elmuraeec

January 4, 2023

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

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: opencv-python in /usr/local/lib/python3.8/dist-
packages (4.6.0.66)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.8/dist-
packages (from opencv-python) (1.21.6)
```

[4]:
```
# Import opencv
import cv2

# Import uuid
import uuid

# Import Operating System
import os

# Import time
import time
```

[ ]:
```
## Training and detection
```

[ ]:
```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'm'
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/
 ↪tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'labelmap.txt'
```

[120]:
```
paths = {
    'WORKSPACE_PATH': os.path.join('/Users/abhishekshastry/Documents/
 ↪Interview_preparation/DeepLearning/TFTrained/', 'workspace'),
    'SCRIPTS_PATH': os.path.join('/Users/abhishekshastry/Documents/
 ↪Interview_preparation/DeepLearning/TFTrained/','scripts'),
    'APIMODEL_PATH': os.path.join('/Users/abhishekshastry/Documents/
 ↪Interview_preparation/DeepLearning/TFTrained/','models'),
```

```
    'ANNOTATION_PATH': os.path.join(paths['WORKSPACE_PATH'],'annotations'),
    'IMAGE_PATH': os.path.join(paths['WORKSPACE_PATH'],'images'),
    'LABELMAP':os.path.join(paths['WORKSPACE_PATH'],'label')
    #  'MODEL_PATH': os.path.join(paths['WORKSPACE_PATH'],'models')
    # 'PRETRAINED_MODEL_PATH': os.path.
 ↪join(paths['WORKSPACE_PATH'],'pre-trained-models'),
    #  'CHECKPOINT_PATH': os.path.join(paths['MODEL_PATH'],CUSTOM_MODEL_NAME),
    # 'OUTPUT_PATH': os.path.join(paths['CHECKPOINT_PATH'], 'export'),
    # 'TFJS_PATH':os.path.join(paths['CHECKPOINT_PATH'], 'tfjsexport'),
    # 'TFLITE_PATH':os.path.join(paths['CHECKPOINT_PATH'],'tfliteexport'),
    # 'PROTOC_PATH':os.path.join('/Users/abhishekshastry/Documents/
 ↪Interview_preparation/DeepLearning/TFTrained/','protoc')
}
```

[122]:
```
files = {
    'PIPELINE_CONFIG':os.path.join('/Users/abhishekshastry/Documents/
 ↪Interview_preparation/DeepLearning/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)
}
```

[ ]:
```
## this is where the code of obejct detection API gets cloned in local. This is␣
 ↪a pretarined model.
```

[2]:
```
if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research',␣
 ↪'object_detection')):
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
```

[ ]:
```
VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research',␣
 ↪'object_detection', 'builders', 'model_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}
```

[1]:
```
!pip install tensorflow --upgrade
```

[123]:
```
import tensorflow
```

[124]:
```
import tensorflow as tf
```

[125]:
```
pip list
```

```
Package                         Version
------------------------------- ----------------------
absl-py                         1.3.0
aeppl                           0.0.33
```

aesara	2.7.9
aiohttp	3.8.3
aiosignal	1.3.1
alabaster	0.7.12
albumentations	1.2.1
altair	4.2.0
appdirs	1.4.4
arviz	0.12.1
astor	0.8.1
astropy	4.3.1
astunparse	1.6.3
async-timeout	4.0.2
atari-py	0.2.9
atomicwrites	1.4.1
attrs	22.2.0
audioread	3.0.0
autograd	1.5
Babel	2.11.0
backcall	0.2.0
beautifulsoup4	4.6.3
bleach	5.0.1
blis	0.7.9
bokeh	2.3.3
branca	0.6.0
bs4	0.0.1
CacheControl	0.12.11
cachetools	5.2.0
catalogue	2.0.8
certifi	2022.12.7
cffi	1.15.1
cftime	1.6.2
chardet	4.0.0
charset-normalizer	2.1.1
click	7.1.2
clikit	0.6.2
cloudpickle	1.5.0
cmake	3.22.6
cmdstanpy	1.0.8
colorcet	3.0.1
colorlover	0.3.0
commonmark	0.9.1
community	1.0.0b1
confection	0.0.3
cons	0.4.5
contextlib2	0.5.5
convertdate	2.4.0
crashtest	0.3.1
crcmod	1.7

cryptography	39.0.0
cufflinks	0.17.3
cvxopt	1.3.0
cvxpy	1.2.2
cycler	0.11.0
cymem	2.0.7
Cython	0.29.32
daft	0.0.4
dask	2022.2.1
datascience	0.17.5
db-dtypes	1.0.5
debugpy	1.0.0
decorator	4.4.2
defusedxml	0.7.1
descartes	1.1.0
dill	0.3.6
distributed	2022.2.1
dlib	19.24.0
dm-tree	0.1.8
dnspython	2.2.1
docutils	0.17.1
dopamine-rl	1.0.5
earthengine-api	0.1.335
easydict	1.10
ecos	2.0.11
editdistance	0.5.3
en-core-web-sm	3.4.1
entrypoints	0.4
ephem	4.1.4
et-xmlfile	1.1.0
etils	0.9.0
etuples	0.3.8
fa2	0.3.5
fastai	2.7.10
fastcore	1.5.27
fastdownload	0.0.7
fastdtw	0.3.4
fastjsonschema	2.16.2
fastprogress	1.0.3
fastrlock	0.8.1
feather-format	0.4.1
filelock	3.8.2
firebase-admin	5.3.0
fix-yahoo-finance	0.0.22
Flask	1.1.4
flatbuffers	1.12
folium	0.12.1.post1
frozenlist	1.3.3

fsspec	2022.11.0
future	0.16.0
gast	0.4.0
GDAL	2.2.3
gdown	4.4.0
gensim	3.6.0
geographiclib	1.52
geopy	1.17.0
gin-config	0.5.0
glob2	0.7
google	2.0.3
google-api-core	2.11.0
google-api-python-client	2.70.0
google-auth	2.15.0
google-auth-httplib2	0.1.0
google-auth-oauthlib	0.4.6
google-cloud-bigquery	3.4.1
google-cloud-bigquery-storage	2.17.0
google-cloud-core	2.3.2
google-cloud-datastore	2.11.0
google-cloud-firestore	2.7.3
google-cloud-language	2.6.1
google-cloud-storage	2.7.0
google-cloud-translate	3.8.4
google-colab	1.0.0
google-crc32c	1.5.0
google-pasta	0.2.0
google-resumable-media	2.4.0
googleapis-common-protos	1.57.0
googledrivedownloader	0.4
graphviz	0.10.1
greenlet	2.0.1
grpcio	1.51.1
grpcio-status	1.48.2
gspread	3.4.2
gspread-dataframe	3.0.8
gym	0.25.2
gym-notices	0.0.8
h5py	3.1.0
HeapDict	1.0.1
hijri-converter	2.2.4
holidays	0.17.2
holoviews	1.14.9
html5lib	1.0.1
httpimport	0.5.18
httplib2	0.17.4
httpstan	4.6.1
humanize	0.5.1

hyperopt	0.1.2
idna	2.10
imageio	2.9.0
imagesize	1.4.1
imbalanced-learn	0.8.1
imblearn	0.0
imgaug	0.4.0
importlib-metadata	5.2.0
importlib-resources	5.10.1
imutils	0.5.4
inflect	2.1.0
intel-openmp	2023.0.0
intervaltree	2.1.0
ipykernel	5.3.4
ipython	7.9.0
ipython-genutils	0.2.0
ipython-sql	0.3.9
ipywidgets	7.7.1
itsdangerous	1.1.0
jaraco.classes	3.2.3
jax	0.3.25
jaxlib	0.3.25+cuda11.cudnn805
jedi	0.18.2
jeepney	0.8.0
jieba	0.42.1
Jinja2	2.11.3
joblib	1.2.0
jpeg4py	0.1.4
jsonschema	4.3.3
jupyter	1.0.0
jupyter-client	6.1.12
jupyter-console	6.1.0
jupyter_core	5.1.1
jupyterlab-widgets	3.0.5
kaggle	1.5.12
kapre	0.3.7
keras	2.9.0
Keras-Preprocessing	1.1.2
keras-vis	0.4.1
keyring	23.13.1
kiwisolver	1.4.4
korean-lunar-calendar	0.3.1
langcodes	3.3.0
libclang	14.0.6
librosa	0.8.1
lightgbm	2.2.3
llvmlite	0.39.1
lmdb	0.99

locket	1.0.0
logical-unification	0.4.5
LunarCalendar	0.0.9
lxml	4.9.2
Markdown	3.4.1
MarkupSafe	2.0.1
marshmallow	3.19.0
matplotlib	3.2.2
matplotlib-venn	0.11.7
miniKanren	1.0.3
missingno	0.5.1
mistune	0.8.4
mizani	0.7.3
mkl	2019.0
mlxtend	0.14.0
more-itertools	9.0.0
moviepy	0.2.3.5
mpmath	1.2.1
msgpack	1.0.4
multidict	6.0.3
multipledispatch	0.6.0
multitasking	0.0.11
murmurhash	1.0.9
music21	5.5.0
natsort	5.5.0
nbconvert	5.6.1
nbformat	5.7.1
netCDF4	1.6.2
networkx	2.8.8
nibabel	3.0.2
nltk	3.7
notebook	5.7.16
numba	0.56.4
numexpr	2.8.4
numpy	1.21.6
oauth2client	4.1.3
oauthlib	3.2.2
okgrade	0.4.3
opencv-contrib-python	4.6.0.66
opencv-python	4.6.0.66
opencv-python-headless	4.6.0.66
openpyxl	3.0.10
opt-einsum	3.3.0
osqp	0.6.2.post0
packaging	21.3
palettable	3.3.0
pandas	1.3.5
pandas-datareader	0.9.0

pandas-gbq	0.17.9
pandas-profiling	1.4.1
pandocfilters	1.5.0
panel	0.12.1
param	1.12.3
parso	0.8.3
partd	1.3.0
pastel	0.2.1
pathlib	1.0.1
pathy	0.10.1
patsy	0.5.3
pep517	0.13.0
pexpect	4.8.0
pickleshare	0.7.5
Pillow	7.1.2
pip	22.0.4
pip-tools	6.6.2
pkginfo	1.9.3
platformdirs	2.6.0
plotly	5.5.0
plotnine	0.8.0
pluggy	0.7.1
pooch	1.6.0
portpicker	1.3.9
prefetch-generator	1.0.3
preshed	3.0.8
prettytable	3.5.0
progressbar2	3.38.0
prometheus-client	0.15.0
promise	2.3
prompt-toolkit	2.0.10
prophet	1.1.1
proto-plus	1.22.1
protobuf	3.19.6
psutil	5.4.8
psycopg2	2.9.5
ptyprocess	0.7.0
py	1.11.0
pyarrow	9.0.0
pyasn1	0.4.8
pyasn1-modules	0.2.8
pycocotools	2.0.6
pycparser	2.21
pyct	0.4.8
pydantic	1.10.2
pydata-google-auth	1.4.0
pydot	1.3.0
pydot-ng	2.0.0

pydotplus	2.0.2
PyDrive	1.3.1
pyemd	0.5.1
pyerfa	2.0.0.1
Pygments	2.6.1
PyGObject	3.26.1
pylev	1.4.0
pymc	4.1.4
PyMeeus	0.5.12
pymongo	4.3.3
pymystem3	0.2.0
PyOpenGL	3.1.6
pyparsing	3.0.9
pyrsistent	0.19.2
pysimdjson	3.2.0
pysndfile	1.3.8
PySocks	1.7.1
pystan	3.3.0
pytest	3.6.4
python-apt	0.0.0
python-dateutil	2.8.2
python-louvain	0.16
python-slugify	7.0.0
python-utils	3.4.5
pytz	2022.7
pyviz-comms	2.2.1
PyWavelets	1.4.1
PyYAML	6.0
pyzmq	23.2.1
qdldl	0.1.5.post2
qtconsole	5.4.0
QtPy	2.3.0
qudida	0.0.4
readme-renderer	37.3
regex	2022.6.2
requests	2.25.1
requests-oauthlib	1.3.1
requests-toolbelt	0.10.1
resampy	0.4.2
rfc3986	2.0.0
rich	13.0.0
rpy2	3.5.5
rsa	4.9
scikit-image	0.18.3
scikit-learn	1.0.2
scipy	1.7.3
screen-resolution-extra	0.0.0
scs	3.2.2

seaborn	0.11.2
SecretStorage	3.3.3
Send2Trash	1.8.0
setuptools	57.4.0
setuptools-git	1.2
shapely	2.0.0
six	1.15.0
sklearn-pandas	1.8.0
smart-open	6.3.0
snowballstemmer	2.2.0
sortedcontainers	2.4.0
soundfile	0.11.0
spacy	3.4.4
spacy-legacy	3.0.10
spacy-loggers	1.0.4
Sphinx	1.8.6
sphinxcontrib-serializinghtml	1.1.5
sphinxcontrib-websupport	1.2.4
SQLAlchemy	1.4.45
sqlparse	0.4.3
srsly	2.4.5
statsmodels	0.12.2
sympy	1.7.1
tables	3.7.0
tabulate	0.8.10
tblib	1.7.0
tenacity	8.1.0
tensorboard	2.9.1
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.1
tensorflow	2.9.2
tensorflow-datasets	4.6.0
tensorflow-estimator	2.9.0
tensorflow-gcs-config	2.9.1
tensorflow-hub	0.12.0
tensorflow-io-gcs-filesystem	0.29.0
tensorflow-metadata	1.12.0
tensorflow-object-detection-api	0.1.1
tensorflow-probability	0.17.0
termcolor	2.1.1
terminado	0.13.3
testpath	0.6.0
text-unidecode	1.3
textblob	0.15.3
thinc	8.1.6
threadpoolctl	3.1.0
tifffile	2022.10.10
toml	0.10.2

```
tomli                           2.0.1
toolz                           0.12.0
torch                           1.13.0+cu116
torchaudio                      0.13.0+cu116
torchsummary                    1.5.1
torchtext                       0.14.0
torchvision                     0.14.0+cu116
tornado                         6.0.4
tqdm                            4.64.1
traitlets                       5.7.1
tweepy                          3.10.0
twine                           4.0.2
typeguard                       2.7.1
typer                           0.7.0
typing_extensions               4.4.0
tzlocal                         1.5.1
uritemplate                     4.1.1
urllib3                         1.26.13
vega-datasets                   0.9.0
wasabi                          0.10.1
wcwidth                         0.2.5
webargs                         8.2.0
webencodings                    0.5.1
Werkzeug                        1.0.1
wheel                           0.38.4
widgetsnbextension              3.6.1
wordcloud                       1.8.2.2
wrapt                           1.14.1
xarray                          2022.12.0
xarray-einstats                 0.4.0
xgboost                         0.90
xkit                            0.0.0
xlrd                            1.2.0
xlwt                            1.3.0
yarl                            1.8.2
yellowbrick                     1.5
zict                            2.2.0
zipp                            3.11.0
```

[126]:
```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
```

```
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.
↪scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))

  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename
```

[127]:
```
from PIL import Image
import PIL
# Import PyDrive and associated libraries.
# This only needs to be done once in a notebook.
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authenticate and create the PyDrive client.
```

```
# This only needs to be done once in a notebook.
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

[21]:
```
from PIL import Image
import PIL
import glob, os


for label in range(0,5):
    #cap = cv2.VideoCapture(2)
    # filename = take_photo()
    # print('Collecting images for {}'.format(label))
    # time.sleep(2)
    # for imgnum in range(number_imgs):
        print('Collecting image {}'.format(imgnum))
       # ret, frame = filename.read()
        # imgname = os.path.join(IMAGES_PATH,label)
        #imgname = os.path.join(IMAGES_PATH,label,label+'.jpg')
        # print(imgname)
        # print(imgname+filename)
        filename = take_photo()
        print('Collecting images for {}'.format(label))
        time.sleep(2)
        # Create & upload a text file.
        uploaded = drive.CreateFile({filename:filename})
        uploaded.SetContentFile(filename)
        uploaded.Upload()
        print('Uploaded file with ID {}'.format(uploaded.get('id')))

        # for infile in glob.glob(filename):
        #       #file, ext = os.path.splitext(infile)
        #       imgname = os.path.join(IMAGES_PATH,label)
        # with Image.open(infile) as im:
        #       im.save(imgname+infile)

        time.sleep(2)
```

```
Collecting image 0
```

```
<IPython.core.display.Javascript object>
```

```
Collecting images for 0
Uploaded file with ID 15b2MzqpOm-_Yjij0o0cGZhXPSPlnEcrQ
Collecting image 0
```

```
<IPython.core.display.Javascript object>
```

```
Collecting images for 1
Uploaded file with ID 1rrcfKjvkj9VUgaNJR7dww8Ba4RCjODkf
Collecting image 0

<IPython.core.display.Javascript object>

Collecting images for 2
Uploaded file with ID 1EeIr1A3oPxvuaeVY1pBYJ69bQHN1dqyX
Collecting image 0

<IPython.core.display.Javascript object>

Collecting images for 3
Uploaded file with ID 1nnHVK6hIN1YfKtXmYXzbgC0kFcvntBGc
Collecting image 0

<IPython.core.display.Javascript object>

Collecting images for 4
Uploaded file with ID 1-QHRX5PCvayqe80GhCs_VaEoG-uyzBxx
```

[128]:
```
!pip uninstall protobuf matplotlib -y
!pip install protobuf matplotlib==3.2
```

```
Found existing installation: protobuf 3.19.6
Uninstalling protobuf-3.19.6:
  Successfully uninstalled protobuf-3.19.6
Found existing installation: matplotlib 3.2.2
Uninstalling matplotlib-3.2.2:
  Successfully uninstalled matplotlib-3.2.2
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting protobuf
  Downloading protobuf-4.21.12-cp37-abi3-manylinux2014_x86_64.whl (409 kB)
                             409.8/409.8

KB 8.3 MB/s eta 0:00:00
Collecting matplotlib==3.2
  Downloading matplotlib-3.2.0-cp38-cp38-manylinux1_x86_64.whl (12.4 MB)
                              12.4/12.4 MB
53.8 MB/s eta 0:00:00
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib==3.2) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-
packages (from matplotlib==3.2) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib==3.2) (1.4.4)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.8/dist-
packages (from matplotlib==3.2) (1.21.6)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.8/dist-packages (from matplotlib==3.2) (2.8.2)
```

```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-
packages (from python-dateutil>=2.1->matplotlib==3.2) (1.15.0)
Installing collected packages: protobuf, matplotlib
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
tensorflow 2.9.2 requires protobuf<3.20,>=3.9.2, but you have protobuf 4.21.12
which is incompatible.
tensorflow-metadata 1.12.0 requires protobuf<4,>=3.13, but you have protobuf
4.21.12 which is incompatible.
tensorboard 2.9.1 requires protobuf<3.20,>=3.9.2, but you have protobuf 4.21.12
which is incompatible.
Successfully installed matplotlib-3.2.0 protobuf-4.21.12
```

[18]:
```
pip install tensorflow-object-detection-api
```

[16]:
```
import object_detection
```

[6]:
```
!cd {"/Users/abhishekshastry/Documents/Interview_preparation/DeepLearning/
  ↪Images/Labelling"} && python labelImg.py
```

```
/bin/bash: line 0: cd: /Users/abhishekshastry/Documents/Interview_preparation/De
epLearning/Images/Labelling: No such file or directory
```

[14]:
```
## ## Labbeling images
```

[ ]:
```
!pip install --upgrade pyqt5 lxml
```

[ ]:
```
LABELIMG_PATH = os.path.join('Tensorflow', 'labelimg')
```

[ ]:
```
if not os.path.exists(LABELIMG_PATH):
    !mkdir {LABELIMG_PATH}
    !git clone https://github.com/tzutalin/labelImg {LABELIMG_PATH}
```

[ ]:
```
!cd {LABELIMG_PATH} && python labelImg.py
```

[15]:
```
##  Create Label Map
```

[36]:
```
from google.colab import drive
drive.mount('/content/gdrive')
labels = [{'name':'victory1', 'id':1}, {'name':'victory2', 'id':2}, {'name':
  ↪'victory3', 'id':3}, {'name':'victory4', 'id':4}]
with open('/content/gdrive/My Drive/labelmap.txt', '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')
```

```
Mounted at /content/gdrive
```

[92]:
```
#Create TF records
```

[ ]:
```
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'],␣
↪'train')} -l {files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'],␣
↪'train.record')}
```

[ ]:
```
Successfully created the TFRecord file: Tensorflow\workspace\annotations\train.
↪record
Successfully created the TFRecord file: Tensorflow\workspace\annotations\test.
↪record
```

[ ]:
```
## Update Config For Transfer Learning
```

[91]:
```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

[90]:
```
config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
```

[93]:
```
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)
```

[94]:
```
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, 'checkpoint',␣
↪'ckpt-0')
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')]
```

[95]:
```
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)
```

[100]:
```
## Train the model...
```

[96]:
```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research',␣
 ↪'object_detection', 'model_main_tf2.py')
```

[97]:
```
command = "python {} --model_dir={} --pipeline_config_path={}␣
 ↪--num_train_steps=2000".format(TRAINING_SCRIPT,␣
 ↪paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])
```

[99]:
```
## Evaluate the model
```

[98]:
```
command = "python {} --model_dir={} --pipeline_config_path={}␣
 ↪--checkpoint_dir={}".format(TRAINING_SCRIPT,␣
 ↪paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'], paths['CHECKPOINT_PATH'])
```

[ ]:
```
## Load Train Model From Checkpoint
```

[102]:
```
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.utils import config_util
```

[103]:
```
# Load pipeline config and build a detection model
```

[105]:
```
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'],␣
 ↪is_training=False)
```

[107]:
```
# Restore checkpoint
```

[108]:
```
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-5')).expect_partial()

@tf.function
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
```

[109]:
```
## Detect from an Image
```

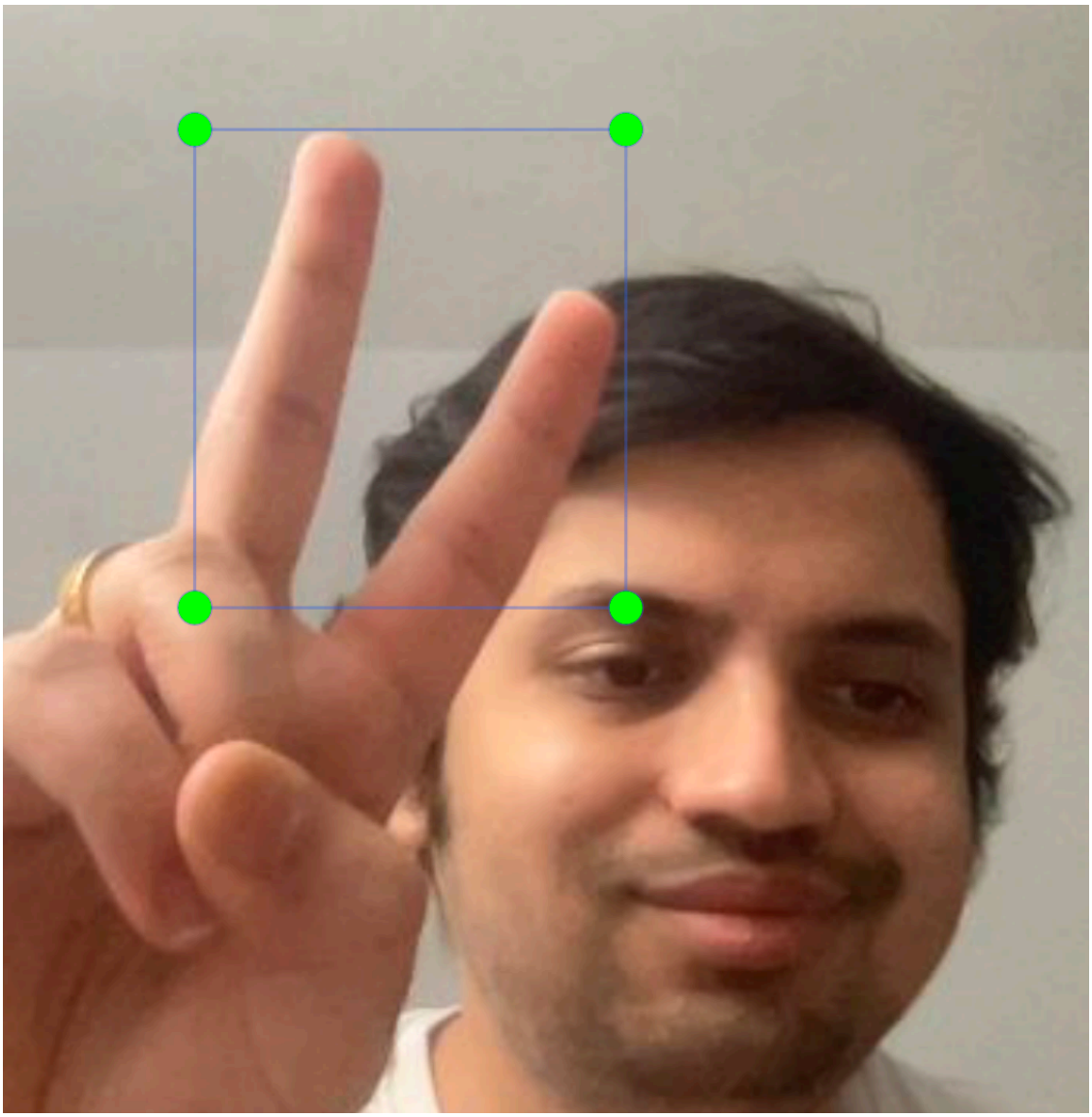
[110]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

[111]:
```
category_index = label_map_util.
 ↪create_category_index_from_labelmap(files['LABELMAP'])
```

[112]:
```
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'livelong.
 ↪02533422-940e-11eb-9dbd-5cf3709bbcc6.jpg')
```

[113]:
```
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()
```

[132]:
```

```



[ ]: