Autonomous_driving_application_Car_detection_2022_05_04_09_36_33-Copy1

May 4, 2022

[1]:
```
import argparse
import os
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import scipy.io
import scipy.misc
import numpy as np
import pandas as pd
import PIL
from PIL import ImageFont, ImageDraw, Image
import tensorflow as tf
from tensorflow.python.framework.ops import EagerTensor

from tensorflow.keras.models import load_model
from yad2k.models.keras_yolo import yolo_head
from yad2k.utils.utils import draw_boxes, get_colors_for_classes, scale_boxes,␣
↪read_classes, read_anchors, preprocess_image

%matplotlib inline
```

[2]:
```
# UNQ_C1 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
# GRADED FUNCTION: yolo_filter_boxes

def yolo_filter_boxes(boxes, box_confidence, box_class_probs, threshold = .6):
    box_scores = box_confidence * box_class_probs
    box_classes = tf.math.argmax(box_scores, axis = -1)
    box_class_scores = tf.math.reduce_max(box_scores, axis = -1)


    filtering_mask = box_class_scores >= threshold


    scores = tf.boolean_mask(box_class_scores, filtering_mask)
    boxes = tf.boolean_mask(boxes, filtering_mask)
```

```
    classes = tf.boolean_mask(box_classes, filtering_mask)
    return scores, boxes, classes
```

[3]:
```
# BEGIN UNIT TEST
tf.random.set_seed(10)
box_confidence = tf.random.normal([19, 19, 5, 1], mean=1, stddev=4, seed = 1)
boxes = tf.random.normal([19, 19, 5, 4], mean=1, stddev=4, seed = 1)
box_class_probs = tf.random.normal([19, 19, 5, 80], mean=1, stddev=4, seed = 1)
scores, boxes, classes = yolo_filter_boxes(boxes, box_confidence,␣
 ↪box_class_probs, threshold = 0.5)
print("scores[2] = " + str(scores[2].numpy()))
print("boxes[2] = " + str(boxes[2].numpy()))
print("classes[2] = " + str(classes[2].numpy()))
print("scores.shape = " + str(scores.shape))
print("boxes.shape = " + str(boxes.shape))
print("classes.shape = " + str(classes.shape))

assert type(scores) == EagerTensor, "Use tensorflow functions"
assert type(boxes) == EagerTensor, "Use tensorflow functions"
assert type(classes) == EagerTensor, "Use tensorflow functions"

assert scores.shape == (1789,), "Wrong shape in scores"
assert boxes.shape == (1789, 4), "Wrong shape in boxes"
assert classes.shape == (1789,), "Wrong shape in classes"

assert np.isclose(scores[2].numpy(), 9.270486), "Values are wrong on scores"
assert np.allclose(boxes[2].numpy(), [4.6399336, 3.2303846, 4.431282, -2.
 ↪202031]), "Values are wrong on boxes"
assert classes[2].numpy() == 8, "Values are wrong on classes"

print("\033[92m All tests passed!")
# END UNIT TEST
```

```
scores[2] = 9.270486
boxes[2] = [ 4.6399336  3.2303846  4.431282  -2.202031 ]
classes[2] = 8
scores.shape = (1789,)
boxes.shape = (1789, 4)
classes.shape = (1789,)
 All tests passed!
```

[6]:
```
# UNQ_C2 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
# GRADED FUNCTION: iou

def iou(box1, box2):
```

```
    (box1_x1, box1_y1, box1_x2, box1_y2) = box1
    (box2_x1, box2_y1, box2_x2, box2_y2) = box2


    xi1 = max(box1[0],box2[0])
    yi1 = max(box1[1],box2[1])
    xi2 = min(box1[2],box2[2])
    yi2 = min(box1[3],box2[3])
    inter_width = max(xi2 - xi1,0)
    inter_height = max(yi2 - yi1,0)
    inter_area = inter_width*inter_height


    box1_area = (box1[3] - box1[1])*(box1[2] - box1[0])
    box2_area = (box2[3] - box2[1])*(box2[2] - box2[0])
    union_area = box1_area + box2_area - inter_area

    # compute the IoU
    iou = inter_area/union_area
    ### END CODE HERE

    return iou
```

[7]:
```
# BEGIN UNIT TEST
## Test case 1: boxes intersect
box1 = (2, 1, 4, 3)
box2 = (1, 2, 3, 4)

print("iou for intersecting boxes = " + str(iou(box1, box2)))
assert iou(box1, box2) < 1, "The intersection area must be always smaller or␣
 ↪equal than the union area."
assert np.isclose(iou(box1, box2), 0.14285714), "Wrong value. Check your␣
 ↪implementation. Problem with intersecting boxes"

## Test case 2: boxes do not intersect
box1 = (1,2,3,4)
box2 = (5,6,7,8)
print("iou for non-intersecting boxes = " + str(iou(box1,box2)))
assert iou(box1, box2) == 0, "Intersection must be 0"

## Test case 3: boxes intersect at vertices only
box1 = (1,1,2,2)
box2 = (2,2,3,3)
print("iou for boxes that only touch at vertices = " + str(iou(box1,box2)))
assert iou(box1, box2) == 0, "Intersection at vertices must be 0"

## Test case 4: boxes intersect at edge only
```

```
box1 = (1,1,3,3)
box2 = (2,3,3,4)
print("iou for boxes that only touch at edges = " + str(iou(box1,box2)))
assert iou(box1, box2) == 0, "Intersection at edges must be 0"

print("\033[92m All tests passed!")
# END UNIT TEST
```

```
iou for intersecting boxes = 0.14285714285714285
iou for non-intersecting boxes = 0.0
iou for boxes that only touch at vertices = 0.0
iou for boxes that only touch at edges = 0.0
 All tests passed!
```

[12]:
```
# UNQ_C3 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
# GRADED FUNCTION: yolo_non_max_suppression

def yolo_non_max_suppression(scores, boxes, classes, max_boxes = 10,␣
↪iou_threshold = 0.5):
    max_boxes_tensor = tf.Variable(max_boxes, dtype='int32')     # tensor to be␣
↪used in tf.image.non_max_suppression()

    nms_indices = tf.image.
↪non_max_suppression(boxes,scores,max_boxes_tensor,iou_threshold =␣
↪iou_threshold)

    scores = tf.gather(scores,nms_indices)
    boxes = tf.gather(boxes,nms_indices)
    classes = tf.gather(classes,nms_indices)


    return scores, boxes, classes
```

[13]:
```
# BEGIN UNIT TEST
tf.random.set_seed(10)
scores = tf.random.normal([54,], mean=1, stddev=4, seed = 1)
boxes = tf.random.normal([54, 4], mean=1, stddev=4, seed = 1)
classes = tf.random.normal([54,], mean=1, stddev=4, seed = 1)
scores, boxes, classes = yolo_non_max_suppression(scores, boxes, classes)

assert type(scores) == EagerTensor, "Use tensoflow functions"
print("scores[2] = " + str(scores[2].numpy()))
print("boxes[2] = " + str(boxes[2].numpy()))
print("classes[2] = " + str(classes[2].numpy()))
print("scores.shape = " + str(scores.numpy().shape))
print("boxes.shape = " + str(boxes.numpy().shape))
print("classes.shape = " + str(classes.numpy().shape))
```

```
assert type(scores) == EagerTensor, "Use tensoflow functions"
assert type(boxes) == EagerTensor, "Use tensoflow functions"
assert type(classes) == EagerTensor, "Use tensoflow functions"

assert scores.shape == (10,), "Wrong shape"
assert boxes.shape == (10, 4), "Wrong shape"
assert classes.shape == (10,), "Wrong shape"

assert np.isclose(scores[2].numpy(), 8.147684), "Wrong value on scores"
assert np.allclose(boxes[2].numpy(), [ 6.0797963, 3.743308, 1.3914018, -0.
↪34089637]), "Wrong value on boxes"
assert np.isclose(classes[2].numpy(), 1.7079165), "Wrong value on classes"

print("\033[92m All tests passed!")
# END UNIT TEST
```

```
scores[2] = 8.147684
boxes[2] = [ 6.0797963   3.743308    1.3914018  -0.34089637]
classes[2] = 1.7079165
scores.shape = (10,)
boxes.shape = (10, 4)
classes.shape = (10,)
 All tests passed!
```

[14]:
```
def yolo_boxes_to_corners(box_xy, box_wh):
    """Convert YOLO box predictions to bounding box corners."""
    box_mins = box_xy - (box_wh / 2.)
    box_maxes = box_xy + (box_wh / 2.)

    return tf.keras.backend.concatenate([
        box_mins[..., 1:2],  # y_min
        box_mins[..., 0:1],  # x_min
        box_maxes[..., 1:2],  # y_max
        box_maxes[..., 0:1]  # x_max
    ])
```

[21]:
```
# UNQ_C4 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
# GRADED FUNCTION: yolo_eval

def yolo_eval(yolo_outputs, image_shape = (720, 1280), max_boxes=10,␣
↪score_threshold=.6, iou_threshold=.5):

    # Retrieve outputs of the YOLO model (1 line)
    box_xy, box_wh, box_confidence, box_class_probs = yolo_outputs
```

```
    # Convert boxes to be ready for filtering functions (convert boxes box_xy␣
↪and box_wh to corner coordinates)
    boxes = yolo_boxes_to_corners(box_xy, box_wh)

    # Use one of the functions you've implemented to perform Score-filtering␣
↪with a threshold of score_threshold (1 line)
    scores, boxes, classes = yolo_filter_boxes(boxes, box_confidence,␣
↪box_class_probs, threshold = score_threshold)

    # Scale boxes back to original image shape.
    boxes = scale_boxes(boxes, image_shape)

    # Use one of the functions you've implemented to perform Non-max␣
↪suppression with a threshold of iou_threshold (1 line)
    scores, boxes, classes = yolo_non_max_suppression(scores, boxes, classes,␣
↪max_boxes = max_boxes, iou_threshold = iou_threshold)

    ### END CODE HERE ###

    return scores, boxes, classes
```

[22]:
```
# BEGIN UNIT TEST
tf.random.set_seed(10)
yolo_outputs = (tf.random.normal([19, 19, 5, 2], mean=1, stddev=4, seed = 1),
                tf.random.normal([19, 19, 5, 2], mean=1, stddev=4, seed = 1),
                tf.random.normal([19, 19, 5, 1], mean=1, stddev=4, seed = 1),
                tf.random.normal([19, 19, 5, 80], mean=1, stddev=4, seed = 1))
scores, boxes, classes = yolo_eval(yolo_outputs)
print("scores[2] = " + str(scores[2].numpy()))
print("boxes[2] = " + str(boxes[2].numpy()))
print("classes[2] = " + str(classes[2].numpy()))
print("scores.shape = " + str(scores.numpy().shape))
print("boxes.shape = " + str(boxes.numpy().shape))
print("classes.shape = " + str(classes.numpy().shape))

assert type(scores) == EagerTensor, "Use tensoflow functions"
assert type(boxes) == EagerTensor, "Use tensoflow functions"
assert type(classes) == EagerTensor, "Use tensoflow functions"

assert scores.shape == (10,), "Wrong shape"
assert boxes.shape == (10, 4), "Wrong shape"
assert classes.shape == (10,), "Wrong shape"

assert np.isclose(scores[2].numpy(), 171.60194), "Wrong value on scores"
assert np.allclose(boxes[2].numpy(), [-1240.3483, -3212.5881, -645.78, 2024.
↪3052]), "Wrong value on boxes"
```

```
assert np.isclose(classes[2].numpy(), 16), "Wrong value on classes"

print("\033[92m All tests passed!")
# END UNIT TEST
```

```
scores[2] = 171.60194
boxes[2] = [-1240.3483 -3212.5881  -645.78    2024.3052]
classes[2] = 16
scores.shape = (10,)
boxes.shape = (10, 4)
classes.shape = (10,)
 All tests passed!
```

[9]:
```
class_names = read_classes("model_data/coco_classes.txt")
anchors = read_anchors("model_data/yolo_anchors.txt")
model_image_size = (608, 608) # Same as yolo_model input layer size
```

[18]:
```
yolo_model = load_model("model_data/" ,compile=True)
```

```
WARNING:tensorflow:No training configuration found in save file, so the model
was *not* compiled. Compile it manually.
```

[19]:
```
yolo_model.summary()
```

```
Model: "functional_1"
__________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
==================
input_1 (InputLayer)            [(None, 608, 608, 3) 0
__________________________________________________________________________________
__________________
conv2d (Conv2D)                 (None, 608, 608, 32) 864         input_1[0][0]
__________________________________________________________________________________
__________________
batch_normalization (BatchNorma (None, 608, 608, 32) 128         conv2d[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu (LeakyReLU)         (None, 608, 608, 32) 0
batch_normalization[0][0]
__________________________________________________________________________________
__________________
max_pooling2d (MaxPooling2D)    (None, 304, 304, 32) 0
leaky_re_lu[0][0]
__________________________________________________________________________________
__________________
conv2d_1 (Conv2D)               (None, 304, 304, 64) 18432
```

```
max_pooling2d[0][0]
__________________________________________________________________________________
__________________
batch_normalization_1 (BatchNor (None, 304, 304, 64) 256         conv2d_1[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_1 (LeakyReLU)       (None, 304, 304, 64) 0
batch_normalization_1[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_1 (MaxPooling2D)  (None, 152, 152, 64) 0
leaky_re_lu_1[0][0]
__________________________________________________________________________________
__________________
conv2d_2 (Conv2D)               (None, 152, 152, 128 73728
max_pooling2d_1[0][0]
__________________________________________________________________________________
__________________
batch_normalization_2 (BatchNor (None, 152, 152, 128 512         conv2d_2[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_2 (LeakyReLU)       (None, 152, 152, 128 0
batch_normalization_2[0][0]
__________________________________________________________________________________
__________________
conv2d_3 (Conv2D)               (None, 152, 152, 64) 8192
leaky_re_lu_2[0][0]
__________________________________________________________________________________
__________________
batch_normalization_3 (BatchNor (None, 152, 152, 64) 256         conv2d_3[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_3 (LeakyReLU)       (None, 152, 152, 64) 0
batch_normalization_3[0][0]
__________________________________________________________________________________
__________________
conv2d_4 (Conv2D)               (None, 152, 152, 128 73728
leaky_re_lu_3[0][0]
__________________________________________________________________________________
__________________
batch_normalization_4 (BatchNor (None, 152, 152, 128 512         conv2d_4[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_4 (LeakyReLU)       (None, 152, 152, 128 0
batch_normalization_4[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_2 (MaxPooling2D)  (None, 76, 76, 128)  0
```

```
leaky_re_lu_4[0][0]
__________________________________________________________________________________
__________________
conv2d_5 (Conv2D)               (None, 76, 76, 256)  294912
max_pooling2d_2[0][0]
__________________________________________________________________________________
__________________
batch_normalization_5 (BatchNor (None, 76, 76, 256)  1024        conv2d_5[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_5 (LeakyReLU)       (None, 76, 76, 256)  0
batch_normalization_5[0][0]
__________________________________________________________________________________
__________________
conv2d_6 (Conv2D)               (None, 76, 76, 128)  32768
leaky_re_lu_5[0][0]
__________________________________________________________________________________
__________________
batch_normalization_6 (BatchNor (None, 76, 76, 128)  512         conv2d_6[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_6 (LeakyReLU)       (None, 76, 76, 128)  0
batch_normalization_6[0][0]
__________________________________________________________________________________
__________________
conv2d_7 (Conv2D)               (None, 76, 76, 256)  294912
leaky_re_lu_6[0][0]
__________________________________________________________________________________
__________________
batch_normalization_7 (BatchNor (None, 76, 76, 256)  1024        conv2d_7[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_7 (LeakyReLU)       (None, 76, 76, 256)  0
batch_normalization_7[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_3 (MaxPooling2D)  (None, 38, 38, 256)  0
leaky_re_lu_7[0][0]
__________________________________________________________________________________
__________________
conv2d_8 (Conv2D)               (None, 38, 38, 512)  1179648
max_pooling2d_3[0][0]
__________________________________________________________________________________
__________________
batch_normalization_8 (BatchNor (None, 38, 38, 512)  2048        conv2d_8[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_8 (LeakyReLU)       (None, 38, 38, 512)  0
```

```
batch_normalization_8[0][0]
__________________________________________________________________________________
__________________
conv2d_9 (Conv2D)               (None, 38, 38, 256)  131072
leaky_re_lu_8[0][0]
__________________________________________________________________________________
__________________
batch_normalization_9 (BatchNor (None, 38, 38, 256)  1024        conv2d_9[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_9 (LeakyReLU)       (None, 38, 38, 256)  0
batch_normalization_9[0][0]
__________________________________________________________________________________
__________________
conv2d_10 (Conv2D)              (None, 38, 38, 512)  1179648
leaky_re_lu_9[0][0]
__________________________________________________________________________________
__________________
batch_normalization_10 (BatchNo (None, 38, 38, 512)  2048        conv2d_10[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_10 (LeakyReLU)      (None, 38, 38, 512)  0
batch_normalization_10[0][0]
__________________________________________________________________________________
__________________
conv2d_11 (Conv2D)              (None, 38, 38, 256)  131072
leaky_re_lu_10[0][0]
__________________________________________________________________________________
__________________
batch_normalization_11 (BatchNo (None, 38, 38, 256)  1024        conv2d_11[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_11 (LeakyReLU)      (None, 38, 38, 256)  0
batch_normalization_11[0][0]
__________________________________________________________________________________
__________________
conv2d_12 (Conv2D)              (None, 38, 38, 512)  1179648
leaky_re_lu_11[0][0]
__________________________________________________________________________________
__________________
batch_normalization_12 (BatchNo (None, 38, 38, 512)  2048        conv2d_12[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_12 (LeakyReLU)      (None, 38, 38, 512)  0
batch_normalization_12[0][0]
__________________________________________________________________________________
__________________
max_pooling2d_4 (MaxPooling2D)  (None, 19, 19, 512)  0
```

```
leaky_re_lu_12[0][0]
__________________________________________________________________________________
__________________
conv2d_13 (Conv2D)              (None, 19, 19, 1024) 4718592
max_pooling2d_4[0][0]
__________________________________________________________________________________
__________________
batch_normalization_13 (BatchNo (None, 19, 19, 1024) 4096        conv2d_13[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_13 (LeakyReLU)      (None, 19, 19, 1024) 0
batch_normalization_13[0][0]
__________________________________________________________________________________
__________________
conv2d_14 (Conv2D)              (None, 19, 19, 512)  524288
leaky_re_lu_13[0][0]
__________________________________________________________________________________
__________________
batch_normalization_14 (BatchNo (None, 19, 19, 512)  2048        conv2d_14[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_14 (LeakyReLU)      (None, 19, 19, 512)  0
batch_normalization_14[0][0]
__________________________________________________________________________________
__________________
conv2d_15 (Conv2D)              (None, 19, 19, 1024) 4718592
leaky_re_lu_14[0][0]
__________________________________________________________________________________
__________________
batch_normalization_15 (BatchNo (None, 19, 19, 1024) 4096        conv2d_15[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_15 (LeakyReLU)      (None, 19, 19, 1024) 0
batch_normalization_15[0][0]
__________________________________________________________________________________
__________________
conv2d_16 (Conv2D)              (None, 19, 19, 512)  524288
leaky_re_lu_15[0][0]
__________________________________________________________________________________
__________________
batch_normalization_16 (BatchNo (None, 19, 19, 512)  2048        conv2d_16[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_16 (LeakyReLU)      (None, 19, 19, 512)  0
batch_normalization_16[0][0]
__________________________________________________________________________________
__________________
conv2d_17 (Conv2D)              (None, 19, 19, 1024) 4718592
```

```
leaky_re_lu_16[0][0]
__________________________________________________________________________________
__________________
batch_normalization_17 (BatchNo (None, 19, 19, 1024) 4096        conv2d_17[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_17 (LeakyReLU)      (None, 19, 19, 1024) 0
batch_normalization_17[0][0]
__________________________________________________________________________________
__________________
conv2d_18 (Conv2D)              (None, 19, 19, 1024) 9437184
leaky_re_lu_17[0][0]
__________________________________________________________________________________
__________________
batch_normalization_18 (BatchNo (None, 19, 19, 1024) 4096        conv2d_18[0][0]
__________________________________________________________________________________
__________________
conv2d_20 (Conv2D)              (None, 38, 38, 64)   32768
leaky_re_lu_12[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_18 (LeakyReLU)      (None, 19, 19, 1024) 0
batch_normalization_18[0][0]
__________________________________________________________________________________
__________________
batch_normalization_20 (BatchNo (None, 38, 38, 64)   256         conv2d_20[0][0]
__________________________________________________________________________________
__________________
conv2d_19 (Conv2D)              (None, 19, 19, 1024) 9437184
leaky_re_lu_18[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_20 (LeakyReLU)      (None, 38, 38, 64)   0
batch_normalization_20[0][0]
__________________________________________________________________________________
__________________
batch_normalization_19 (BatchNo (None, 19, 19, 1024) 4096        conv2d_19[0][0]
__________________________________________________________________________________
__________________
space_to_depth_x2 (Lambda)      (None, 19, 19, 256)  0
leaky_re_lu_20[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_19 (LeakyReLU)      (None, 19, 19, 1024) 0
batch_normalization_19[0][0]
__________________________________________________________________________________
__________________
concatenate (Concatenate)       (None, 19, 19, 1280) 0
```

```
space_to_depth_x2[0][0]
leaky_re_lu_19[0][0]
__________________________________________________________________________________
__________________
conv2d_21 (Conv2D)              (None, 19, 19, 1024) 11796480
concatenate[0][0]
__________________________________________________________________________________
__________________
batch_normalization_21 (BatchNo (None, 19, 19, 1024) 4096        conv2d_21[0][0]
__________________________________________________________________________________
__________________
leaky_re_lu_21 (LeakyReLU)      (None, 19, 19, 1024) 0
batch_normalization_21[0][0]
__________________________________________________________________________________
__________________
conv2d_22 (Conv2D)              (None, 19, 19, 425)  435625
leaky_re_lu_21[0][0]
==================================================================================
==================
Total params: 50,983,561
Trainable params: 50,962,889
Non-trainable params: 20,672
__________________________________________________________________________________
__________________
```

[23]:
```
def predict(image_file):

    # Preprocess your image
    image, image_data = preprocess_image("images/" + image_file,␣
↪model_image_size = (608, 608))

    yolo_model_outputs = yolo_model(image_data)
    yolo_outputs = yolo_head(yolo_model_outputs, anchors, len(class_names))

    out_scores, out_boxes, out_classes = yolo_eval(yolo_outputs, [image.
↪size[1],  image.size[0]], 10, 0.3, 0.5)

    # Print predictions info
    print('Found {} boxes for {}'.format(len(out_boxes), "images/" +␣
↪image_file))
    # Generate colors for drawing bounding boxes.
    colors = get_colors_for_classes(len(class_names))
    # Draw bounding boxes on the image file
    #draw_boxes2(image, out_scores, out_boxes, out_classes, class_names,␣
↪colors, image_shape)
    draw_boxes(image, out_boxes, out_classes, class_names, out_scores)
    # Save the predicted bounding box on the image
```

```
    image.save(os.path.join("out", image_file), quality=100)
    # Display the results in the notebook
    output_image = Image.open(os.path.join("out", image_file))
    imshow(output_image)

    return out_scores, out_boxes, out_classes
```

[27]:
```
out_scores, out_boxes, out_classes = predict("Transpo-Mumbaitraffic-1079622456.
↪jpg")
```

```
Found 10 boxes for images/Transpo-Mumbaitraffic-1079622456.jpg
person 0.71 (1310, 394) (1431, 798)
person 0.68 (1127, 444) (1268, 907)
person 0.67 (1672, 493) (1818, 970)
person 0.67 (974, 379) (1087, 841)
person 0.67 (753, 311) (900, 736)
person 0.66 (828, 1012) (956, 1200)
person 0.64 (2089, 1015) (2273, 1200)
person 0.63 (1879, 495) (2034, 937)
person 0.63 (906, 400) (1046, 837)
car 0.62 (1050, 1063) (1527, 1189)
```