```
pip install -U ultralytics
```

Show hidden output

```
import os
from ultralytics import YOLO
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt


# Step 3: Train the YOLOv5 Model
def train_model(data_yaml, epochs=10, batch_size=8, image_size=128):
    """
    Train YOLOv5 model on a custom dataset.

    Args:
        data_yaml (str): Path to the dataset YAML file.
        epochs (int): Number of training epochs.
        batch_size (int): Training batch size.
        image_size (int): Size of input images.

    Returns:
        model: Trained YOLO model.
    """
    model = YOLO('yolov5su.pt')  # Pre-trained YOLOv5 model
    model.train(data=data_yaml, epochs=epochs, batch=batch_size, imgsz=image_size)
    return model


# Train the model
data_yaml = "/content/sample_data/Automobile axle number detection.v9i.yolov5pytorch.zip"  # Replace with your YAML file path
trained_model = train_model(data_yaml, epochs=10, batch_size=8, image_size=128)

# Save the trained model weights
trained_model.export(format='torchscript')  # Save as TorchScript or other formats
print("Model training completed and saved.")
```

Show hidden output

```
def detect_wheels(image_path, model_path="/content/runs/detect/train/weights/best.pt"):
    """
    Detect wheels in a vehicle image using a YOLO model.

    Args:
        image_path (str): Path to the input image.
        model_path (str): Path to the trained YOLO model weights.

    Returns:
        list: Detected wheel bounding boxes [(x1, y1, x2, y2), ...].
        np.array: Image in RGB format.
    """
    # Load the YOLO model
    model = YOLO(model_path)

    # Load and preprocess the image
    try:
        pil_image = Image.open(image_path).convert("RGB")
    except Exception as e:
        raise ValueError(f"Error loading image: {e}")

    image_rgb = np.array(pil_image)

    # Run inference
    results = model(image_rgb)[0]

    # Extract wheel detections
    wheels = []
    count=0
    if results.boxes:
        for box, cls in zip(results.boxes.xyxy.cpu().numpy(), results.boxes.cls.cpu().numpy()):
            label = model.names[int(cls)]
            print(f"Detected: {label} with box: {box}")  # Debugging line
            count=count+1
```

```
            count=count+1
            if label.lower() == "axel":  # Change "wheel" to "axel"
                wheels.append(tuple(map(int, box)))  # Convert box to integers


    return wheels, image_rgb,count


def estimate_axles(wheels, image_rgb):
    """
    Estimate the number of axles based on detected wheels.

    Args:
        wheels (list): List of detected wheel bounding boxes [(x1, y1, x2, y2), ...].
        image_rgb: RGB image with wheels drawn.

    Returns:
        int: Estimated number of axles.
    """
    # Draw wheels on the image for visualization
    plt.imshow(image_rgb)
    for (x1, y1, x2, y2) in wheels:
        plt.gca().add_patch(plt.Rectangle((x1, y1), x2 - x1, y2 - y1, edgecolor="green", facecolor="none", linewidth=2))
    plt.axis("off")
    plt.show()

    # Estimate axles by grouping wheels based on vertical alignment
    #wheels.sort(key=lambda box: box[1])  # Sort by vertical position (y1)
    #print(f"Sorted wheels: {wheels}")  # Debugging line

    return count


if __name__ == "__main__":
    # Path to the input image
    image_path = r"/content/sample_data/MH12UM4603 B.jpg"  # Replace with your image path

    # Detect wheels
    wheels, image_rgb,count = detect_wheels(image_path, model_path="/content/runs/detect/train/weights/best.pt")

    # Estimate axles
    axle_count = estimate_axles(wheels, image_rgb)
    print(f"Estimated Axle Count: {count}")
```

```
0: 96x128 5 axelss, 37.5ms
Speed: 0.8ms preprocess, 37.5ms inference, 1.0ms postprocess per image at shape (1, 3, 96, 128)
Detected: axels with box: [     773.85       440.4      1038.9      772.75]
Detected: axels with box: [     523.03      416.49      718.05       646.6]
Detected: axels with box: [     391.94       411.7      506.04      577.65]
Detected: axels with box: [     212.54      406.09      275.29       502.4]
Detected: axels with box: [     142.81      402.53      195.62      489.86]
```


```
Estimated Axle Count: 5
```

Start coding or generate with AI.