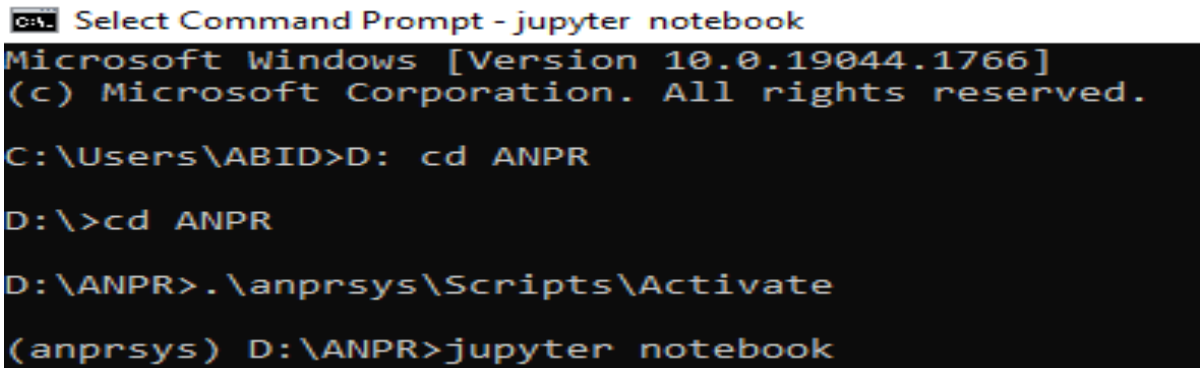
STEPS

1



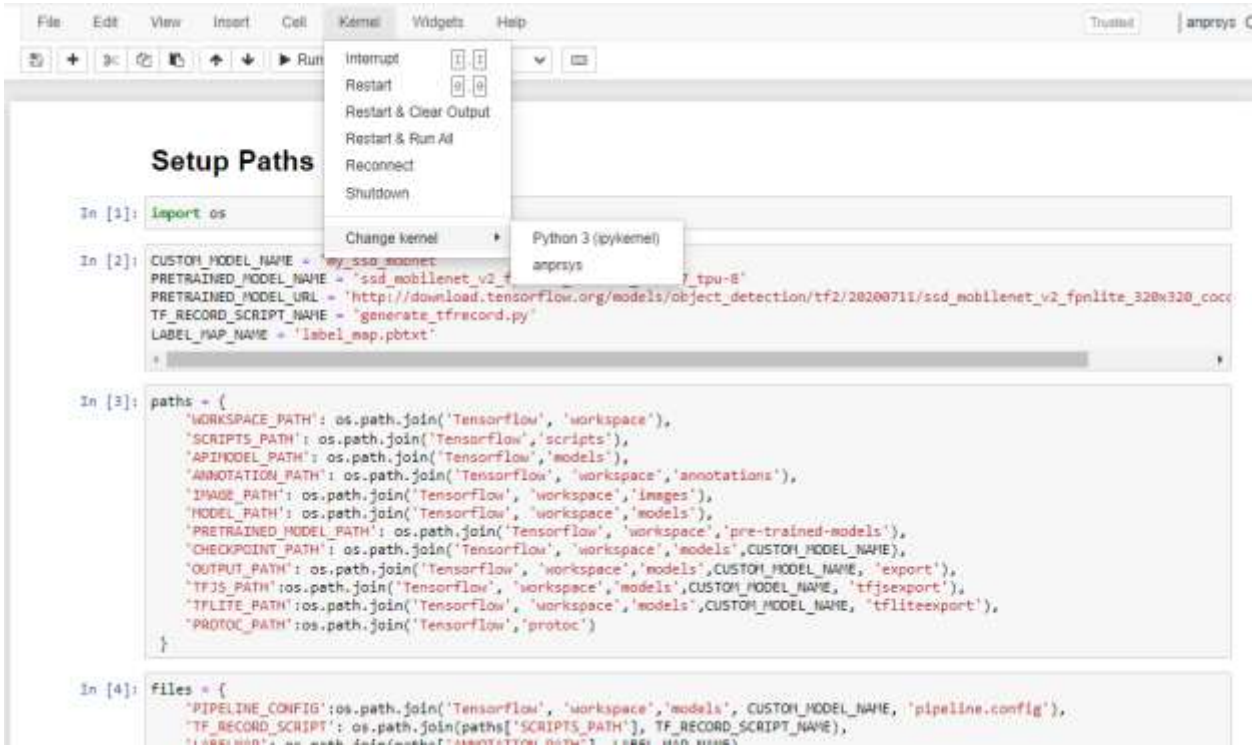
2.

Install dependencies and add virtual environment to the Python Kernel

```
pip install ipykernel

python -m pip install --upgrade pip

python -m ipykernel install --user --name=anprsys
```

3

ensure you change the kernel to the virtual environment as shown below



File Edit View Insert Cell Kernel Widgets Help Trusted anprsys

Run Code

Setup Paths

In [1]:
```
import os
```

In [2]:
```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coc
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
```

In [3]:
```
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')
 }
```

In [4]:
```
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)
}
```

In [5]:
```
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
```

In [ ]:
```
#!pip install --upgrade pip
```

Download TF Models Pretrained Models from Tensorflow Model Zoo and Install TFOD

In [ ]:
```
if os.name=='nt':
    !pip install wget
    import wget
```

In [ ]:
```
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection')):
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
```

In [ ]:
```

if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && cp object_detection/packages/tf2/

if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
    !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
    os.environ['PATH'] += os.pathsep + os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --python_out=. && copy object_detection\\packages\\
    !cd Tensorflow/models/research/slim && pip install -e .
```

In [ ]:
```
#!pip install opencv-python
```

In [ ]:
```
#pip install tf-nightly
```

In [ ]:
```
#!pip install tf-nightly-gpu
```

In [ ]:
```
#!pip install pyparsing==2.4.7
```

In [ ]:
```
#!pip install  matplotlib
```

In [ ]:
```
#!pip uninstall protobuf matplotlib -y
#!pip install protobuf matplotlib==3.2
```

File Edit View Insert Cell Kernel Widgets Help Trusted anprsys

▶ Run Code

Verify Installation

In [ ]:
```
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'builders', 'model_builder_tf2_test.p

!python {VERIFICATION_SCRIPT}
```

In [ ]:
```
#!pip install pycocotools
```

In [ ]:
```
import object_detection
```

In [ ]:
```
if os.name =='posix':
    !wget {PRETRAINED_MODEL_URL}
    !mv {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
if os.name == 'nt':
    wget.download(PRETRAINED_MODEL_URL)
    !move {PRETRAINED_MODEL_NAME+'.tar.gz'} {paths['PRETRAINED_MODEL_PATH']}
    !cd {paths['PRETRAINED_MODEL_PATH']} && tar -zxvf {PRETRAINED_MODEL_NAME+'.tar.gz'}
```

Create Label Map

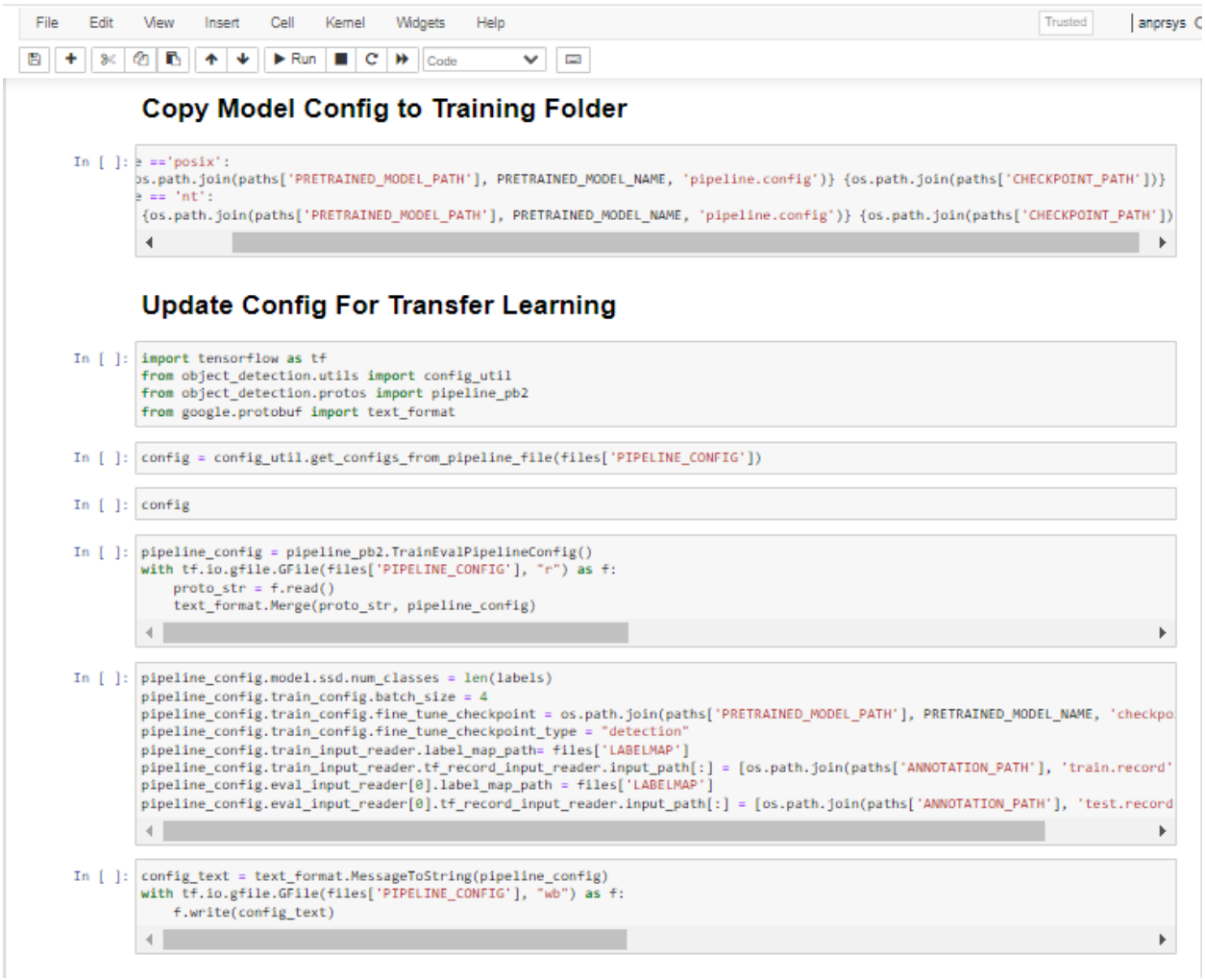
In [ ]:
```
labels = [{'name':'licence', 'id':1}]

with open(files['LABELMAP'], 'w') as f:
    for label in labels:
        f.write('item { \n')
        f.write('\tname:\'{}\'\n'.format(label['name']))
        f.write('\tid:{}\n'.format(label['id']))
        f.write('}\n')
```

Create TF records

In [ ]:
```
if not os.path.exists(files['TF_RECORD_SCRIPT']):
    !git clone https://github.com/nicknochnack/GenerateTFRecord {paths['SCRIPTS_PATH']}
```

In [ ]:
```
#!pip install protobuf==3.19.0
```

In [ ]:
```
x {os.path.join(paths['IMAGE_PATH'], 'train')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'train.record')}
x {os.path.join(paths['IMAGE_PATH'], 'test')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'test.record')}
```


File Edit View Insert Cell Kernel Widgets Help
Trusted
anprsys
Run
Code
Copy Model Config to Training Folder
In [ ]:
e =='posix':
os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
e == 'nt':
{os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])
Update Config For Transfer Learning
In [ ]:
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
In [ ]:
config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
In [ ]:
config
In [ ]:
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
proto_str = f.read()
text_format.Merge(proto_str, pipeline_config)
In [ ]:
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'checkpo
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'train.record'
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'test.record
In [ ]:
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
f.write(config_text)

File Edit View Insert Cell Kernel Widgets Help Trusted anprsys

Run Code

Train the model

In [ ]:
```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection', 'model_main_tf2.py')
```

In [ ]:
```
command = "python {} --model_dir={} --pipeline_config_path={} --num_train_steps=10000".format(TRAINING_SCRIPT, paths['CHECKPOINT
```

In [ ]:
```
print(command)
```

In [ ]:
```
!{command}
```

In [ ]:
```
#!pip install pycocotools
```

Evaluate the Model

In [ ]:
```
command = "python {} --model_dir={} --pipeline_config_path={} --checkpoint_dir={}".format(TRAINING_SCRIPT, paths['CHECKPOINT_PATH
```

In [ ]:
```
print(command)
```

In [ ]:
```
!{command}
```

Load Train Model From Checkpoint

In [ ]:
```
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
```

In [ ]:
```
#!pip install pip install tf-models-official
```

Load pipeline config and build a detection model & Restore checkpoint

In [ ]:
```
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-11')).expect_partial()
```

```
(anprsys) D:\ANPR>python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace\models\my_ssd_mobnet --pipeline_config_path=Tensorflow\wo
rkspace\models\my_ssd_mobnet\pipeline.config --num_train_steps=5000
(anprsys) D:\ANPR>python Tensorflow\models\research\object_detection\model_main_tf2.py --model_dir=Tensorflow\workspace\models\my_ssd_mobnet --pipeline_config_path=Tensorflow\wo
rkspace\models\my_ssd_mobnet\pipeline.config --num_train_steps=5000
2022-07-03 13:15:20.133343: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'cudart64_110.dll'; dlerror: cudart64_110.dll not foun
d
2022-07-03 13:15:20.133472: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
D:\ANPR\anprsys\lib\site-packages\tensorflow_addons\utils\ensure_tf_install.py:37: UserWarning: You are currently using a nightly version of TensorFlow (2.10.0-dev20220702).
TensorFlow Addons offers no support for the nightly versions of TensorFlow. Some things might work, some other might not.
If you encounter a bug, do not file an issue on GitHub.
  warnings.warn(
2022-07-03 13:15:26.588438: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'nvcuda.dll'; dlerror: nvcuda.dll not found
2022-07-03 13:15:26.588535: W tensorflow/stream_executor/cuda/cuda_driver.cc:263] Failed call to cuInit: UNKNOWN ERROR (303)
2022-07-03 13:15:26.592251: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:169] retrieving CUDA diagnostic information for host: DESKTOP-7305C0G
2022-07-03 13:15:26.592403: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:176] hostname: DESKTOP-7305C0G
2022-07-03 13:15:26.593001: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the
following CPU instructions in performance-critical operations:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
WARNING:tensorflow:There are non-GPU devices in `tf.distribute.Strategy`, not using nccl allreduce.
W0703 13:15:26.593598 11608 cross_device_ops.py:1387] There are non-GPU devices in `tf.distribute.Strategy`, not using nccl allreduce.
INFO:tensorflow:Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/device:CPU:0',)
I0703 13:15:26.613833 11608 mirrored_strategy.py:374] Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/device:CPU:0',)
INFO:tensorflow:Maybe overwriting train_steps: 5000
I0703 13:15:26.629454 11608 config_util.py:552] Maybe overwriting train_steps: 5000
INFO:tensorflow:Maybe overwriting use_bfloat16: False
I0703 13:15:26.629454 11608 config_util.py:552] Maybe overwriting use_bfloat16: False
WARNING:tensorflow:From D:\ANPR\anprsys\lib\site-packages\object_detection-0.1-py3.10.egg\object_detection\model_lib_v2.py:563: StrategyBase.experimental_distribute_datasets_fro
m_function (from tensorflow.python.distribute.distribute_lib) is deprecated and will be removed in a future version.
Instructions for updating:
rename to distribute_datasets_from_function
W0703 13:15:26.778272 11608 deprecation.py:350] From D:\ANPR\anprsys\lib\site-packages\object_detection-0.1-py3.10.egg\object_detection\model_lib_v2.py:563: StrategyBase.experim
ental_distribute_datasets_from_function (from tensorflow.python.distribute.distribute_lib) is deprecated and will be removed in a future version.
Instructions for updating:
rename to distribute_datasets_from_function
INFO:tensorflow:Reading unweighted datasets: ['Tensorflow\\workspace\\annotations\\train.record']
I0703 13:15:26.885579 11608 dataset_builder.py:162] Reading unweighted datasets: ['Tensorflow\\workspace\\annotations\\train.record']
INFO:tensorflow:Reading record datasets for input file: ['Tensorflow\\workspace\\annotations\\train.record']
I0703 13:15:26.886584 11608 dataset_builder.py:79] Reading record datasets for input file: ['Tensorflow\\workspace\\annotations\\train.record']
INFO:tensorflow:Number of filenames to read: 1
I0703 13:15:26.887576 11608 dataset_builder.py:80] Number of filenames to read: 1
WARNING:tensorflow:num_readers has been reduced to 1 to match input file shards.
W0703 13:15:26.888576 11608 dataset_builder.py:86] num_readers has been reduced to 1 to match input file shards.
WARNING:tensorflow:From D:\ANPR\anprsys\lib\site-packages\object_detection-0.1-py3.10.egg\object_detection\builders\dataset_builder.py:100: parallel_interleave (from tensorflow.
python.data.experimental.ops.interleave_ops) is deprecated and will be removed in a future version.
Instructions for updating:
use `tf.data.Dataset.interleave(map_func, cycle_length, block_length, num_parallel_calls=tf.data.AUTOTUNE)` instead. If sloppy execution is desired, use `tf.data.Options.determi
stic`.
W0703 13:15:27.200266 11608 deprecation.py:350] From D:\ANPR\anprsys\lib\site-packages\object_detection-0.1-py3.10.egg\object_detection\builders\dataset_builder.py:100: parallel
_interleave (from tensorflow.python.data.experimental.ops.interleave_ops) is deprecated and will be removed in a future version.
Instructions for updating:
use `tf.data.Dataset.interleave(map_func, cycle_length, block_length, num_parallel_calls=tf.data.AUTOTUNE)` instead. If sloppy execution is desired, use `tf.data.Options.determi
stic`.
WARNING:tensorflow:From D:\ANPR\anprsys\lib\site-packages\object_detection-0.1-py3.10.egg\object_detection\builders\dataset_builder.py:235: DatasetV1.map_with_legacy_function (f
rom tensorflow.python.data.ops.dataset_ops) is deprecated and will be removed in a future version.
```

File Edit View Insert Cell Kernel Widgets Help

Trusted | anpsys

Run Code

Detect from an Image

In [ ]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [ ]:
```
category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
```

In [ ]:
```
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'cars432.png')
```

In [ ]:
```
img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)

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()

viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=5,
            min_score_thresh=.8,
            agnostic_mode=False)

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

In [ ]:
```
detections.keys()
```

In [ ]:
```
import easyocr
```

In [ ]:
```
detection_threshold = 0.5
```

In [ ]:
```
image = image_np_with_detections

scores = list(filter(lambda x: x> detection_threshold, detections['detection_scores']))
boxes = detections['detection_boxes'][:len(scores)]
classes = detections['detection_classes'][:len(scores)]
```

In [ ]:
```
scores
```

In [ ]:
```
detections['detection_scores']
```

In [ ]:
```
scores
```

In [ ]:
```
width = image.shape[1]
height = image.shape[0]
```

In [ ]:
```
width
```

In [ ]:
```
# Apply ROI filtering and OCR
for idx, box in enumerate(boxes):
    print(box)
    roi = box*[height, width, height, width]
    region = image [int(roi[0]):int(roi[2]),int(roi[1]):int(roi[3])]
    print(roi)
    reader = easyocr.Reader(['bn'])

    ocr_result = reader.readtext(region)
    print(ocr_result)
    plt.imshow(cv2.cvtColor(region,cv2.COLOR_BGR2RGB))
```

In [ ]:
```
ocr_result
```

In [ ]:
```
for result in ocr_result:
    print(np.sum(np.subtract(result[0][2],result[0][1])))
    print(result[1])
```

In [ ]:
```
region_threshold = 0.005
```

In [ ]:
```
def filter_text(region, ocr_result, region_threshold):
    rectangle_size = region.shape[0]*region.shape[1]

    plate = []

    for result in ocr_result:
        length = np.sum(np.subtract(result[0][1],result[0][0]))
        height = np.sum(np.subtract (result[0][2],result[0][1]))


        if length*height / rectangle_size > region_threshold:
            plate.append(result[1])
        return plate
```

In [ ]:
```
filter_text(region, ocr_result, region_threshold)
```

In [ ]:
```
region_threshold
```

In [ ]:
```
def ocr_it(image, detections, detection_threshold, region_threshold):
    scores= list(filter (lambda x: x> detection_threshold, detections['detection_scores']))
    boxes = detections['detection_boxes'][:len(scores)]
    classes = detections['detection_classes'][:len(scores)]

    width = image.shape[1]
    height =  image.shape[0]


    for idx, box in enumerate(boxes):
        print(box)
        roi = box*[height, width, height, width]
        print(roi)
        region = image[int(roi[0]):int(roi[2]), int(roi[1]):int(roi[3])]
        reader = easyocr.Reader(['bn'])
```

In [ ]:
```
text,region = ocr_it(image_np_with_detections, detections, detection_threshold, region_threshold)
```

Saving Files

In [ ]:
```
import csv
```

In [ ]:
```
import uuid
```

In [ ]:
```
'{}.png'.format(uuid.uuid1())
```

In [ ]:
```
def save_results(text, region, csv_filename, folder_path):
    img_name = '{}.png'.format(uuid.uuid1())

    cv2.imwrite(os.path.join(folder_path, img_name), region)

    with open( csv_filename, mode='a', newline='') as f:
        csv_writer = csv.writer ( f, delimiter = ',' , quotechar='"',quoting=csv.QUOTE_MINIMAL )
        csv_writer.writerow([(img_name, text)])
```

In [ ]:
```
region
```

In [ ]:
```
text
```

In [ ]:
```
save_results(text, region,'detection_results.csv','Detection_images')
```

Real Time Detections from your Webcam

In [ ]:
```
#!pip install opencv-contrib-python
```

In [ ]:
```
#!pip uninstall opencv-contrib-python-headless -y
#!pip uninstall opencv-python  -y
#!pip uninstall opencv-python-headless  -y
```

In [ ]:
```
#!pip list opencv
```

In [ ]:
```
#!pip install opencv-python==3.4.18.65
```

In [ ]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [ ]:
```
cap = cv2.VideoCapture(0)
```

Real Time Detections from your Webcam

In [ ]:
```
#!pip install opencv-contrib-python
```

In [ ]:
```
#!pip uninstall opencv-contrib-python-headless -y
#!pip uninstall opencv-python  -y
#!pip uninstall opencv-python-headless  -y
```

In [ ]:
```
#!pip list opencv
```

In [ ]:
```
#!pip install opencv-python==3.4.18.65
```

In [ ]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

In [ ]:
```
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
while cap.isOpened():
    ret, frame = cap.read()
    image_np = np.array(frame)

    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()

    viz_utils.visualize_boxes_and_labels_on_image_array(
                image_np_with_detections,
                detections['detection_boxes'],
                detections['detection_classes']+label_id_offset,
                detections['detection_scores'],
                category_index,
                use_normalized_coordinates=True,
                max_boxes_to_draw=5,
                min_score_thresh=.8,
                agnostic_mode=False)

     try:
```