《计算机视觉》实验报告

姓名：戴枫源 学号：**19120199**

实验六

一. 任务 1

a) 核心代码：

```
# *_*coding:utf-8 *_*

import os
import sys
import cv2
import logging
import numpy as np

def logger_init():
    logger = logging.getLogger("PedestranDetect")
    formatter = logging.Formatter('%(asctime)s %(levelname)-8s: %(message)s')
    console_handler = logging.StreamHandler(sys.stdout)
    console_handler.formatter = formatter
    logger.addHandler(console_handler)
    logger.setLevel(logging.INFO)

    return logger

def load_data_set(logger):
```

```
    logger.info('Checking data path!')
    pwd = os.getcwd()
    logger.info('Current path is:{}'.format(pwd))

    # 提取正样本
    pos_dir = os.path.join(pwd, 'Positive')
    if os.path.exists(pos_dir):
        logger.info('Positive data path is:{}'.format(pos_dir))
        pos = os.listdir(pos_dir)
        logger.info('Positive samples number:{}'.format(len(pos)))

    # 提取负样本
    neg_dir = os.path.join(pwd, 'Negative')
    if os.path.exists(neg_dir):
        logger.info('Negative data path is:{}'.format(neg_dir))
        neg = os.listdir(neg_dir)
        logger.info('Negative samples number:{}'.format(len(neg)))

    # 提取测试集
    test_dir = os.path.join(pwd, 'TestData')
    if os.path.exists(test_dir):
        logger.info('Test data path is:{}'.format(test_dir))
        test = os.listdir(test_dir)
        logger.info('Test samples number:{}'.format(len(test)))

    return pos, neg, test

def load_train_samples(pos, neg):
```

```
    pwd = os.getcwd()
    pos_dir = os.path.join(pwd, 'Positive')
    neg_dir = os.path.join(pwd, 'Negative')

    samples = []
    labels = []
    for f in pos:
        file_path = os.path.join(pos_dir, f)
        if os.path.exists(file_path):
            samples.append(file_path)
            labels.append(1.)

    for f in neg:
        file_path = os.path.join(neg_dir, f)
        if os.path.exists(file_path):
            samples.append(file_path)
            labels.append(-1.)

    # labels  要转换成 numpy 数组，类型为 np.int32
    labels = np.int32(labels)
    labels_len = len(pos) + len(neg)
    labels = np.resize(labels, (labels_len, 1))

    return samples, labels

def extract_hog(samples, logger):

    train = []
```

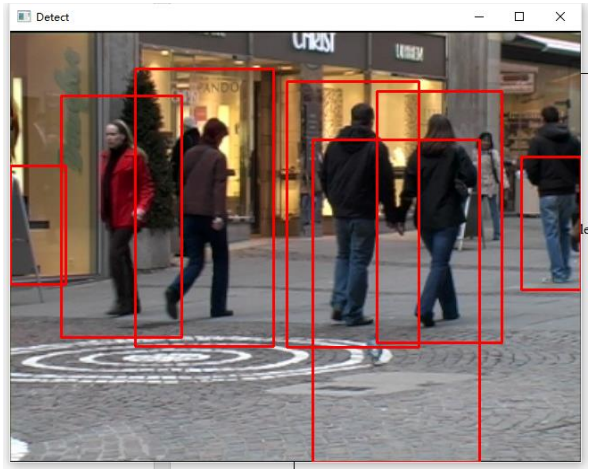
```
    logger.info('Extracting HOG Descriptors...')
    num = 0.
    total = len(samples)
    for f in samples:
        num += 1.
        logger.info('Processing {} {:2.1f}%'.format(f, num/total*100))
        hog = cv2.HOGDescriptor((64,128), (16,16), (8,8), (8,8), 9)
        # hog = cv2.HOGDescriptor()
        img = cv2.imread(f, -1)
        img = cv2.resize(img, (64,128))
        descriptors = hog.compute(img)
        logger.info('hog feature descriptor size: {}'.format(descriptors.shape))       #
(3780, 1)
        train.append(descriptors)

    train = np.float32(train)
    train = np.resize(train, (total, 3780))

    return train

def get_svm_detector(svm):

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

def train_svm(train, labels, logger):
```

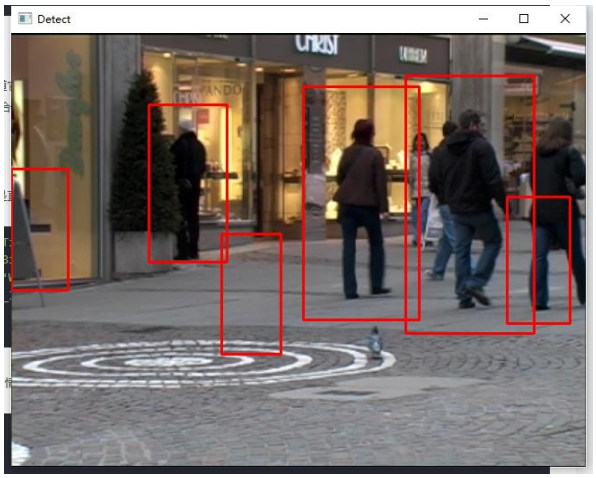
```
    logger.info('Configuring SVM classifier.')
    svm = cv2.ml.SVM_create()
    svm.setCoef0(0.0)
    svm.setDegree(3)
    criteria = (cv2.TERM_CRITERIA_MAX_ITER + cv2.TERM_CRITERIA_EPS,
1000, 1e-3)
    svm.setTermCriteria(criteria)
    svm.setGamma(0)
    svm.setKernel(cv2.ml.SVM_LINEAR)
    svm.setNu(0.5)
    svm.setP(0.1)    # for EPSILON_SVR, epsilon in loss function?
    svm.setC(0.01)    # From paper, soft classifier
    svm.setType(cv2.ml.SVM_EPS_SVR)

    logger.info('Starting training svm.')
    svm.train(train, cv2.ml.ROW_SAMPLE, labels)
    logger.info('Training done.')

    pwd = os.getcwd()
    model_path = os.path.join(pwd, 'svm.xml')
    svm.save(model_path)
    logger.info('Trained SVM classifier is saved as: {}'.format(model_path))

    return get_svm_detector(svm)

def test_hog_detect(test, svm_detector, logger):

    hog = cv2.HOGDescriptor()
    hog.setSVMDetector(svm_detector)
```

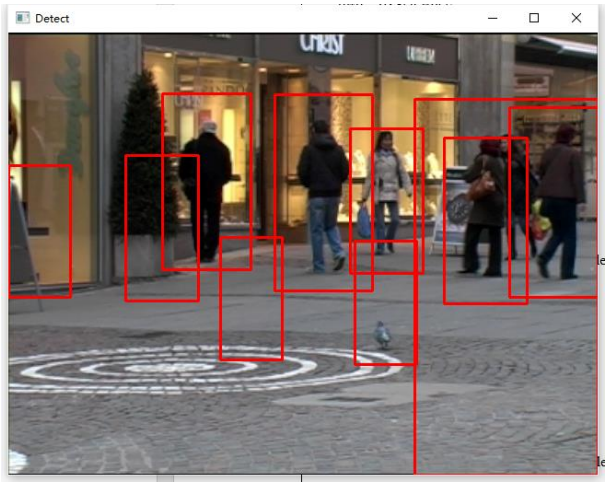
```
    pwd = os.getcwd()
    test_dir = os.path.join(pwd, 'TestData')
    cv2.namedWindow('Detect')
    for f in test:
        file_path = os.path.join(test_dir, f)
        logger.info('Processing {}'.format(file_path))
        img = cv2.imread(file_path)
        rects, _ = hog.detectMultiScale(img, winStride=(4,4), padding=(8,8), scale=1.05)
        for (x,y,w,h) in rects:
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255), 2)
        cv2.imshow('Detect', img)
        c = cv2.waitKey(0) & 0xff
        if c == 27:
            break
    cv2.destroyAllWindows()



if __name__ == '__main__':
    logger = logger_init()
    pos, neg, test = load_data_set(logger=logger)
    samples, labels = load_train_samples(pos, neg)
    train = extract_hog(samples, logger=logger)
    logger.info('Size of feature vectors of samples: {}'.format(train.shape))
    logger.info('Size of labels of samples: {}'.format(labels.shape))
    svm_detector = train_svm(train, labels, logger=logger)
    test_hog_detect(test, svm_detector, logger)
```

b) 实验结果截图

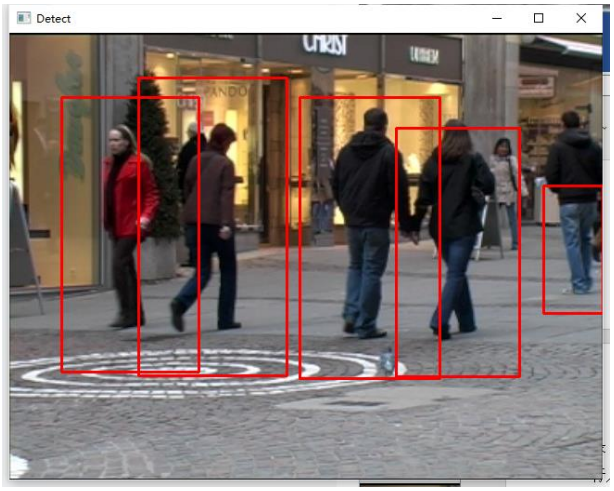
用自己训练的 svm

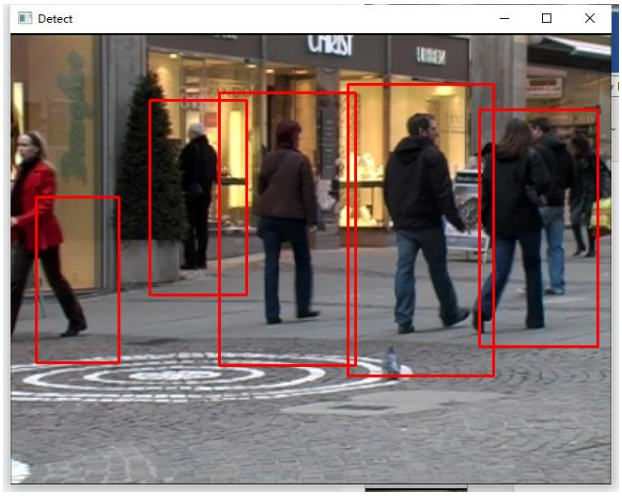


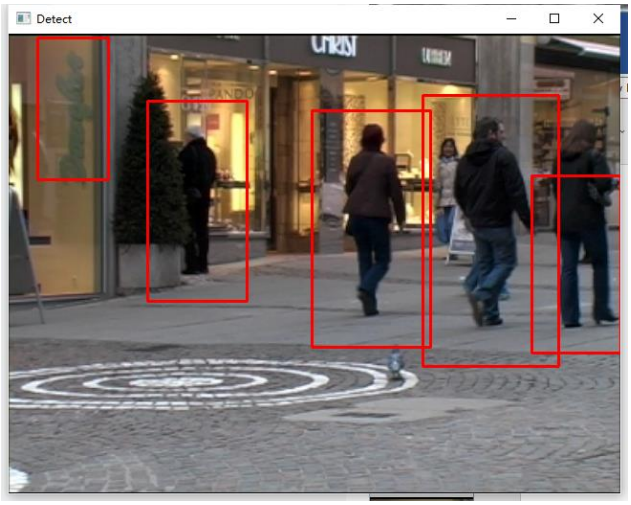




用 **openCV 自带的检测器**







c) 实验小结

本实验的训练数据集是在同目录文件夹 Negative 和 Positive 下，而测试数据在同目

录 TestData 下，其来源均来自网络。在进行行人检测的时候，经常会判定一些背景作为行人，这是因为负向数据集太小，没有足够的各种背景图片进行训练导致的，而行人检测基本比较准确，只是在行人重叠时模型会进行误判。总结而言是训练集太小，模型训练得不是很完善。

在调参方面，经过几次试验，我选用线性核函数，参数 C 设置为 0.01，其他参数均采用了推荐的值，训练过后效果最佳，其效果如上面前三张图片所示。而 OpenCV 中也有自带的行人检测器，只需将代码中的 hog.setSVMDetector(svm_detector)换成 hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())，运行完毕后的效果如上面后三张图片所示，可见效果比自己训练的好得多。

二．任务 2

a) 核心代码：

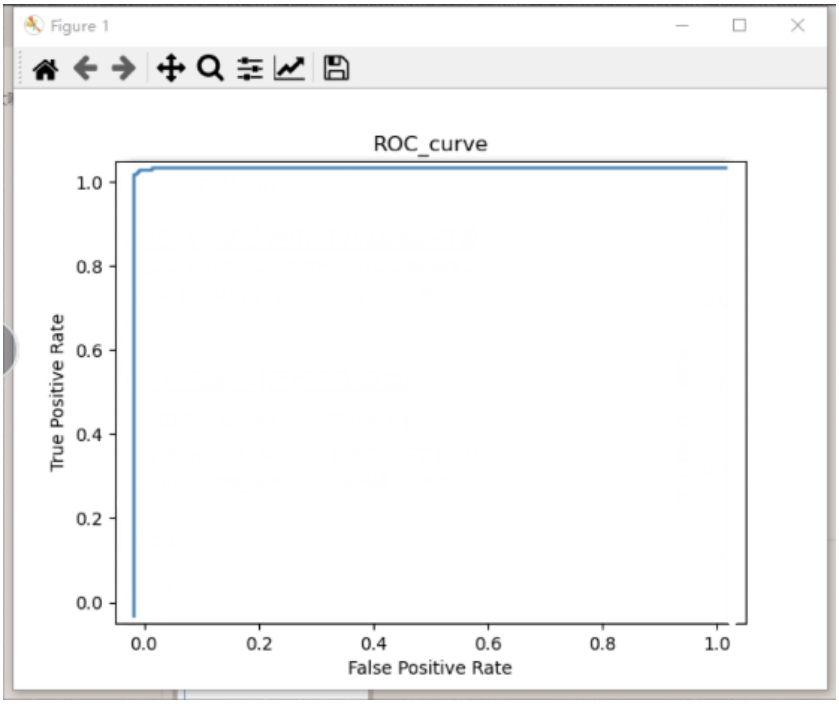
绘制 ROC 曲线函数

```
def draw_roc(svm_detector):
    hog = cv2.HOGDescriptor()
    hog.setSVMDetector(svm_detector)
    pwd = os.getcwd()

    test_dir = []
    # 提取正样本
    pos_dir = os.path.join(pwd, 'Positive')
    pos_files = os.listdir(pos_dir)
    print(pos_files)
    shuffle(pos_files)
    for file in pos_files[:50]:
        test_dir.append(os.path.join(pos_dir, file))

    # 提取负样本
    neg_dir = os.path.join(pwd, 'Negative')
    neg_files = os.listdir(neg_dir)
```

```
    shuffle(neg_files)
    for file in neg_files[:50]:
        test_dir.append(os.path.join(neg_dir, file))



    label1 = [1 for _ in range(50)]
    label2 = [0 for _ in range(50)]
    label_dir = label1+label2
    # test  包含测试图片和标签
    test = list(zip(test_dir,label_dir))
    shuffle(test)

    test_dir[:],label_dir[:]= zip(*test)
    res = []
    for i in range(100):
        img = cv2.imread(test_dir[i])
        rects, _ = hog.detectMultiScale(img, winStride=(4,4), padding=(8,8),
scale=1.05)
        print(rects)
        if rects != ():
            res.append(1)
        else:
            res.append(0)
    fpr, tpr, thresholds= roc_curve(res, label_dir, pos_label=1)

    plt.plot(fpr, tpr)
    plt.title('ROC_curve')
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
```

```
    plt.show()
```

b) 实验结果截图



c) 实验小结

本任务的主要思路是根据任务 1 训练出来的 svm，从训练集中随机抽取正向图片和负向图片各二十张，进行分类处理。如果能在图片中找到行人则置该图片的标签为 1，否则为 0，然后根据分类结果绘制 ROC 曲线。严格来说训练集的数据是不能作为测试集使用的，但是因为我从训练集中选用的数据比较小，因此可以得到一个相对误差并不是很大的结果。