COMSATS University Islamabad Wah Campus

Program:	**BS(CS)**
Date:	**12-12-2-24**
Subject:	**Pattern Recognition**
Instructor:	**Samia Riaz**
Name:	**Hussain Ali**
Reg #:	**FA21-BCS-066**

Q.NO.1.Explain YOLO algorithm.

YOLO (You Only Look Once) Algorithm

YOLO (You Only Look Once) is a popular, real-time object detection algorithm used for detecting objects in images and videos. It is known for its speed and efficiency in processing images, making it suitable for real-time applications like surveillance, autonomous vehicles, and robotics.

Key Features of YOLO:

1. **Real-Time-Performance**:
   YOLO can process images in real time (i.e., very fast).

This makes it ideal for use in applications that require quick object detection, such as video surveillance or autonomous systems.

2. **Unified-Architecture**:
   Unlike traditional object detection methods that treat object detection as separate classification and localization tasks, YOLO performs both classification and localization (finding bounding boxes) in one unified neural network. This reduces computational complexity.
3. **Single-Forward-Pass**:
   YOLO predicts the positions and classes of objects in a single forward pass through the network, which contributes to its speed. It divides the input image into a grid and predicts bounding boxes and class probabilities for each grid cell.
4. **End-to-End-Training**:
   YOLO is trained as a single network that optimizes all parameters jointly. This contrasts with methods that require separate networks for object detection and classification.
5. **Generalization-to-Multiple-Objects**:
   YOLO can detect a wide range of objects at once, such as cars, people, dogs, etc., making it highly versatile.

Steps in YOLO Algorithm:

1. **Input-Image**:
   The image is passed through the network, which has a fixed input size (typically 416x416 pixels for YOLOv3).
2. **Grid-Division**:
   The image is divided into a grid of SxS cells, where each cell will predict multiple bounding boxes.
3. **Bounding-Box-and-Class-Predictions**:
   For each cell, YOLO predicts several bounding boxes, each with:
   - Coordinates of the center (x, y)
   - Width and height (w, h)
   - Confidence score (probability that the box contains an object)
   - Class probabilities (for detecting object types like a person, dog, car, etc.)
4. **Detection-Output**:
   The network outputs a tensor containing all bounding boxes, class scores, and confidence scores for each grid cell.
5. **Post-Processing**:
   Non-Maximum Suppression is applied to filter out multiple boxes predicting the same object. Only the most confident bounding box is retained for each object.

Q.NO.2.

Take data of person and perform pose detection using yolo. Include screen shots / code / comments to explain the process

- **The objective of this assignment is to learn how to use YOLO (You Only Look Once) for pose detection. You will take an image of a person, process it through a pre-trained YOLO model to detect keypoints, and visualize the results.**

+ Code + Text

```
!pip install opencv-python opencv-python-headless numpy
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
```

```
[2]  import cv2
     import numpy as np
     import requests
     from io import BytesIO
     import matplotlib.pyplot as plt
```

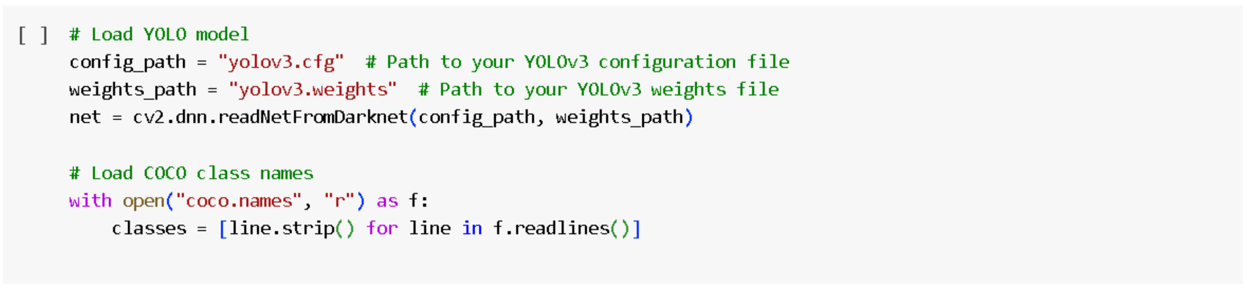
```
Start coding or generate with AI.
```

```
[4]  from google.colab import files
     uploaded = files.upload()  # This will allow you to upload the YOLO files
```

Choose Files 3 files

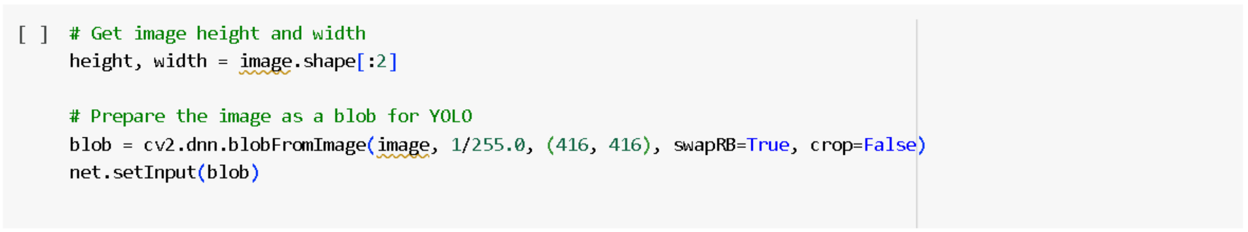
- **yolov3.weights**(n/a) - 134 bytes, last modified: 12/12/2024 - 100% done
- **coco.names**(n/a) - 625 bytes, last modified: 12/12/2024 - 100% done
- **yolov3.cfg**(n/a) - 8342 bytes, last modified: 12/12/2024 - 100% done

```
Saving yolov3.weights to yolov3.weights
Saving coco.names to coco.names
Saving yolov3.cfg to yolov3.cfg
```



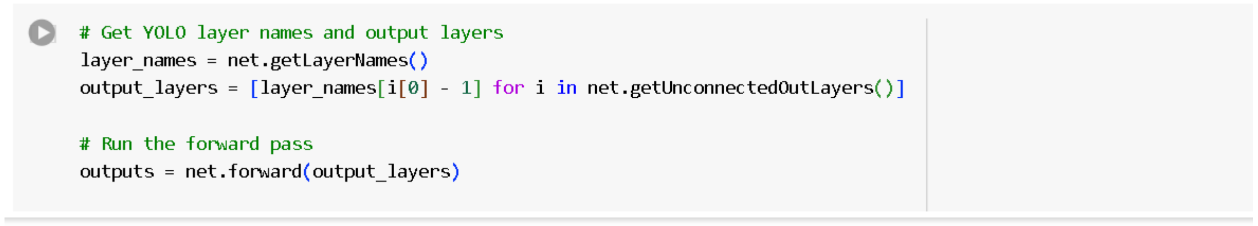
```
# Load YOLO model
config_path = "yolov3.cfg"  # Path to your YOLOv3 configuration file
weights_path = "yolov3.weights"  # Path to your YOLOv3 weights file
net = cv2.dnn.readNetFromDarknet(config_path, weights_path)

# Load COCO class names
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
```



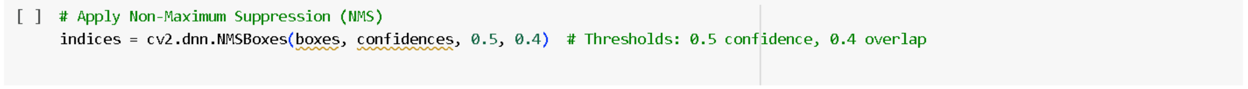
```
# Get image height and width
height, width = image.shape[:2]

# Prepare the image as a blob for YOLO
blob = cv2.dnn.blobFromImage(image, 1/255.0, (416, 416), swapRB=True, crop=False)
net.setInput(blob)
```

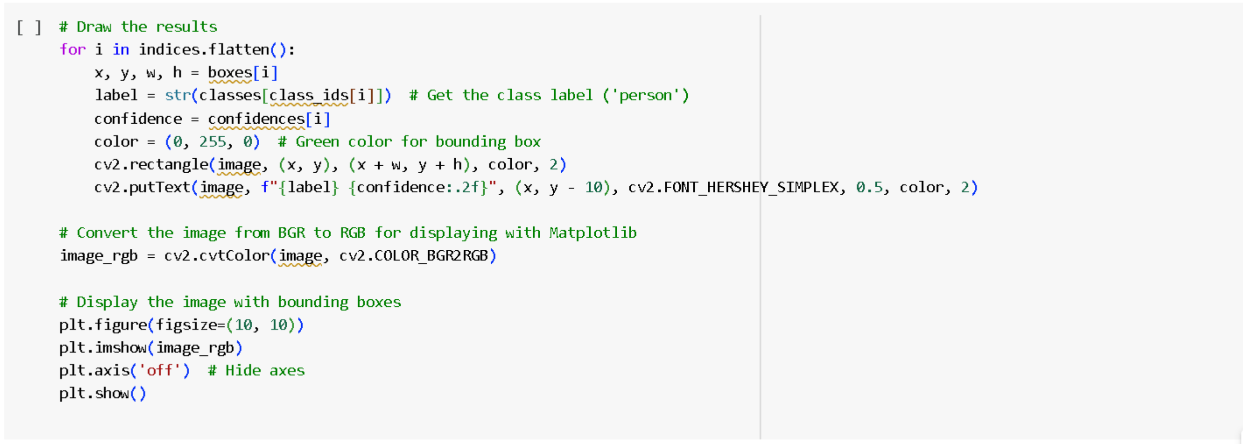


```
# Get YOLO layer names and output layers
layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

# Run the forward pass
outputs = net.forward(output_layers)
```



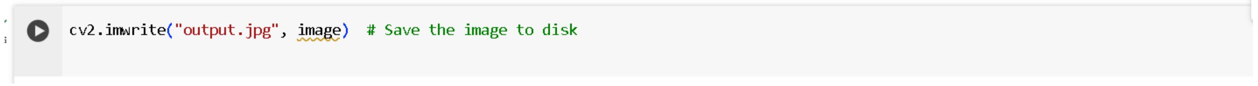
```
# Apply Non-Maximum Suppression (NMS)
indices = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)  # Thresholds: 0.5 confidence, 0.4 overlap
```



```
# Draw the results
for i in indices.flatten():
    x, y, w, h = boxes[i]
    label = str(classes[class_ids[i]])  # Get the class label ('person')
    confidence = confidences[i]
    color = (0, 255, 0)  # Green color for bounding box
    cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)
    cv2.putText(image, f"{label} {confidence:.2f}", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

# Convert the image from BGR to RGB for displaying with Matplotlib
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Display the image with bounding boxes
plt.figure(figsize=(10, 10))
plt.imshow(image_rgb)
plt.axis('off')  # Hide axes
plt.show()
```



```
cv2.imwrite("output.jpg", image)  # Save the image to disk
```