predict_YOLOv3-tiny

May 5, 2020

1 Preparation for Darknet

Source: https://pjreddie.com/darknet/yolo/

gets configuration files

```
git clone https://github.com/pjreddie/darknet
copy darknet/cfg/yolov3.cfg -P ./cfg
copy darknet/cfg/yolo3-tiny.cfg -P ./cfg
```

gets weights files

```
wget https://pjreddie.com/media/files/yolov3.weights -P ./weights
wget https://pjreddie.com/media/files/yolov3-tiny.weights -P ./weights
```

2 YOLOv3 & YOLOv3-TINY

Source: Object Detection using YoloV3 and OpenCV

[1]:
```
import cv2
import numpy as np
import time
```

[2]:
```
def load_yolo(weights_file, cfg_file):
    net = cv2.dnn.readNet(weights_file, cfg_file)
    classes = []
    with open("data/coco.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]
    layers_names = net.getLayerNames()
    output_layers = [layers_names[i[0]-1] for i in net.
 ↪getUnconnectedOutLayers()]
    colors = np.random.uniform(0, 255, size=(len(classes), 3))
    return net, classes, colors, output_layers
```

[3]:
```
def load_image(img_file):
    # image loading
    img = cv2.imread(img_file)
    height, width, channels = img.shape
    return img, height, width, channels
```

[4]:
```
def detect_objects(img, net, outputLayers, size):
    blob = cv2.dnn.blobFromImage(img, scalefactor=0.00392, size=size, mean=(0,␣
↪0, 0), swapRB=True, crop=False)
    net.setInput(blob)
    outputs = net.forward(outputLayers)
    return blob, outputs
```

[5]:
```
def get_box_dimensions(outputs, height, width):
    boxes = []
    confs = []
    class_ids = []
    for output in outputs:
        for detect in output:
            scores = detect[5:]
            class_id = np.argmax(scores)
            conf = scores[class_id]
            if conf > 0.3:
                center_x = int(detect[0] * width)
                center_y = int(detect[1] * height)
                w = int(detect[2] * width)
                h = int(detect[3] * height)
                x = int(center_x - w/2)
                y = int(center_y - h / 2)
                boxes.append([x, y, w, h])
                confs.append(float(conf))
                class_ids.append(class_id)
    return boxes, confs, class_ids
```

3 Result

[6]:
```
from os import listdir
from os.path import isfile, join
import json
```

[7]:
```
def convert_points(p):
    p1 = p[:2]
    p2 = [p[0] + p[2], p[1] + p[3]]
    return p[:2] + [p[0] + p[2], p[1] + p[3]]

def batch_prediction(yolo_version, print_progress = True):

    # loads a model
    weights_path = "/Users/chanho/Documents/GitLab/niceface/evaluation/weights/"
    weights_name = yolo_version + '.weights'
    weights_file = weights_path + weights_name

    cfg_path = "/Users/chanho/Documents/GitLab/niceface/evaluation/cfg/"
    cfg_name = yolo_version + '.cfg'
    cfg_file = cfg_path + cfg_name

    if yolo_version == 'yolov3':
        img_size = (416, 416)
    elif yolo_version == 'yolov3-tiny':
        img_size = (320, 320)

    model, classes, colors, output_layers = load_yolo(weights_file, cfg_file)
    if print_progress:
        print('%s is loaded.' % (yolo_version))

    # detects objects
    img_path = '/Users/chanho/Documents/GitLab/niceface/evaluation/testset-img/'
    img_names = [f for f in listdir(img_path) if f.endswith('.jpg')]

    result_dict = {}
    label = ['person', 'car']
    for l in label:
        result_dict[l] = {}
        for f in img_names:
            img_file = img_path + f
            image, height, width, channels = load_image(img_file)
            blob, outputs = detect_objects(image, model, output_layers,␣
↪img_size)
            boxes, confs, class_ids = get_box_dimensions(outputs, height, width)
            indexes = cv2.dnn.NMSBoxes(boxes, confs, 0.5, 0.4)

            result_dict[l][f] = {}
            result_dict[l][f]['boxes'] = []
            result_dict[l][f]['scores'] = []
            for i, c in enumerate(class_ids):
                if i in indexes:
                    if (l == 'person' and c == 0) or (l == 'car' and c in [2,␣
↪5, 7]): # car, bus, truck
                        result_dict[l][f]['boxes'].
↪append(convert_points(boxes[i]))
                        result_dict[l][f]['scores'].append(confs[i])
        if print_progress:
            print('%s is predicted in %d images.' % (l, len(img_names)))

    # writes results
    det_dir = '/Users/chanho/Documents/GitLab/niceface/evaluation/
↪predicted_boxes/'
```

```
    for l in label:
        with open(det_dir+'/predicted_boxes-'+yolo_version+'-'+l+'.json', 'w')␣
↪as fp:
            json.dump(result_dict[l], fp)
        if print_progress:
            print('%s is writeen.' % (l))
```

[8]:
```
batch_prediction('yolov3', print_progress = True)
batch_prediction('yolov3-tiny', print_progress = True)
```

```
yolov3 is loaded.
person is predicted in 100 images.
car is predicted in 100 images.
person is writeen.
car is writeen.
yolov3-tiny is loaded.
person is predicted in 100 images.
car is predicted in 100 images.
person is writeen.
car is writeen.
```