Top of Form

Bottom of Form

Top of Form

Bottom of Form

|  |  |
| --- | --- |
|  | import numpy as np |
|  | import os |
|  | import sys |
|  | import tensorflow as tf |
|  | from matplotlib import pyplot as plt |
|  | from PIL import Image |
|  | import cv2 |
|  | import pytesseract |
|  |  |
|  | from custom\_plate import allow\_needed\_values as anv |
|  | from custom\_plate import do\_image\_conversion as dic |
|  |  |
|  | get\_ipython().magic('matplotlib inline') |
|  | sys.path.append("..") |
|  |  |
|  | from utils import label\_map\_util |
|  | from utils import visualization\_utils as vis\_util |
|  |  |
|  | MODEL\_NAME = 'numplate' |
|  | PATH\_TO\_CKPT = MODEL\_NAME + '/graph-200000/frozen\_inference\_graph.pb' |
|  | PATH\_TO\_LABELS = os.path.join('training', 'object-detection.pbtxt') |
|  | NUM\_CLASSES = 1 |
|  |  |
|  |  |
|  | detection\_graph = tf.Graph() |
|  | with detection\_graph.as\_default(): |
|  | od\_graph\_def = tf.GraphDef() |
|  | with tf.gfile.GFile(PATH\_TO\_CKPT, 'rb') as fid: |
|  | serialized\_graph = fid.read() |
|  | od\_graph\_def.ParseFromString(serialized\_graph) |
|  | tf.import\_graph\_def(od\_graph\_def, name='') |
|  |  |
|  |  |
|  | label\_map = label\_map\_util.load\_labelmap(PATH\_TO\_LABELS) |
|  | categories = label\_map\_util.convert\_label\_map\_to\_categories(label\_map, max\_num\_classes=NUM\_CLASSES, use\_display\_name=True) |
|  | category\_index = label\_map\_util.create\_category\_index(categories) |
|  |  |
|  |  |
|  | def load\_image\_into\_numpy\_array(image): |
|  | (im\_width, im\_height) = image.size |
|  | return np.array(image.getdata()).reshape( |
|  | (im\_height, im\_width, 3)).astype(np.uint8) |
|  |  |
|  |  |
|  | PATH\_TO\_TEST\_IMAGES\_DIR = 'png\_tesseract/test\_ram' |
|  | TEST\_IMAGE\_PATHS = [ os.path.join(PATH\_TO\_TEST\_IMAGES\_DIR, 'image{}.jpg'.format(i)) for i in range(5, 6) ] |
|  | IMAGE\_SIZE = (12, 8) |
|  | TEST\_ARUNAA=os.path.join('numplate') |
|  | count = 0 |
|  |  |
|  |  |
|  | with detection\_graph.as\_default(): |
|  | with tf.Session(graph=detection\_graph) as sess: |
|  | image\_tensor = detection\_graph.get\_tensor\_by\_name('image\_tensor:0') |
|  | detection\_boxes = detection\_graph.get\_tensor\_by\_name('detection\_boxes:0') |
|  | detection\_scores = detection\_graph.get\_tensor\_by\_name('detection\_scores:0') |
|  | detection\_classes = detection\_graph.get\_tensor\_by\_name('detection\_classes:0') |
|  | num\_detections = detection\_graph.get\_tensor\_by\_name('num\_detections:0') |
|  | for image\_path in TEST\_IMAGE\_PATHS: |
|  | image = Image.open(image\_path) |
|  | image\_np = load\_image\_into\_numpy\_array(image) |
|  | image\_np\_expanded = np.expand\_dims(image\_np, axis=0) |
|  | (boxes, scores, classes, num) = sess.run( |
|  | [detection\_boxes, detection\_scores, detection\_classes, num\_detections], |
|  | feed\_dict={image\_tensor: image\_np\_expanded}) |
|  | ymin = boxes[0,0,0] |
|  | xmin = boxes[0,0,1] |
|  | ymax = boxes[0,0,2] |
|  | xmax = boxes[0,0,3] |
|  | (im\_width, im\_height) = image.size |
|  | (xminn, xmaxx, yminn, ymaxx) = (xmin \* im\_width, xmax \* im\_width, ymin \* im\_height, ymax \* im\_height) |
|  | cropped\_image = tf.image.crop\_to\_bounding\_box(image\_np, int(yminn), int(xminn),int(ymaxx - yminn), int(xmaxx - xminn)) |
|  | img\_data = sess.run(cropped\_image) |
|  | count = 0 |
|  | filename = dic.yo\_make\_the\_conversion(img\_data, count) |
|  | pytesseract.tesseract\_cmd = '/home/tensorflow-cuda/dharun\_custom/models/research/object\_detection/tessdata/' |
|  | text = pytesseract.image\_to\_string(Image.open(filename),lang=None) |
|  | print('CHARCTER RECOGNITION : ',anv.catch\_rectify\_plate\_characters(text)) |
|  | vis\_util.visualize\_boxes\_and\_labels\_on\_image\_array( |
|  | image\_np, |
|  | np.squeeze(boxes), |
|  | np.squeeze(classes).astype(np.int32), |
|  | np.squeeze(scores), |
|  | category\_index, |
|  | use\_normalized\_coordinates=True, |
|  | line\_thickness=5) |
|  |  |
|  |  |
|  | plt.figure(figsize=IMAGE\_SIZE) |
|  | # plt.imshow(img\_data) |
|  | # plt.show() |