**TECHPLEMENT PROJECT**

Name: Amaravadi Dheeraj

Project team - 02

**Team Name: Py 02**

**Project Statement**: Image Recognition or Object Detection Description: Develop a Python program that can recognize objects in images or perform image classification tasks. Implementation: Use Python libraries like OpenCV or TensorFlow to train and deploy image recognition models. Utilize pre-trained models or train your own with labeled datasets.

**Code:**

import cv2

net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")

classes = []

with open("coco.names", "r") as f:

classes = [line.strip() for line in f.readlines()]

# Load image

img = cv2.imread("image.jpg")

height, width, \_ = img.shape

blob = cv2.dnn.blobFromImage(img, 1/255.0, (416, 416), swapRB=True, crop=False)

net.setInput(blob)

outs = net.forward()

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0.5:

center\_x = int(detection[0] \* width)

center\_y = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

x = int(center\_x - w / 2)

y = int(center\_y - h / 2)

cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.putText(img, classes[class\_id], (x, y - 5), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

cv2.imshow("Object Detection", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Object Detection with YOLO (You Only Look Once):**

**Model Loading:**

We load the pre-trained YOLO model using cv2.dnn.readNet, which reads the model weights and configuration files.

**Class Loading:**

We load the class names used by the YOLO model from a file called "coco.names", which contains the names of 80 common objects in the COCO dataset.

**Image Loading and Preprocessing:**

We load an image using cv2.imread from OpenCV.

The image is preprocessed for input to the YOLO model using cv2.dnn.blobFromImage, which performs resizing, scaling, and mean subtraction.

**Prediction:**

We pass the preprocessed image through the YOLO model using net.forward().

The detection results are processed to extract bounding box coordinates, class IDs, and confidence scores.

Bounding boxes are drawn around detected objects, and class labels are annotated with bounding boxes.

**Displaying Results:**

The annotated image with detected objects and class labels is displayed using cv2.imshow.

The YOLO model processes images in real-time by dividing them into a grid and predicting bounding boxes and class probabilities for each grid cell.

Confidence threshold is set to 0.5, meaning that only detections with confidence scores above this threshold are considered.