# Prerequisites

Libraries Required:  
- `opencv-python-headless`: Used for computer vision tasks, specifically for image processing and feature extraction.  
- `numpy`: For numerical operations and handling arrays.  
- `matplotlib`: Used for visualizing images and results.  
- `tensorflow`: For loading pre-trained object detection models (SSD MobileNet V2).  
- `skimage`: To use the HOG feature extraction method.

Install dependencies if necessary:  
```bash  
pip install opencv-python-headless numpy matplotlib tensorflow scikit-image  
```

# Step-by-Step Breakdown

## 1. HOG Feature Extraction with SVM for Pedestrian Detection

Objective: Use HOG features with SVM to detect pedestrians.  
Functions:  
- `cv2.imread()`: Load an image.  
- `cv2.cvtColor()`: Convert the image to grayscale.  
- `hog()`: Extract HOG features and visualize them.  
- `cv2.HOGDescriptor()`: Initialize a pre-trained HOG+SVM descriptor for pedestrian detection.  
- `detectMultiScale()`: Detect pedestrians in the image using HOG.  
- `cv2.rectangle()`: Draw bounding boxes around detected objects.  
  
Example Code:  
```python  
# Load and preprocess image  
image = cv2.imread('/content/image1.jpeg')  
gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
features, hog\_image = hog(gray\_image, orientations=9, pixels\_per\_cell=(8, 8), cells\_per\_block=(2, 2), visualize=True)  
```

## 2. YOLOv3 Object Detection

Objective: Detect multiple classes of objects using YOLOv3.  
Functions:  
- `wget`: Download pre-trained weights, configuration, and class labels.  
- `cv2.dnn.readNet()`: Load YOLO model.  
- `cv2.dnn.blobFromImage()`: Preprocess the image for YOLO input.  
- `net.forward()`: Perform a forward pass and extract detections.  
- `cv2.rectangle()`: Draw bounding boxes around detected objects with class labels.  
  
Example Code:  
```python  
# Download YOLO files  
!wget https://pjreddie.com/media/files/yolov3.weights  
!wget https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov3.cfg  
!wget https://raw.githubusercontent.com/pjreddie/darknet/master/data/coco.names  
  
# Load YOLO model  
net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')  
```

## 3. SSD MobileNet V2 Object Detection

Objective: Use SSD MobileNet V2 for detecting objects in real-time applications.  
Functions:  
- `wget`: Download pre-trained SSD MobileNet V2 model and label map.  
- `tf.saved\_model.load()`: Load the TensorFlow SSD model.  
- `tf.convert\_to\_tensor()`: Convert images to tensors for TensorFlow processing.  
- `detections[]`: Parse model output to get bounding box, class, and confidence scores.  
- `cv2.rectangle()`: Draw bounding boxes and class labels for detected objects.  
  
Example Code:  
```python  
# Download SSD MobileNet model and label map  
!wget http://download.tensorflow.org/models/object\_detection/tf2/20200711/ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8.tar.gz  
!wget https://raw.githubusercontent.com/tensorflow/models/master/research/object\_detection/data/mscoco\_label\_map.pbtxt  
  
# Load SSD model  
ssd\_model = tf.saved\_model.load('ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8/saved\_model')  
```

## 4. Result Visualization and Comparison

Objective: Display and compare results from HOG (traditional) and deep learning-based detections (YOLO, SSD).  
Functions:  
- `plt.imshow()`: Display images with detections and visualizations.  
- `plt.subplots()`: Organize multiple images in a single figure for comparison.  
  
Example Code:  
```python  
# Concatenate and display traditional vs deep learning results  
deep\_learning\_result = np.concatenate((yolo\_result, ssd\_result), axis=1)  
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 7))  
ax1.imshow(hog\_result)  
ax1.set\_title('Traditional (HOG)')  
ax2.imshow(deep\_learning\_result)  
ax2.set\_title('Deep Learning (YOLO + SSD)')  
plt.show()  
```

# Key Points

- \*\*HOG + SVM\*\* is a traditional method mainly used for pedestrian detection. It’s less computationally intensive but limited in accuracy and class diversity.  
- \*\*YOLOv3\*\* is a deep learning-based approach optimized for real-time object detection. It can detect a wide variety of objects.  
- \*\*SSD MobileNet V2\*\* is optimized for embedded and mobile devices, enabling fast detection with reasonable accuracy.