Nguyễn Hoàng Thanh – 2151013087

LAB 5: PEDESTRIAN RECOGNITION

Assignment 1:

Code:

```
    from skimage.feature import hog
import joblib, glob, os, cv2
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn import svm
import numpy as np
from sklearn.preprocessing import LabelEncoder
from matplotlib import pyplot as plt
from sklearn.metrics import confusion_matrix,
classification_report

# Training model
X = []
Y = []

positive_img_path = "datasets/positive"
negative_img_path = "datasets/negative"

# load positive
for filename in glob.glob(os.path.join(positive_img_path,
"*.png")):
    file_load = cv2.imread(filename, 0)
    file_load = cv2.resize(file_load, (96, 160))
    hog_features = hog(
        file_load,
        orientations=9,
        pixels_per_cell=(8, 8),
        cells_per_block=(2, 2),
        visualize=False,
        block_norm="L2-Hys",
```

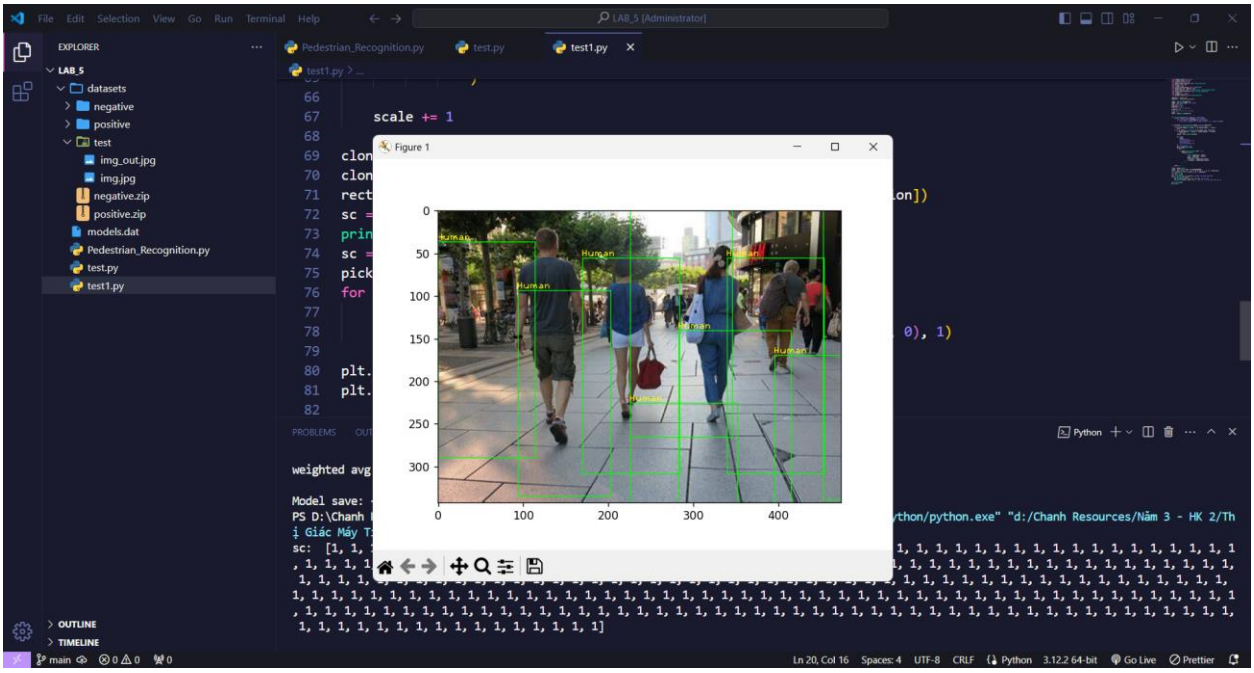
```
        transform_sqrt=True,
    )
    X.append(hog_features)  # Append the HOG features
    Y.append(1)

# load negative
for filename in glob.glob(os.path.join(negative_img_path,
"*.png")):
    file_load = cv2.imread(filename, 0)
    file_load = cv2.resize(file_load, (96, 160))
    hog_features = hog(
        file_load,
        orientations=9,
        pixels_per_cell=(8, 8),
        cells_per_block=(2, 2),
        visualize=False,
        block_norm="L2-Hys",
        transform_sqrt=True,
    )
    X.append(hog_features)  # Append the HOG features
    Y.append(0)


X = np.float32(X)
Y = np.array(Y)

X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size=0.2
)
print("Train Data: ", len(X_train))
print("Train Labels (1, 0)", len(y_train))

model = SVC (kernel='rbf')
model.fit(X_train, y_train)

y_predict = model.predict(X_test)
```

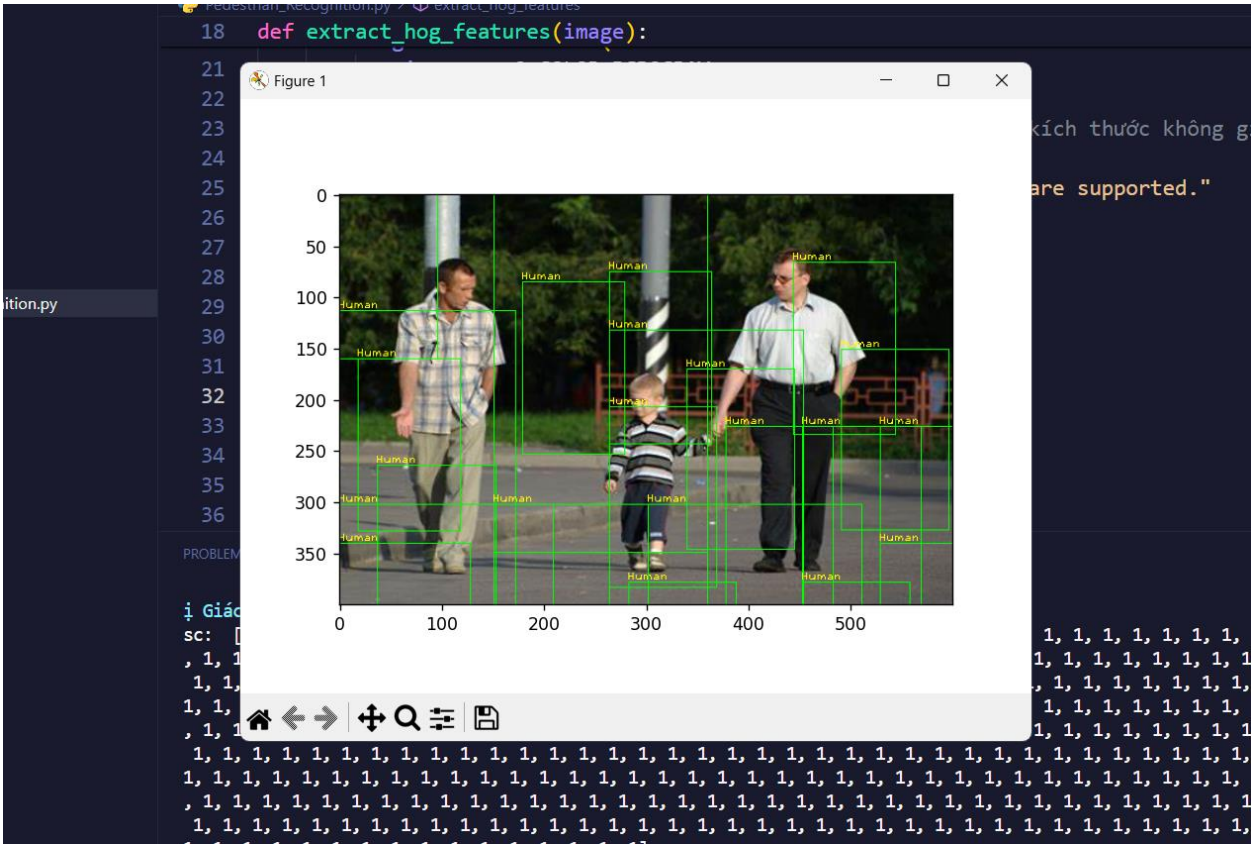
```
conf_matrix = confusion_matrix(y_test, y_predict)
print("Confusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_report(y_test, y_predict))

joblib.dump(model, "models.dat")
print("Model save: {}'.format ('models.dat)")
import imutils
from skimage import color
from skimage.transform import pyramid_gaussian

modelFile = "models.dat"
inputFile = "datasets/test/img1.jfif"

image = cv2.imread(inputFile)
image = cv2.resize(image, (600, 400))
size = (96, 160)
step_size = (9, 9)
downscale = 1.05
# List to store the detection
detection = []
# The current scale off the image
scale = 0
model = joblib.load(modelFile)


def sliding_window(image, window_size, step_size):
    for y in range(0, image.shape[0], step_size[1]):
        for x in range(0, image.shape[1], step_size[0]):
            yield (x, y, image[y : y + window_size[1], x
: x + window_size[0]])
```

```
for im_scaled in pyramid_gaussian(image,
downscale=downscale):
    # The lisst contains detection at the current scale
    if im_scaled.shape[0] < size[1] or im_scaled.shape[1]
< size[0]:
        break
    for x, y, window in sliding_window(im_scaled, size,
step_size):
        if window.shape[0] != size[1] or window.shape[1]
!= size[0]:
            continue
        window = color.rgb2gray(window)

        fd = hog(
            window,
            orientations=9,
            pixels_per_cell=(8, 8),
            visualize=False,
            cells_per_block=(2, 2),
        )
        fd = fd.reshape(1, -1)
        pred = model.predict(fd)
        if pred == 1:

            if model.decision_function(fd) > 0.5:
                detection.append(
                    (
                        int(x * (downscale * scale)),
                        int(y * (downscale * scale)),
                        model.predict(fd),
                        int(size[0] *
(downscale**scale)),
                        int(size[1] *
(downscale**scale)),
                    )
                )
```

```
    scale += 1

clone = image.copy()
clone = cv2.cvtColor(clone, cv2.COLOR_BGR2RGB)
rects = np.array([[x, y, x + w, y + h] for [x, y, _, w,
h] in detection])
sc = [score[0] for (x, y, score, w, h) in detection]
print("sc: ", sc)
sc = np.array(sc)
pick = non_max_suppression(rects, probs=sc,
overlapThresh=0.45)
for x1, y1, x2, y2 in pick:
    cv2.rectangle(clone, (x1, y1), (x2, y2), (0, 255, 0))
    cv2.putText(clone, "Human", (x1 - 2, y1 - 2), 1,
0.75, (255, 255, 0), 1)

plt.imshow(clone)
plt.show()
```

Result:





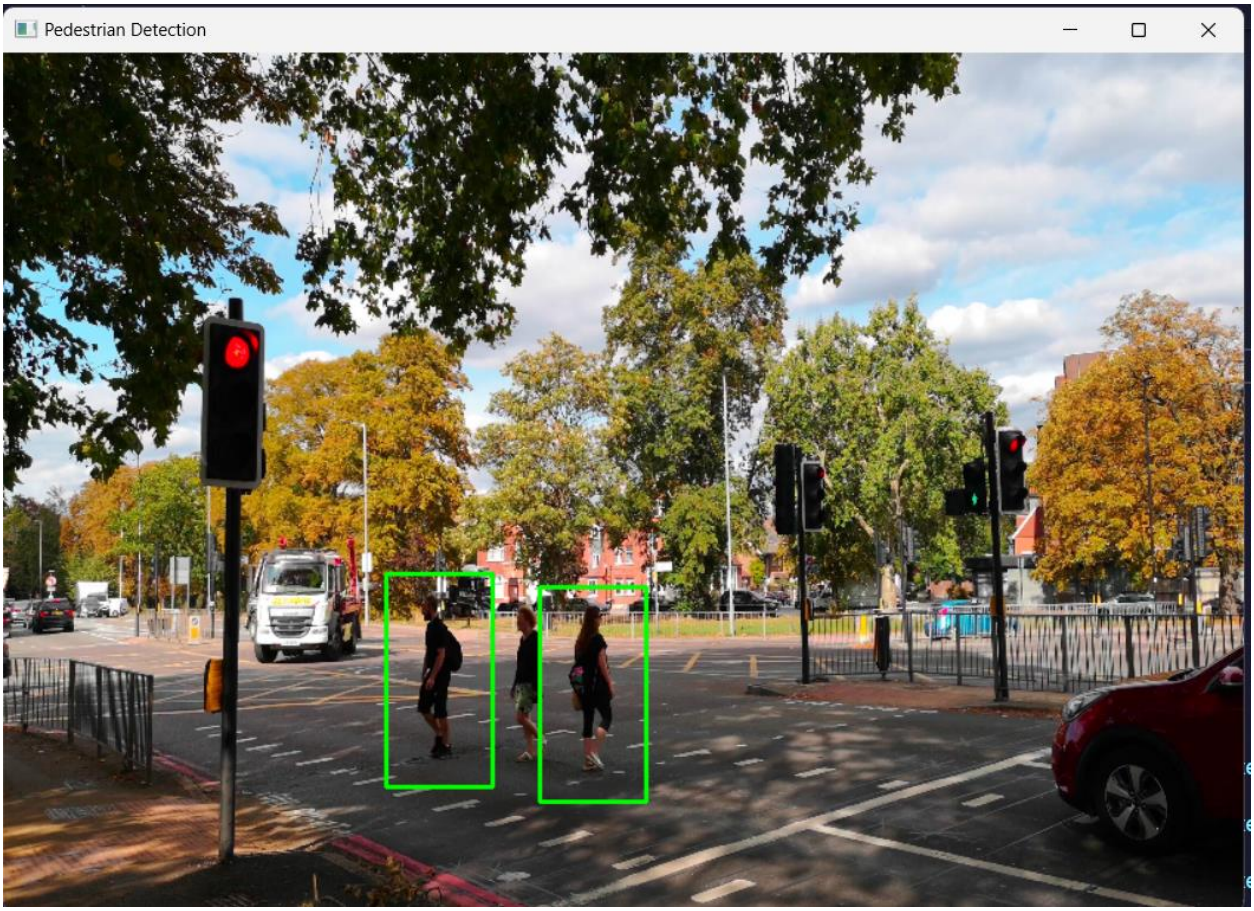
Assignment 2:

Code:

```
import numpy as np
import cv2

hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDete
ctor())
cv2.startWindowThread()

input_video_path = "datasets/test/video2.mp4"

cap = cv2.VideoCapture(input_video_path)

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    if not ret:
        break

    frame = cv2.resize(frame, (840, 580))

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    boxes, weights = hog.detectMultiScale(frame,
winStride=(8, 8))

    boxes = np.array([[x, y, x + w, y + h] for (x, y, w,
h) in boxes])

    for xA, yA, xB, yB in boxes:
        cv2.rectangle(frame, (xA, yA), (xB, yB), (0, 255,
0), 2)

    cv2.imshow("Pedestrian Detection", frame)
```

```
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
cv2.waitKey(1)
```

Result:


Pedestrian Detection


Pedestrian Detection