```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Download pret-trained YOLOv8n model
!pip install --upgrade ultralytics
from ultralytics import YOLO

# Load the YOLOv8n model
model = YOLO('yolov8n.pt')
```

```
Collecting ultralytics
  Downloading ultralytics-8.4.6-py3-none-any.whl.metadata (38 kB)
Requirement already satisfied: numpy>=1.23.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.12
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.12/dis
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.12/dis
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.12/
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.12/dist
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.1
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.12/dis
Requirement already satisfied: polars>=0.20.0 in /usr/local/lib/python3.12/di
Collecting ultralytics-thop>=2.0.18 (from ultralytics)
  Downloading ultralytics_thop-2.0.18-py3-none-any.whl.metadata (14 kB)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/py
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dis
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/d
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packa
Requirement already satisfied: fsspec>=0.8.5 in /usr/local/lib/python3.12/dis
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-pac
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.1
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/d
Downloading ultralytics-8.4.6-py3-none-any.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 41.4 MB/s eta 0:00:00
Downloading ultralytics_thop-2.0.18-py3-none-any.whl (28 kB)
Installing collected packages: ultralytics-thop, ultralytics
Successfully installed ultralytics-8.4.6 ultralytics-thop-2.0.18
Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralyti
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.4.0/yo
```

Inputs to the Yolo network:

Images are of different sizes such as:

768 X 1024 pixels

1024 X 768 pixels

1024 X 683 pixels

Do we resize the images? In fact, Yolo will do that once we specify the image size in config params: INPUT_WIDTH = 640 INPUT_HEIGHT = 640

Outputs of the Yolo network:

There is only 1 class i.e. the name plate of the vehicle. There are only two possibilities, either the name plate is present in the box or not present and this is covered by the Confidence score. Thus no output is needed for the class. If confidence score is 1, it means that name plate is present in the box. If confidence score is 0, it means that name plate is NOT present in the box. Thus Confidence score and 4 parameters representing bounding box, are the total 5 outputs of the network.

```
# Config for the model
import cv2
import numpy as np

# input image width and height
INPUT_WIDTH = 640
INPUT_HEIGHT = 640
#[0.9 , bb1,bb2,bb3,bb4, pc1, pc2,....,pc80]
# probability threshold to filter boxes with object or no object
OBJECT_SCORE_THRESHOLD = 0.5
#3 Feature Maps -> SxS grids -> 3 Anchor boxes
# # probability threshold to detect and Assign Class
CLASS_CONFIDENCE_THRESHOLD = 0.45

# IOU AREA Threshold to suppress redundant boxes using NMS
NMS_THRESHOLD = 0.45 #Non Max

# Text parameters used for annotating label on Image
FONT_FACE = cv2.FONT_HERSHEY_SIMPLEX
FONT_SCALE = 0.4
THICKNESS = 1
BOX_COLOR = (0,255,255)
FONT_COLOR= (0,0,0)
```

```
# Train the model
model = YOLO("yolov8n.pt")
results = model.train(data="dataset.yaml", epochs=10, imgsz=640, batch=-1)
```

```
Results saved to /content/runs/detect/train
```

```
# Print metrics for the trained model
print(results)
```

```
save_dir: PosixPath('/content/runs/detect/train')
speed: {'preprocess': 2.1072334000564297, 'inference': 190.83674429994062,
stats: {'tp': [], 'conf': [], 'pred_cls': [], 'target_cls': [], 'target_img
task: 'detect'
```

```
# Function to predict bounding box using the model
def predict_bbox(filename):
  #source = '/content/drive/MyDrive/License/images/test/0cacb08195a3e2d7.jpg

  results = model.predict(filename, save=True, conf=0.5)

  return results[0].boxes
  #print(boxes)
```

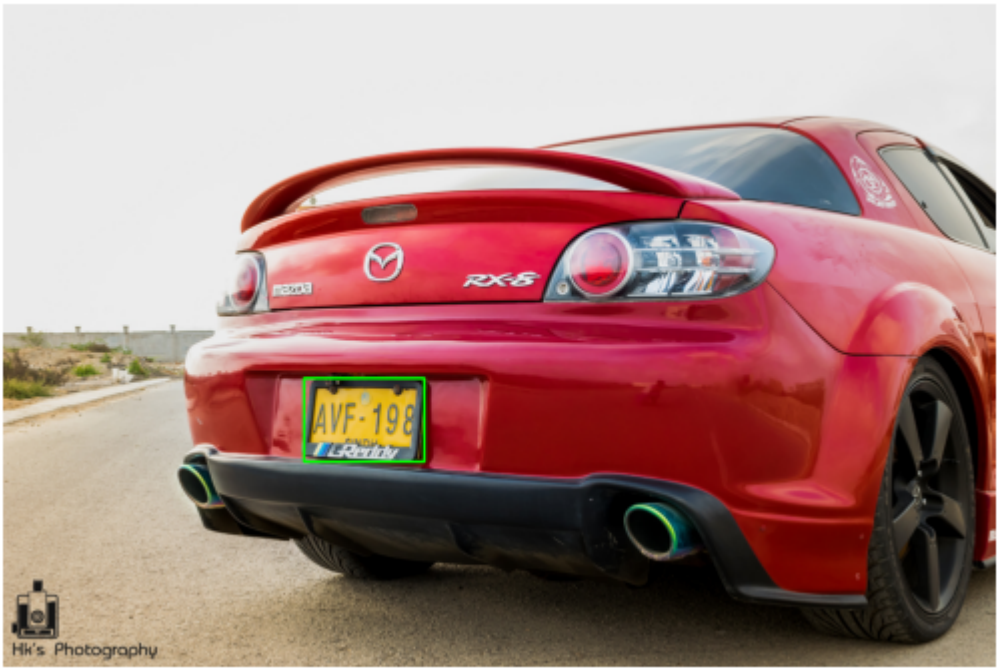
```
import cv2
import matplotlib.pyplot as plt

# Function to show bounding box on the image
def show_bbox(filename, boxes):

  # Get coordinates of bounding box
  coords = boxes[0].xyxy[0].tolist()
  x1, y1, x2, y2 = [round(x) for x in coords]

  # Read image
  image = cv2.imread(filename)

  # OpenCV reads in BGR, matplotlib displays in RGB
  image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

  # Draw box (ensure it's on the RGB image if you converted it)
  cv2.rectangle(image_rgb, (x1, y1), (x2, y2), (0, 255, 0), 2)

  # Display
  plt.imshow(image_rgb)
  plt.axis('off') # Hide axes
  plt.show()
```

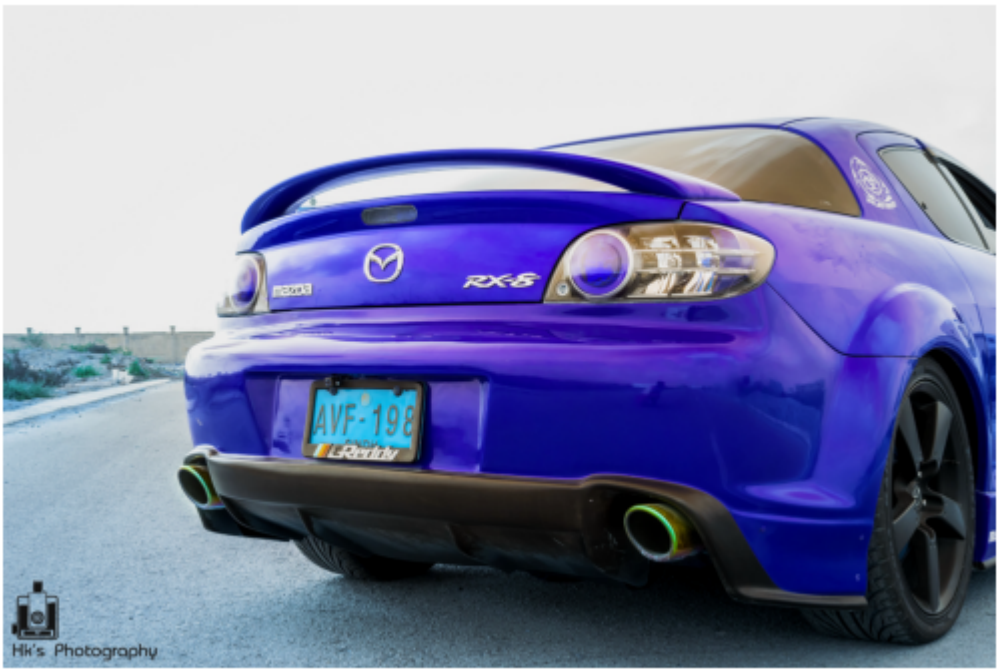
```
# Function to apply Gaussian blur
def apply_gaussian_blur(filename, boxes):

  # Get coordinates of bounding box
  coords = boxes[0].xyxy[0].tolist()
  x1, y1, x2, y2 = [round(x) for x in coords]

  x1= 10
  x2 = 20
  y1 = 10
  y2 = 20

  # Read image
  image = cv2.imread(filename)
```

```
    # Extract the ROI from the image
    roi = image[y1:y2, x1:x2]

    # Apply Gaussian blur to the ROI
    blurred_roi = cv2.GaussianBlur(roi, (11, 11), 0)

    image[y1:y2, x1:x2] = blurred_roi

    # Display the image with the blurred ROI
    plt.imshow(image)
    plt.axis('off')
    plt.show()
```

```
# Test file 1
filename = '/content/drive/MyDrive/License/images/test/0cacb08195a3e2d7.jpg'
boxes = predict_bbox(filename)
show_bbox(filename, boxes)
apply_gaussian_blur(filename, boxes)
```

```
image 1/1 /content/drive/MyDrive/License/images/test/0cacb08195a3e2d7.jpg: 44
Speed: 3.2ms preprocess, 165.2ms inference, 1.2ms postprocess per image at sh
Results saved to /content/runs/detect/predict
```



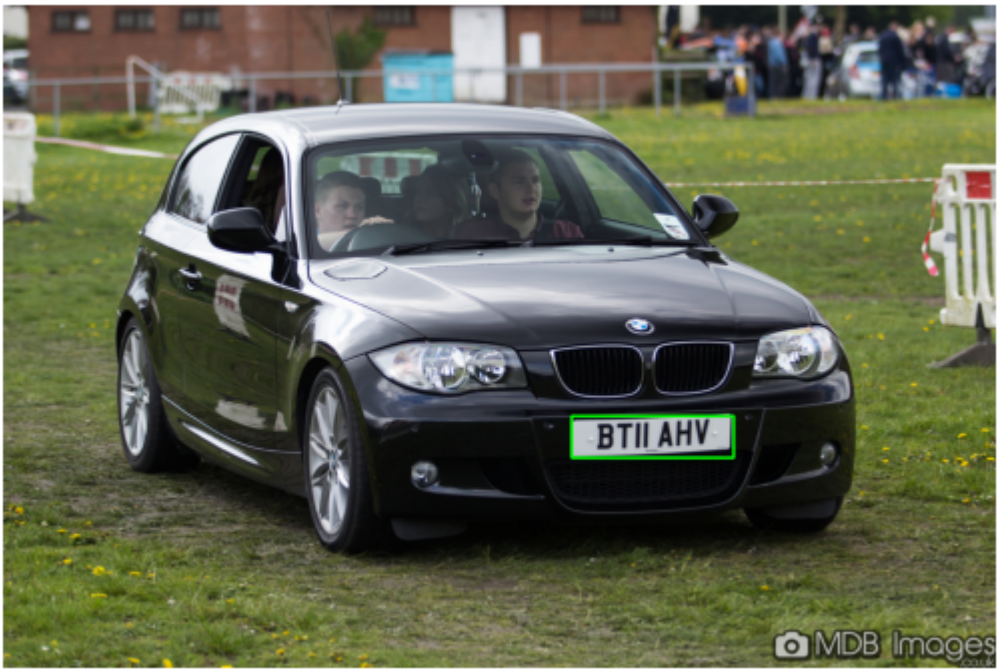


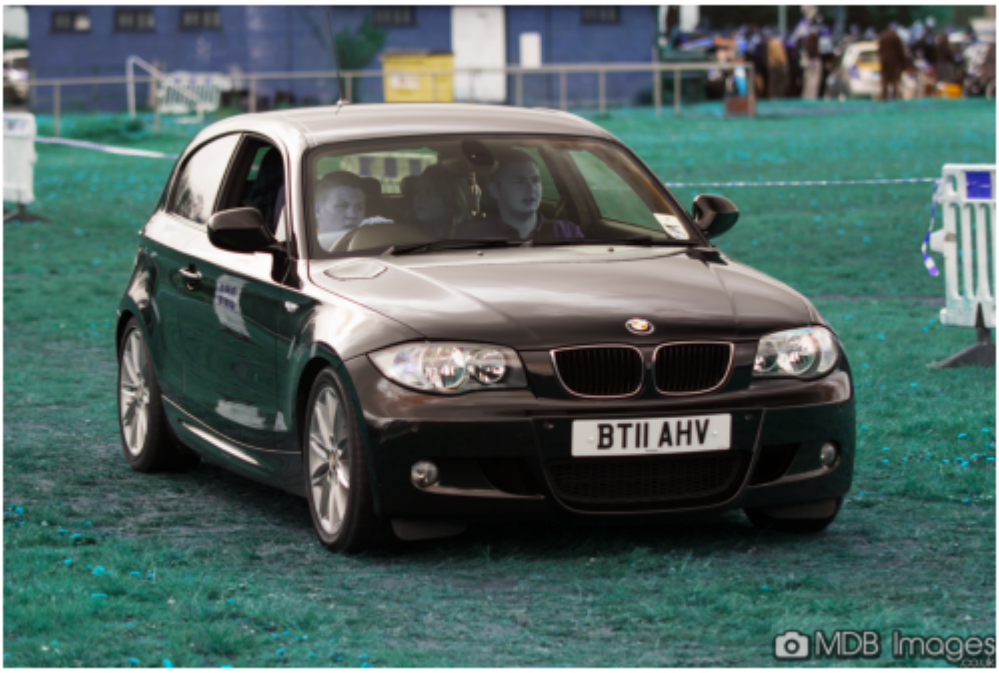
Please note that the first image shows correct bounding box (green color) around number plate. The 2nd image was an attempt to blur the plate using same box coordinates, but the code snippet actually affects the entire image (which even TA tried to look at). We tried another library for blurring, still did not succeed.

Similarly, for next all test images, the bounding box is clearly visible, but the blurring part (2nd image) is not successful.

```
# Test file 2
filename = '/content/drive/MyDrive/License/images/test/0e50ea14c4fc1353.jpg'
boxes = predict_bbox(filename)
show_bbox(filename, boxes)
apply_gaussian_blur(filename, boxes)
```

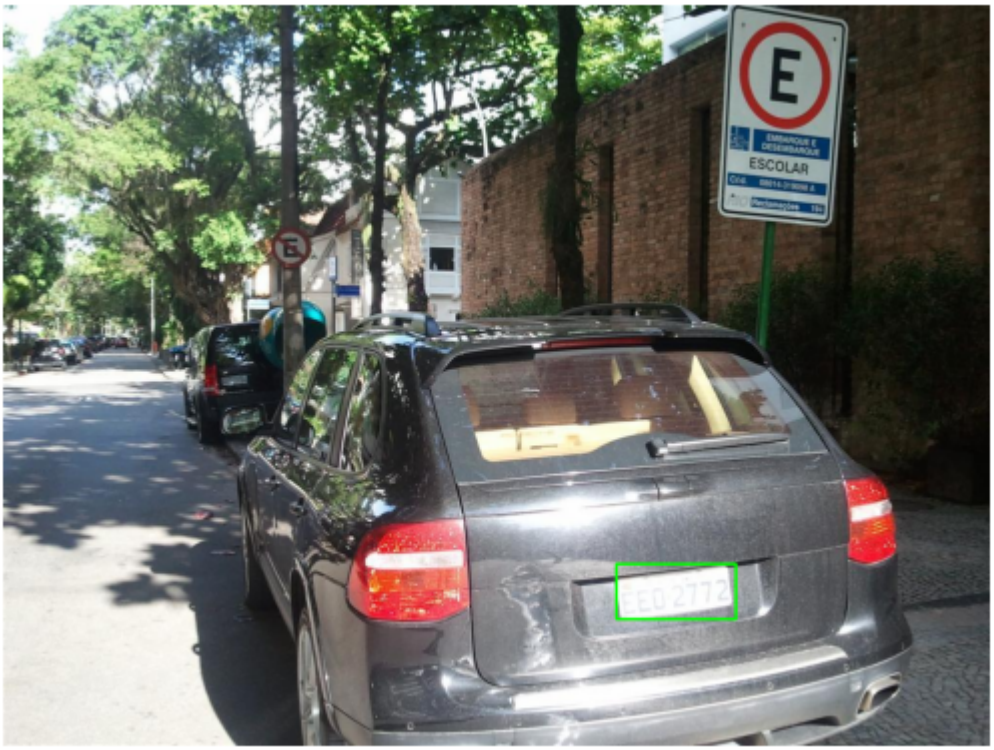
```
image 1/1 /content/drive/MyDrive/License/images/test/0e50ea14c4fc1353.jpg: 44
Speed: 29.4ms preprocess, 509.2ms inference, 13.1ms postprocess per image at
Results saved to /content/runs/detect/predict
```

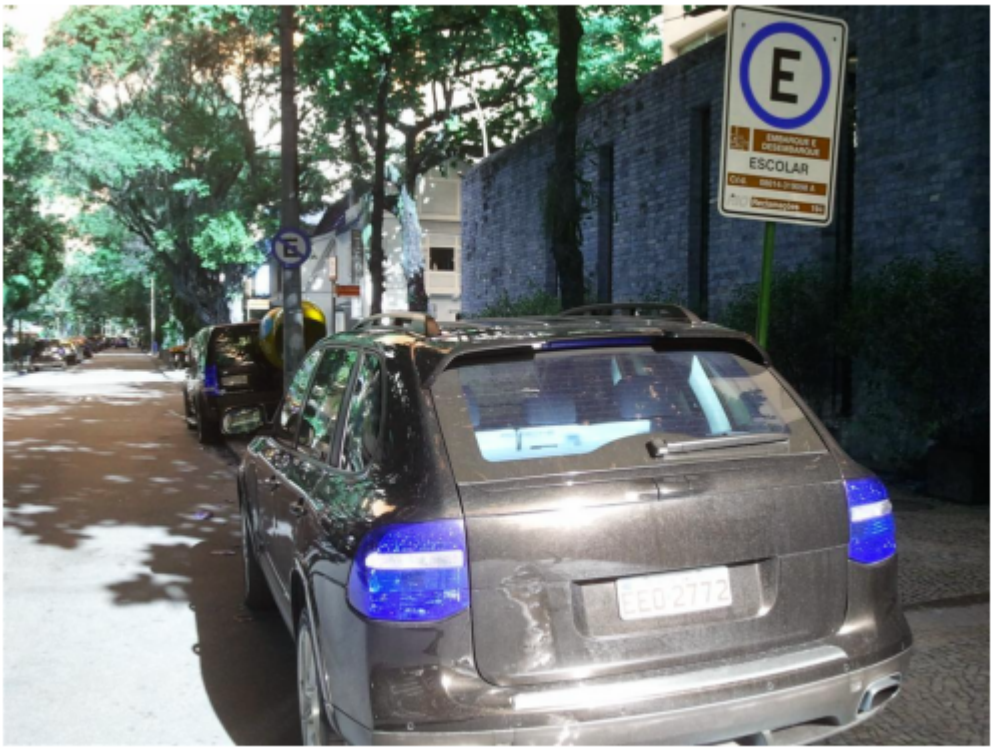





```
# Test file 3
filename = '/content/drive/MyDrive/License/images/test/0c756c9366a8cb10.jpg'
boxes = predict_bbox(filename)
show_bbox(filename, boxes)
apply_gaussian_blur(filename, boxes)
```

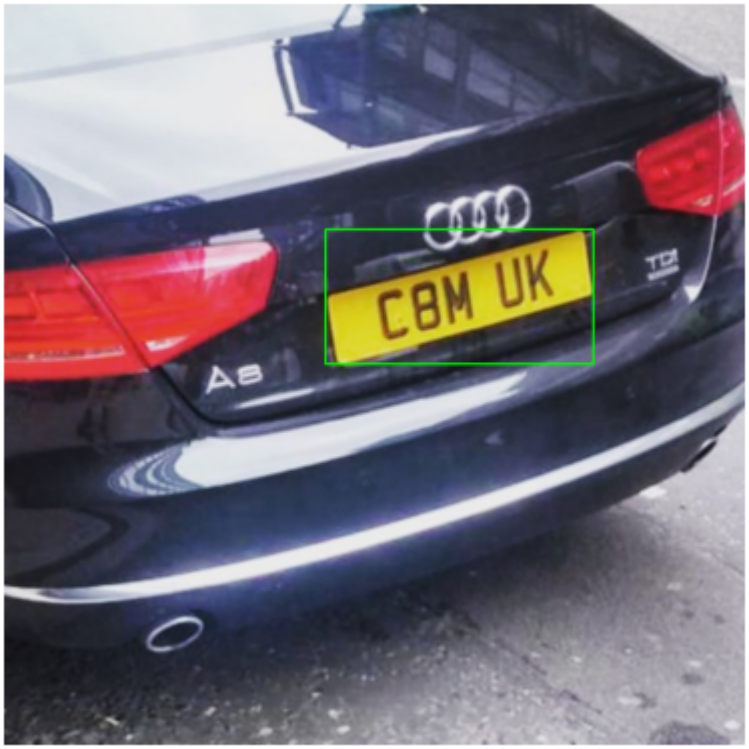
```
image 1/1 /content/drive/MyDrive/License/images/test/0c756c9366a8cb10.jpg: 48
Speed: 3.7ms preprocess, 193.1ms inference, 1.2ms postprocess per image at sh
Results saved to /content/runs/detect/predict
```

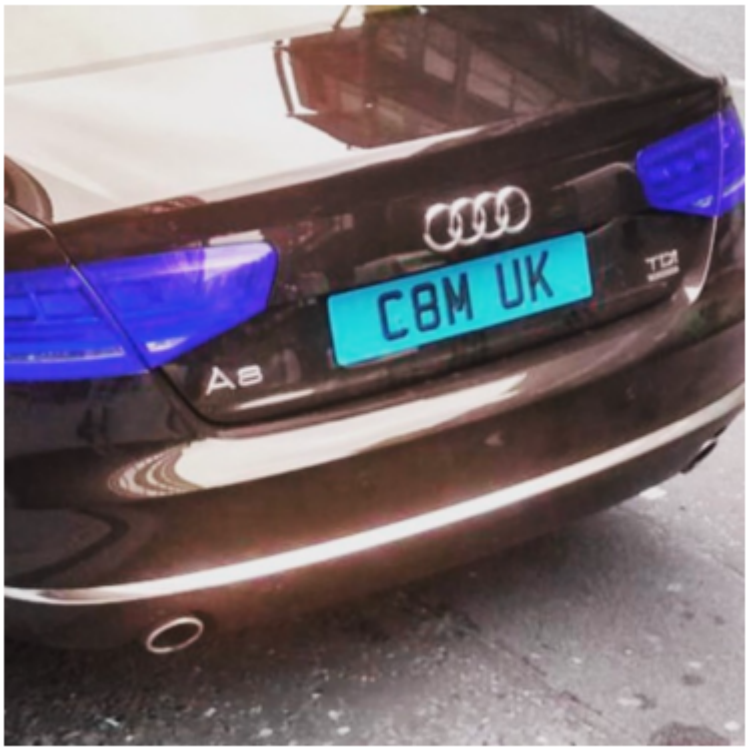





```
# Test file 4
filename = '/content/drive/MyDrive/License/images/test/0f4bfc46402a9f52.jpg'
boxes = predict_bbox(filename)
show_bbox(filename, boxes)
apply_gaussian_blur(filename, boxes)
```

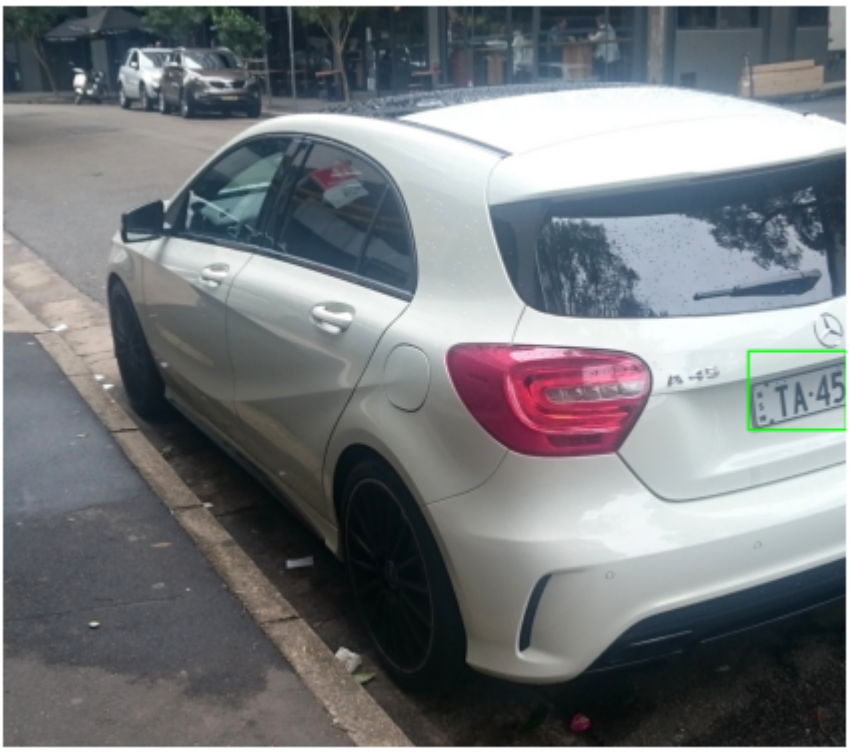
```
image 1/1 /content/drive/MyDrive/License/images/test/0f4bfc46402a9f52.jpg: 64
Speed: 6.5ms preprocess, 454.6ms inference, 2.7ms postprocess per image at sh
Results saved to /content/runs/detect/predict
```

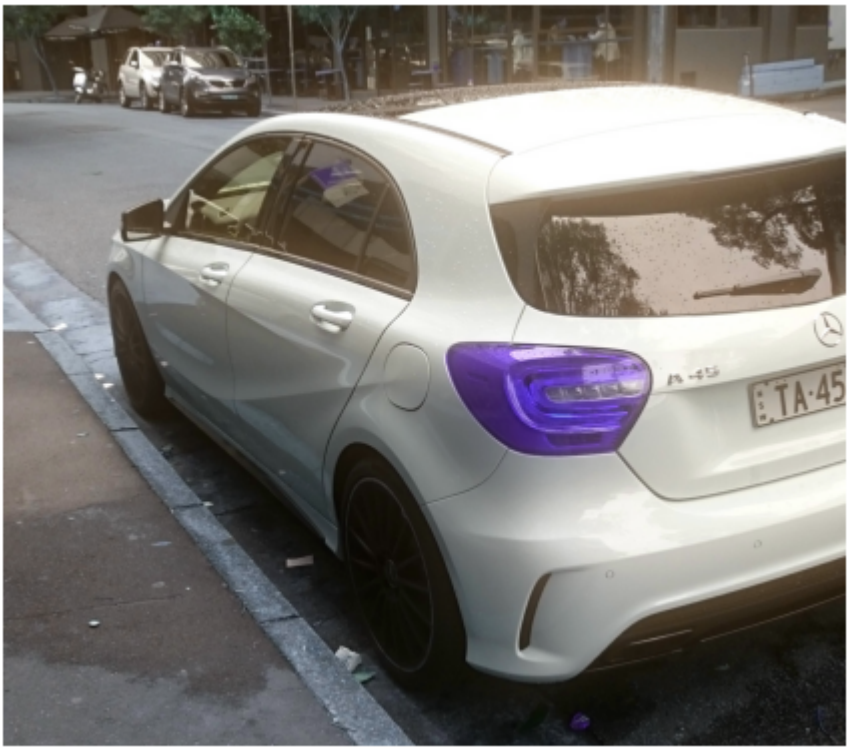





```
# Test file 5
filename = '/content/drive/MyDrive/License/images/test/0f0596b1c511e071.jpg'
boxes = predict_bbox(filename)
show_bbox(filename, boxes)
apply_gaussian_blur(filename, boxes)
```

```
image 1/1 /content/drive/MyDrive/License/images/test/0f0596b1c511e071.jpg: 57
Speed: 6.0ms preprocess, 289.6ms inference, 1.1ms postprocess per image at sh
Results saved to /content/runs/detect/predict
```





The yolo model was trained on 345 images due to execution time constraints. Looks like it is predicting bounding box well. Will explore blurring part in future, if TA can guide. Thanks to all help so far from TA, who made the project succesful.

Close

```
End
End
```

> End End End End End End End End End