🚀 Deploying Your YOLO Model on Raspberry Pi 4 model B

Since you're deploying a **fine-tuned YOLO model** on Raspberry Pi 4, we'll focus on **installing dependencies, transferring your model, and running inference** on images, videos, and a USB camera feed.

🔗 Step 1: Set Up Raspberry Pi 4

✅ 1. Update and Upgrade Packages

Run the following to ensure everything is up to date:

```
sudo apt update && sudo apt upgrade -y
```

✅ 2. Install Python and Virtual Environment

```
sudo apt install python3 python3-pip python3-venv -y
```

Create a dedicated virtual environment for YOLO:

```
mkdir ~/yolo_project && cd ~/yolo_project

python3 -m venv venv

source venv/bin/activate
```

📦 Step 3: Install Required Dependencies

```
pip install torch torchvision numpy opencv-python pillow ultralytics
```

📂 Step 4: Transfer Your Trained Model (best.pt)

If your model (best.pt) is on your computer, **transfer it to Raspberry Pi 4**.

Option 1: Use SCP (from your PC)

Run this from your **PC terminal** (replace <your_pi_ip> with your Raspberry Pi's IP address):

scp best.pt pi@<your_pi_ip>:~/yolo_project/

Option 2: Use a USB Drive

1. Copy best.pt to a USB drive from your **PC**.

2. Plug the USB into **Raspberry Pi 4**.
3. Mount and copy it to ~/yolo_project/.

💻 Step 5: Run YOLO Model for Inference

Create a script **detect.py** inside ~/yolo_project/:

nano detect.py

Paste this Python code and save it (**CTRL + X → Y → ENTER**):

```
import torch
from ultralytics import YOLO
import cv2
import sys

# Load the trained model
model = YOLO("best.pt")  # Ensure the model file is in the same directory

# Function to run inference
def run_inference(source):
    cap = cv2.VideoCapture(source)

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Run YOLOv8 inference
        results = model(frame)

        # Draw bounding boxes
```

```
        for result in results:
            boxes = result.boxes.xyxy.numpy()
            for box in boxes:
                x1, y1, x2, y2 = map(int, box[:4])
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.imshow("YOLOv8 Detection", frame)
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    cap.release()
    cv2.destroyAllWindows()
# Get source from command-line arguments
if len(sys.argv) > 1:
    source = sys.argv[1]
    if source.isdigit():  # Check if source is a camera index (e.g., 0, 1)
        source = int(source)
    run_inference(source)
else:
    print("Usage: python detect.py <source>")
    print("Example: python detect.py 0 (for live camera)")
    print("Example: python detect.py sample_video.mp4 (for a video file)")
    print("Example: python detect.py sample.jpg (for an image)")
```

▶️ Step 6: Run Detection on Images, Videos, and USB Camera

🔵 1. Run on an Image

```
python detect.py sample.jpg
```

🔹 Replace sample.jpg with the actual image file.

🟢 2. Run on a Video

```
python detect.py test_video.mp4
```

🔹 Replace test_video.mp4 with your actual video file.

🔴 3. Run on a Live USB Camera

```
python detect.py 0
```

🔹 Change 0 to 1 if the USB camera is not detected.