In [1]:
```
import os
import pathlib
import random
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [2]:
```
from object_detection.utils import visualization_utils as viz_utils
```

In [3]:
```
import tensorflow as tf
```

In [4]:
```
from object_detection.utils import config_util
from object_detection.builders import model_builder
```

In [5]:
```
model_name = 'ssd_resnet50_v1_fpn_640x640_coco17_tpu-8'
pipeline_config = os.path.join('C:/object_detection_tensorflow/tensorflow_garden/resea
model_dir = 'C:/object_detection_tensorflow/workspace/training_demo/pre-trained-models
```

In [6]:
```
configs = config_util.get_configs_from_pipeline_file(pipeline_config)
model_config = configs['model']
detection_model = model_builder.build(model_config=model_config, is_training=False)
```

In [7]:
```
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(model_dir, 'ckpt-0')).expect_partial()
```

Out[7]:
```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x173d99a86a0>
```

In [8]:
```
def get_model_detection_function(model):

  @tf.function
  def detect_fn(image):
    image, shape = model.preprocess(image)
    prediction_dict = model.predict(image, shape)
    detections = model.postprocess(prediction_dict, shape)

    return detections

  return detect_fn

detect_fn = get_model_detection_function(detection_model)
```

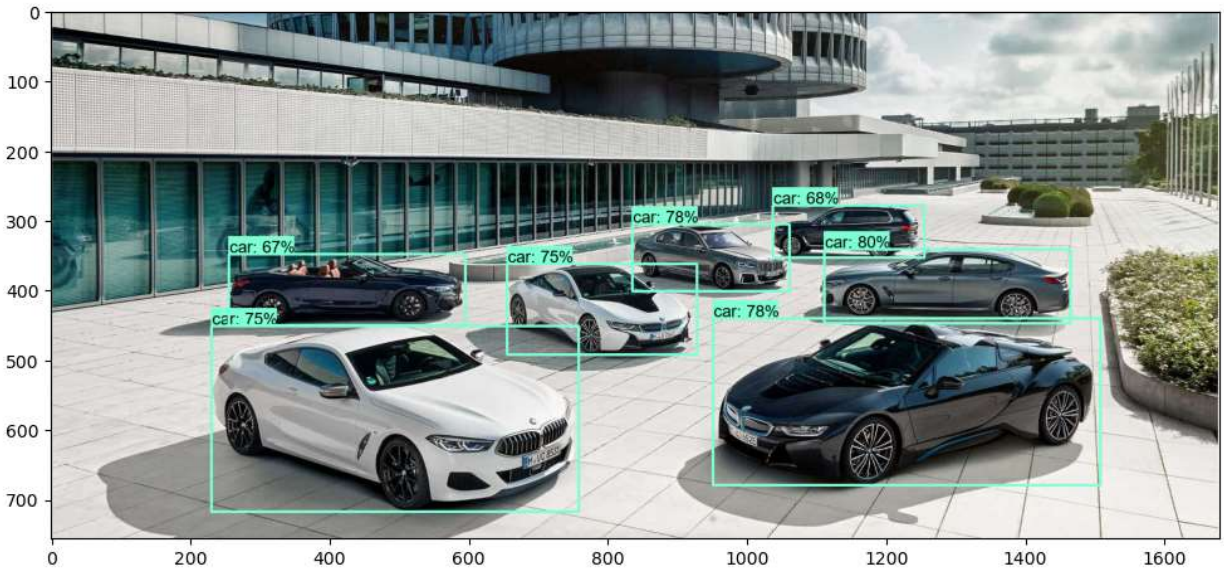
In [9]:
```
from object_detection.utils import label_map_util

label_map_path = configs['eval_input_config'].label_map_path
label_map_path = 'C:/object_detection_tensorflow/tensorflow_garden/research/object_det

label_map = label_map_util.load_labelmap(label_map_path)
categories = label_map_util.convert_label_map_to_categories(
    label_map,
    max_num_classes=label_map_util.get_max_label_map_index(label_map),
    use_display_name=True
)
```

```
category_index = label_map_util.create_category_index(categories)
label_map_dict = label_map_util.get_label_map_dict(label_map, use_display_name=True)
```

In [10]:
```
%matplotlib inline

img = cv2.imread('C:/Users/yesal/Pictures/gama-bmw.jpg')
image_np = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

In [11]:
```
input_tensor = tf.convert_to_tensor(
    np.expand_dims(image_np, 0), dtype=tf.float32
)
detections = detect_fn(input_tensor)
```

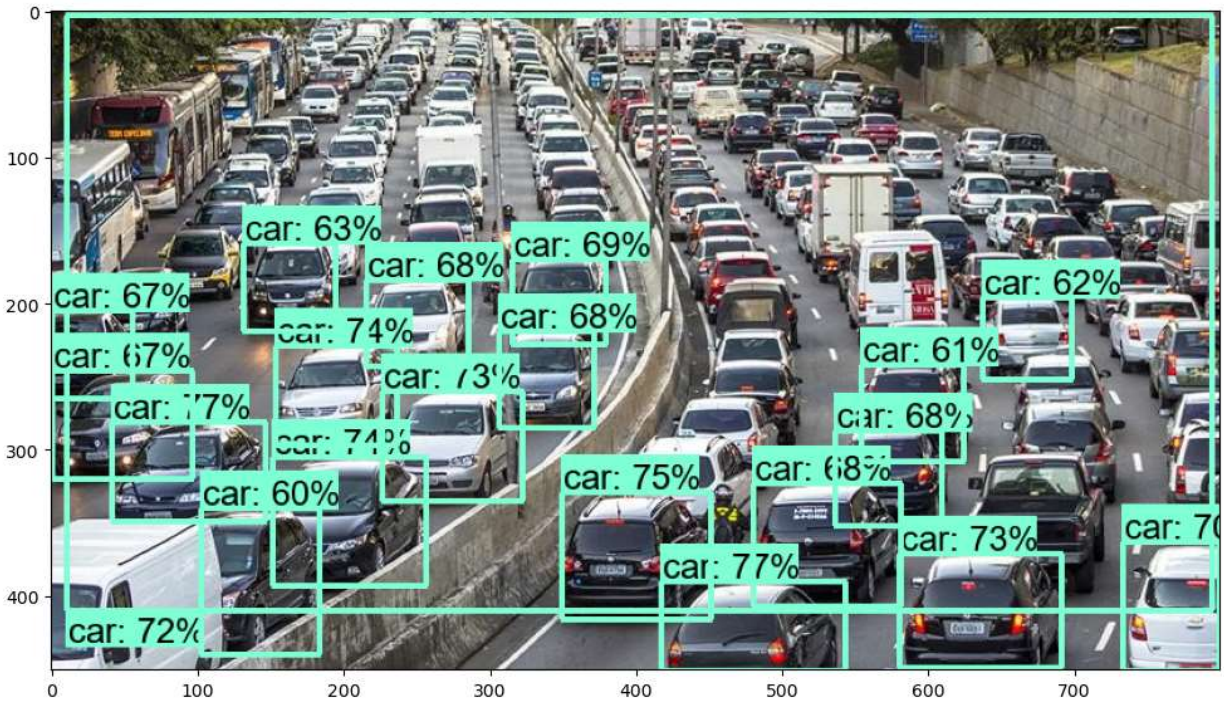
In [12]:
```
label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections['detection_boxes'][0].numpy(),
    (detections['detection_classes'][0].numpy() + label_id_offset).astype(int),
    detections['detection_scores'][0].numpy(),
    category_index,
    use_normalized_coordinates=True,
    min_score_thresh=0.5
)

plt.figure(figsize=(12, 16))
plt.imshow(image_np_with_detections)
plt.show()
```



In [13]:
```
%matplotlib inline

img2 = cv2.imread('C:/Users/yesal/Pictures/trafico.jpg')
image_np2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)
```

In [14]:
```
input_tensor = tf.convert_to_tensor(
    np.expand_dims(image_np2, 0), dtype=tf.float32
)
detections = detect_fn(input_tensor)
```

In [15]:
```
label_id_offset = 1
image_np_with_detections = image_np2.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections['detection_boxes'][0].numpy(),
    (detections['detection_classes'][0].numpy() + label_id_offset).astype(int),
    detections['detection_scores'][0].numpy(),
    category_index,
    use_normalized_coordinates=True,
    min_score_thresh=0.4
)

plt.figure(figsize=(12, 16))
plt.imshow(image_np_with_detections)
plt.show()
```



In [16]:
```
%matplotlib inline

img3 = cv2.imread('C:/Users/yesal/Pictures/Family.jpg')
image_np3 = cv2.cvtColor(img3, cv2.COLOR_BGR2RGB)
```

In [17]:
```
input_tensor = tf.convert_to_tensor(
    np.expand_dims(image_np3, 0), dtype=tf.float32
)
detections = detect_fn(input_tensor)
```

In [18]:
```
label_id_offset = 1
image_np_with_detections = image_np3.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections['detection_boxes'][0].numpy(),
    (detections['detection_classes'][0].numpy() + label_id_offset).astype(int),
    detections['detection_scores'][0].numpy(),
```

```
    category_index,
    use_normalized_coordinates=True,
    min_score_thresh=0.5
)

plt.figure(figsize=(12, 16))
plt.imshow(image_np_with_detections)
plt.show()
```

In [19]:
```
%matplotlib inline

img4 = cv2.imread('C:/Users/yesal/Pictures/animal.jpg')
image_np4 = cv2.cvtColor(img4, cv2.COLOR_BGR2RGB)
```

In [20]:
```
input_tensor = tf.convert_to_tensor(
    np.expand_dims(image_np4, 0), dtype=tf.float32
)
detections = detect_fn(input_tensor)
```

In [22]:
```
label_id_offset = 1
image_np_with_detections = image_np4.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections['detection_boxes'][0].numpy(),
    (detections['detection_classes'][0].numpy() + label_id_offset).astype(int),
    detections['detection_scores'][0].numpy(),
    category_index,
    use_normalized_coordinates=True,
    min_score_thresh=0.5
```

```
)

plt.figure(figsize=(12, 16))
plt.imshow(image_np_with_detections)
plt.show()
```