武汉大学国家网络安全学院实验报告

课程名称	内容安全实验	实验日期	2024-3-25
实验名称	实验 1：图像处理与分析		
姓名	学号	专业	班级
邓鹏	2021302181152	网络空间安全	6 班

目录

一、实验目的及实验内容

（本次实验的具体内容；必要的原理分析）

1.1 实验目的

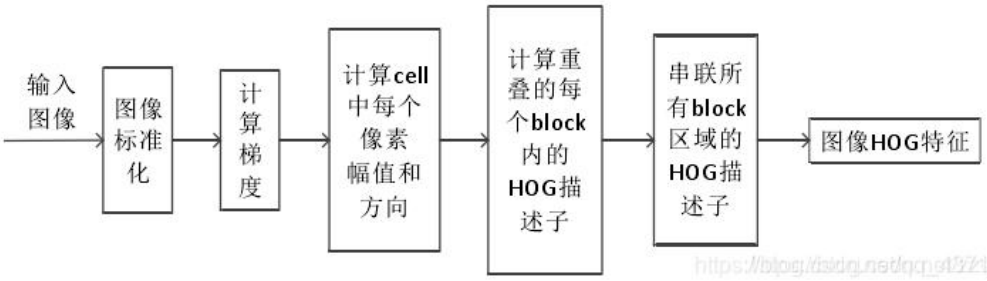
1. 理解 HOG 算法，使用 SVM\KNN 完成目标行人检测
2. 本地部署 yolo V8-PyTorch，任选数据集，完成图像目标检测任务

1.2 实验内容

1.2.1 HOG 特征提取原理

HOG 就是梯度方向直方图 (Histogram of Oriented Gradient, HOG) ，HOG 特征是直接将图像像素点的方向梯度作为图像特征，包括梯度大小和方向。

HOG 特征提取的步骤：



1. 图像标准化，在其他地方可能也会成为对图像进行伽马处理。就是对每个图像中的像素点进行 n 次方，其中的 n 称为伽马系数，一般 n=0.5。伽马值，是对图像的优化调整，是亮度和对比度的辅助功能。

2. 计算梯度，梯度的计算是要求每个像素点水平和垂直方向的梯度值

$$G_{\mathrm{x}}(x,y)=I(\mathrm{x}+1,y)-I(x,y)$$

$$G_{\mathrm{y}}(x,y)=I(\mathrm{x},y+1)-I(x,y)$$

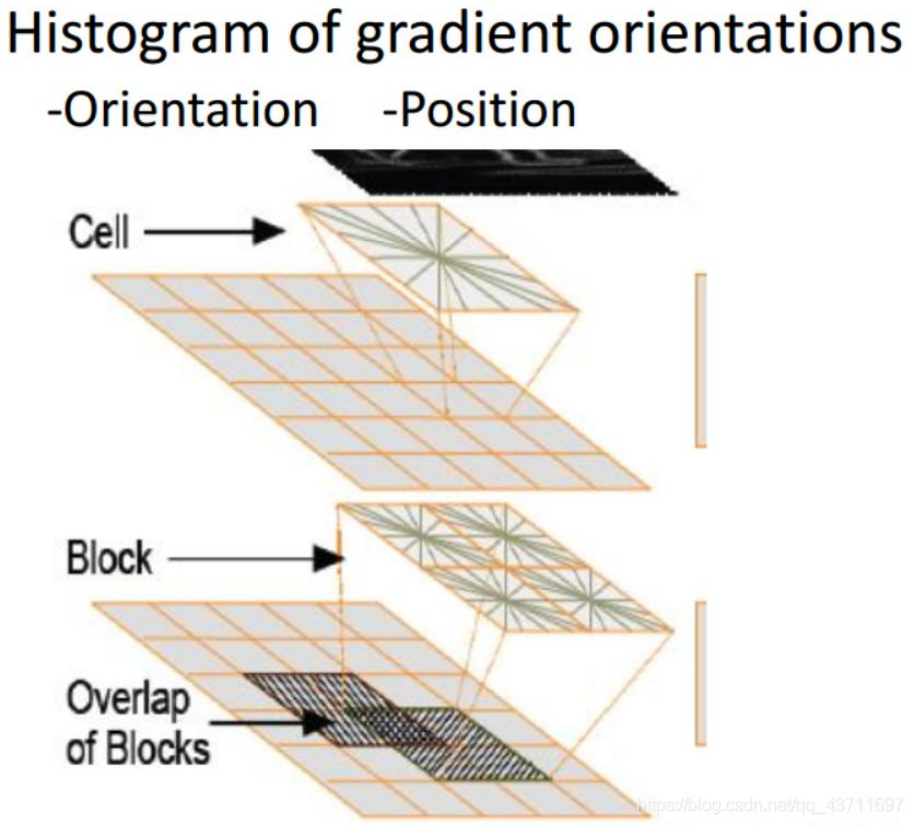
I(x,y)是指在(x,y)处的像素点的值，然后就得到了 x 和 y 方向上的梯度值

然后就可以求梯度方向和幅值了

$$G(x,y)=\sqrt{G_{\mathrm{x}}(x,y)^2+G_{\mathrm{y}}(x,y)^2}$$

$$\alpha(x,y)=\tan^{-1}\left(\frac{G_{\mathrm{y}}(x,y)}{G_{\mathrm{x}}(x,y)}\right)$$

3. 接下来要说的是重点就是 Win，block，blockStride，cell，cellBin



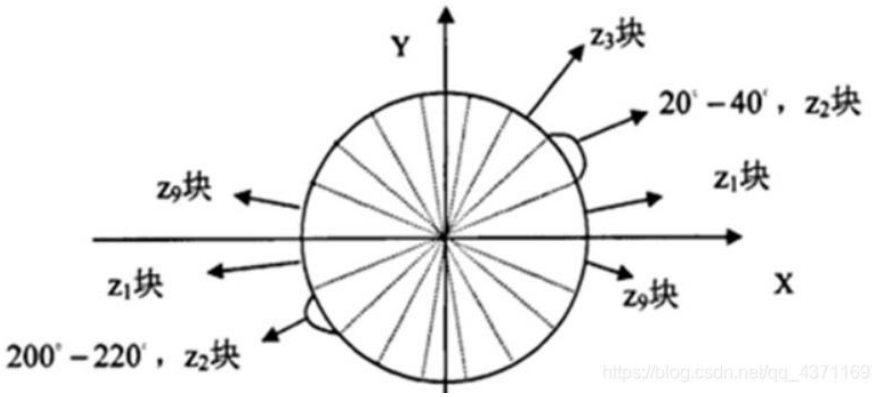
• win 就是一个检测窗口一般的化我们都是习惯上采用(64,128),因为这是用来检测行人或车辆上研究出来比较适合的一个参数值 1：2 或 2：1。

• block 是在 win 窗口里面移动的一个(16,16)的块

• blockStride 就是块每次移动的步长了一般为(8,8)

• cell 就是 block 里面一个(8,8)的元组，一个块里面含有四个 cell

• 一个 cell 里面一般分为 9 个 bin 也就是 9 个方向，下面这个图是方向分类的一个圆图



4. 然后就像以上那样求出所有像素点得特征后我们来计算一下。一张图像一般裁剪成(64,128)（行人检测）然后一共有（（64-16）/8+1）（（128-16）/8+1）=105 个块，一个块有 4 个 cell，一个 cell 有 9 个 bin，那么一共有 105x4x9=3780 个 bin，也就是一张图像会得到 3780 个数字也就是这张图象特征维度。然后我们会把这个 3780 个数放在一个数组里面。

1.2.2 SVM 原理概述

支持向量机（SVM）是一种强大的监督学习算法，用于分类和回归任务，尤其在分类问题中表现突出。SVM 的核心思想是寻找一个最优超平面，使得不同类别的数据在该超平面两侧，并且各自距离超平面尽可能远。这样的超平面能够最大化两类数据之间的间隔，从而提高分类的准确性和泛化能力。

• 线性可分 SVM

假设数据集线性可分，即存在一个超平面能够完美地将两类数据分开，超平面的数学表示为：

$$w^T x + b = 0$$

其中，w 是超平面的法向量，决定了超平面的方向；b 是超平面的截距；x 是数据点。$w^T x + b$ 的符号决定了点 x 位于超平面的哪一侧。

SVM 的目标是找到 w 和 b，使得最近的数据点（即支持向量）到超平面的距离最大化。这个最小距离称为间隔，SVM 尝试最大化这个间隔，从而得到最优超平面。这个优化问题可以表示为：

$$\begin{aligned} &\min_{w,b} \frac{1}{2}||w||^2 \\ &s.t. \quad y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n \end{aligned}$$

• 核技巧

在实际应用中，很多数据集不是线性可分的。SVM 通过引入核函数来处理非线性可分的情况。核函数能够将数据映射到一个高维空间，在这个高维空间中，数据可能变得线性可分。常用的核函数包括线性核、多项式核、径向基核等。

使用核函数后，SVM 的决策函数变为：

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b\right)$$

其中，$K(x_i, x)$ 是核函数，它计算映射到高维空间后的两个点的内积；αi 是由训练过程决定的拉格朗日乘子，非零的αi 对应的 xi 即为支持向量。

1.2.3 HOG+SVM 行人检测

1. 特征提取：首先，从训练图像中提取 HOG 特征。这些特征能够有效地捕捉行人的轮廓和形状信息，对光照变化和小的形状变形具有鲁棒性。

2. 训练 SVM：使用提取的 HOG 特征和对应的标签（行人或非行人）训练 SVM 模型。训练过程中，SVM 学习如何根据 HOG 特征将图像区域分类为包含行人或不包含行人。

3. 行人检测：对测试图像进行同样的 HOG 特征提取，然后使用训练好的 SVM 模型对这些特征进行分类，以确定图像的哪些区域包含行人。

4. 后处理：行人检测后，可能需要进行非最大抑制等后处理步骤，以合并重叠的检测框，从而改进最终的检测结果。

1.2.4 YOLO 工作原理

1. 将图像分格：YOLO 将输入图像划分成一个个小格子（例如 13x13），每个格子负责预测中心点在该格子内的目标。
2. 边界框预测：对于每个格子，模型会预测多个边界框。每个边界框包含五个预测值：边界框的中心（x, y）坐标、宽度、高度，以及边界框包含目标的置信度。置信度表示边界框中确实包含目标的概率与该预测框与实际框的 IOU（交并比）的乘积。
3. 类别概率预测：每个格子还会预测每个类别的条件概率，即在边界框包含目标的前提下，该目标属于每个类别的概率。
4. 模型训练：YOLO 使用整个图像进行训练，并直接从图像像素优化预测的边界框和类别概率。训练过程中使用损失函数来惩罚模型对边界框位置、大小、置信度和类别概率的错误预测。
5. 非最大抑制（NMS）：在预测阶段，为了移除多余的重叠框，YOLO 使用非最大抑制技术。这个过程会保留置信度最高的边界框，并移除与其高度重叠的其他边界框。

二、实验环境

（本次实验所使用的开发环境、依赖包等的情况）

1. Windows10 + Anaconda3
2. 参考 在 anaconda 下安装 OpenCV-Python 的最简单方法 _conda opencv-python-CSDN 博客安装 opnecv-python
3. 数据集 INRIAPerson，参考【计算机视觉】INRIA 行人数据集 （INRIA Person Dataset） _inria 数据集-CSDN 博客
数据集下载地址 ftp://ftp.inrialpes.fr/pub/lear/douze/data/INRIAPerson.tar
4. YOLOv8，参考 YOLOv8(2023 年 8 月版本)安装配置！一条龙傻瓜式安装，遇到问题评论区提问_yolov8 下载-CSDN 博客

三、实验步骤及结果分析

（详细描述实验步骤，并根据具体实验，记录、整理相应的数据表格等，对实验结果进行分析）

4.1 HOG + SVM 完成目标行人检测

1. 导入数据集

```
import cv2
import numpy as np
import os

# 定义样本图片目录和样本数量
pos_img_dir = "D:/Content_Secu/INRIAPerson/INRIAPerson/Train/pos"  # 正样本图片目录
neg_img_dir = "D:/Content_Secu/INRIAPerson/INRIAPerson/Train/neg"  # 负样本图片目录
pos_samples = 2400  # 指定正样本数量
neg_samples = 12000  # 指定负样本数量
central_crop = True  # 是否对正样本进行中心裁剪

model_path = "svm_hog.xml"  # 模型保存路径
```

2. 创建 HOG 描述符

初始化了一个 cv2.HOGDescriptor 对象，这是用于提取图像中的 HOG 特征的关键工具。

```
# 创建HOG描述符
hog = cv2.HOGDescriptor()
```

3. 计算正样本和负样本的 HOG 特征

compute_hog_features 函数，用于从指定目录中的图像提取 HOG 特征。

每张图像的 HOG 特征被存储在 features 列表中，对应的标签（正样本为 1，负样本为-1）被存储在 labels 列表中。

```
def compute_hog_features(img_dir, num_samples, img_size=(64, 128), central_crop=False):
    features = []
    labels = []
    for img_name in sorted(os.listdir(img_dir))[:num_samples]:
        img_path = os.path.join(img_dir, img_name)
        img = cv2.imread(img_path)
        if img is None:
            continue  # 如果图像未正确加载，则跳过
        if central_crop:
            img = img[16:-16, 16:-16]  # 假设原图是96x160，裁剪至64x128
        img = cv2.resize(img, img_size)  # 确保图像是一致的尺寸
        descriptor = hog.compute(img)
        if descriptor is not None:
            descriptor = descriptor.reshape(-1)
            features.append(descriptor)
            label = 1 if img_dir == pos_img_dir else -1
            labels.append(label)
    return np.array(features), np.array(labels)

# 计算正样本和负样本的HOG特征
pos_features, pos_labels = compute_hog_features(pos_img_dir, pos_samples, central_crop=central_crop)
neg_features, neg_labels = compute_hog_features(neg_img_dir, neg_samples, central_crop=False)
```

4. 合并正样本和负样本的 HOG 特征
使用 np.vstack 和 np.hstack 将正负样本的特征和标签合并成一个大的特征矩阵和一个标签向量，以便用于训练 SVM 分类器。

```
# 合并特征和标签
features = np.vstack((pos_features, neg_features))
labels = np.hstack((pos_labels, neg_labels))
```

5. 训练 SVM 分类器并存储模型
初始化了一个 SVM 分类器，并设置了其类型为 C-SVC（C-支持向量分类），核函数为线性核，惩罚系数 C 为 0.01。然后使用合并后的特征矩阵和标签向量对 SVM 进行训练。

```
# 训练SVM分类器
svm = cv2.ml.SVM_create()
svm.setType(cv2.ml.SVM_C_SVC)
svm.setKernel(cv2.ml.SVM_LINEAR)
svm.setC(0.01)
svm.train(features, cv2.ml.ROW_SAMPLE, labels)
svm.save(model_path)
print("SVM training completed and model saved.")
```

6. 行人检测

```
import cv2
import numpy as np

def get_svm_detector(svm):
    sv = svm.getSupportVectors()
    rho, _, _ = svm.getDecisionFunction(0)
    sv = np.transpose(sv)
    return np.append(sv, [[-rho]], 0)

# 加载训练好的SVM模型
model_path = "svm_hog.xml"
svm = cv2.ml.SVM_load(model_path)

# 创建HOG描述符
win_size = (64, 128)
block_size = (16, 16)
block_stride = (8, 8)
cell_size = (8, 8)
n_bins = 9
hog = cv2.HOGDescriptor(win_size, block_size, block_stride, cell_size, n_bins)

# 设置HOG描述符的检测器为训练好的SVM模型
hog.setSVMDetector(get_svm_detector(svm))

# 读取测试图片
test_img_path = "D:/Content_Secu/INRIAPerson/INRIAPerson/Test/pos/person_122.png"
img = cv2.imread(test_img_path)

# 在测试图片上进行行人检测
(rects, _) = hog.detectMultiScale(img, winStride=(8, 8), padding=(16, 16), scale=1.05)

# 画出检测到的行人
for (x, y, w, h) in rects:
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

# 显示结果
cv2.imshow("Detection", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

7. 效果展示

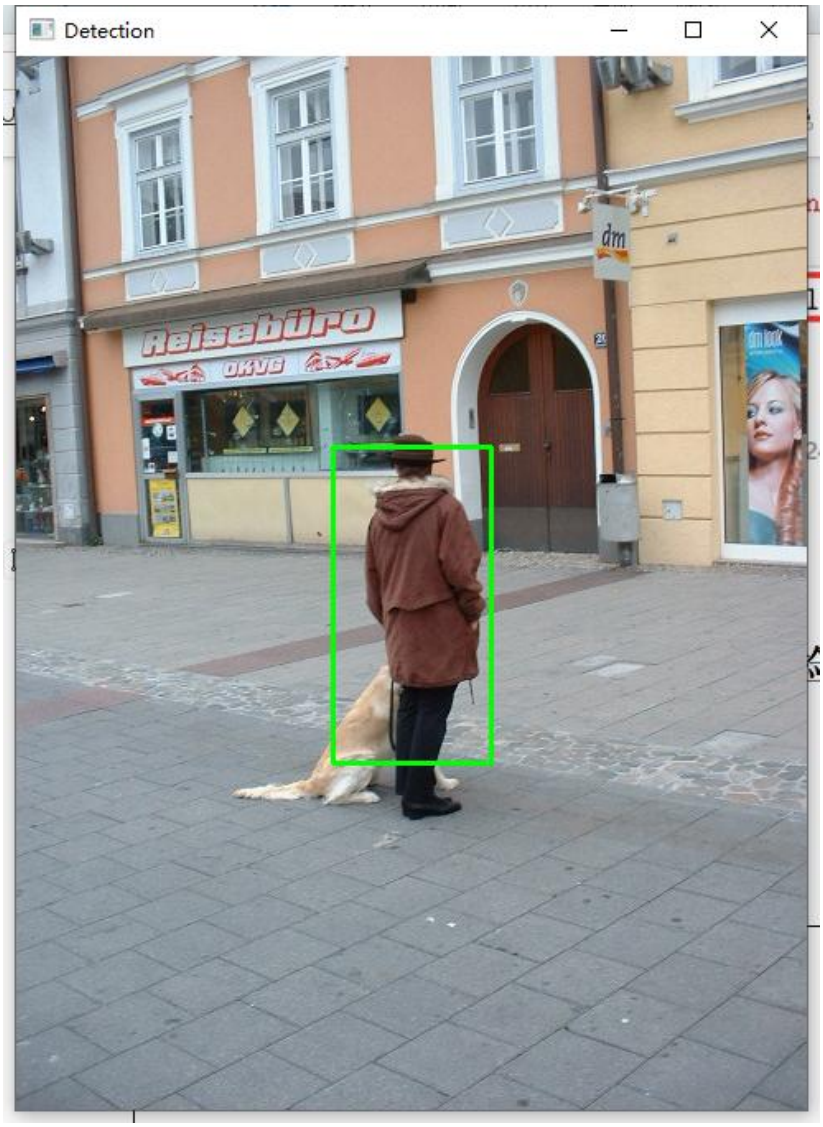
- 训练模型

```
# 训练SVM分类器
svm = cv2.ml.SVM_create()
svm.setType(cv2.ml.SVM_C_SVC)
svm.setKernel(cv2.ml.SVM_LINEAR)
svm.setC(0.01)
svm.train(features, cv2.ml.ROW_SAMPLE, labels)
svm.save(model_path)
print("SVM training completed and model saved.")
```

```
SVM training completed and model saved.
```

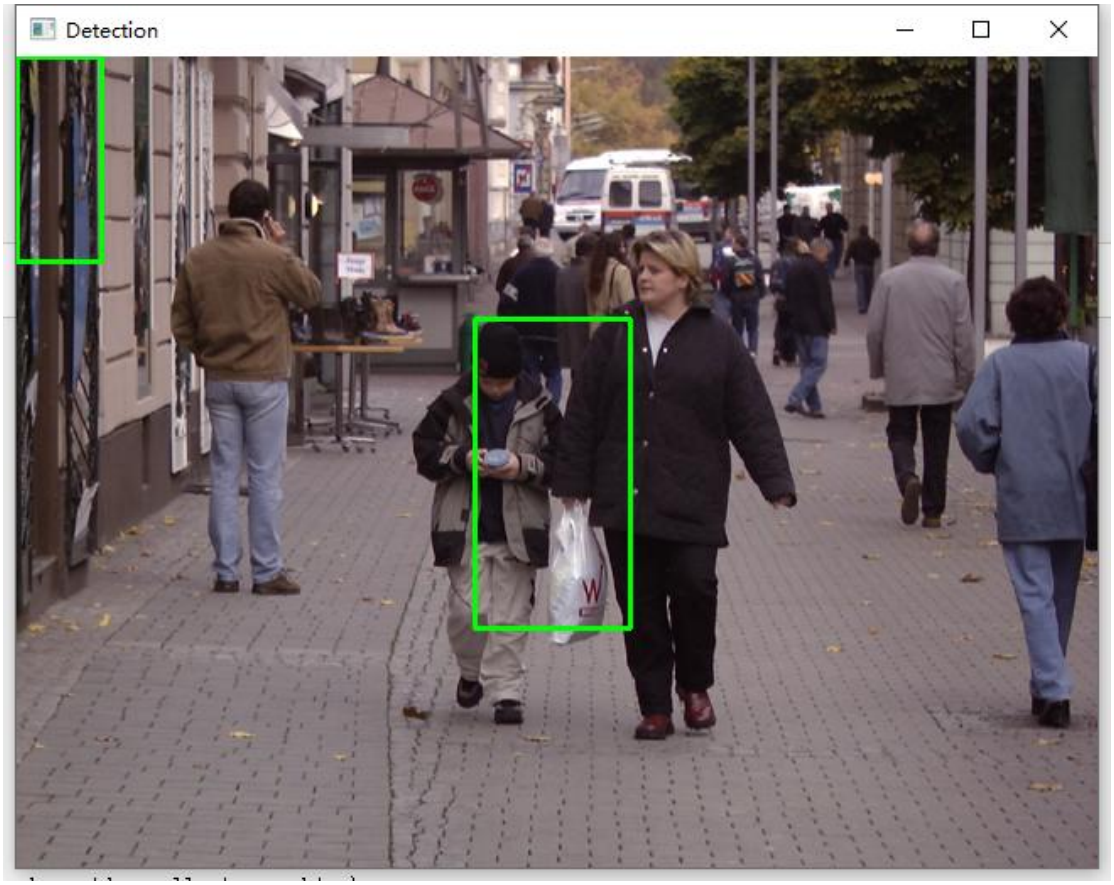
且根目录下生成 svm_hog.xml

- 使用模型并测试



成功！

但是对于多个人体，效果不佳。



3.2 yolo V8-PyTorch 完成图像目标检测任务

1. 导出 onnx
加载一个预训练的 YOLO 模型，并将这个模型导出为 ONNX 格式

```python
from ultralytics import YOLO
# Load a model
model = YOLO('yolov8n.pt')  # load an official model
# Export the model
model.export(format='onnx', opset=12)
```

终端执行

```
(yo8) PS D:\Content_Secu> python onn.py
Ultralytics YOLOv8.1.34 🚀 Python-3.8.19 torch-2.2.1+cu118 CPU (11th Gen Intel Core(TM) i5-11400H 2.70GHz)
YOLOv8n summary (fused): 168 layers, 3151904 parameters, 0 gradients, 8.7 GFLOPs

PyTorch: starting from 'yolov8n.pt' with input shape (1, 3, 640, 640) BCHW and output shape(s) (1, 84, 8400) (6.2 MB)

ONNX: starting export with onnx 1.15.0 opset 12...
ONNX: export success ✅ 0.9s, saved as 'yolov8n.onnx' (12.2 MB)

Export complete (2.8s)
Results saved to D:\Content_Secu
Predict:         yolo predict task=detect model=yolov8n.onnx imgsz=640
Validate:        yolo val task=detect model=yolov8n.onnx imgsz=640 data=coco.yaml
Visualize:       https://netron.app
```

2. 创建 COCO 数据集的 80 个类别并为每个类别分配颜色

```python
# 预定义的COCO数据集的80个类别
CLASSES = {0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5: 'bus', 6: 'train',
           7: 'truck', 8: 'boat', 9: 'traffic light', 10: 'fire hydrant', 11: 'stop sign', 12: 'parking meter',
           13: 'bench', 14: 'bird', 15: 'cat', 16: 'dog', 17: 'horse', 18: 'sheep', 19: 'cow', 20: 'elephant',
           21: 'bear', 22: 'zebra', 23: 'giraffe', 24: 'backpack', 25: 'umbrella', 26: 'handbag', 27: 'tie',
           28: 'suitcase', 29: 'frisbee', 30: 'skis', 31: 'snowboard', 32: 'sports ball', 33: 'kite', 34: 'baseball bat',
           35: 'baseball glove', 36: 'skateboard', 37: 'surfboard', 38: 'tennis racket', 39: 'bottle', 40: 'wine glass',
           41: 'cup', 42: 'fork', 43: 'knife', 44: 'spoon', 45: 'bowl', 46: 'banana', 47: 'apple', 48: 'sandwich', 49: 'orange',
           50: 'broccoli', 51: 'carrot', 52: 'hot dog', 53: 'pizza', 54: 'donut', 55: 'cake', 56: 'chair', 57: 'couch',
           58: 'potted plant', 59: 'bed', 60: 'dining table', 61: 'toilet', 62: 'tv', 63: 'laptop', 64: 'mouse', 65: 'remote',
           66: 'keyboard', 67: 'cell phone', 68: 'microwave', 69: 'oven', 70: 'toaster', 71: 'sink', 72: 'refrigerator',
           73: 'book', 74: 'clock', 75: 'vase', 76: 'scissors', 77: 'teddy bear', 78: 'hair drier', 79: 'toothbrush'}
# 为每个类别分配随机颜色
colors = np.random.uniform( low: 0,  high: 255, size=(len(CLASSES), 3))
```

3. 绘制边界框和类别标签

```python
# 绘制边界框和类别标签
1个用法
def draw_bounding_box(img, class_id, confidence, x, y, x_plus_w, y_plus_h):
    # 准备标签文本，包括类别名称和置信度
    label = f'{CLASSES[class_id]} ({confidence:.2f})'
    # 获取对应类别的颜色
    color = colors[class_id]
    # 在图像上绘制边界框
    cv2.rectangle(img, (x, y), (x_plus_w, y_plus_h), color, 2)
    # 在图像上绘制标签文本
    cv2.putText(img, label,  org: (x - 10, y - 10), cv2.FONT_HERSHEY_SIMPLEX,  fontScale: 0.5, color,  thickness: 2)
```

4. 加载模型

```python
# 从ONNX文件加载模型
model: cv2.dnn.Net = cv2.dnn.readNetFromONNX(onnx_model)
```

5. 图像预处理

```python
# 读取输入图像
original_image: np.ndarray = cv2.imread(input_image)
[height, width, _] = original_image.shape
length = max((height, width))
# 创建方形图像，以适应YOLOv8模型的输入需求
image = np.zeros( shape: (length, length, 3), np.uint8)
image[0:height, 0:width] = original_image
# 计算缩放比例，以还原边界框坐标到原图尺寸
scale = length / 640
```

6. 模型推理

```python
# 为模型准备输入blob
blob = cv2.dnn.blobFromImage(image, scalefactor=1 / 255, size=(640, 640), swapRB=True)
model.setInput(blob)
# 执行模型推理，获取输出
outputs = model.forward() # output: 1 X 8400 x 84
outputs = np.array([cv2.transpose(outputs[0])])
rows = outputs.shape[1]
```

7. 结果处理

```python
# 初始化列表，用于存储检测结果
boxes = []
scores = []
class_ids = []
# 解析模型输出，筛选出置信度高于阈值的检测结果
for i in range(rows):
    classes_scores = outputs[0][i][4:]  # 提取当前行的类别得分
    (minScore, maxScore, minClassLoc, (x, maxClassIndex)) = cv2.minMaxLoc(classes_scores)
    if maxScore >= 0.25:  # 置信度阈值
        box = [
            outputs[0][i][0] - (0.5 * outputs[0][i][2]), outputs[0][i][1] - (0.5 * outputs[0][i][3]),
            outputs[0][i][2], outputs[0][i][3]]
        boxes.append(box)  # 添加边界框
        scores.append(maxScore)  # 添加置信度
        class_ids.append(maxClassIndex)  # 添加类别索引
    # 使用非极大值抑制（NMS）进一步筛选边界框
    result_boxes = cv2.dnn.NMSBoxes(boxes, scores,  score_threshold: 0.25,  nms_threshold: 0.45,  eta: 0.5)
```

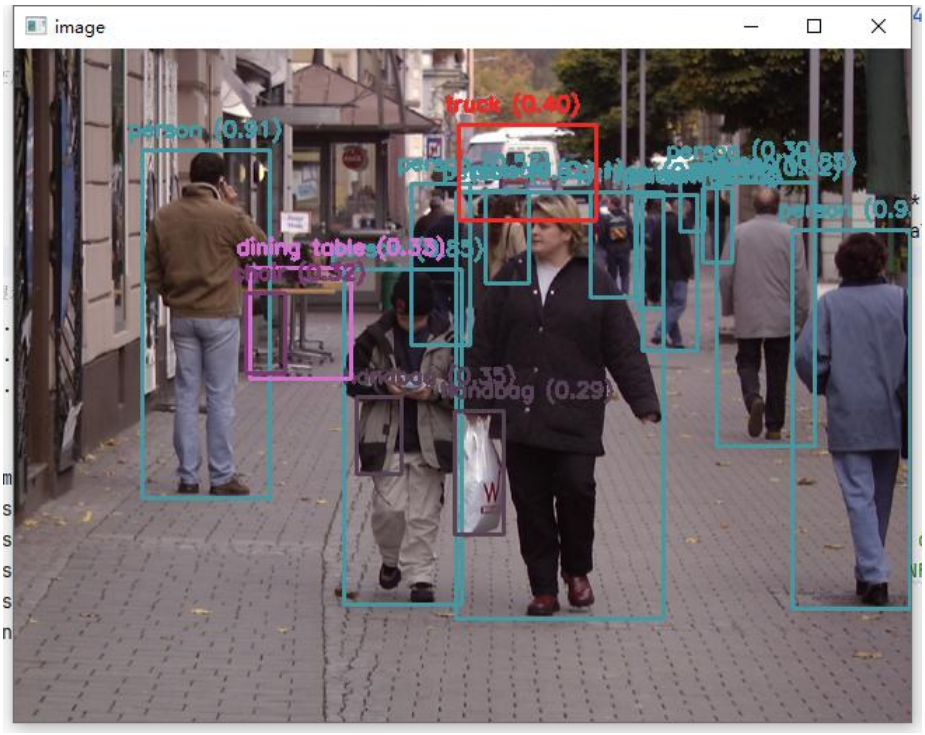
8. 绘制边界框和标签

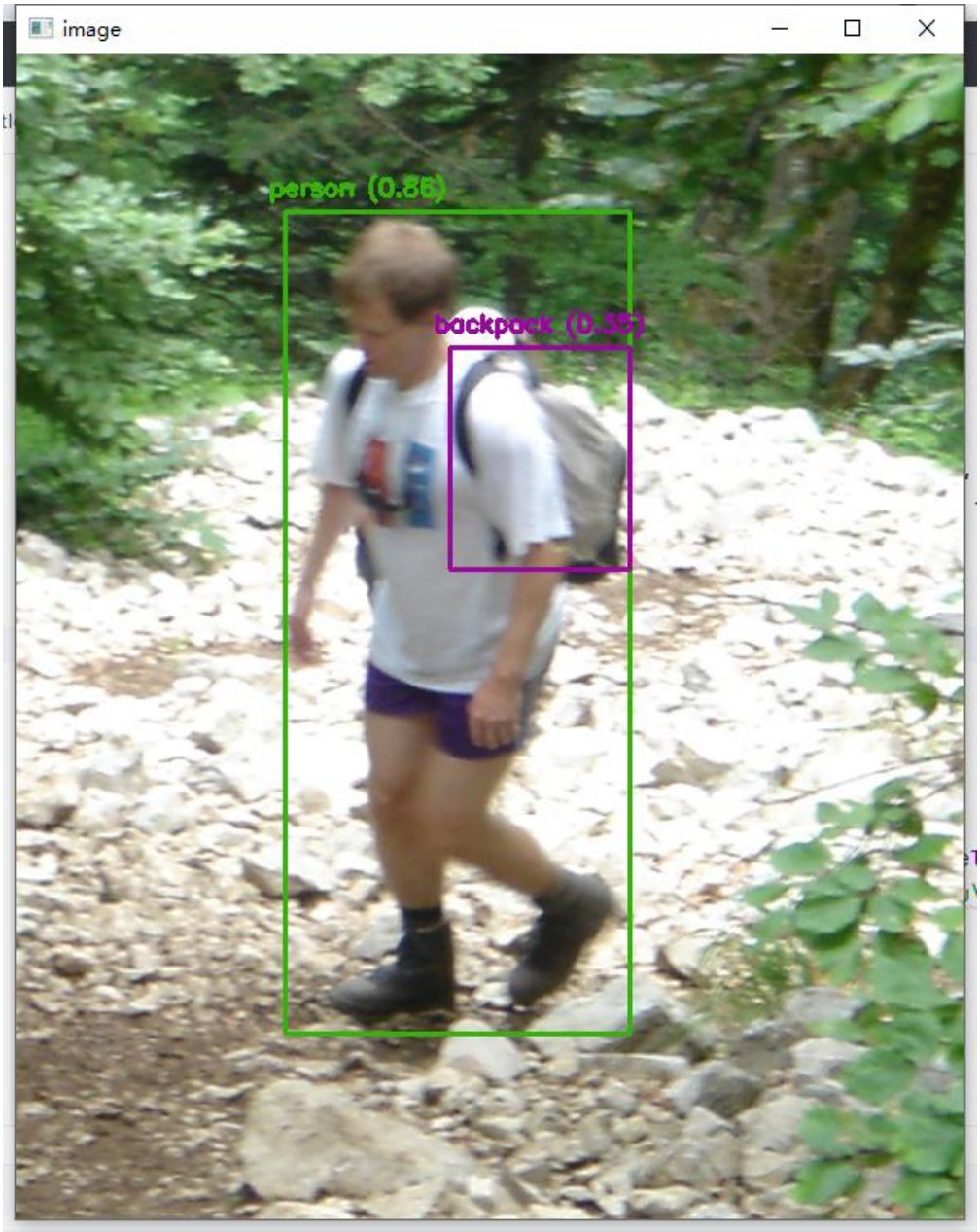
```python
# 在原图上绘制最终筛选出的边界框和类别标签
for i in range(len(result_boxes)):
    index = result_boxes[i]
    box = boxes[index]
    draw_bounding_box(original_image, class_ids[index], scores[index], round(box[0] * scale), round(box[1] * scale),
                      round((box[0] + box[2]) * scale), round((box[1] + box[3]) * scale))
```

9. 显示结果

```python
# 显示结果图像
cv2.imshow( winname: 'image', original_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

10. 执行代码





效果很不错！

四、出现的问题及解决方法

（详细记录实验过程中发生的故障和问题，进行故障分析，说明故障排除的过程及方法）

问题主要涉及 yolov8 的部署

1. yolov8 的源代码在不断更新，参考教程进行部署的过程中会发现部署失败。

解决：

选择提供了 yolov8 源代码的教程进行部署

YOLOv8(2023 年 8 月版本)安装配置！一条龙傻瓜式安装，遇到问题评论区提问

_yolov8 下载-CSDN 博客
【免费】yolov8(2023 年 8 月版本),已经下好 yolov8s.pt 和 yolov8n.pt_yolov8n.pt 文件怎么安装资源-CSDN 文库

2. 到处 onnx 模型的时候遇到如下报错

```
(yo8) PS D:\Content_Secu> python onnx.py
Ultralytics YOLOv8.1.34 🚀 Python-3.8.19 torch-2.2.1+cu118 CPU (11th Gen Intel Core(TM) i5-11400H 2.70GHz)
YOLOv8n summary (fused): 168 layers, 3151904 parameters, 0 gradients, 8.7 GFLOPs

  File "C:\Users\35083\.conda\envs\yo8\lib\site-packages\torch\utils\_contextlib.py", line 115, in decorate_context
    return func(*args, **kwargs)
  File "C:\Users\35083\.conda\envs\yo8\lib\site-packages\ultralytics\engine\exporter.py", line 287, in __call__
    f[2], _ = self.export_onnx()
  File "C:\Users\35083\.conda\envs\yo8\lib\site-packages\ultralytics\engine\exporter.py", line 138, in outer_func
    raise e
  File "C:\Users\35083\.conda\envs\yo8\lib\site-packages\ultralytics\engine\exporter.py", line 133, in outer_func
    f, model = inner_func(*args, **kwargs)
  File "C:\Users\35083\.conda\envs\yo8\lib\site-packages\ultralytics\engine\exporter.py", line 362, in export_onnx
    LOGGER.info(f"\n{prefix} starting export with onnx {onnx.__version__} opset {opset_version}...")
AttributeError: partially initialized module 'onnx' has no attribute '__version__' (most likely due to a circular import)
```

解决：文件不要以 onnx 命名

五、个人小结

（描述实验心得，可提出实验的改进意见）

通过深入学习 HOG 特征提取原理和 SVM 的工作机制，我不仅掌握了行人检测的技术细节，也对支持向量机（SVM）在图像分类问题上的应用有了更深的认识。此外，通过部署和应用 YOLO V8-PyTorch，我体会到了深度学习在目标检测领域的强大力量，尤其是 YOLO 系列算法的高效和精确。

六、教师评语及评分

教师签名：

年　　月　　日