**Python Code (Object Detection):**

# Object Detection

## Setup

"""

import tensorflow as tf

import matplotlib.pyplot as plt

import tensorflow\_hub as hub

import tempfile

from six.moves.urllib.request import urlopen

from six import BytesIO

import time

import numpy as np

from PIL import Image

from PIL import ImageColor

from PIL import ImageDraw

from PIL import ImageFont

from PIL import ImageOps

def show\_image(image1):

fig = plt.figure(figsize=(20, 15))

plt.grid(False)

plt.imshow(image1)

def resize\_download\_image(web\_image, new\_width=256, new\_height=256,

display=False):

\_, filename = tempfile.mkstemp(suffix=".jpg")

response = urlopen(web\_image)

data\_of\_image = response.read()

data\_of\_image = BytesIO(data\_of\_image)

image\_pils = Image.open(data\_of\_image)

image\_pils = ImageOps.fit(image\_pils, (new\_width, new\_height), Image.ANTIALIAS)

image\_pils\_rgb = image\_pils.convert("RGB")

image\_pils\_rgb.save(filename, format="JPEG", quality=90)

print("Image downloaded to %s." % filename)

if display:

show\_image(image\_pils)

return filename

def image\_bounding\_box(image2,

ymin1,

xmin1,

ymax1,

xmax1,

color,

font,

thickness=4,

display\_str\_list1=()):

draw = ImageDraw.Draw(image2)

im\_width1, im\_height1 = image2.size

(left\_side, right\_side, top\_side, bottom\_side) = (xmin1 \* im\_width1, xmax1 \* im\_width1,

ymin1 \* im\_height1, ymax1 \* im\_height1)

draw.line([(left\_side, top\_side), (left\_side, bottom\_side), (right\_side, bottom\_side),

(right\_side, top\_side),

(left\_side, top\_side)],

width=thickness,

fill=color)

display\_str\_heights12 = [font.getsize(ds)[1] for ds in display\_str\_list1]

total\_display\_str\_height = (1 + 2 \* 0.05) \* sum(display\_str\_heights12)

if top\_side > total\_display\_str\_height:

text\_bottom\_side = top\_side

else:

text\_bottom\_side = top\_side + total\_display\_str\_height

for display\_str in display\_str\_list1[::-1]:

text\_width, text\_height = font.getsize(display\_str)

margin = np.ceil(0.05 \* text\_height)

draw.rectangle([(left\_side, text\_bottom\_side - text\_height - 2 \* margin),

(left\_side + text\_width, text\_bottom\_side)],

fill=color)

draw.text((left\_side + margin, text\_bottom\_side - text\_height - margin),

display\_str,

fill="black",

font=font)

text\_bottom\_side -= text\_height - 2 \* margin

def box\_draw(image1, boxes1, class\_names1, scores1, max\_boxes1=10, min\_score1=0.1):

colr = list(ImageColor.colormap.values())

try:

font = ImageFont.truetype("/usr/share/fonts/truetype/liberation/LiberationSansNarrowRegular.ttf",

25)

except IOError:

print("Font not found, using default font.")

font = ImageFont.load\_default()

for i in range(min(boxes1.shape[0], max\_boxes1)):

if scores1[i] >= min\_score1:

ymin1, xmin1, ymax1, xmax1 = tuple(boxes1[i])

display\_str = "{}: {}%".format(class\_names1[i].decode("ascii"),

int(100 \* scores1[i]))

color1 = colr[hash(class\_names1[i]) % len(colr)]

image\_pil = Image.fromarray(np.uint8(image1)).convert("RGB")

image\_bounding\_box(

image\_pil,

ymin1,

xmin1,

ymax1,

xmax1,

color1,

font,

display\_str\_list1=[display\_str])

np.copyto(image1, np.array(image\_pil))

return image1

image\_web\_image = "https://imagevars.gulfnews.com/2020/01/11/191101-sheikh-zayedroad\_16f944d7361\_original-ratio.jpg" #@param

downloaded\_image\_path = resize\_download\_image(image\_web\_image, 1280, 856, True)

module\_transfer =

"https://tfhub.dev/google/faster\_rcnn/openimages\_v4/inception\_resnet\_v2/1" #@param

["https://tfhub.dev/google/openimages\_v4/ssd/mobilenet\_v2/1",

"https://tfhub.dev/google/faster\_rcnn/openimages\_v4/inception\_resnet\_v2/1"]

Image\_classifier\_detect = hub.load(module\_transfer).signatures['default']

def image\_load(path1):

img1 = tf.io.read\_file(path1)

img1 = tf.image.decode\_jpeg(img1, channels=3)

return img1

def detector\_func(Image\_classifier\_detect, path1):

img = image\_load(path1)

converted\_img = tf.image.convert\_image\_dtype(img, tf.float32)[tf.newaxis, ...]

start\_time1 = time.time()

res = Image\_classifier\_detect(converted\_img)

end\_time = time.time()

res = {key:value.numpy() for key,value in res.items()}

print("Found %d objects." % len(res["detection\_scores"]))

print("Inference time: ", end\_time-start\_time1)

image\_with\_boxes = box\_draw(

img.numpy(), res["detection\_boxes"],

res["detection\_class\_entities"], res["detection\_scores"])

show\_image(image\_with\_boxes)

detector\_func(Image\_classifier\_detect, downloaded\_image\_path)