```
import os
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
from google.colab import files
import numpy as np
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.utils import to_categorical
import torch
from torchvision import datasets, transforms, models
from torch.utils.data import DataLoader
import torch.nn as nn
import torch.optim as optim
from torch.optim.lr_scheduler import StepLR
from tqdm import tqdm
import json
import matplotlib.pyplot as plt
```

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.6.17)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.8.30)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.6)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.2.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.10)
```

```
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"davjethedev","key":"5b7d16bb638fa832dd9a17dba9c0a33b"}'}
```

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d kshitij192/cars-image-dataset
```

```
!unzip cars-image-dataset.zip -d /content/car_images
```

Show hidden output

```
dataset_path = '/content/car_images/Cars Dataset'

train_path = os.path.join(dataset_path, 'train')
test_path = os.path.join(dataset_path, 'test')
```

```
for folder in os.listdir(train_path):
    print(f"{folder} -> {len(os.listdir(os.path.join(train_path, folder)))} images")
```

```
Tata Safari -> 441 images
Toyota Innova -> 775 images
Hyundai Creta -> 271 images
Rolls Royce -> 311 images
Swift -> 424 images
Mahindra Scorpio -> 316 images
Audi -> 814 images
```

```
def load_dataset(folder_path):
    records = []
    for label in os.listdir(folder_path):
        label_path = os.path.join(folder_path, label)
```

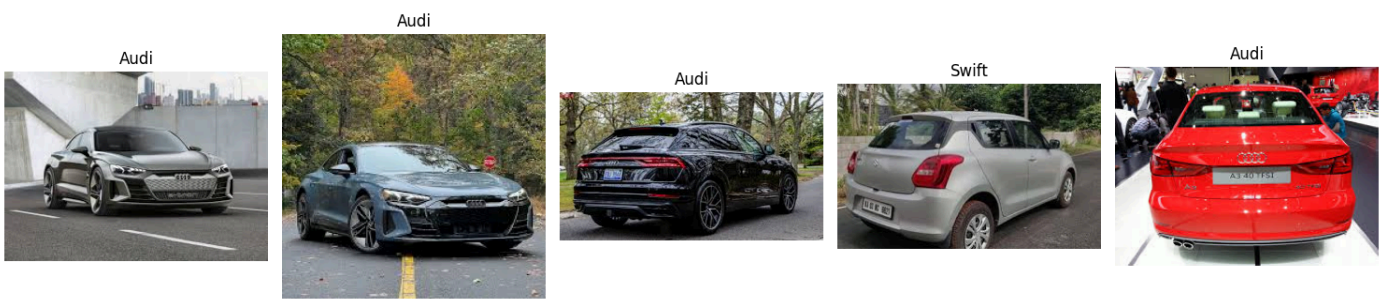
```
        if os.path.isdir(label_path):
            for image_file in os.listdir(label_path):
                image_path = os.path.join(label_path, image_file)
                records.append([image_path, label])
    return pd.DataFrame(records, columns=['file_path', 'label'])
```

```
train_df = load_dataset(train_path)
test_df = load_dataset(test_path)

print("Training Data:")
print(train_df.head())
print("\nTesting Data:")
print(test_df.head())
```

```
Training Data:
                                           file_path        label
0  /content/car_images/Cars Dataset/train/Tata Sa...  Tata Safari
1  /content/car_images/Cars Dataset/train/Tata Sa...  Tata Safari
2  /content/car_images/Cars Dataset/train/Tata Sa...  Tata Safari
3  /content/car_images/Cars Dataset/train/Tata Sa...  Tata Safari
4  /content/car_images/Cars Dataset/train/Tata Sa...  Tata Safari

Testing Data:
                                           file_path        label
0  /content/car_images/Cars Dataset/test/Tata Saf...  Tata Safari
1  /content/car_images/Cars Dataset/test/Tata Saf...  Tata Safari
2  /content/car_images/Cars Dataset/test/Tata Saf...  Tata Safari
3  /content/car_images/Cars Dataset/test/Tata Saf...  Tata Safari
4  /content/car_images/Cars Dataset/test/Tata Saf...  Tata Safari
```

```
def visualize_images(df, num_images=5):
    plt.figure(figsize=(15, 10))
    sample_data = df.sample(num_images)
    for i, (_, row) in enumerate(sample_data.iterrows()):
        img = Image.open(row['file_path'])
        plt.subplot(1, num_images, i + 1)
        plt.imshow(img)
        plt.axis('off')
        plt.title(row['label'])
    plt.tight_layout()
    plt.show()
```

```
visualize_images(train_df, num_images=5)
```




```
def preprocess_images(df, img_size):
    images = []
    for file_path in df['file_path']:
        img = load_img(file_path, target_size=img_size)
        img_array = img_to_array(img) / 255.0
        images.append(img_array)
    return np.array(images)
```

```
IMG_SIZE = (224, 224)

label_encoder = LabelEncoder()
train_df['label_encoded'] = label_encoder.fit_transform(train_df['label'])
test_df['label_encoded'] = label_encoder.transform(test_df['label'])
```

```
train_labels = to_categorical(train_df['label_encoded'])
test_labels = to_categorical(test_df['label_encoded'])


train_images = preprocess_images(train_df, IMG_SIZE)
test_images = preprocess_images(test_df, IMG_SIZE)


print(f"Train Images Shape: {train_images.shape}")
print(f"Test Images Shape: {test_images.shape}")
print(f"Train Labels Shape: {train_labels.shape}")
print(f"Test Labels Shape: {test_labels.shape}")
```

```
Train Images Shape: (3352, 224, 224, 3)
Test Images Shape: (813, 224, 224, 3)
Train Labels Shape: (3352, 7)
Test Labels Shape: (813, 7)
```

```
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
])


train_dir = '/content/car_images/Cars Dataset/train'
test_dir = '/content/car_images/Cars Dataset/test'


train_dataset = datasets.ImageFolder(root=train_dir, transform=transform)
test_dataset = datasets.ImageFolder(root=test_dir, transform=transform)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)


print(f"Classes: {train_dataset.classes}")
```

```
Classes: ['Audi', 'Hyundai Creta', 'Mahindra Scorpio', 'Rolls Royce', 'Swift', 'Tata Safari', 'Toyota Innova']
```

## Functions

[ ] ↳ *5 cells hidden*

## Model 1: ResNet18 Baseline for Car Class Prediction

### Objective

Train a ResNet18 model for car class prediction and integrate gas station-specific recommendations.

---

### Model Overview

- **Architecture**: ResNet18 pretrained on ImageNet.
- **Output Classes**: 7 car classes.
- **Loss Function**: CrossEntropyLoss.
- **Optimizer**: Adam (LR=0.001).
- **Scheduler**: StepLR (step size=5, gamma=0.1).

---

### Dataset Summary

- **Train Size**: Preprocessed car images (224x224, 32 batches).
- **Test Size**: Similar preprocessing as train data.

---

### Results

- **Training Accuracy**: Logged per batch.
- **Test Accuracy**: **92.50%**.

---

## Key Features

1. **Pretrained Transfer Learning**: Fine-tuned ResNet18.
2. **JSON-Based Recommendations**: Predict car class and display car needs, upsell items, and fun facts.

---

[ ] ↳ *8 cells hidden*

# Model 2: Enhanced ResNet18 for Car Classification

## Objective

To improve the classification accuracy and robustness of the ResNet18 model for identifying car types, leveraging advanced techniques like fine-tuning, data augmentation, and optimized training strategies.

---

## Model Overview

- **Base Architecture**: ResNet18 (Pre-trained on ImageNet).
- **Fine-Tuning Strategy**:
  - Unfrozen additional layers for better feature extraction.
  - Fine-tuned the fully connected (FC) layer for classification across 7 car types.
- **Optimization**:
  - **Learning Rate**: Reduced for finer adjustments during training.
  - **Regularization**: Applied weight decay to prevent overfitting.
  - **Scheduler**: CosineAnnealingLR for smoother learning rate adjustments.
- **Data Augmentation**: Random rotation, horizontal flips, and color jitter to increase data diversity and model robustness.

---

## Dataset Summary

- **Classes**: 7 unique car categories (e.g., Audi, Toyota Innova, Rolls Royce).
- **Training Images**: 3,352.
- **Testing Images**: 813.
- **Resolution**: Resized to (224, 224) for input consistency.

---

## Results

- **Test Accuracy**: 96.31%.
- **Enhanced Robustness**: Improved generalization to unseen data due to augmentation and fine-tuning.
- **Performance Stability**: Smooth and consistent learning process through advanced scheduling techniques.

---

## Key Features

1. **Unfrozen Layers**: Leveraged the deeper layers of ResNet18 for domain-specific learning.
2. **Robust Training**: Data augmentation and weight decay reduced overfitting significantly.
3. **Efficient Learning Rate Management**: CosineAnnealingLR facilitated smoother convergence.
4. **High Accuracy**: Achieved 96.31% on test data, surpassing baseline performance.
5. **Scalability**: Framework is adaptable for integration into object detection pipelines (e.g., YOLO for video-based analysis).

Below is the step-by-step implementation of Model 2 with the suggested changes.

```
model_2 = models.resnet18(pretrained=True)
```

```
for param in model_2.layer4.parameters():
    param.requires_grad = True
```

```
num_ftrs = model_2.fc.in_features
model_2.fc = nn.Linear(num_ftrs, len(train_dataset.classes))
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model_2 = model_2.to(device)
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is depreca
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f370
100%|██████████| 44.7M/44.7M [00:00<00:00, 139MB/s]
```

Apply augmentations to enhance the dataset's variability and improve generalization.

```
from torchvision import transforms


train_transforms = transforms.Compose([
    transforms.RandomResizedCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
])

test_transforms = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),
])


train_dataset = datasets.ImageFolder(train_dir, transform=train_transforms)
test_dataset = datasets.ImageFolder(test_dir, transform=test_transforms)

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

Lower the learning rate and add weight decay for regularization.

```
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model_2.parameters(), lr=0.0001, weight_decay=1e-4)


scheduler = optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=10)
```

Use the same training and evaluation functions but apply them to `model_2`.

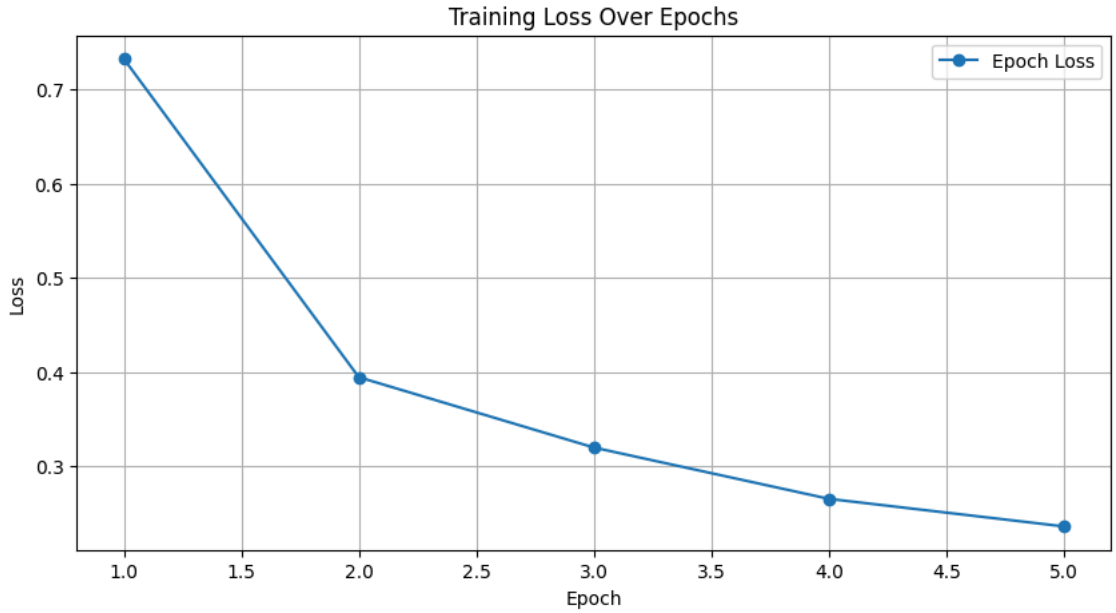
```
epoch_loss_2, batch_loss_history_2, epoch_accuracy_2 = train_model(
    model_2, train_loader, criterion, optimizer, device, epochs=5, print_every=5
)
```

Show hidden output

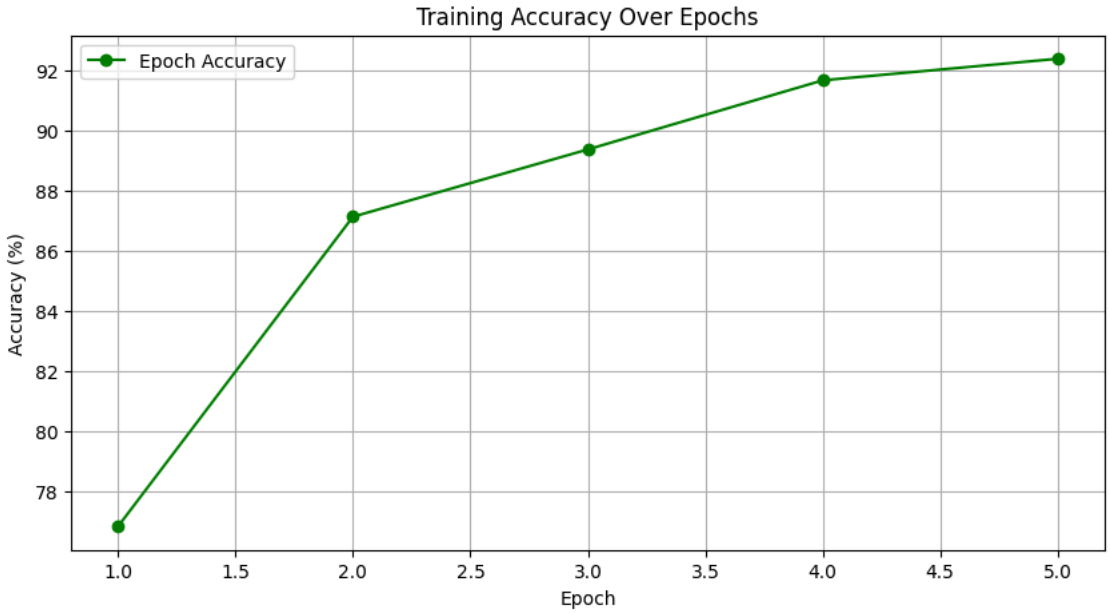
```
torch.save(model_2.state_dict(), "/content/model_2_state_dict.pth")


evaluate_model(model_2, test_loader, device)
```

```
Test Accuracy: 96.31%
```

```
plot_training_metrics(epoch_loss_2, batch_loss_history_2, epoch_accuracy_2)
```


Training Loss Over Epochs
Epoch Loss
Loss
Epoch
0.7
0.6
0.5
0.4
0.3
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

Training Loss Per Batch
Batch Loss
Loss
Batch
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500


Training Accuracy Over Epochs
Epoch Accuracy
Accuracy (%)
Epoch
92
90
88
86
84
82
80
78
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

Prediction car

```
with open('/content/car_recommendations.json', 'r') as f:
    car_recommendations = json.load(f)
```

```
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving 116.jpg to 116.jpg
```

```
image_path = next(iter(uploaded.keys()))
class_names = ["Audi", "Hyundai Creta", "Mahindra Scorpio", "Rolls Royce", "Swift", "Tata Safari", "Toyota Innova"]
```

```
predicted_label = predict_car_class(model_2, image_path, class_names, device)
display_recommendation(predicted_label, car_recommendations)
```

```
Predicted Class: Tata Safari
Needs:
- Recommended oil: 15W-50
- Heavy-duty windshield washer fluid

Upsell Recommendations:
- Portable tire inflator
- Emergency roadside toolkit

Fun Fact: The Tata Safari is an iconic Indian SUV that blends luxury with off-road capability.
```

## YOLOv5 Model Implementation - First Try

```
!git clone https://github.com/ultralytics/yolov5.git
%cd yolov5
!pip install -r requirements.txt
```

Show hidden output

```
import torch
import cv2
from google.colab import files
from IPython.display import display, Video
from pathlib import Path
import json
```

```
!curl -L -o yolov5s.pt https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s.pt
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 14.0M  100 14.0M    0     0  12.0M      0  0:00:01  0:00:01 --:--:-- 36.5M
```

```
uploaded = files.upload()
video_path = next(iter(uploaded.keys()))
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving test_video.mp4 to test_video.mp4
```

```
with open('/content/car_recommendations.json', 'r') as f:
    car_recommendations = json.load(f)
```

```
def detect_and_annotate(video_path, output_path="output.mp4"):

    model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5s.pt')
    model.conf = 0.25

    cap = cv2.VideoCapture(video_path)
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    fps = int(cap.get(cv2.CAP_PROP_FPS))
```

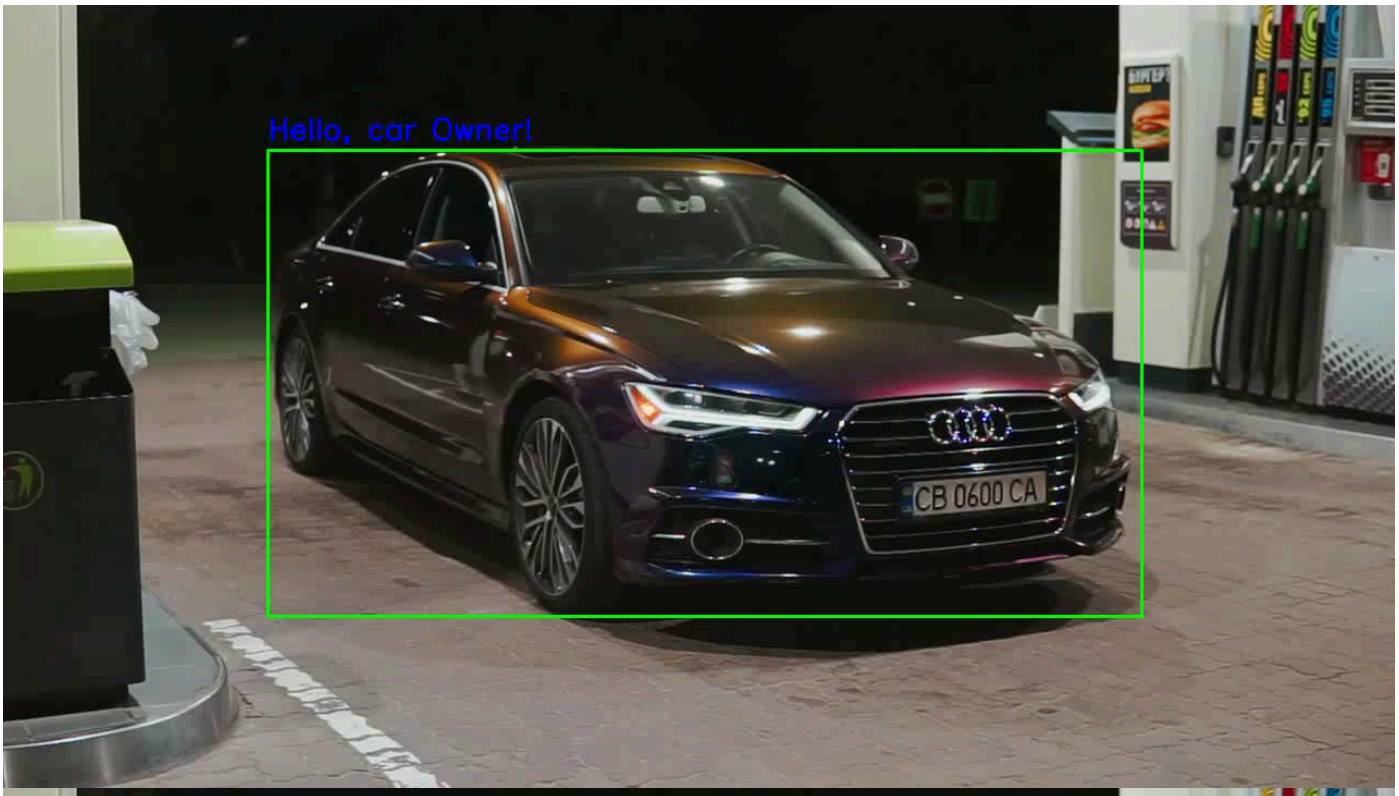
```
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

    with open("/content/car_recommendations.json", "r") as f:
        recommendations = json.load(f)

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

        results = model(frame)


        for *box, conf, cls in results.xyxy[0]:
            label = results.names[int(cls)]
            if label == "car":
                x1, y1, x2, y2 = map(int, box)
                cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(frame, f"Hello, {label} Owner!", (x1, y1 - 10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0), 2)

                if label in recommendations:
                    rec = recommendations[label]
                    cv2.putText(frame, f"Needs: {', '.join(rec['needs'])}", (x1, y1 + 30),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 255), 1)
                    cv2.putText(frame, f"Upsell: {', '.join(rec['upsell'])}", (x1, y1 + 50),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 255), 1)

        out.write(frame)

    cap.release()
    out.release()
    print(f"Video processed. Saved at {output_path}")
```

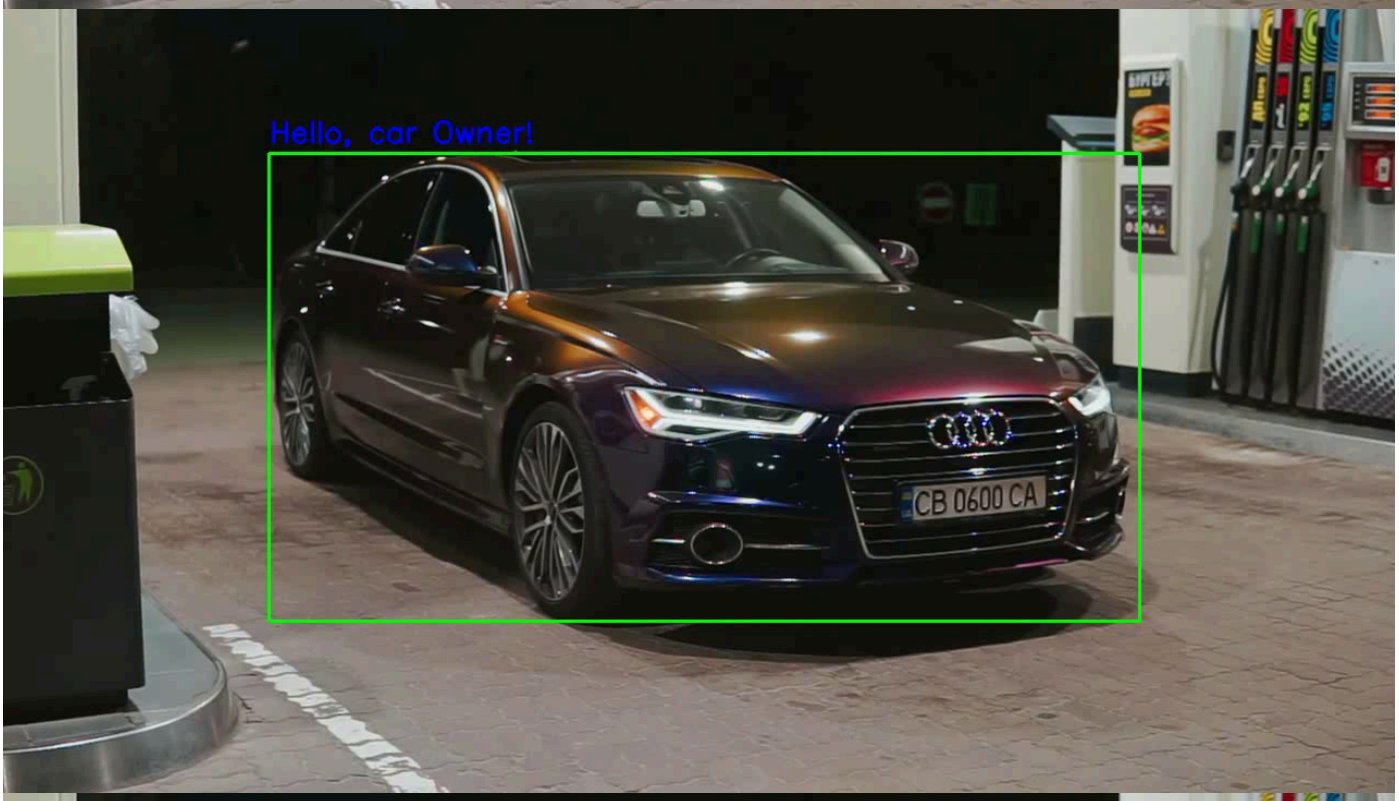
```
detect_and_annotate(video_path, output_path="annotated_video.mp4")
```

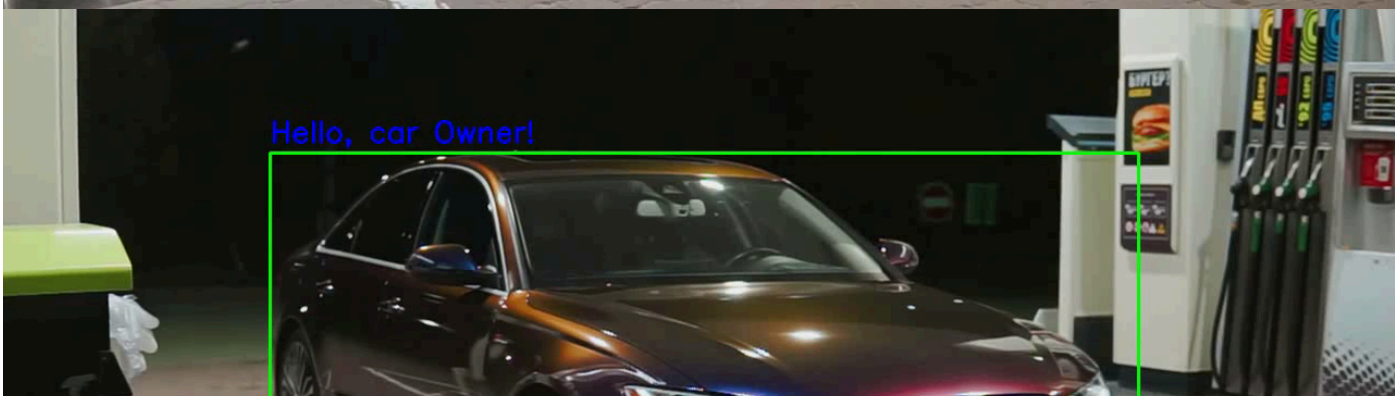
```
Using cache found in /root/.cache/torch/hub/ultralytics_yolov5_master
YOLOv5 🚀  2024-11-14 Python-3.10.12 torch-2.5.0+cu121 CPU

Fusing layers...
YOLOv5s_v6 summary: 213 layers, 7225885 parameters, 0 gradients
Adding AutoShape...
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
```

```
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
/root/.cache/torch/hub/ultralytics_yolov5_master/models/common.py:892: FutureWarning: `torch.cuda.amp.autocast(args...)` is
  with amp.autocast(autocast):
```

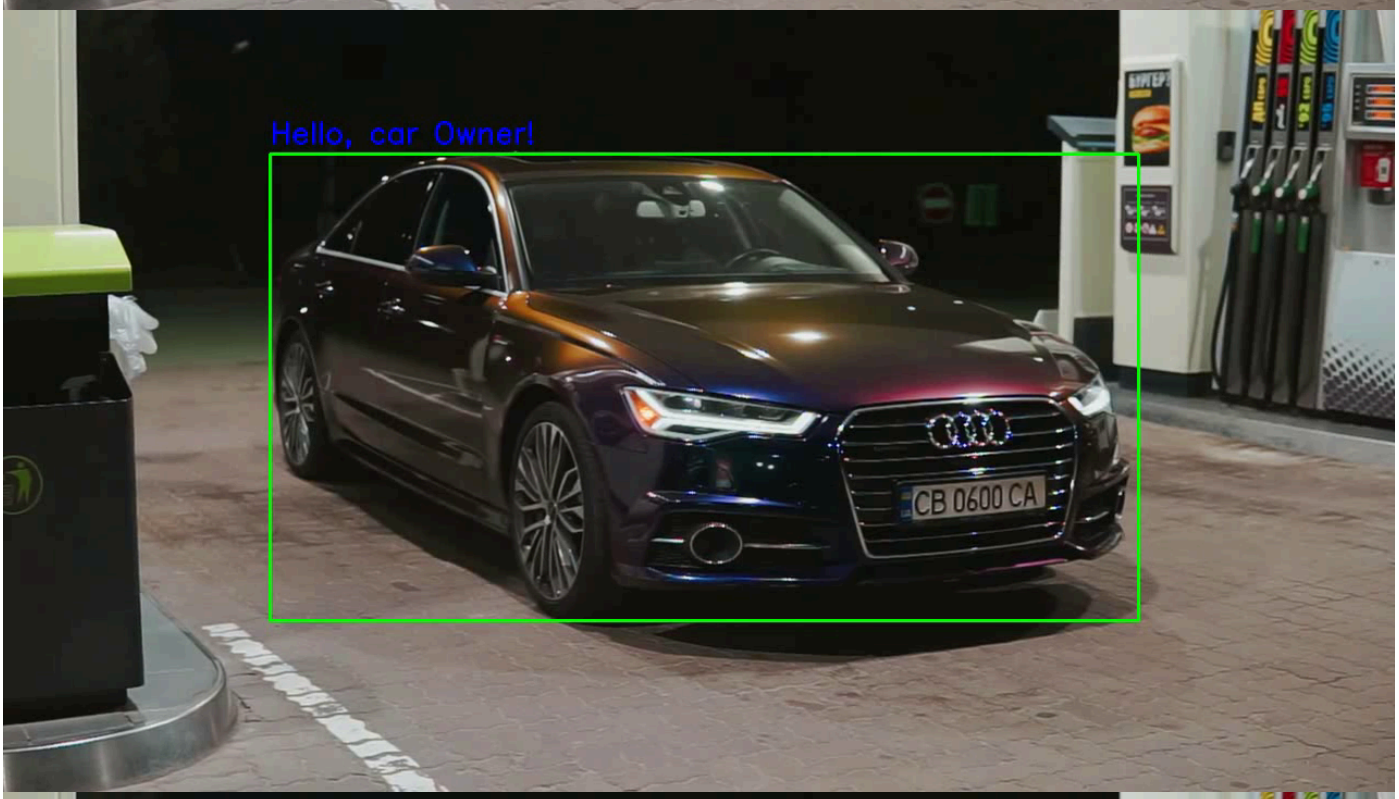
```
from google.colab.patches import cv2_imshow

cap = cv2.VideoCapture("annotated_video.mp4")
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
    cv2_imshow(frame)

cap.release()
```

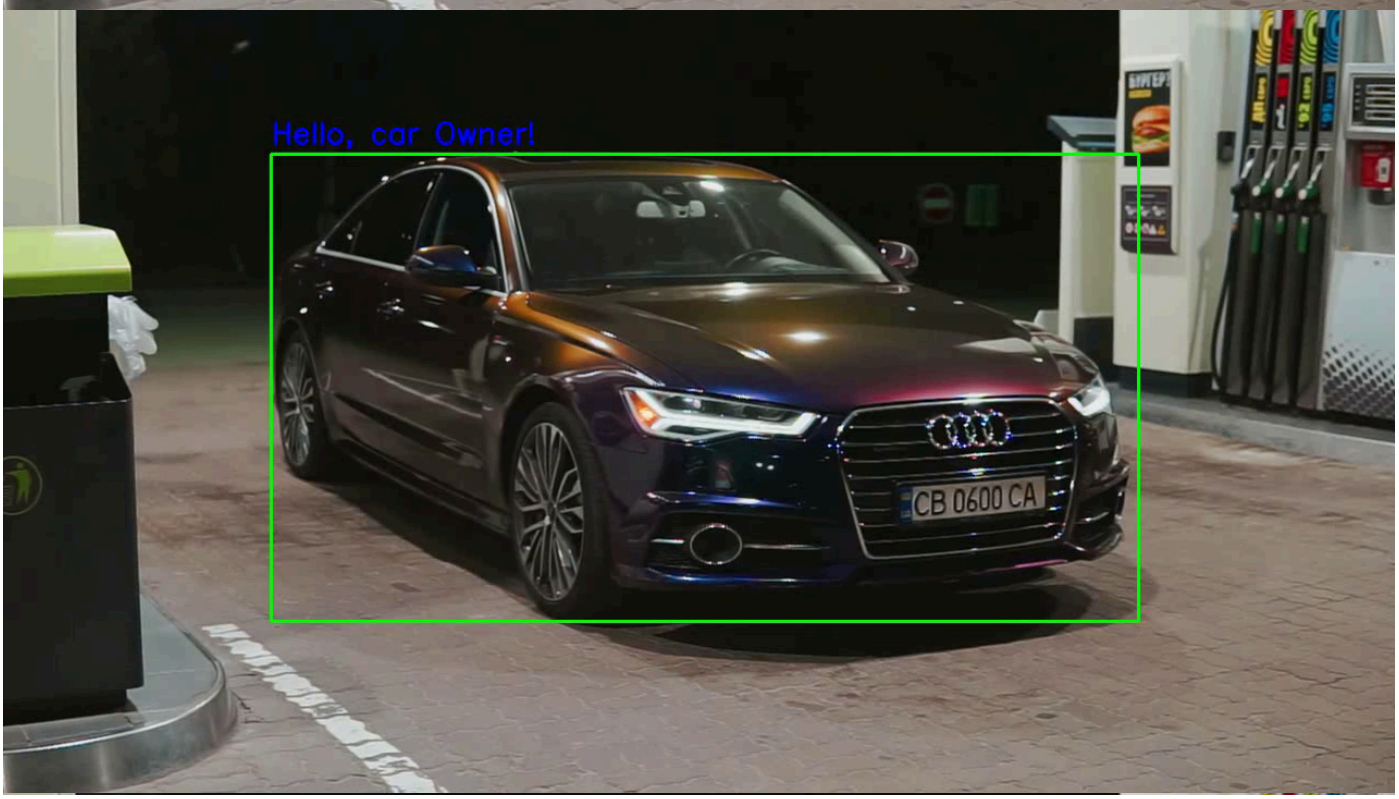

Hello, car Owner!
CB 0600 CA

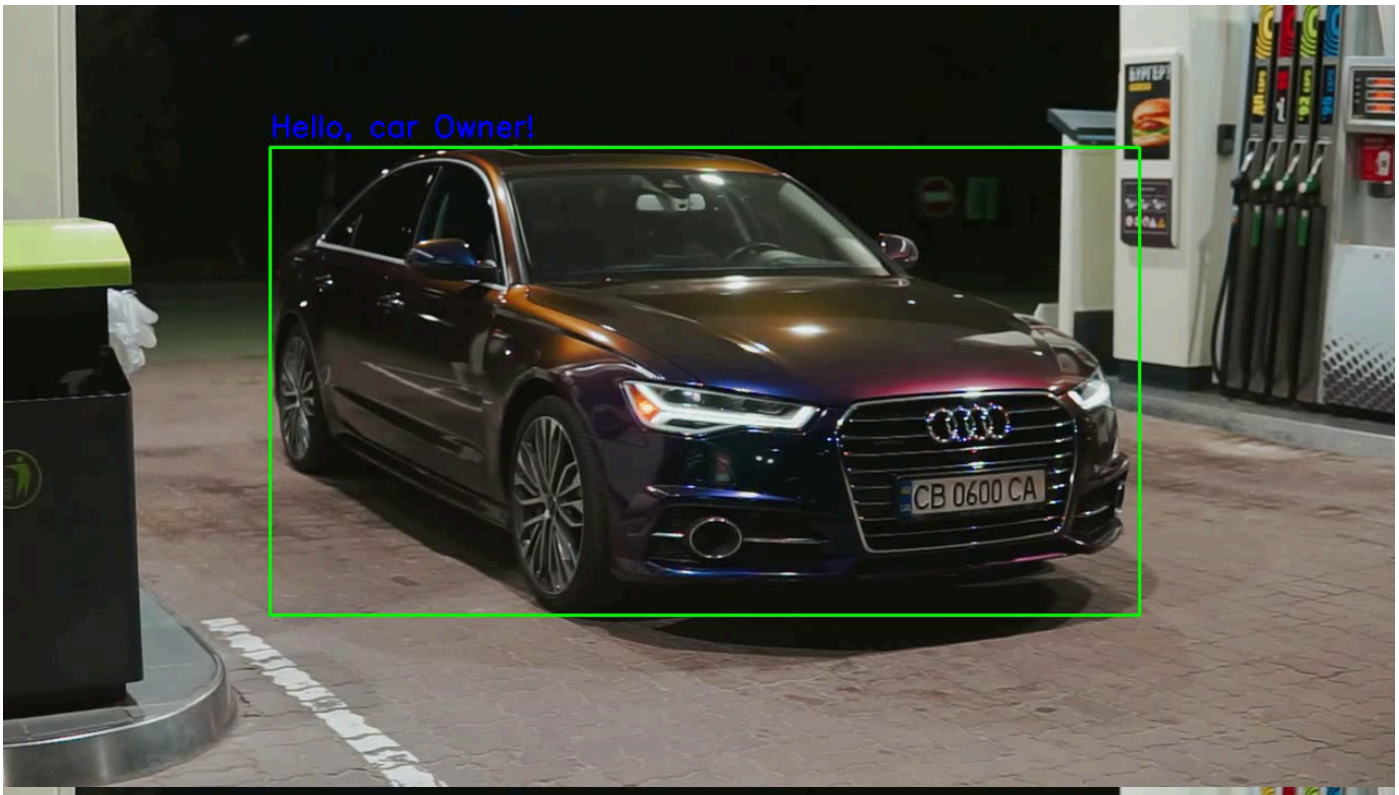

Hello, car Owner!
CB 0600 CA

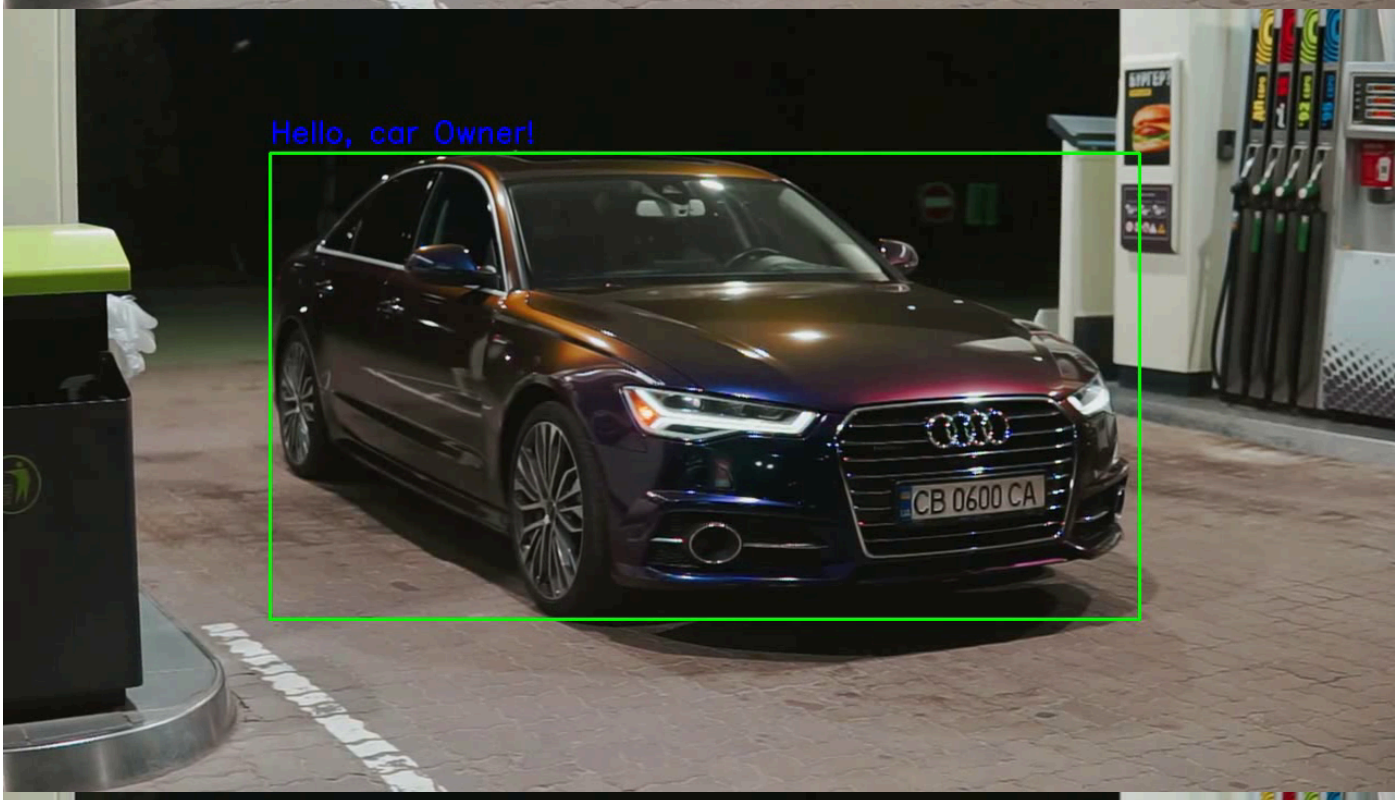

Hello, car Owner!

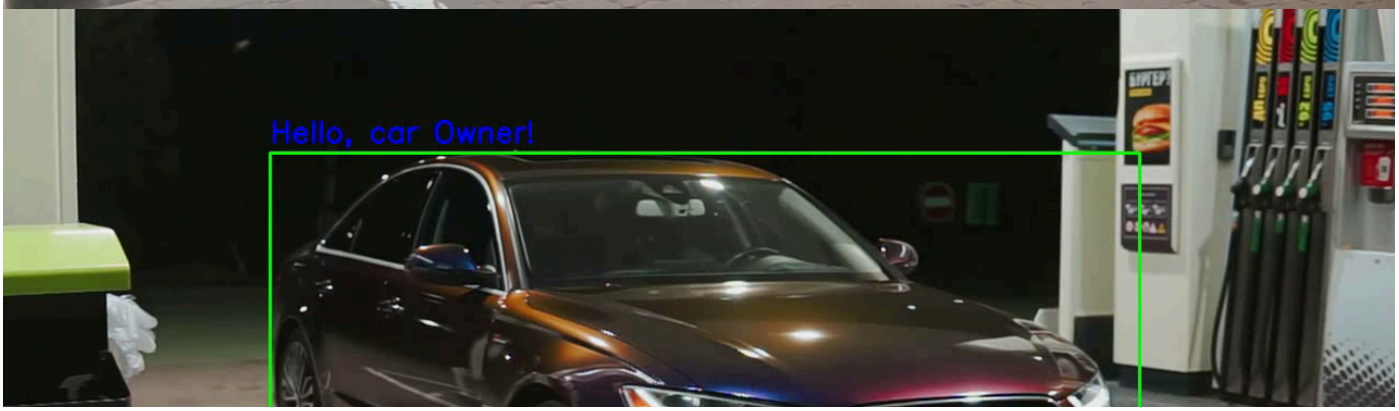
CB 0600 CA

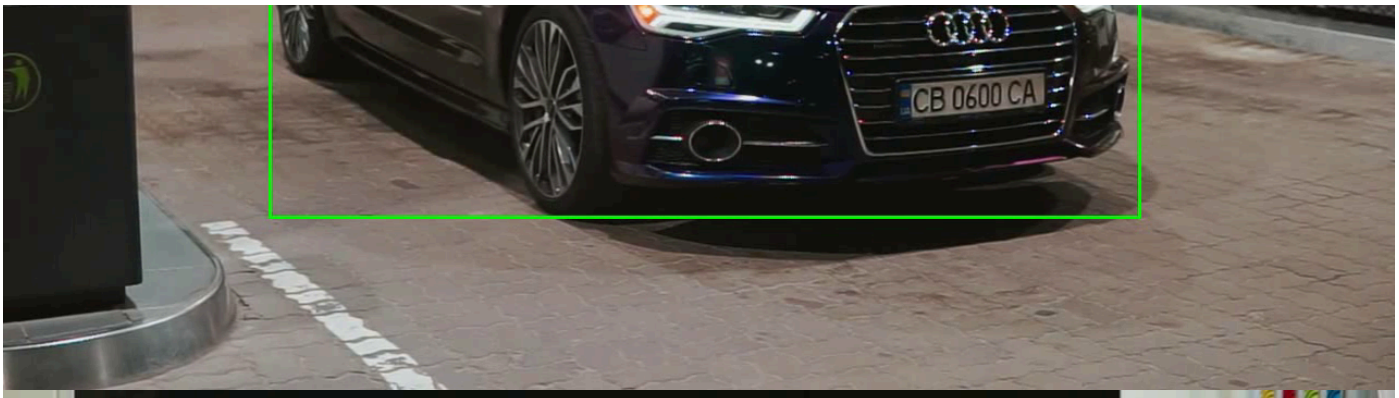

Hello, car Owner!
CB 0600 CA

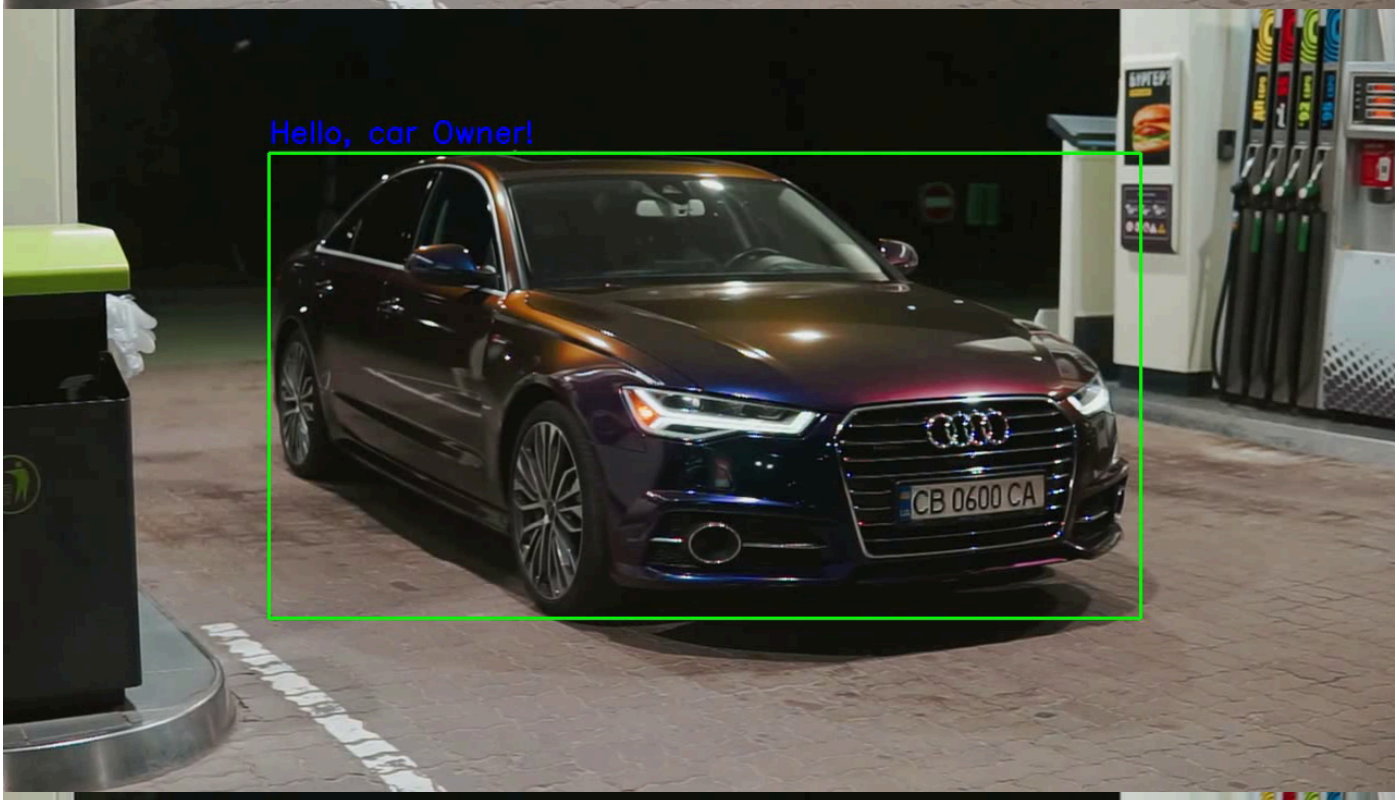

Hello, car Owner!
CB 0600 CA

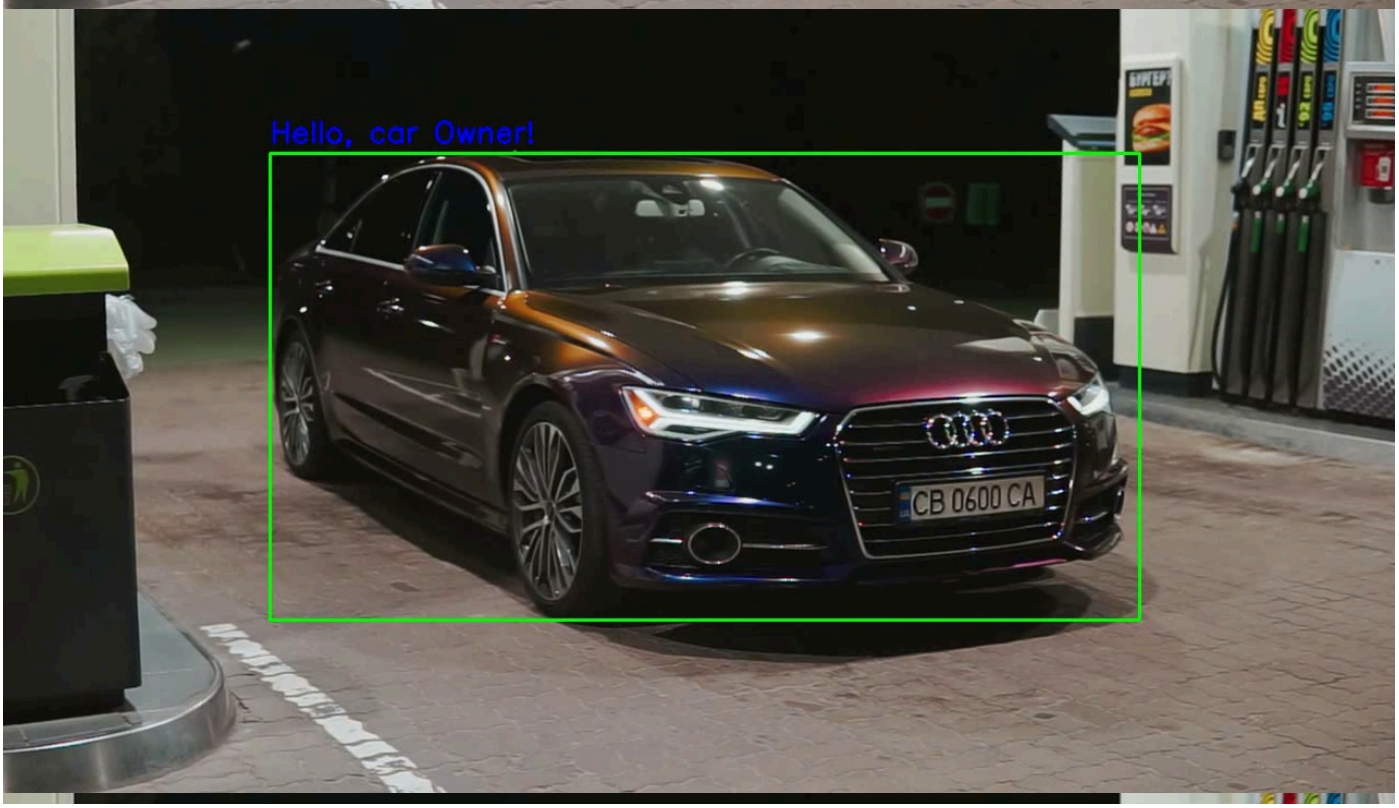

Hello, car Owner!
CB 0600 CA

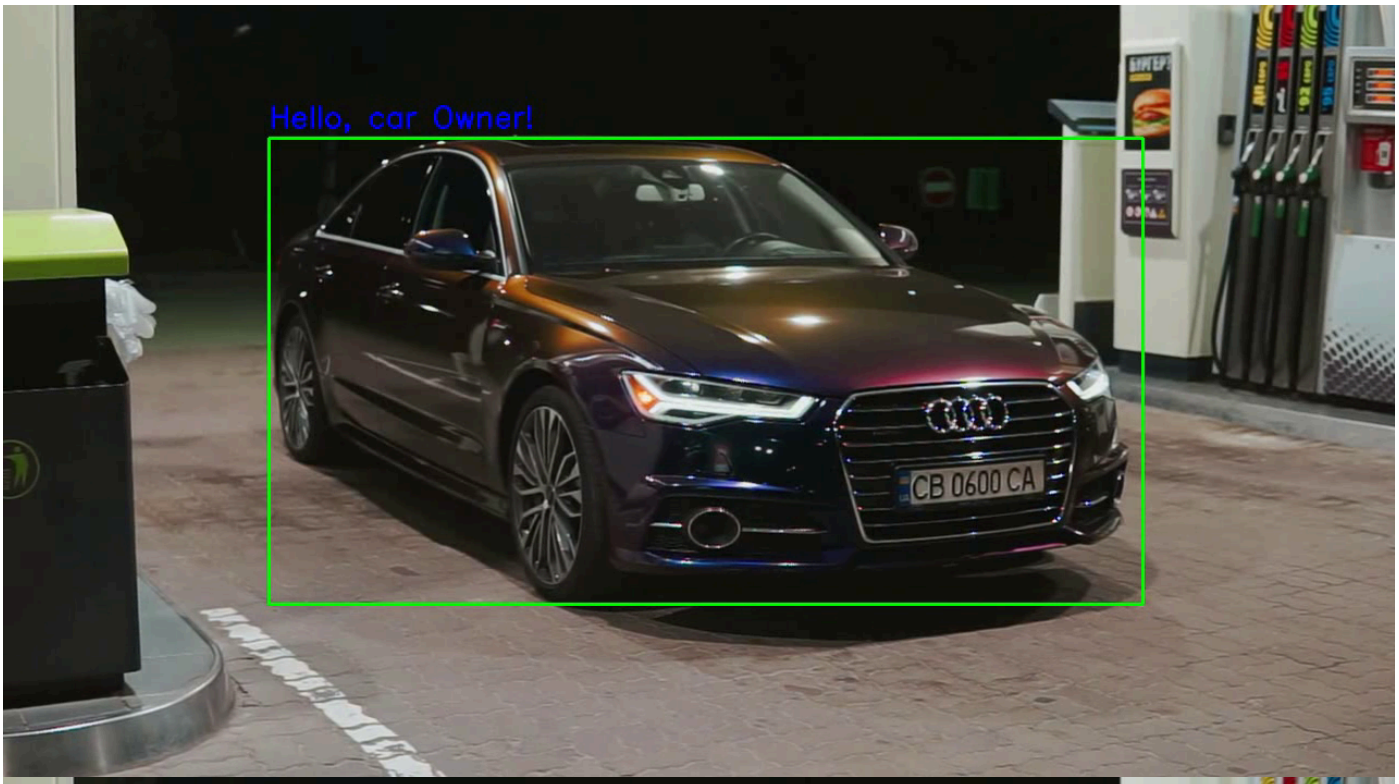

Hello, car Owner!
CB 0600 CA

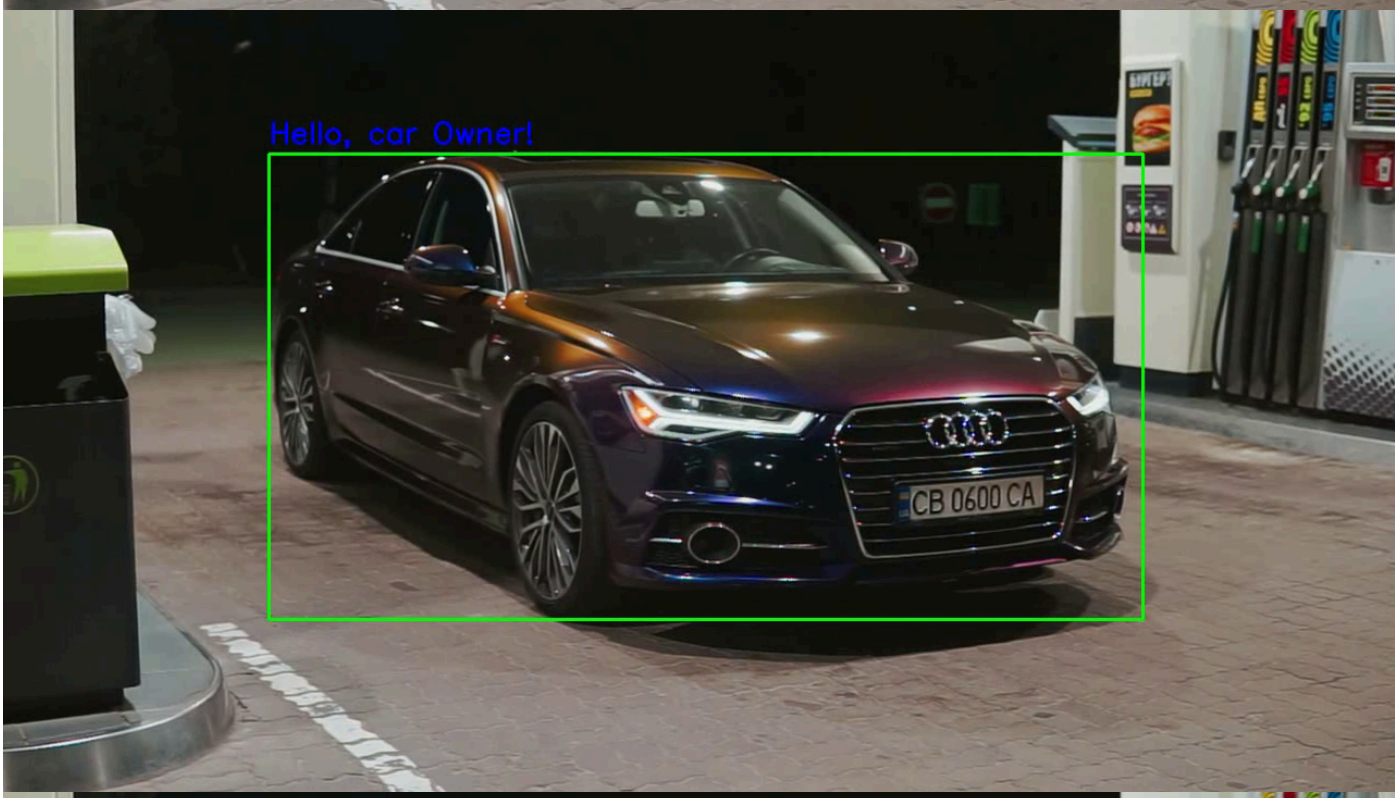

Hello, car Owner!

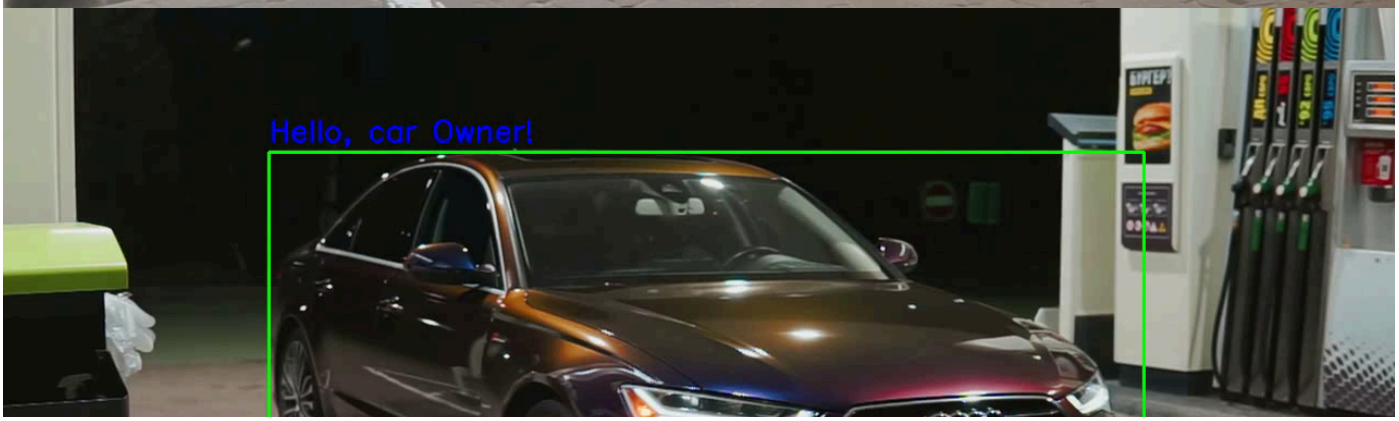

CB 0600 CA

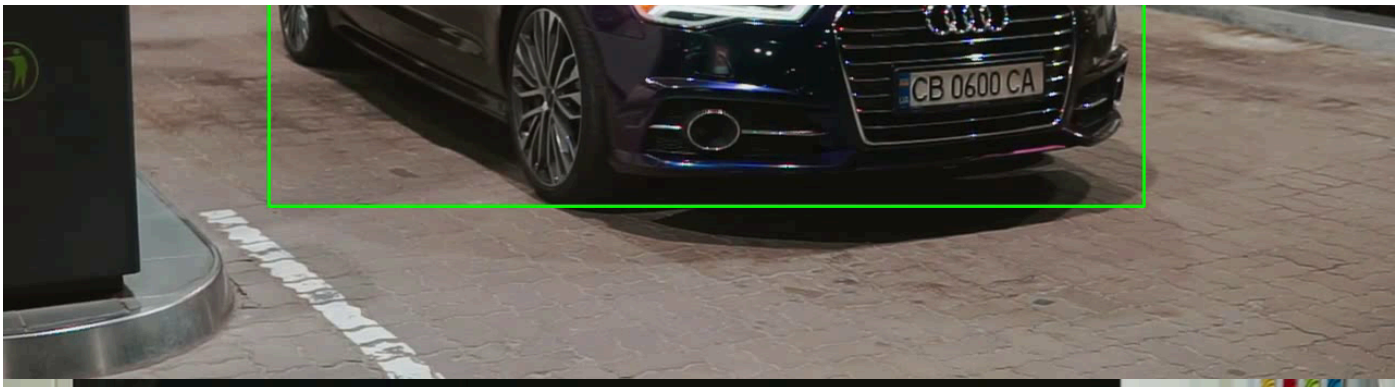

Hello, car Owner!
CB 0600 CA

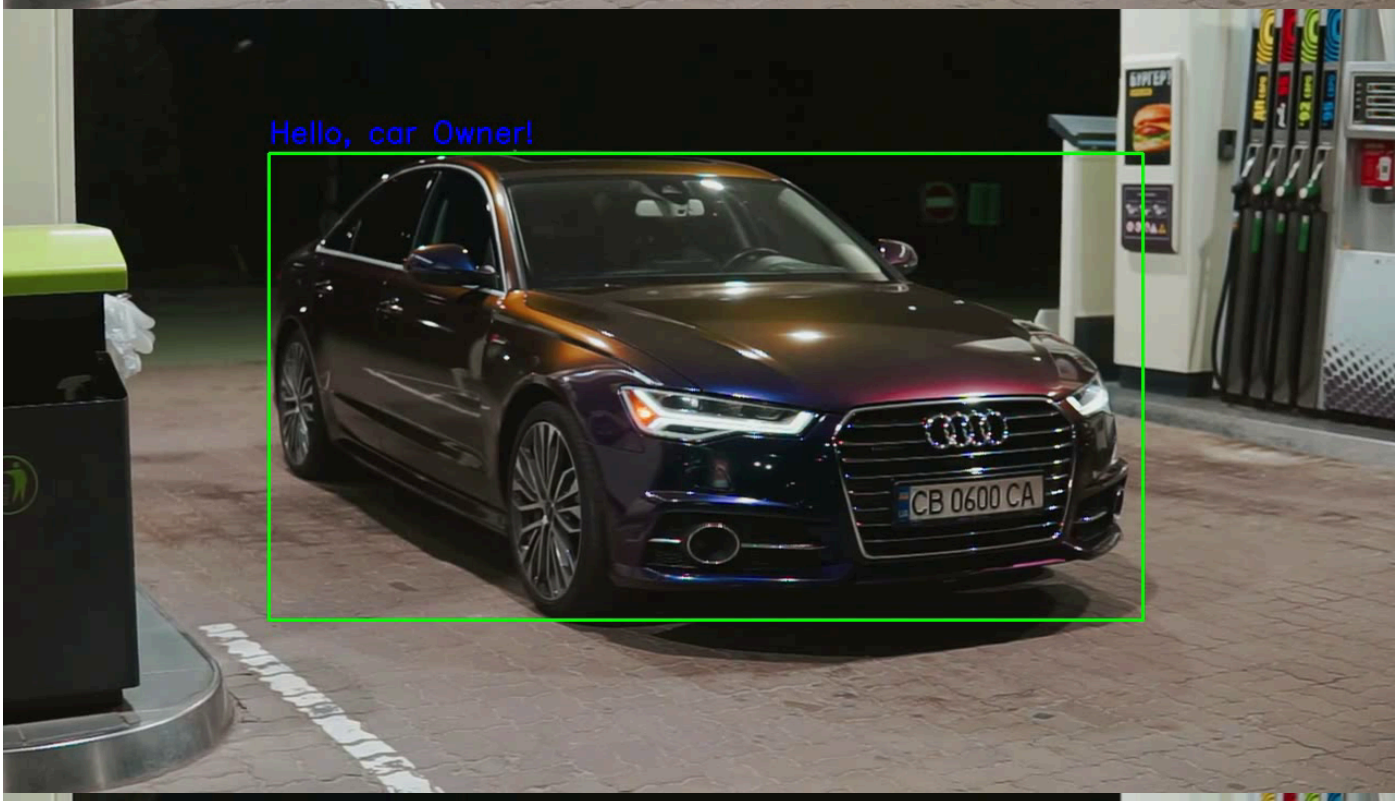

Hello, car Owner!
CB 0600 CA

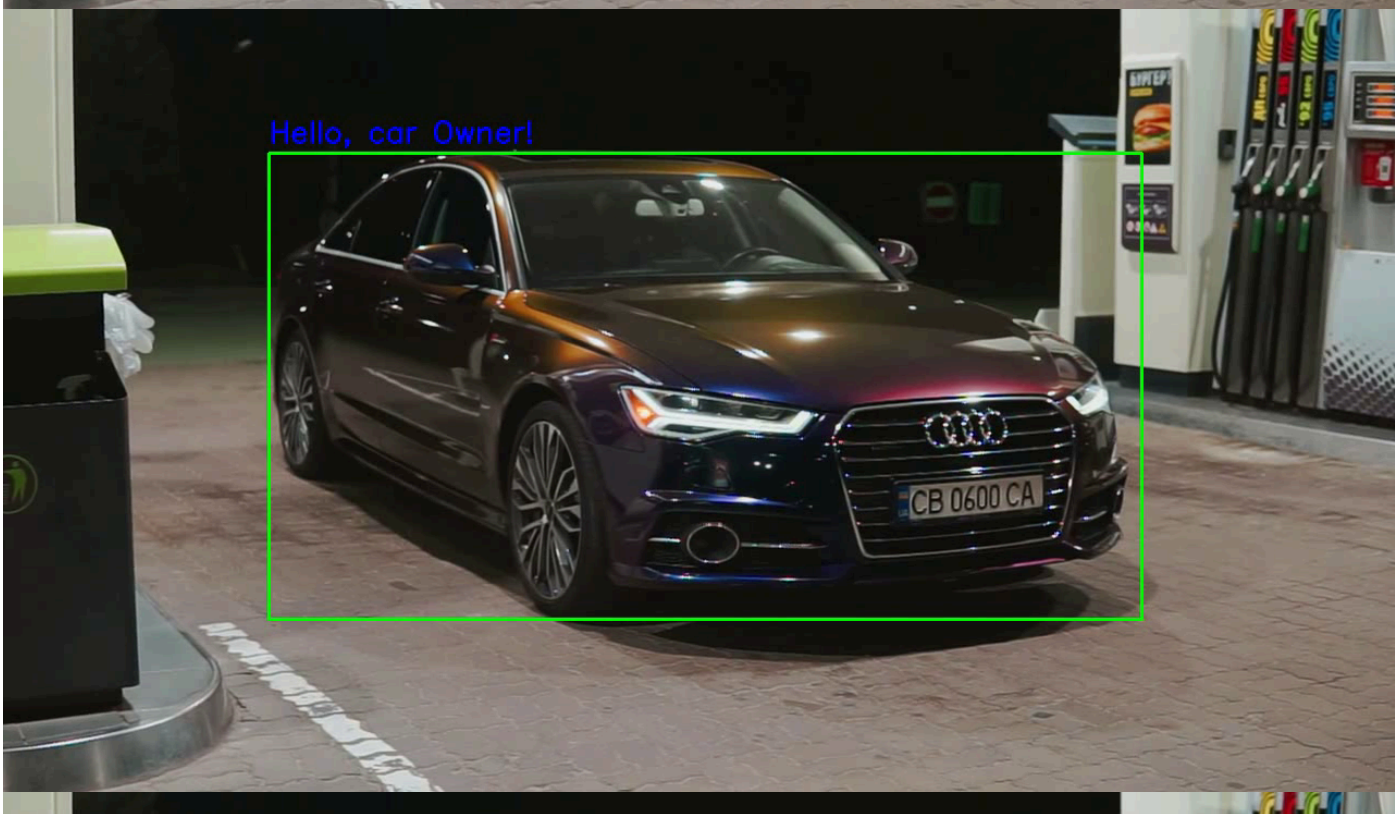

Hello, car Owner!
CB 0600 CA

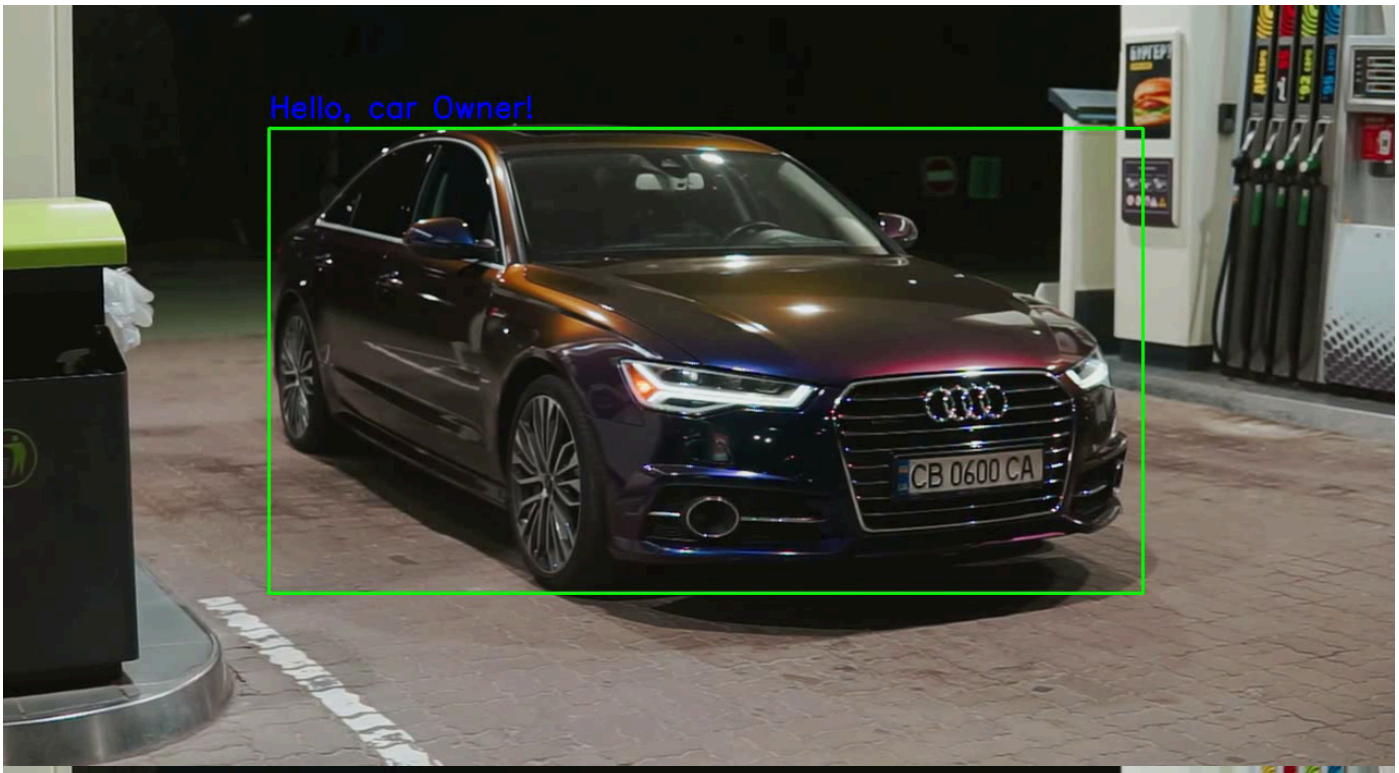

Hello, car Owner!
CB 0600 CA

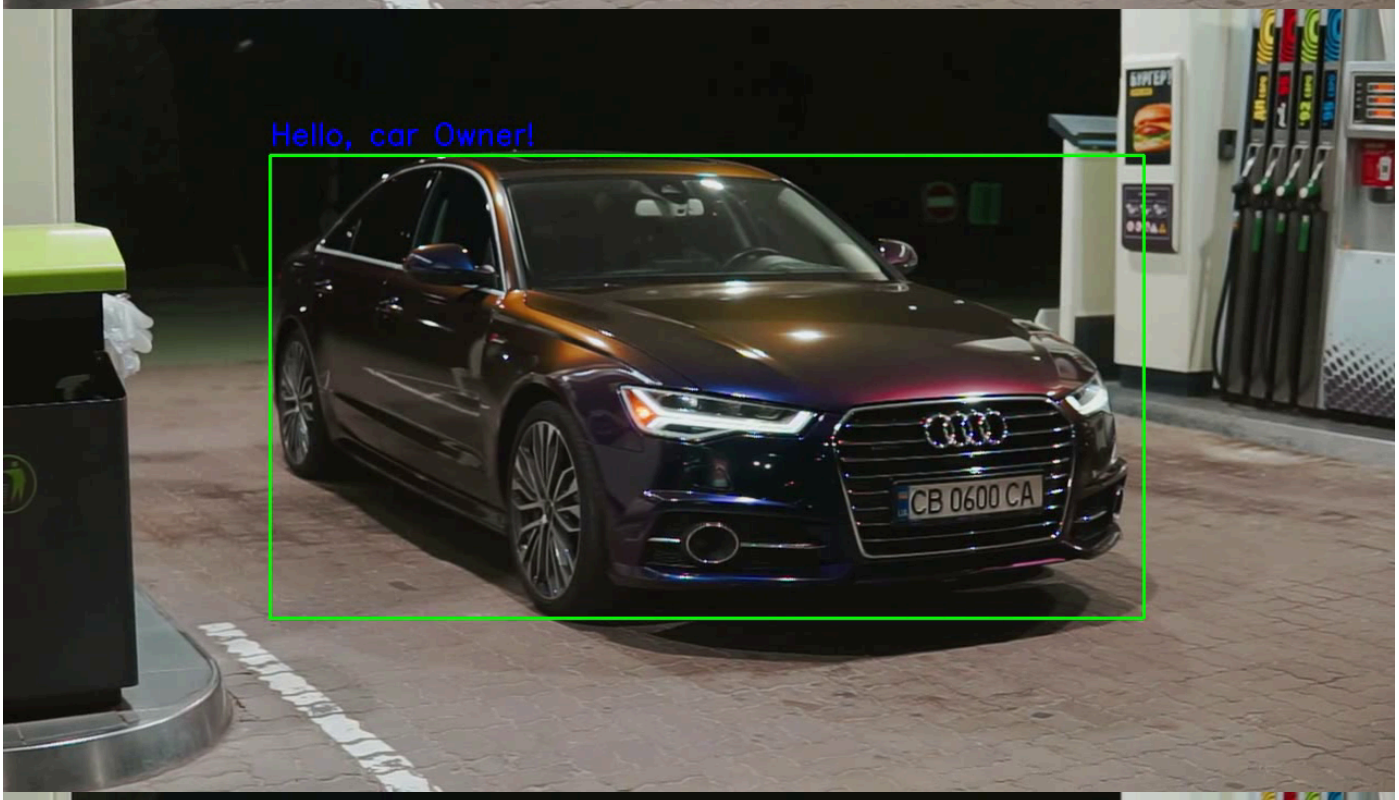

Hello, car Owner!

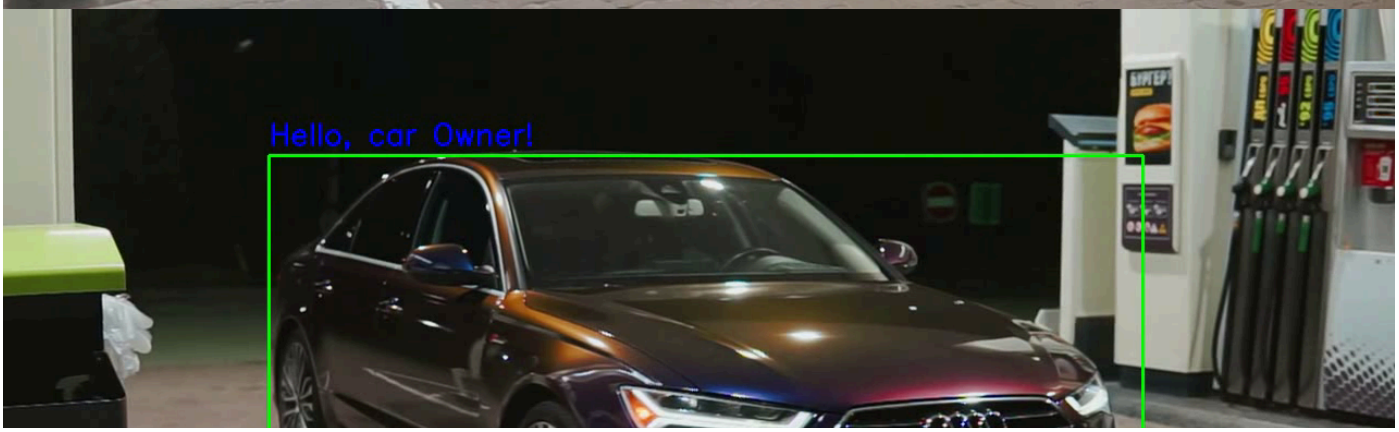

CB 0600 CA


Hello, car Owner!
CB 0600 CA


Hello, car Owner!
CB 0600 CA


Hello, car Owner!
CB 0600 CA


Hello, car Owner!
CB 0600 CA


Hello, car Owner!