predict_SSD-MobileNet-with-OpenCV

May 6, 2020

1 Preparation for SSD MobileNet with OpenCV

Source: Object detection with deep learning and OpenCV

get resources https://app.monstercampaigns.com/c/tortsem7qkvyuxc4cyfi

2 SSD MobileNet with OpenCV

[1]:

```
# import the necessary packages
import numpy as np
import cv2
from os import listdir
from os.path import isfile, join
import json
```

[2]:

```
args = {}
args["prototxt"] = 'models/MobileNetSSD_deploy.prototxt.txt'
args["model"] = 'models/MobileNetSSD_deploy.caffemodel'
args["confidence"] = 0.2

img_path = 'testset-img/'
args["image"] = [f for f in listdir(img_path) if f.endswith('.jpg')]

det_dir = 'predicted_boxes/'
net_type = 'ssd-mobilenet-opencv'
```

[3]:

```
# initialize the list of class labels MobileNet SSD was trained to
# detect, then generate a set of bounding box colors for each class
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
        "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
        "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
        "sofa", "train", "tvmonitor"]
```

[4]:

```
# load our serialized model from disk
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
```

```
[INFO] loading model…
```

[5]:

```
# load the input image and construct an input blob for the image
# by resizing to a fixed 300x300 pixels and then normalizing it
# (note: normalization is done via the authors of the MobileNet SSD
# implementation)
result_dict = {}
label = ['person', 'car']
for l in label:
    result_dict[l] = {}
    for f in args["image"]:
        image = cv2.imread(img_path + f)
        (h, w) = image.shape[:2]
        blob = cv2.dnn.blobFromImage(cv2.resize(image, (300, 300)), 0.007843,␣
 ↪(300, 300), 127.5)

        result_dict[l][f] = {}
        result_dict[l][f]['boxes'] = []
        result_dict[l][f]['scores'] = []

        # pass the blob through the network and obtain the detections and
        # predictions
        net.setInput(blob)
        detections = net.forward()

        # loop over the detections
        for i in np.arange(0, detections.shape[2]):
            # extract the confidence (i.e., probability) associated with the
            # prediction
            confidence = detections[0, 0, i, 2]

            # filter out weak detections by ensuring the `confidence` is
            # greater than the minimum confidence
            if confidence > args["confidence"]:
                # extract the index of the class label from the `detections`,
                # then compute the (x, y)-coordinates of the bounding box for
                # the object
                idx = int(detections[0, 0, i, 1])
                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")

                # display the prediction
                c = CLASSES[idx]
                if (l == 'person' and idx == 15) or (l == 'car' and idx in [6,␣
 ↪7]):
                    result_dict[l][f]['boxes'].append([int(startX),␣
 ↪int(startY), int(endX), int(endY)])
```

```
                        result_dict[l][f]['scores'].append(float(confidence))
```

```
[6]: for l in label:
         with open(det_dir+'predicted_boxes-'+net_type+'-'+l+'.json', 'w') as fp:
             json.dump(result_dict[l], fp)
```