In [1]:
```
from roboflow import Roboflow
rf = Roboflow(api_key="BI1tQqHk4ASPaxBOOYf4")
project = rf.workspace("universitas-diponegoro-mb10s").project("car-detection-r
version = project.version(1)
dataset = version.download("yolov8")
```

```
loading Roboflow workspace...
loading Roboflow project...
Dependency ultralytics==8.0.196 is required but found version=8.1.45, to fix:
`pip install ultralytics==8.0.196`
```

In [2]:
```
!git clone https://github.com/ultralytics/ultralytics
```

```
fatal: destination path 'ultralytics' already exists and is not an empty dire
ctory.
```

In [3]:
```
import os

dataset_path = 'Car-detection-1'  # Adjust this to the path of your dataset

# List the directories within the dataset
for dirpath, dirnames, filenames in os.walk(dataset_path):
    print(f"Found directory: {dirpath}")
    for filename in filenames[:5]:  # Just print the first 5 files for brevity
        print(f"  {filename}")
```

```
Found directory: Car-detection-1
  data.yaml
  README.dataset.txt
  README.roboflow.txt
Found directory: Car-detection-1\test
Found directory: Car-detection-1\test\images
  youtube-44_jpg.rf.4a9b4bd6780237f50038cf6d47e121ba.jpg
  youtube-59_jpg.rf.d5d4cce3b92916006b85a365ef419c13.jpg
  youtube-62_jpg.rf.eeb271a7bb61e742698397b1b5cf7deb.jpg
Found directory: Car-detection-1\test\labels
  youtube-44_jpg.rf.4a9b4bd6780237f50038cf6d47e121ba.txt
  youtube-59_jpg.rf.d5d4cce3b92916006b85a365ef419c13.txt
  youtube-62_jpg.rf.eeb271a7bb61e742698397b1b5cf7deb.txt
Found directory: Car-detection-1\train
Found directory: Car-detection-1\train\images
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.513cdbac98dc8698c9fea99e25
866f7c.jpg
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.51fe3b159fe8fd5caeba316c3f
de341d.jpg
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.6f47f4ab2d0759d687130cb755
269dfe.jpg
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.7d98bdf3d0a3391cc3160baf7b
c13a9b.jpg
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.831af8a40cbadcf3ec0f400869
11bceb.jpg
Found directory: Car-detection-1\train\labels
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.513cdbac98dc8698c9fea99e25
866f7c.txt
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.51fe3b159fe8fd5caeba316c3f
de341d.txt
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.6f47f4ab2d0759d687130cb755
269dfe.txt
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.7d98bdf3d0a3391cc3160baf7b
c13a9b.txt
  -0378D390-5AE2-4D09-84BB-387E850F5E5A-png_jpg.rf.831af8a40cbadcf3ec0f400869
11bceb.txt
Found directory: Car-detection-1\valid
Found directory: Car-detection-1\valid\images
  -0573F94C-289D-4503-A571-94C6C8C5674C-png_jpg.rf.7e1fe70bdd94c393484f0f03d2
00f58c.jpg
  -0ACAD937-70A4-46F0-8E49-0CAE6F942CF2-png_jpg.rf.72951820c6124c65c29b8c2a3f
b50f72.jpg
  -0AE13AAD-2BD5-490E-B245-D51927B1EBBC-png_jpg.rf.5eca6871cf297a0dd148d24c48
bdec58.jpg
  -0B43DB53-2377-4D6D-B5B8-60CBBAB90F88-png_jpg.rf.e9f572574f92cfe4b504a9bbad
2b2c0d.jpg
  -0FB0E465-1451-4618-84CD-1A1A88293B08-png_jpg.rf.4da9235f70835c8951a038bffd
cc87a2.jpg
Found directory: Car-detection-1\valid\labels
  -0573F94C-289D-4503-A571-94C6C8C5674C-png_jpg.rf.7e1fe70bdd94c393484f0f03d2
00f58c.txt
  -0ACAD937-70A4-46F0-8E49-0CAE6F942CF2-png_jpg.rf.72951820c6124c65c29b8c2a3f
b50f72.txt
  -0AE13AAD-2BD5-490E-B245-D51927B1EBBC-png_jpg.rf.5eca6871cf297a0dd148d24c48
bdec58.txt
  -0B43DB53-2377-4D6D-B5B8-60CBBAB90F88-png_jpg.rf.e9f572574f92cfe4b504a9bbad
2b2c0d.txt
```

```
  -0FB0E465-1451-4618-84CD-1A1A88293B08-png_jpg.rf.4da9235f70835c8951a038bffd
cc87a2.txt
```

In [17]:
```
with open(os.path.join(dataset_path, 'data.yaml'), 'r') as file:
    print(file.read())
```

```
names:
- car
nc: 1
roboflow:
  license: CC BY 4.0
  project: car-detection-nxsxm
  url: https://universe.roboflow.com/universitas-diponegoro-mb10s/car-detecti
on-nxsxm/dataset/1 (https://universe.roboflow.com/universitas-diponegoro-mb10
s/car-detection-nxsxm/dataset/1)
  version: 1
  workspace: universitas-diponegoro-mb10s
test: C:/Users/shane/Car_Detection/Car-detection-1/test/images
train: C:/Users/shane/Car_Detection/Car-detection-1/train/images
val: C:/Users/shane/Car_Detection/Car-detection-1/val/images
```

In [18]:
```
import os

# Print the current working directory
print("Current Working Directory:", os.getcwd())

# Set or correct the dataset_path variable
dataset_path = 'Car-detection-1'  # Adjust based on your setup
print("Dataset Path:", dataset_path)
```

```
Current Working Directory: C:\Users\shane\Car_Detection
Dataset Path: Car-detection-1
```

In [19]:
```
# List files in the specified directory to verify the image file's presence
image_directory = os.path.join(dataset_path, 'train/images')
print(f"Listing files in {image_directory}:\n")
!dir "{image_directory}"  # For Windows
```

```
Listing files in Car-detection-1\train/images:

 Volume in drive C has no label.
 Volume Serial Number is DEB0-8041

 Directory of C:\Users\shane\Car_Detection\Car-detection-1\train\images

2024-04-08  12:30 AM    <DIR>          .
2024-04-08  12:30 AM    <DIR>          ..
2024-04-08  12:30 AM            46,545 -0378D390-5AE2-4D09-84BB-387E850F5E5
A-png_jpg.rf.513cdbac98dc8698c9fea99e25866f7c.jpg
2024-04-08  12:30 AM            46,545 -0378D390-5AE2-4D09-84BB-387E850F5E5
A-png_jpg.rf.51fe3b159fe8fd5caeba316c3fde341d.jpg
2024-04-08  12:30 AM            19,551 -0378D390-5AE2-4D09-84BB-387E850F5E5
A-png_jpg.rf.6f47f4ab2d0759d687130cb755269dfe.jpg
2024-04-08  12:30 AM            41,734 -0378D390-5AE2-4D09-84BB-387E850F5E5
A-png_jpg.rf.7d98bdf3d0a3391cc3160baf7bc13a9b.jpg
2024-04-08  12:30 AM            39,152 -0378D390-5AE2-4D09-84BB-387E850F5E5
A-png_jpg.rf.831af8a40cbadcf3ec0f40086911bceb.jpg
```

In [20]:
```
from PIL import Image, ImageDraw
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle

def display_image_with_boxes(image_path, label_path):
    # Load the image
    img = Image.open(image_path)
    fig, ax = plt.subplots()
    ax.imshow(img)

    # Open the label file and draw boxes
    with open(label_path, 'r') as f:
        for line in f.readlines():
            # Assuming the format: class x_center y_center width height
            _, x_center, y_center, width, height = map(float, line.split())

            # Convert to pixel coordinates
            img_width, img_height = img.size
            x_center, y_center, width, height = x_center * img_width, y_center
            left, top = x_center - width / 2, y_center - height / 2

            # Create a Rectangle patch
            rect = Rectangle((left, top), width, height, linewidth=2, edgecolor

            # Add the patch to the Axes
            ax.add_patch(rect)

    plt.axis('off')  # Turn off axis
    plt.show()
```

In [21]:
```
image_filename = 'youtube-21_jpg.rf.723924ba40644beb42eb488edaec1c69.jpg'  # C
image_path = os.path.join('Car-detection-1', 'train/images', image_filename)
label_path = os.path.join('Car-detection-1', 'train/labels', image_filename.rep

display_image_with_boxes(image_path, label_path)
```

In [22]:
```
from ultralytics import YOLO
```

In [36]:
```
model = YOLO("yolov8s.yaml")
```

In [37]:
```
model.train(data="C:/Users/shane/Car_Detection/Car-detection-1/data.yaml", epoc
metrics = model.val()  # Evaluate model performance on the validation set
results = model.predict("C:/Users/shane/Car_Detection/Car-detection-1/test/imag
path = model.export(format="onnx")  # Export the model to ONNX format

print("Training complete.")
print("Validation metrics:", metrics)
print("Prediction results:", results)
print("Model exported to:", path)
```

```
Ultralytics YOLOv8.1.45  Python-3.10.13 torch-2.2.1+cpu CPU (Intel Core(TM)
i7-4790K 4.00GHz)
engine\trainer: task=detect, mode=train, model=yolov8s.yaml, data=C:/Users/
shane/Car_Detection/Car-detection-1/data.yaml, epochs=3, time=None, patienc
e=100, batch=16, imgsz=640, save=True, save_period=-1, cache=False, device=
None, workers=8, project=None, name=train7, exist_ok=False, pretrained=Tru
e, optimizer=auto, verbose=True, seed=0, deterministic=True, single_cls=Fal
se, rect=False, cos_lr=False, close_mosaic=10, resume=False, amp=True, frac
tion=1.0, profile=False, freeze=None, multi_scale=False, overlap_mask=True,
mask_ratio=4, dropout=0.0, val=True, split=val, save_json=False, save_hybri
d=False, conf=None, iou=0.7, max_det=300, half=False, dnn=False, plots=Tru
e, source=None, vid_stride=1, stream_buffer=False, visualize=False, augment
=False, agnostic_nms=False, classes=None, retina_masks=False, embed=None, s
how=False, save_frames=False, save_txt=False, save_conf=False, save_crop=Fa
lse, show_labels=True, show_conf=True, show_boxes=True, line_width=None, fo
rmat=torchscript, keras=False, optimize=False, int8=False, dynamic=False, s
implify=False, opset=None, workspace=4, nms=False, lr0=0.01, lrf=0.01, mome
ntum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, wa
rmup_bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smo
```

In [40]:
```
model_yolov8s = YOLO("yolov8s.pt")
```

In [27]:
```
# model.train(data="C:/Users/shane/Car_Detection/Car-detection-1/data.yaml", ep
# metrics = model.val()  # Evaluate model performance on the validation set
# results = model("C:/Users/shane/Car_Detection/Car-detection-1/test/images")
# path = model.export(format="onnx")  # Export the model to ONNX format

# print("Training complete.")
# print("Validation metrics:", metrics)
# print("Prediction results:", results)
# print("Model exported to:", path)
```

```
0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_
bias_lr=0.1, box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0, label_smoothin
g=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.
1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, bgr=0.0,
mosaic=1.0, mixup=0.0, copy_paste=0.0, auto_augment=randaugment, erasing=0.
4, crop_fraction=1.0, cfg=None, tracker=botsort.yaml, save_dir=runs\detect
\train6
Overriding model.yaml nc=80 with nc=1

                   from  n    params  module
arguments
  0                  -1  1       928  ultralytics.nn.modules.conv.Conv
[3, 32, 3, 2]
  1                  -1  1     18560  ultralytics.nn.modules.conv.Conv
[32, 64, 3, 2]
  2                  -1  1     29056  ultralytics.nn.modules.block.C2f
[64, 64, 1, True]
  3                  -1  1     73984  ultralytics.nn.modules.conv.Conv
[64, 128, 3, 2]
  4                  -1  2    197632  ultralytics.nn.modules.block.C2f
```

In [42]:
```
model_yolov8s.train(data="C:/Users/shane/Car_Detection/Car-detection-1/data.yar

metrics = model_yolov8s.val()  # Evaluate model performance on the validation s
results = model_yolov8s.predict("C:/Users/shane/Car_Detection/Car-detection-1/t
path = model_yolov8s(format="onnx")  # Export the model to ONNX format

print("Training complete.")
print("Validation metrics:", metrics)
print("Prediction results:", results)
print("Model exported to:", path)
```

```
                 Class     Images  Instances      Box(P          R      mAP
50  mAP50-95): 100%|██████████| 2/2 [00:13<00:00,  6.98s/it]

                   all         38         41      0.904      0.921      0.9
53      0.609
Speed: 3.7ms preprocess, 353.2ms inference, 0.0ms loss, 0.4ms postprocess p
er image
Results saved to CarDetectionProject\YOLOv8s_Training2
Ultralytics YOLOv8.1.45  Python-3.10.13 torch-2.2.1+cpu CPU (Intel Core(TM)
i7-4790K 4.00GHz)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 G
FLOPs

val: Scanning C:\Users\shane\Car_Detection\Car-detection-1\valid\labels.cac
he... 38 images, 0 backgrounds, 0 corrupt: 100%|██████████| 38/38 [00:00<?,
?it/s]
                 Class     Images  Instances      Box(P          R      mAP
50  mAP50-95): 100%|██████████| 3/3 [00:14<00:00,  4.70s/it]

                   all         38         41      0.904      0.921      0.9
53      0.609
```

In [46]:
```
# Path to your dataset's YAML file
data_yaml_path = "C:/Users/shane/Car_Detection/Car-detection-1/data.yaml"
```

In [47]:
```
# Path to the latest checkpoint
checkpoint_path_yolo = "C:/Users/shane/Car_Detection/CarDetectionProject/YOLOv8
# Initialize the YOLO model from the checkpoint
model_yolov8s_fine = YOLO(checkpoint_path_yolo)
```

In [49]:
```
# Fine-tuning configuration
train_args = {
    "data": data_yaml_path,  # Make sure this is the correct path
    "epochs": 5,  # More epochs for fine-tuning
    "batch": 16,  # Batch size
    "imgsz": 640,  # Image size
    "device": 'cpu',  # or "cuda" for GPU
    "optimizer": 'Adam',
    "lr0": 0.001,  # Learning rate
    # "patience": 3,  # This parameter may not be directly supported, depending
}

# Start fine-tuning
model_yolov8s_fine.train(**train_args,project='CarDetectionProject', name='YOLO

# Evaluate model performance on the validation set
metrics = model_yolov8s_fine.val()

# Predict on test set
results = model_yolov8s_fine.predict("C:/Users/shane/Car_Detection/Car-detectio

# Print validation metrics and results
print("Validation metrics:", metrics)
print("Prediction results:", results)
```

```
ze

        4/5         0G     0.8804     0.6324      1.236         24        6
40: 100%|██████████| 30/30 [09:09<00:00, 18.32s/it]
                 Class     Images  Instances      Box(P          R      mAP
50  mAP50-95): 100%|██████████| 2/2 [00:15<00:00,  7.88s/it]

                   all         38         41      0.995          1      0.9
95      0.718



      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Si
ze

        5/5         0G     0.7925     0.5902      1.174         34        6
40: 100%|██████████| 30/30 [09:08<00:00, 18.28s/it]
                 Class     Images  Instances      Box(P          R      mAP
50  mAP50-95): 100%|██████████| 2/2 [00:16<00:00,  8.29s/it]

                   all         38         41      0.994          1      0.9
```

In [43]:
```
# Initialize model with custom pretrained weights
model_custom = YOLO("C:/Users/shane/Car_Detection/multiple_vehicle_detection_we
```

In [45]:
```
# Train model on your dataset
model_custom.train(data="C:/Users/shane/Car_Detection/Car-detection-1/data.yaml
metrics = model_custom.val()  # Evaluate model performance on the validation se
results = model_custom.predict("C:/Users/shane/Car_Detection/Car-detection-1/te
path = model_custom.export(format="onnx")  # Export the model to ONNX format

print("Training complete.")
print("Validation metrics:", metrics)
print("Prediction results:", results)
print("Model exported to:", path)
```

```
                   all         38         41      0.933      0.927      0.
97      0.597



      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Si
ze

        3/3         0G     0.9328     0.7435      1.281         25        6
40: 100%|██████████| 30/30 [08:43<00:00, 17.44s/it]
                 Class     Images  Instances      Box(P          R      mAP
50  mAP50-95): 100%|██████████| 2/2 [00:15<00:00,  7.75s/it]

                   all         38         41      0.904      0.921      0.9
53      0.609



3 epochs completed in 0.465 hours
```

In [48]:
```
# Path to the latest checkpoint
checkpoint_path_custom = "C:/Users/shane/Car_Detection/CarDetectionProject/Cust
# Initialize the YOLO model from the checkpoint
model_custom_fine = YOLO(checkpoint_path_custom)
```

In [50]:
```
# Fine-tuning configuration
train_args = {
    "data": data_yaml_path,  # Make sure this is the correct path
    "epochs": 5,  # More epochs for fine-tuning
    "batch": 16,  # Batch size
    "imgsz": 640,  # Image size
    "device": 'cpu',  # or "cuda" for GPU
    "optimizer": 'Adam',
    "lr0": 0.001,  # Learning rate
    # "patience": 3,  # This parameter may not be directly supported, depending
}

# Start fine-tuning
model_custom_fine.train(**train_args, project='CarDetectionProject', name='Cust

# Evaluate model performance on the validation set
metrics = model_custom_fine.val()

# Predict on test set
results = model_custom_fine.predict("C:/Users/shane/Car_Detection/Car-detectior

# Print validation metrics and results
print("Validation metrics:", metrics)
print("Prediction results:", results)
```

```
40: 100%|██████████| 30/30 [09:06<00:00, 18.21s/it]
                 Class     Images  Instances      Box(P          R      mAP
50  mAP50-95): 100%|██████████| 2/2 [00:16<00:00,  8.05s/it]

                   all         38         41      0.995          1      0.9
95      0.718



      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Si
ze

        5/5         0G     0.7925     0.5902      1.174         34        6
40: 100%|██████████| 30/30 [09:14<00:00, 18.50s/it]
                 Class     Images  Instances      Box(P          R      mAP
50  mAP50-95): 100%|██████████| 2/2 [00:15<00:00,  7.97s/it]

                   all         38         41      0.994          1      0.9
95      0.769
```

In [52]:
```
from PIL import Image
from IPython.display import display

# Generate predictions for each model
results_custom_fine = model_custom_fine.predict("C:/Users/shane/Car_Detection/C
results_yolov8s = model_yolov8s.predict("C:/Users/shane/Car_Detection/Car-detec
results_yolov8s_fine = model_yolov8s_fine.predict("C:/Users/shane/Car_Detectior
results_custom = model_custom.predict("C:/Users/shane/Car_Detection/Car-detecti

# Assuming each result set can be visualized in the same manner
# And each `results` item is a list of prediction objects that can be plotted
models_results = {
    "YOLOv8s": results_yolov8s,
    "YOLOv8s Fine": results_yolov8s_fine,
    "Custom": results_custom,
    "Custom Fine": results_custom_fine,
}
```

```
image 1/3 C:\Users\shane\Car_Detection\Car-detection-1\test\images\youtube-
44_jpg.rf.4a9b4bd6780237f50038cf6d47e121ba.jpg: 640x640 1 car, 380.2ms
image 2/3 C:\Users\shane\Car_Detection\Car-detection-1\test\images\youtube-
59_jpg.rf.d5d4cce3b92916006b85a365ef419c13.jpg: 640x640 1 car, 448.3ms
image 3/3 C:\Users\shane\Car_Detection\Car-detection-1\test\images\youtube-
62_jpg.rf.eeb271a7bb61e742698397b1b5cf7deb.jpg: 640x640 1 car, 637.8ms
Speed: 4.8ms preprocess, 488.8ms inference, 1.6ms postprocess per image at
shape (1, 3, 640, 640)

image 1/3 C:\Users\shane\Car_Detection\Car-detection-1\test\images\youtube-
44_jpg.rf.4a9b4bd6780237f50038cf6d47e121ba.jpg: 640x640 1 car, 756.6ms
image 2/3 C:\Users\shane\Car_Detection\Car-detection-1\test\images\youtube-
59_jpg.rf.d5d4cce3b92916006b85a365ef419c13.jpg: 640x640 1 car, 428.4ms
image 3/3 C:\Users\shane\Car_Detection\Car-detection-1\test\images\youtube-
62_jpg.rf.eeb271a7bb61e742698397b1b5cf7deb.jpg: 640x640 1 car, 571.3ms
Speed: 7.4ms preprocess, 585.4ms inference, 1.6ms postprocess per image at
shape (1, 3, 640, 640)
```

In [55]:
```
# Select one image to visualize across models
image_index = 0  # Assuming you want to visualize the first image's results

for model_name, results in models_results.items():
    print(f"Displaying results for {model_name}:")
    # Use the plot method on the selected image's results
    image_with_predictions = results[image_index].plot()  # This returns an ann
    # Convert BGR to RGB
    image_with_predictions_rgb = Image.fromarray(image_with_predictions[..., :
    # Display the image with predictions
    display(image_with_predictions_rgb)
```

```
Displaying results for YOLOv8s:
```

In [66]:
```
image_indices = range(len(next(iter(models_results.values()))))
random_image_index = random.choice(list(image_indices))

for model_name, results in models_results.items():
    print(f"Displaying results for {model_name}:")
    # Use the plot method on the randomly selected image's results
    image_with_predictions = results[random_image_index].plot()  # This returns
    # Convert BGR to RGB
    image_with_predictions_rgb = Image.fromarray(image_with_predictions[..., :
    # Display the image with predictions
    display(image_with_predictions_rgb)
```

```
Displaying results for YOLOv8s:
```

In [67]:
```
# Generate predictions for each model using the new dataset
results_custom_fine_test2 = model_custom_fine.predict("C:/Users/shane/Car_Detec
results_yolov8s_test2 = model_yolov8s.predict("C:/Users/shane/Car_Detection/Car
results_yolov8s_fine_test2 = model_yolov8s_fine.predict("C:/Users/shane/Car_Det
results_custom_test2 = model_custom.predict("C:/Users/shane/Car_Detection/Car-d

# Prepare to display predictions in the same manner as before
models_results_test2 = {
    "YOLOv8s (Test2)": results_yolov8s_test2,
    "YOLOv8s Fine (Test2)": results_yolov8s_fine_test2,
    "Custom (Test2)": results_custom_test2,
    "Custom Fine (Test2)": results_custom_fine_test2,
}
```

```
image 1/175 C:\Users\shane\Car_Detection\Car-detection-1\test\test_images2
\vid_5_25100.jpg: 384x640 (no detections), 201.6ms
image 2/175 C:\Users\shane\Car_Detection\Car-detection-1\test\test_images2
\vid_5_25120.jpg: 384x640 (no detections), 204.5ms
image 3/175 C:\Users\shane\Car_Detection\Car-detection-1\test\test_images2
\vid_5_25140.jpg: 384x640 (no detections), 191.0ms
image 4/175 C:\Users\shane\Car_Detection\Car-detection-1\test\test_images2
\vid_5_25160.jpg: 384x640 (no detections), 220.8ms
image 5/175 C:\Users\shane\Car_Detection\Car-detection-1\test\test_images2
\vid_5_25180.jpg: 384x640 (no detections), 263.0ms
image 6/175 C:\Users\shane\Car_Detection\Car-detection-1\test\test_images2
\vid_5_25200.jpg: 384x640 (no detections), 226.6ms
image 7/175 C:\Users\shane\Car_Detection\Car-detection-1\test\test_images2
\vid_5_25220.jpg: 384x640 (no detections), 208.3ms
image 8/175 C:\Users\shane\Car_Detection\Car-detection-1\test\test_images2
\vid_5_25240.jpg: 384x640 (no detections), 215.0ms
image 9/175 C:\Users\shane\Car_Detection\Car-detection-1\test\test_images2
\vid_5_25260.jpg: 384x640 (no detections), 184.3ms
```

In [68]:
```
# Select one image to visualize across models
image_index_test2 = 0  # Assuming you want to visualize the first image's resul

for model_name, results in models_results_test2.items():
    print(f"Displaying results for {model_name}:")
    # Use the plot method on the selected image's results
    image_with_predictions = results[image_index_test2].plot()  # This returns
    # Convert BGR to RGB
    image_with_predictions_rgb = Image.fromarray(image_with_predictions[..., :
    # Display the image with predictions
    display(image_with_predictions_rgb)
```

```
Displaying results for YOLOv8s (Test2):
```

```
Displaying results for YOLOv8s Fine (Test2):
```

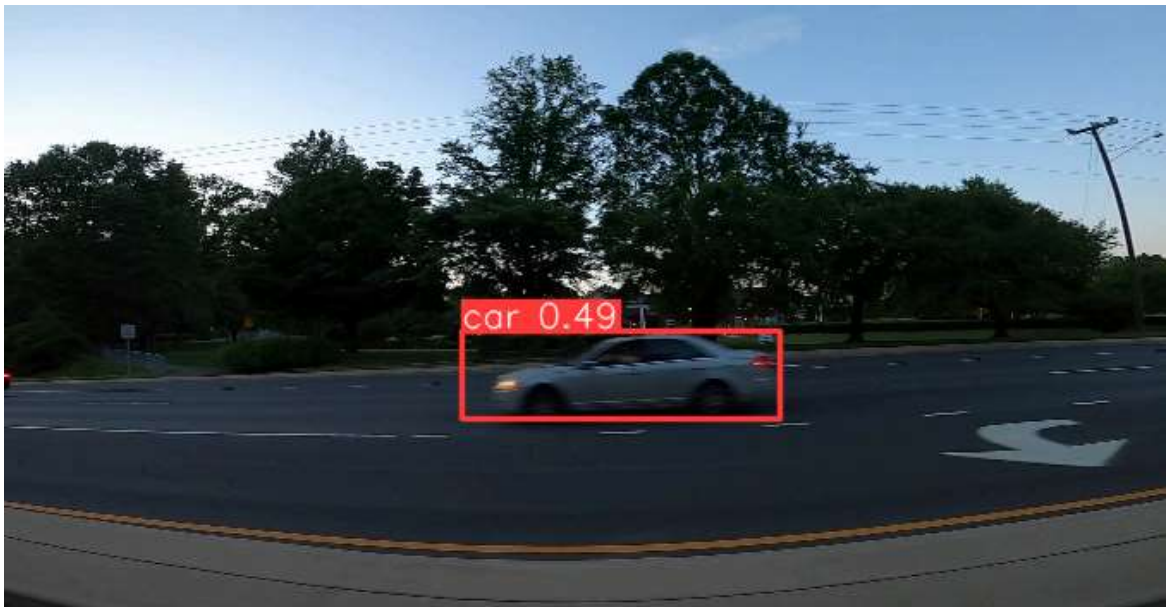
```
Displaying results for Custom (Test2):
```

```
Displaying results for Custom Fine (Test2):
```

In [69]:
```
image_indices_test2 = range(len(models_results_test2[next(iter(models_results_t
random_image_index_test2 = random.choice(list(image_indices_test2))

for model_name, results in models_results_test2.items():
    print(f"Displaying results for {model_name}:")
    # Use the plot method on the randomly selected image's results
    image_with_predictions = results[random_image_index_test2].plot()  # This r
    # Convert BGR to RGB
    image_with_predictions_rgb = Image.fromarray(image_with_predictions[..., :
    # Display the image with predictions
    display(image_with_predictions_rgb)
```

```
Displaying results for YOLOv8s (Test2):
```



In [ ]: