EX.No:1 DATE: 28/1/25	**Write a program to demonstrate the working of CNN architecture to classify images**

AIM:

To perform object detection on an input image using the YOLOv3 model in Google Colab and visualize the detected objects with bounding boxes and class labels.

ALGORITHM:

1. **Setup and Download YOLO Files**:
   Import necessary libraries (`cv2`, `numpy`, `urllib`) and download the YOLOv3 weights, configuration file, and class labels (`coco.names`).
2. **Load YOLO Model and Classes**:
   Load the YOLO model using `cv2.dnn.readNet()` and read the class names from the `coco.names` file.
3. **Preprocess the Input Image**:
   Load the input image, convert it into a YOLO-compatible blob, and pass it into the network.
4. **Detect Objects and Post-process Results**:
   Perform a forward pass to detect objects, filter out low-confidence predictions, and apply Non-Maximum Suppression (NMS) to remove redundant boxes.
5. **Visualize Detections**:
   Draw bounding boxes and class labels on the image and display the result using `cv2_imshow()`.

CODE:

```
import cv2

import numpy as np

import urllib.request

import matplotlib.pyplot as plt

from google.colab.patches import cv2_imshow

yolo_weights_url = "https://pjreddie.com/media/files/yolov3.weights"

yolo_cfg_url = "https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg?raw=true"

yolo_names_url = "https://raw.githubusercontent.com/pjreddie/darknet/master/data/coco.names"

yolo_weights_path = "yolov3.weights"

yolo_cfg_path = "yolov3.cfg"

yolo_names_path = "coco.names"

def download_file(url, path):

    try:
```

```
            urllib.request.urlretrieve(url, path)
            print(f"Downloaded {path}")
        except Exception as e:
            print(f"Error downloading {path}: {e}")
    download_file(yolo_weights_url, yolo_weights_path)
    download_file(yolo_cfg_url, yolo_cfg_path)
    download_file(yolo_names_url, yolo_names_path)
    net = cv2.dnn.readNet(yolo_weights_path, yolo_cfg_path)
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
    with open(yolo_names_path, 'r') as f:
        classes = [line.strip() for line in f.readlines()]
    image_path = '/content/adorable-dog-lifestyle.jpg'
    image = cv2.imread(image_path)
    if image is None:
        print(f"Error: Image at {image_path} not found or failed to load.")
        exit()
    height, width, channels = image.shape
    blob = cv2.dnn.blobFromImage(image, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5:  # Filter weak detections
                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)
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)
indices = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
if len(indices) > 0:
    indices = indices.flatten()
else:
    print("No objects detected.")
    indices = []
for i in indices:
    x, y, w, h = boxes[i]
    label = str(classes[class_ids[i]])  # Get the label
    confidence = confidences[i]
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
    label_text = f"{label} {confidence:.2f}"
    cv2.putText(image, label_text, (x, y - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)
cv2_imshow(image)
```

OUTPUT:

RESULT:

Thus the program has been completed and verified successfully.