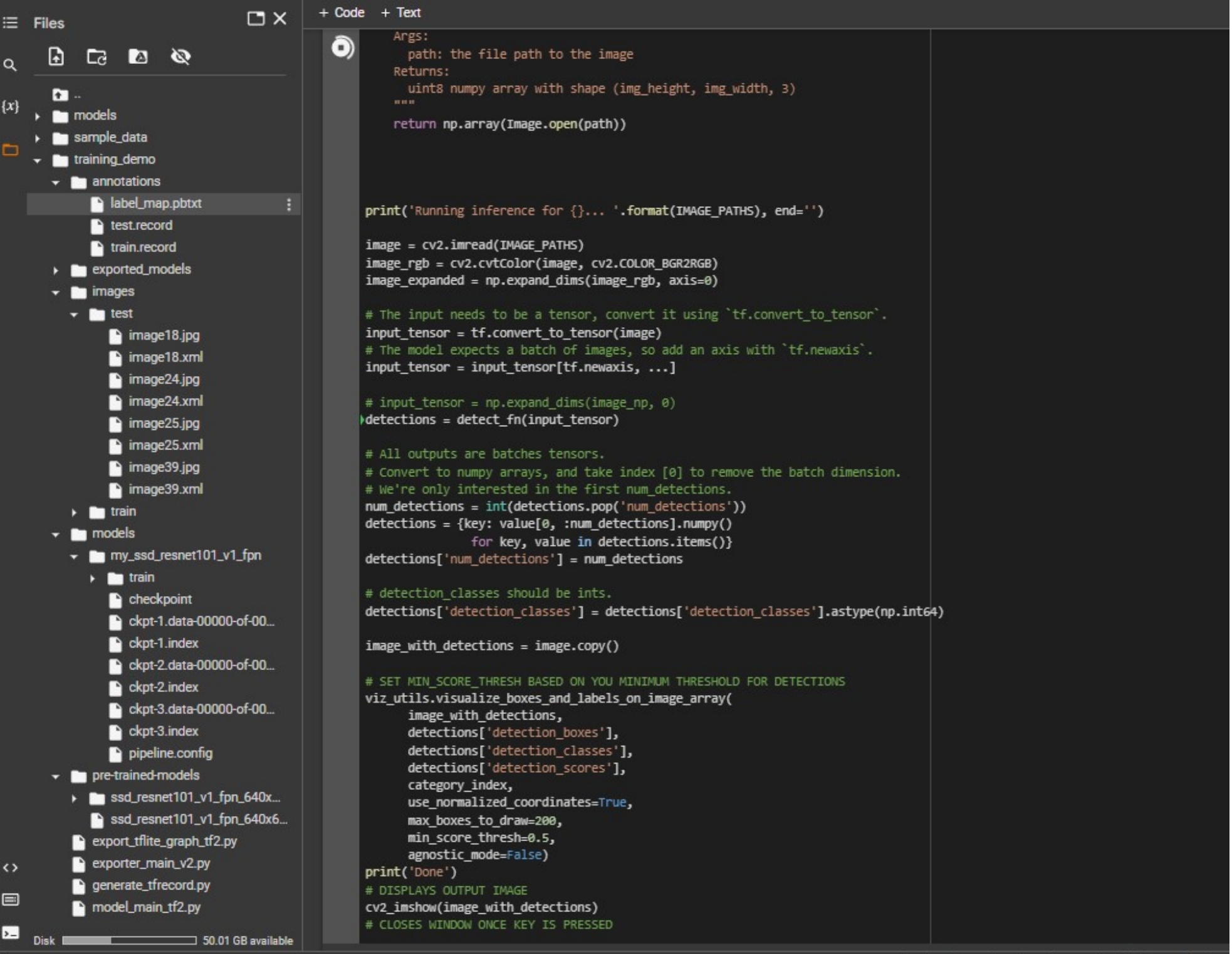
Files
..
models
sample_data
training_demo
annotations
label_map.pbtxt
test.record
train.record
exported_models
images
test
image18.jpg
image18.xml
image24.jpg
image24.xml
image25.jpg
image25.xml
image39.jpg
image39.xml
train
models
my_ssd_resnet101_v1_fpn
train
checkpoint
ckpt-1.data-00000-of-00...
ckpt-1.index
ckpt-2.data-00000-of-00...
ckpt-2.index
ckpt-3.data-00000-of-00...
ckpt-3.index
pipeline.config
pre-trained-models
ssd_resnet101_v1_fpn_640x...
ssd_resnet101_v1_fpn_640x6...
export_tflite_graph_tf2.py
exporter_main_v2.py
generate_tfrecord.py
model_main_tf2.py
Disk
50.01 GB available
+ Code
+ Text
Args:
path: the file path to the image
Returns:
uint8 numpy array with shape (img_height, img_width, 3)
"""
return np.array(Image.open(path))
print('Running inference for {}... '.format(IMAGE_PATHS), end='')
image = cv2.imread(IMAGE_PATHS)
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
image_expanded = np.expand_dims(image_rgb, axis=0)
# The input needs to be a tensor, convert it using `tf.convert_to_tensor`.
input_tensor = tf.convert_to_tensor(image)
# The model expects a batch of images, so add an axis with `tf.newaxis`.
input_tensor = input_tensor[tf.newaxis, ...]
# input_tensor = np.expand_dims(image_np, 0)
detections = detect_fn(input_tensor)
# All outputs are batches tensors.
# Convert to numpy arrays, and take index [0] to remove the batch dimension.
# We're only interested in the first num_detections.
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)
image_with_detections = image.copy()
# SET MIN_SCORE_THRESH BASED ON YOU MINIMUM THRESHOLD FOR DETECTIONS
viz_utils.visualize_boxes_and_labels_on_image_array(
image_with_detections,
detections['detection_boxes'],
detections['detection_classes'],
detections['detection_scores'],
category_index,
use_normalized_coordinates=True,
max_boxes_to_draw=200,
min_score_thresh=0.5,
agnostic_mode=False)
print('Done')
# DISPLAYS OUTPUT IMAGE
cv2_imshow(image_with_detections)
# CLOSES WINDOW ONCE KEY IS PRESSED
Executing (36s) Cell > _call_attribute() > error_handler() > __call__() >
30°C