```
'''
Adarsh Gourab Das
2141004066
'''

import cv2
import os
import numpy as np

def load_yolo_model():
    net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
    layer_names = net.getLayerNames()
    output_layers_indices = net.getUnconnectedOutLayers()
    output_layers = [layer_names[i - 1] for i in output_layers_indices]
    return net, output_layers

# Perform detection
def detect_objects(net, output_layers, image):
    height, width = image.shape[:2]

    # Prepare the image for the model
    blob = cv2.dnn.blobFromImage(image, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)
    outputs = net.forward(output_layers)

    boxes = []
    confidences = []
    class_ids = []

    # Check if outputs is a list or a scalar value
    if isinstance(outputs[0], np.ndarray):
        for output in outputs:
            for detection in output:
                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)

                    # Rectangle coordinates
                    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)
    else:
        detection = outputs[0]
        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)

            # Rectangle coordinates
            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)

    # Apply Non-Maxima Suppression
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

    return boxes, confidences, class_ids, indexes

# Draw bounding boxes on the image
def draw_labels(boxes, confidences, class_ids, indexes, image):
    if isinstance(indexes, np.ndarray):
        indexes = indexes.flatten()

    for i in indexes:
        box = boxes[i]
        x, y, w, h = box
        label = str(classes[class_ids[i]])
        confidence = confidences[i]
        color = (100, 255, 100)  # 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 - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color,
2)

def process_images(input_directory, output_directory):
    net, output_layers = load_yolo_model()

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

    os.makedirs(output_directory, exist_ok=True)

    for filename in os.listdir(input_directory):
        if filename.endswith('.jpg') or filename.endswith('.png'):
            image_path = os.path.join(input_directory, filename)
            image = cv2.imread(image_path)

            boxes, confidences, class_ids, indexes = detect_objects(net, output_layers, image)
            draw_labels(boxes, confidences, class_ids, indexes, image)

            output_path = os.path.join(output_directory, f'detected_{filename}')
            cv2.imwrite(output_path, image)
            print(f'Processed and saved detection for: {filename}')

input_directory = r'D:\Personal Projects\Celebal Technologies\submissions\Week 10\Sample Dataset'
output_directory = r'D:\Personal Projects\Celebal Technologies\submissions\Week 10\Saved Images\T6YOLO'

process_images(input_directory, output_directory)
```