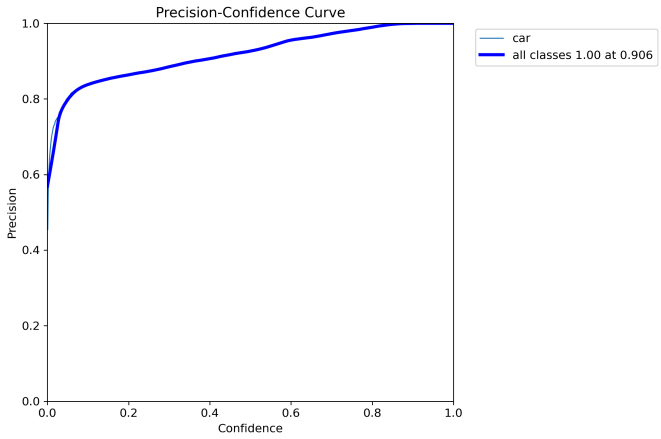


效果很好

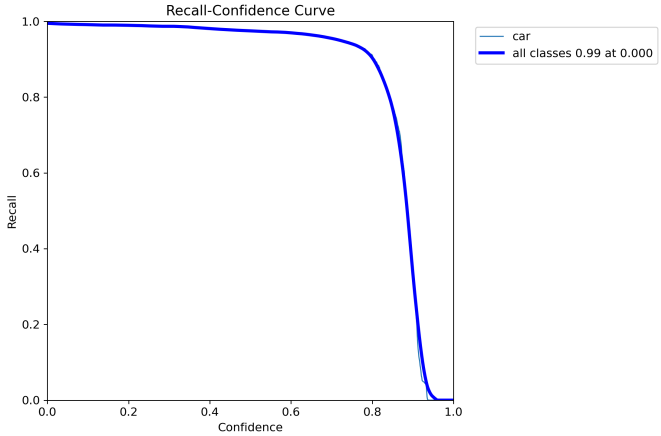
F1 curve



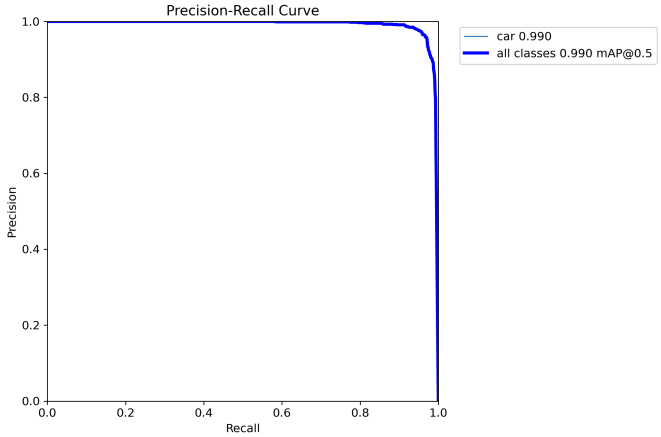
P curve



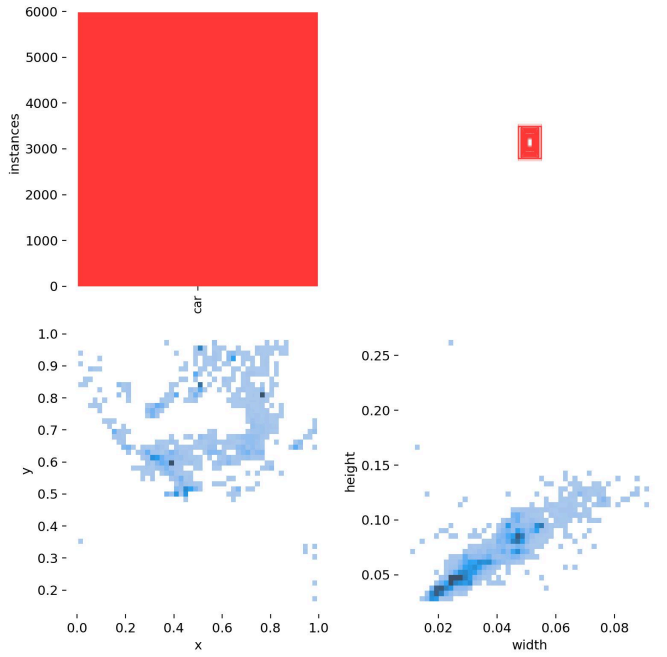
R curve

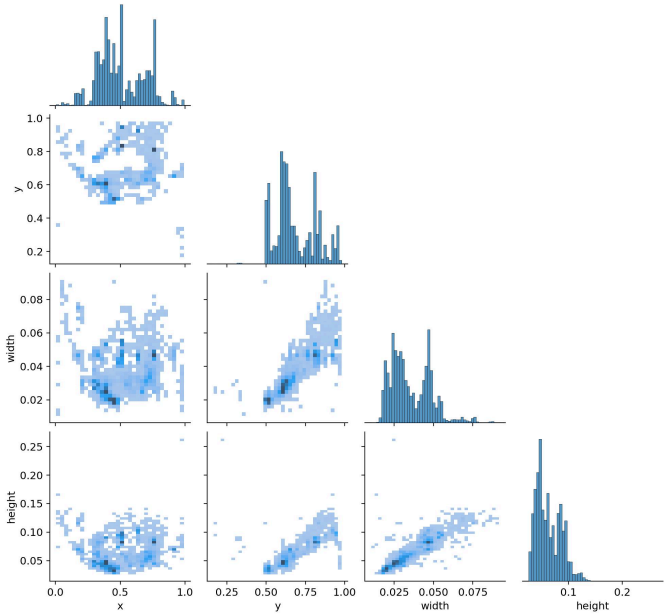


PR curve

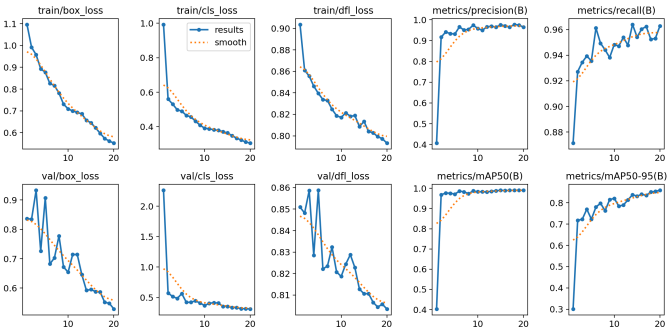


labels



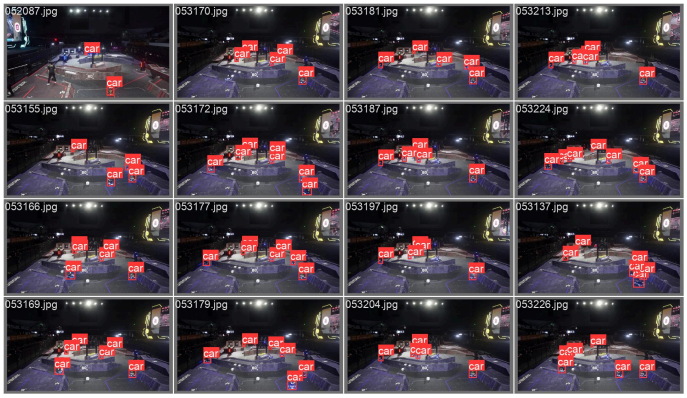


一些结果



验证集推理结果与标注结果对照

标注：



推理：

标注：

推理：

标注： 

推理： 

视频推理代码

```
import cv2
from ultralytics import YOLO
# 加载模型
model = YOLO('./runs/detect/train10/weights/best.pt')
# 打开视频文件
video_path = "./深度学习任务二测试视频.mp4"
cap = cv2.VideoCapture(video_path)
# 获取视频帧的维度
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
#创建VideoWriter对象
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out = cv2.VideoWriter('./output3.mp4', fourcc, 20.0, (frame_width, frame_height))
#循环视频帧
while cap.isOpened():
    # 读取某一帧
    success, frame = cap.read()
    if success:
        # 使用yolov8进行预测
        results = model(frame)
        #可视化结果
        annotated_frame = results[0].plot()
        #将带注释的帧写入视频文件
        out.write(annotated_frame)
```

```
    else:
        # 最后结尾中断视频帧循环
        break
#释放读取和写入对象
cap.release()
out.release()
```