《计算机视觉》实验报告

姓名：孔馨怡 学号：22122128

实验 6 行人检测

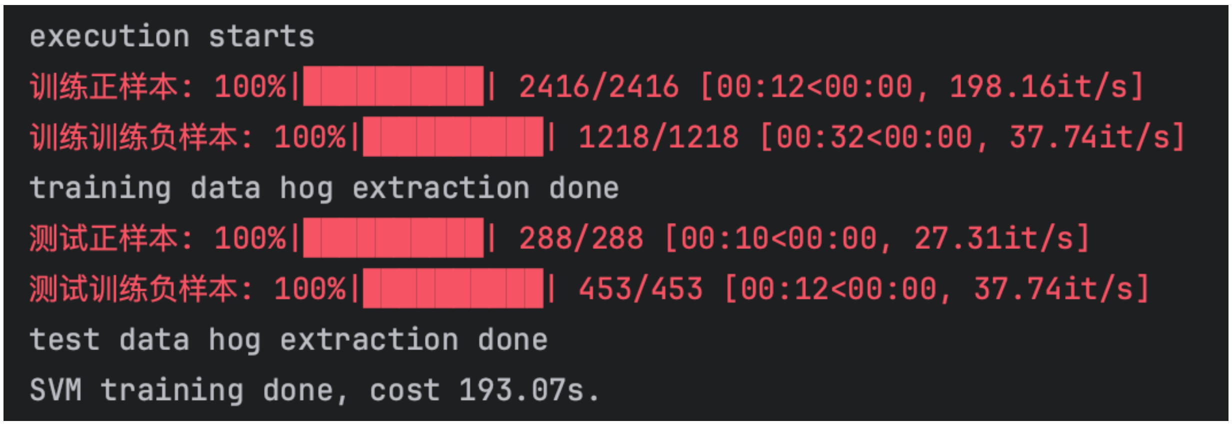
一. 任务 1

a) 核心代码：

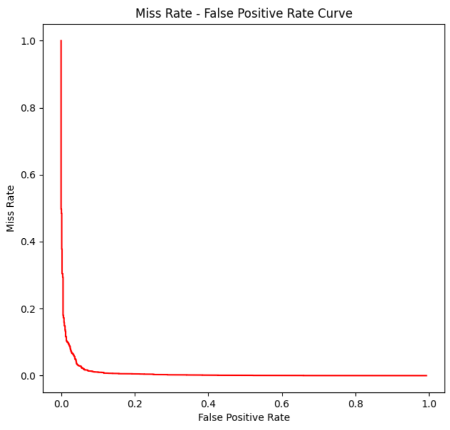
```
def extract_hog_feature(img):
    # 提取单个图像 img 的 HOG 特征
    return hog(
        img,
        orientations=9,
        pixels_per_cell=(16, 16),
        cells_per_block=(2, 2),
        block_norm='L2-Hys',
        visualize=False
    ).astype('float32')

def read_images(pos_dir, neg_dir,
                neg_area_count, description):
    # 读取图片，提取样本 HOG 特征。
    pos_img_files = os.listdir(pos_dir)
    # 正样本文件列表
    neg_img_files = os.listdir(neg_dir)
    # 负样本文件列表

    area_width = 64
    area_height = 128

    x = []  # 图片的 HOG 特征
    y = []  # 图片的分类

    for pos_file in tqdm(pos_img_files,
                         desc=f'{description}正样本'):
        # 读取所有正样本
        pos_path = os.path.join(pos_dir, pos_file)
        pos_img = imread(pos_path, as_gray=True)
        img_height, img_width = pos_img.shape
```

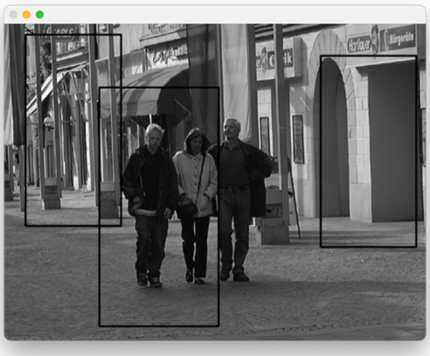
```python
        clip_left = (img_width - area_width) // 2
        clip_top = (img_height - area_height) // 2
        pos_center = clip_image(pos_img,
                                clip_left, clip_top, area_width, area_height)
        # 截取中间部分
        hog_feature = extract_hog_feature(
            pos_center)  # 提取 HOG 特征
        x.append(hog_feature)  # 加入 HOG 向量
        y.append(1)  # 1 代表正类

    for neg_file in tqdm(neg_img_files,
                         desc=f'{description}训练负样本'):
        neg_path = os.path.join(neg_dir, neg_file)
        neg_img = imread(neg_path, as_gray=True)
        img_height, img_width = neg_img.shape
        left_max = img_width - area_width
        top_max = img_height - area_height
        for _ in range(neg_area_count):
            # 随机截取 neg_area_count 个区域
            left = random.randint(0, left_max)
            top = random.randint(0, top_max)
            clipped_area = clip_image(neg_img,
                                      left, top, area_width, area_height)
            # 截取的区域
            hog_feature = extract_hog_feature(
                clipped_area)  # 提取 HOG 特征
            x.append(hog_feature)
            y.append(0)
    return x, y
def train_SVM(x, y):
    # 训练 SVM。
    SVM = SVC(
        tol=1e-6,
        C=0.01,
        max_iter=-1,
        gamma='auto',
        kernel='rbf',
        probability=True
    )  # 创建 SVM 实例
    SVM.fit(x, y)  # 进行训练
    return SVM


def test_SVM(SVM, test_data, show_stats=False):
    # 测试训练好的 SVM
    hog_features = test_data[0]  # 测试数据的 HOG 特征
    labels = test_data[1]  # 数据标签（0=不是人，1=是人）
    prob = SVM.predict_proba(hog_features)[:, 1]
    if show_stats:
        sorted_indices = np.argsort(
            prob, kind="mergesort")[::-1].astype(int) # 转化为 int 类型
        labels = np.array(labels)
        labels = labels[sorted_indices]
        prob = prob[sorted_indices]
        distinct_value_indices = np.where(np.diff(prob))[0]
        threshold_idxs = np.r_[
            distinct_value_indices, labels.size - 1]
        tps = np.cumsum(labels)[threshold_idxs]
        fps = 1 + threshold_idxs - tps
        num_positive = tps[-1]
        recall = tps / num_positive
        # 查全率就是在所有正例中查出了多少真正例
        miss = 1 - recall  # 计算 miss
        num_negative = fps[-1]  # 负例个数
        fpr = fps / num_negative
        # 假阳性率（false positive rate）
        plt.plot(miss, fpr, color='red')
        plt.xlabel('False Positive Rate')
        plt.ylabel('Miss Rate')
        plt.title('Miss Rate - '
                  'False Positive Rate Curve')
        plt.show()
    AUC = metrics.roc_auc_score(labels, prob)
    return AUC

def non_maximum_suppression(pos_box_list, pos_prob,
                            IoU_threshold=0.4):
   #  非极大值抑制（NMS）。
    result = []
    for box1, prob1 in zip(pos_box_list, pos_prob):
        discard = False  # 是否舍弃 box1
        for box2, prob2 in zip(
                pos_box_list, pos_prob):
            if intersection_over_union(
                    box1, box2) > IoU_threshold:
                # IoU 大于阈值
                if prob2 > prob1:  # 舍弃置信度较小的
                    discard = True
                break
        if not discard:  # 未舍弃 box1
            result.append(box1)  # 加入结果列表
    return result


def detect_pedestrian(SVM, filename, show_img=False,
                      threshold=0.99, area_width=64, area_height=128,
                      min_width=48, width_scale=1.25, coord_step=16,
                      ratio=2):
    # 用 SVM 检测 file 文件中的行人，采用非极大值抑制（NMS）
    box_list = []  # 行人边框列表
    hog_list = []  # HOG 特征列表
    with open(filename, 'rb') as file:
        img = imread(file, as_gray=True)
        img_height, img_width = img.shape
        width = min_width
        height = int(width * ratio)
        while width < img_width and height < img_height:
            for left in range(0, img_width - width,
                              coord_step):
                for top in range(0, img_height - height,
                                 coord_step):
                    patch = clip_image(img, left, top,
                                       width, height)
                    resized = resize(patch,
                                     (area_height, area_width))
                    # 缩放图片
                    hog_feature = extract_hog_feature(
                        resized)  # 提取 HOG 特征
                    box_list.append((left, top,
                                     width, height))
                    hog_list.append(hog_feature)
            width = int(width * width_scale)
            height = width * ratio
        prob = SVM.predict_proba(hog_list)[:, 1]
        # 用 SVM 模型进行判断
        mask = (prob >= threshold)
        # 布尔数组, mask[i]代表 prob[i]是否等于阈值
        pos_box_list = np.array(box_list)[mask]
        # 含有人的框
        pos_prob = prob[mask]  # 对应的预测概率
        box_list_after_NMS = non_maximum_suppression(
            pos_box_list, pos_prob)
        # NMS 处理之后的框列表
        if show_img:
            shown_img = np.array(img)
            for box in box_list_after_NMS:
                shown_img = rectangle(shown_img,
                                      pt1=(box[0], box[1]),
                                      pt2=(box[0] + box[2],
                                           box[1] + box[3]),
                                      color=(0, 0, 0),
                                      thickness=2)
            imshow('', shown_img)
            waitKey(0)
        return box_list_after_NMS
```

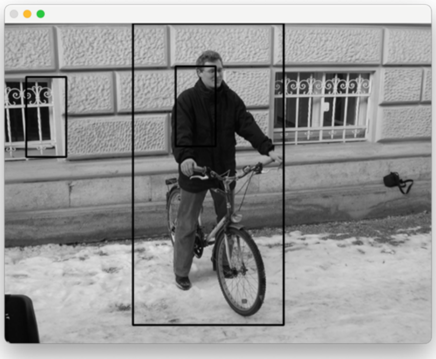
b) 实验结果截图

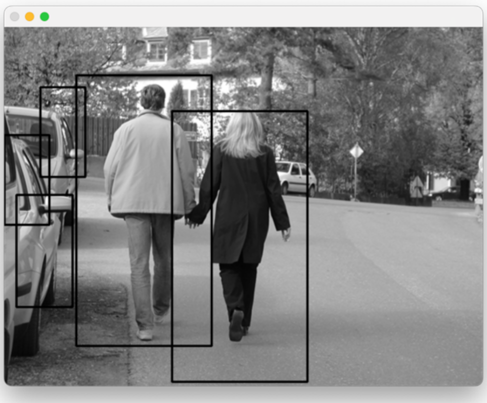


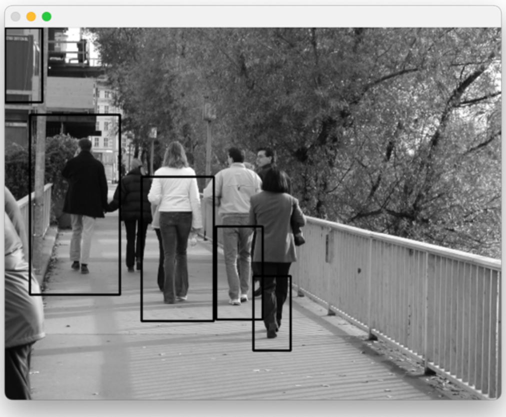
```
execution starts
训练正样本: 100%|██████████| 2416/2416 [00:12<00:00, 198.16it/s]
训练训练负样本: 100%|██████████| 1218/1218 [00:32<00:00, 37.74it/s]
training data hog extraction done
测试正样本: 100%|██████████| 288/288 [00:10<00:00, 27.31it/s]
测试训练负样本: 100%|██████████| 453/453 [00:12<00:00, 37.74it/s]
test data hog extraction done
SVM training done, cost 193.07s.
```











c) 实验小结

下面是一些实验中踩的坑和心得：

在实验过程中，我遇到了一个棘手的问题：当我尝试对标签 `labels` 使用 `sorted_indices` 进行索引操作时，出现了 `TypeError: only integer scalar arrays can be converted to a scalar index` 错误。虽然我已经将 `labels` 转换为了 NumPy 数组，但这个错误仍然阻碍了我的进展。问题的根源可能在于 `sorted_indices` 中包含了非整数标量的数组，导致无法进行索引操作。我将 `sorted_indices` 也转换为整数数组，并确保其中的索引都是整数。我使用了 `astype(int)` 方法将 `sorted_indices` 转换为整数数组