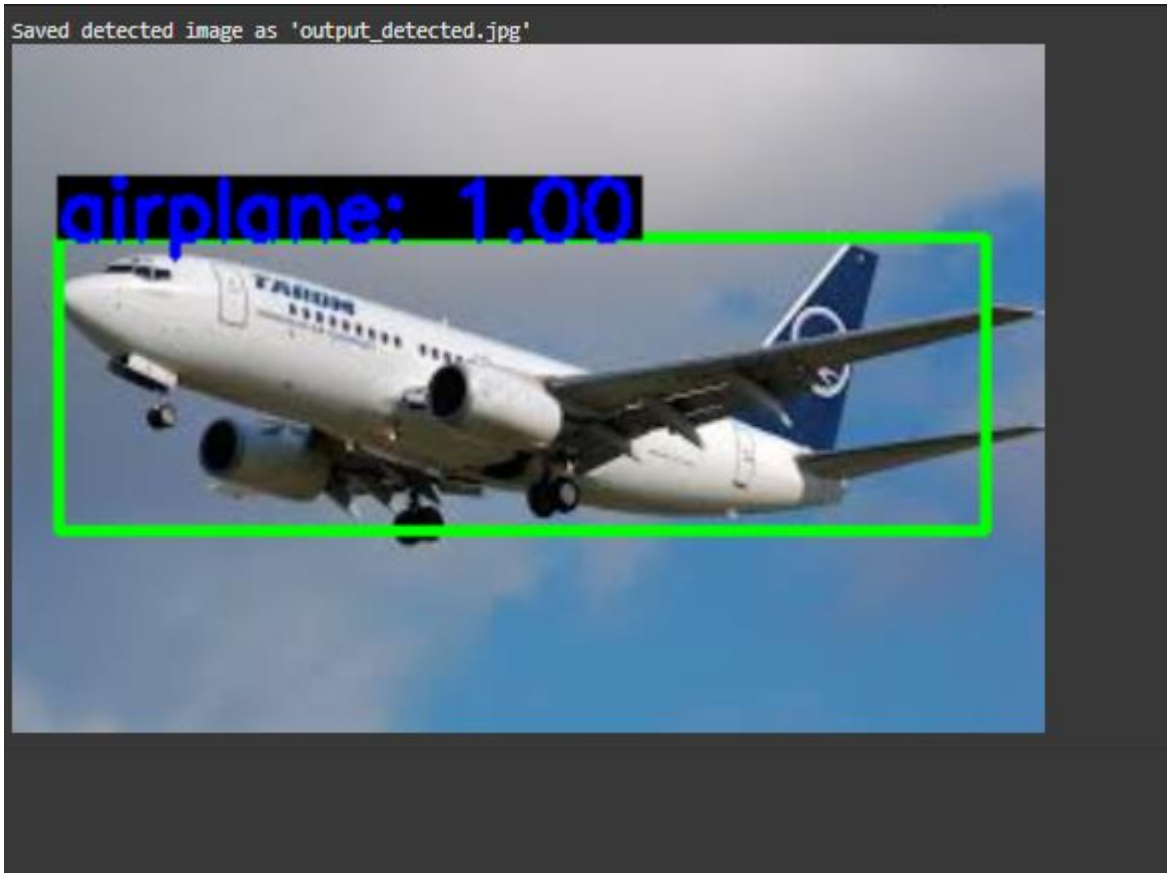
Code:

```
import torch
import torchvision
from torchvision.transforms import functional as F
import cv2
import numpy as np
from google.colab.patches import cv2_imshow  # Only for Google Colab

# Load the pre-trained Faster R-CNN model
model =
torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
model.eval()

COCO_INSTANCE_CATEGORY_NAMES = [
    '__background__', 'person', 'bicycle', 'car', 'motorcycle',
'airplane', 'bus', 'train', 'truck', 'boat',
    'traffic light', 'fire hydrant', 'stop sign', 'parking meter',
'bench', 'bird', 'cat', 'dog', 'horse',
    'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpack',
'umbrella', 'handbag', 'tie', 'suitcase',
    'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball
bat', 'baseball glove', 'skateboard', 'surfboard',
    'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife',
'spoon', 'bowl', 'banana', 'apple', 'sandwich',
    'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut',
'cake', 'chair', 'couch', 'potted plant', 'bed',
    'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote',
'keyboard', 'cell phone', 'microwave', 'oven',
    'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase',
'scissors', 'teddy bear', 'hair drier', 'toothbrush'
]

def detect_objects(image_path, confidence_threshold=0.5,
resize_factor=2.5, save_result=False, save_path="detected_image.jpg",
box_color=(0, 255, 0), text_color=(255, 0, 0)):
    """
    Detect objects in an image using Faster R-CNN and draw bounding
boxes.

    Parameters:
    - image_path (str): Path to the input image.
    - confidence_threshold (float): Minimum confidence score for
detections.
    - resize_factor (float): Factor to resize the output image for
better visibility.
    - save_result (bool): Whether to save the resulting image.
```

```
    - save_path (str): Path to save the image.
    - box_color (tuple): RGB color for bounding boxes.
    - text_color (tuple): RGB color for text labels.

    Returns:
    - detected_image (numpy array): Image with detected objects.
    """
    # Load the image
    image = cv2.imread(image_path)
    if image is None:
        print(f"Error: Could not load image from '{image_path}'. Please
check the file path.")
        return None

    original_image = image.copy()
    image_tensor = F.to_tensor(image)

    # Perform inference
    with torch.no_grad():
        predictions = model([image_tensor])

    boxes = predictions[0]['boxes'].cpu().numpy()
    labels = predictions[0]['labels'].cpu().numpy()
    scores = predictions[0]['scores'].cpu().numpy()

    # Check if there are valid detections
    if len(boxes) == 0:
        print("No objects detected.")
        return original_image

    # Draw bounding boxes for high-confidence detections
    for i, box in enumerate(boxes):
        if scores[i] >= confidence_threshold:
            label_idx = labels[i]

            # Ensure label index is within valid range
            if label_idx < len(COCO_INSTANCE_CATEGORY_NAMES):
                label = COCO_INSTANCE_CATEGORY_NAMES[label_idx]
                score = scores[i]
                start_point = (int(box[0]), int(box[1]))
                end_point = (int(box[2]), int(box[3]))
                cv2.rectangle(original_image, start_point, end_point,
box_color, 2)

                font_scale = 0.7  # Increased font size
                thickness = 2  # Increased text thickness
                text_size, _ = cv2.getTextSize(f"{label}: {score:.2f}",
cv2.FONT_HERSHEY_SIMPLEX, font_scale, thickness)
```

```
                text_width, text_height = text_size

                cv2.rectangle(original_image, start_point,
(start_point[0] + text_width, start_point[1] - text_height), (0, 0, 0),
-1)

                cv2.putText(original_image, f"{label}: {score:.2f}",
start_point, cv2.FONT_HERSHEY_SIMPLEX, font_scale, text_color,
thickness)

    resized_image = cv2.resize(original_image, (0, 0),
fx=resize_factor, fy=resize_factor)

    if save_result:
        cv2.imwrite(save_path, resized_image)
        print(f"Saved detected image as '{save_path}'")

    return resized_image

if __name__ == "__main__":
    IMAGE_PATH = "/download.jpg"
    detected_image = detect_objects(IMAGE_PATH,
confidence_threshold=0.5, resize_factor=2.5, save_result=True,
save_path="output_detected.jpg", box_color=(0, 255, 0),
text_color=(255, 0, 0))

    if detected_image is not None:
        cv2_imshow(detected_image)
```

output:



Modifications:

- **Additional Parameters**:
  - `save_result`: Option to save the detected image.
  - `save_path`: Path to save the result.
  - `box_color` **and** `text_color`: Customizable colors for bounding boxes and text.
- **Text Appearance**:
  - **Increased font size** (0.7) and **text thickness** (2) for better readability.
  - Added a **background rectangle** behind text for contrast.
- **Image Resizing**:
  - Image is resized by a **factor of 2.5** after detection for better visibility.
- **Error Handling**:
  - Checks if no objects are detected and avoids drawing boxes.

- **Main Execution**:
  - The function now has **customizable default values** (e.g., confidence threshold, resize factor).
  - Includes **saving** the result image if `save_result=True`.