《计算机视觉》实验报告

姓名：王子棠 学号：21122897

实验 6

一. 任务 1

a) 核心代码：

```
# 提取单个图像的 HOG 特征
def hog_descriptor(image):
    if (image.max()-image.min()) != 0:
        image = (image - image.min()) / (image.max() - image.min())
        image *= 255
        image = image.astype(np.uint8)
    hog = cv2.HOGDescriptor((64, 128), (16, 16), (8, 8), (8, 8), 9)
    hog_feature = hog.compute(image)
    return hog_feature
# 数据集读取，对于正样本，截取中间 64×128 大小的部分提取 HOG 特征。对于负样本，随机截取 10 个 64×128 的部分，提取 HOG 特征。
# 读取正样本，设置正样本标签为 1
for i in range(len(poslist)):
    posimg = io.imread(os.path.join(pos_dir, poslist[i]))
    posimg = cv2.cvtColor(posimg, cv2.COLOR_RGBA2BGR)
    posimg = cv2.resize(posimg, (64, 128), interpolation=cv2.INTER_NEAREST)
    pos_hog = hog_descriptor(posimg)
    hog_list.append(pos_hog)
    label_list.append(1) # 1 为正样本
```

```
# 读取负样本，标签为 0
for i in range(len(neglist)):
    negimg = io.imread(os.path.join(neg_dir, neglist[i]))
    negimg = cv2.cvtColor(negimg, cv2.COLOR_RGBA2BGR)
    # 在每张 neg 图像中截取 10 张标准大小的图片作为负样本
    for j in range(10):
        y = int(random.random() * (negimg.shape[0] - 128))
        x = int(random.random() * (negimg.shape[1] - 64))
        negimgs = negimg[y:y + 128, x:x + 64]
                                    negimgs = cv2.resize(negimgs, (64, 128),
interpolation=cv2.INTER_NEAREST)
        neg_hog = hog_descriptor(negimgs)
        hog_list.append(neg_hog)
        label_list.append(0)

# 训练 SVM
clf = SVC(
    C = 1.0,  # 正则化参数/惩罚系数。当 C 越大时，分类器的准确性越高，所以容错率越低，泛化能力就变差。当 C 越小时，分类器的准确性降低，但容错率增大，泛化能力越强
    gamma='auto',
    kernel='rbf',
    probability=True
)
clf.fit(hog_list.squeeze(), label_list.squeeze())
joblib.dump(clf, "trained_svm.model") # 保存训练好的模型

# ROC 曲线
prob = clf.predict_proba(test_hog.squeeze())[:, 1]
fpr, tpr, thresholds_2 = metrics.roc_curve(test_label.squeeze(), prob, pos_label=1)
```

```
plt.figure(figsize=(10, 8))
plt.plot(fpr, tpr, c='red')
plt.scatter(fpr, tpr, c='blue')
plt.xlabel("FPR", fontdict={'size': 16})
plt.ylabel("TPR", fontdict={'size': 16})
plt.title("ROC_curve", fontdict={'size': 20})
plt.savefig('ROC.png')
AUC=metrics.roc_auc_score(test_label.squeeze(), prob)
print(AUC)

# 在图像上给行人画框，NMS 算法，去除重叠的边界框
def mynms(box_list,prob_list,threshold=0.4):
    x1 = box_list[:,0]
    y1 = box_list[:,1]
    x2 = box_list[:,2]
    y2 = box_list[:,3]
    areas = (x2-x1+1)*(y2-y1+1)
    box_result = []
    flag = []
    index = prob_list.argsort()[::-1] # 从大到小排序
    while index.size>0:
        i = index[0]
        flag.append(i)
        x11 = np.maximum(x1[i], x1[index[1:]])
        y11 = np.maximum(y1[i], y1[index[1:]])
        x22 = np.minimum(x2[i], x2[index[1:]])
        y22 = np.minimum(y2[i], y2[index[1:]])
        w = np.maximum(0, x22 - x11 + 1)
        h = np.maximum(0, y22 - y11 + 1)
        overlaps = w * h
```

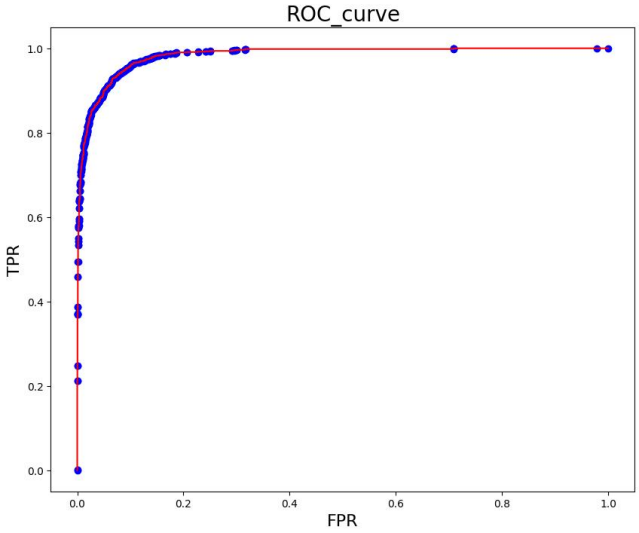
```
        ious = overlaps / (areas[i] + areas[index[1:]] - overlaps)
        #idx = np.where(ious < threshold)[0]
                index = np.delete(index,np.concatenate(([0],np.where(ious < threshold)[0])))
        #index = index[idx + 1]
    return box_list[flag].astype("int")

# 测试集图片行人检测
for i in range(101,len(imglist)):
    img = io.imread(osp.join('INRIADATA/original_images/test/pos', imglist[i]))
    img = cv2.cvtColor(img, cv2.COLOR_RGBA2BGR)
    h,w,c= img.shape
    patch_list = []
    hog_feature = []
    box_list = []
    for j in range(minscale,maxscale+1,1):
        winsize = [j*64,j*128]
        for m in range(0,h-winsize[1],20):
            for n in range(0,w-winsize[0],20):
                patch = img[m:m+winsize[1],n:n+winsize[0]]
                        patch = cv2.resize(patch, (64,128), interpolation = cv2.INTER_NEAREST)
                boxcoord = (m,n,m+winsize[1],n+winsize[0])
                hogfea = hog_descriptor(patch)
                hog_feature.append(hogfea)
                box_list.append(boxcoord)
                patch_list.append(patch)
    hog_feature = np.array(hog_feature).squeeze()
    box_list = np.array(box_list)
    prob = clf.predict_proba(hog_feature)[:, 1]
```

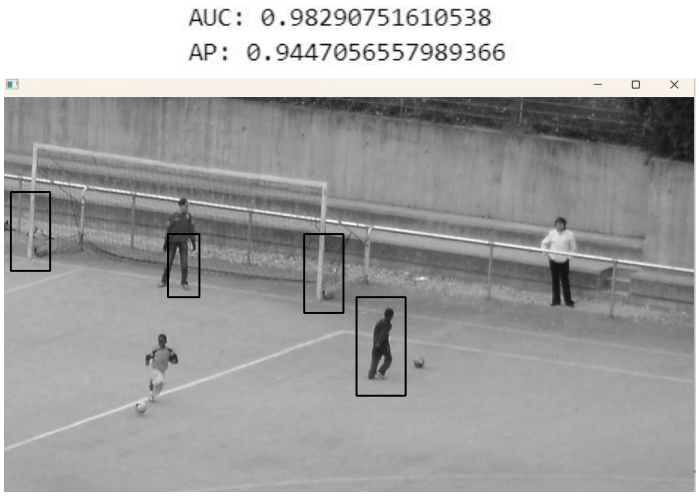
```
mask = (prob>= 0.99)
box_list = box_list[mask]
prob = prob[mask]
boxzhong = mynms(box_list,prob)
for k in range(len(boxzhong)):
        cv2.rectangle(img, (boxzhong[k][1], boxzhong[k][0]), (boxzhong[k][3],
boxzhong[k][2]),(0, 0, 255),3)
cv2.imwrite('result/'+imglist[i]+".jpg",img)
```

b) 实验结果截图

• ROC 曲线如下图所示，AUC=0.983



• 检测结果



c) 实验小结

本次实验通过 HOG 和 SVM 实现了行人检测，从最后测试的结果可以看出 hog 和 svm 的结合很容易把柱状物体识别成行人，这可能是它们的 HOG 特征比较相似的缘故。并且在 NMS 算法中，设置 IoU 阈值为 0.4，用于去除重复的边界框。此外，在 SVM 训练中，将正则化参数设置为 1.0，使得检测效果略有提升。