```
#!/usr/bin/env python
# coding: utf-8

# In[5]:


import cv2
import numpy as np
get_ipython().run_line_magic('matplotlib', 'inline')
from matplotlib import pyplot as fig
import matplotlib
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util

#Object detection, line(20-109), FOUND
HERE:https://github.com/nicknochnack/TFODCourse/blob/main/2.%20Training%20and%20Detection.ipynb


#CUSTOM_MODEL_NAME = 'efficientdet_d1_coco17_tpu-32_V11_40000s'
#PRETRAINED_MODEL_NAME = 'efficientdet_d1_coco17_tpu-32'
#PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/
#tf2/20200711/efficientdet_d1_coco17_tpu-32.tar.gz'
#TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
#LABEL_MAP_NAME = 'label_map.pbtxt'


CUSTOM_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_640x640_V4_sveis_40000s'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_640x640_coco17_tpu-8'
PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_640x640_coco17_tpu-8.tar.
gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'



paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'export'),
```

```
    'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }

files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MODEL_NAME, 'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}

configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-41')).expect_partial()  #The longer a model trains the more chekpoints it returns, important to chose the last checkpoint bechause its traind the most.

def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections


category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'LysSetting', 'LysSetting.d0d0f470-ca0b-11ec-8ce9-a0510b46e00a.jpg')

img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)
print(img.shape)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

label_id_offset = 1

image_np_with_detections = image_np.copy()
det_boxes = detections['detection_boxes'] #laga variabler
det_classes = detections['detection_classes']+label_id_offset
det_scores = detections['detection_scores']

viz_utils.visualize_boxes_and_labels_on_image_array(
```

```
            image_np_with_detections,
            det_boxes,
            det_classes,
            det_scores,
            category_index,
            use_normalized_coordinates=True,
            line_thickness=3,
            max_boxes_to_draw=1,
            min_score_thresh=.1,
            agnostic_mode=False)

fig.imshow(cv2.cvtColor(image_np_with_detections,cv2.COLOR_BGR2RGB))
fig = matplotlib.pyplot.gcf()
fig.show()
fig.savefig("TESTbildeV9.png",dpi =100)



# In[3]:

 ############################################################################################################
 ############################################################
# Find boundingbox cordinates, line(121-140), FOUND HERE: https://github.com/tensorflow/models/issues/4682

width, height = img.shape[:2]
print(width)
print(height)

boxes = detections['detection_boxes']
height, width = img.shape[:2]
box= np.squeeze(boxes)
max_boxes_to_draw=box.shape[0]
scores=np.squeeze(det_scores)
min_score_thresh=0.5

for i in range (min(max_boxes_to_draw, box.shape[0])):

     if scores [i] > min_score_thresh:
        ymin = (int(box[i,0]*height))
        xmin = (int(box[i,1]*width))
        ymax = (int(box[i,2]*height))
        xmax = (int(box[i,3]*width))
        print (xmin,ymin,xmax,ymax)


```