

前言

YOLOv3 应该是大家接触的第一款图像识别算法，依赖于darknet框架，相对于pytorch框架更小巧简单，了解其基础配置，可以为之后的识别算法打基础，比如YOLOv4的配置也是一致的。在熟悉该系列配置后，可以尝试使用pytorch框架的YOLO系列算法，更加精准高效。[ultralytics/yolov3: YOLOv3 in PyTorch > ONNX > CoreML > TFLite \(github.com\)](https://github.com/ultralytics/yolov3)

本次任务具体实现流程请参考以下教程。

默认使用CPU，但是建议各位使用GPU去训练。

默认在ubuntu系统下进行，条件有限的可以使用windows系统。

要求

- 1.使用YOLOv3算法与darknet框架训练
- 2.自主选定 3 种物体，制作数据集，建议每种类不少于200张各角度、背景、远近的图片
- 3.提交测试集识别效果图，不少于20张，识别率不低于0.3。（在实际使用场景，识别率不是最重要的衡量标准，数值也并非越高越好）
- 4.将训练过程中所用到的代码、图片，以及 文字心得（README.md）上传至个人代码托管平台，如github，gitee，gitlab等
- 5.以上内容也务必打包一份，发至QQ群内

*最终上传群文件夹“结业任务提交文件夹”，命名格式：**结业 专业班级-姓名**



本次也是视觉培训的最后一次任务，提交日期为**2.20-2.25**。

教程

一、标注工具 (labelimg)

1.下载地址




注意系统版本

 data	2018/10/22 星期...	文件夹	
 labelImg.exe	2018/10/22 星期...	应用程序	13,179 KB

2.双击运行



3.保存后的文件为xml格式

 p1.xml	2019/9/7 星期六 ...	XML 文件	2 KB
 p2.xml	2019/9/7 星期六 ...	XML 文件	3 KB
 p3.xml	2019/9/7 星期六 ...	XML 文件	3 KB

二、下载编译darknet

1.拉取darknet

```
git clone https://github.com/pjreddie/darknet
cd darknet
```

2.修改配置文件Makefile 如果使用GPU，此处可参考[Ubuntu下GPU模式YOLOv3部署 - 知乎\(zhihu.com\)](https://zhuanlan.zhuanlan.com/p/51111111)

```
GPU=1 #如果使用GPU设置为1，CPU设置为0
CUDNN=1 #如果使用CUDNN设置为1，否则为0
OPENCV=0 #如果调用摄像头，还需要设置OPENCV为1，否则为0
OPENMP=0 #如果使用OPENMP设置为1，否则为0
DEBUG=0 #如果使用DEBUG设置为1，否则为0
```

3.开始编译

```
make
```

4.下载yolov3预训练模型

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

5.测试

```
./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg
```

或者

```
./darknet detector test cfg/coco.data cfg/yolov3.cfg yolov3.weights data/dog.jpg
```

[官网链接](#)

三、准备数据集、训练、测试

1.在darknet目录下创建myData文件夹，目录结构如下，将之前标注好的图片和xml文件放到对应目录下

```
myData
...JPEGImages#存放图像

...Annotations#存放图像对应的xml文件

...ImageSets/Main # 存放训练/验证图像的名字（格式如 000001.jpg或者000001），里面包括
train.txt。这里给出的格式是：000000，因为下面的代码中给出了图像的格式。
```

将自己的数据集图片拷贝到JPEGImages目录下。将数据集label文件拷贝到Annotations目录下。在myData下创建test.py，将下面代码拷贝进去运行，将生成四个文件：train.txt,val.txt,test.txt和trainval.txt。

```
import os
import random

trainval_percent = 0.1
train_percent = 0.9
xmlfilepath = 'Annotations'
txtsavepath = 'ImageSets/Main'
total_xml = os.listdir(xmlfilepath)
num = len(total_xml)
list = range(num)
tv = int(num * trainval_percent)
tr = int(tv * train_percent)
trainval = random.sample(list, tv)
train = random.sample(trainval, tr)

ftrainval = open('ImageSets/Main/trainval.txt', 'w')
ftest = open('ImageSets/Main/test.txt', 'w')
ftrain = open('ImageSets/Main/train.txt', 'w')
fval = open('ImageSets/Main/val.txt', 'w')

for i in list:
    name = total_xml[i][:-4] + '\n'
    if i in trainval:
        ftrainval.write(name)
        if i in train:
            ftest.write(name)
        else:
            fval.write(name)
    else:
        ftrain.write(name)

ftrainval.close()
ftrain.close()
fval.close()
ftest.close()
```

运行test.py

	名称	修改日期	类型	大小
VOC2007				
Annotations	test.txt	2019/9/7 星期六 ...	文本文档	0 KB
ImageSets	train.txt	2019/9/7 星期六 ...	文本文档	1 KB
Main	trainval.txt	2019/9/7 星期六 ...	文本文档	0 KB
JPEGImages	val.txt	2019/9/7 星期六 ...	文本文档	0 KB

2.将数据转换成darknet支持的格式

yolov3提供了将VOC数据集转为YOLO训练所需要的格式的代码，在scripts/voc_label.py文件中。这里提供一个修改版本的。在darknet文件夹下新建一个my_labels.py文件，内容如下：

```
import xml.etree.ElementTree as ET
import pickle
import os
from os import listdir, getcwd
from os.path import join

#源代码sets=[('2012', 'train'), ('2012', 'val'), ('2007', 'train'), ('2007', 'val'), ('2007', 'test')]
sets=[('myData', 'train')] # 改成自己建立的myData

classes = ["person", "foot", "face"] # 改成自己的类别

def convert(size, box):
    dw = 1./(size[0])
    dh = 1./(size[1])
    x = (box[0] + box[1])/2.0 - 1
    y = (box[2] + box[3])/2.0 - 1
    w = box[1] - box[0]
    h = box[3] - box[2]
    x = x*dw
    w = w*dw
    y = y*dh
    h = h*dh
    return (x,y,w,h)

def convert_annotation(year, image_id):
    in_file = open('myData/Annotations/%s.xml'%(image_id)) # 源代码
    out_file = open('myData/labels/%s.txt'%(image_id), 'w') # 源代码
    tree=ET.parse(in_file)
    root = tree.getroot()
    size = root.find('size')
    w = int(size.find('width').text)
    h = int(size.find('height').text)

    for obj in root.iter('object'):
        difficult = obj.find('difficult').text
        cls = obj.find('name').text
        if cls not in classes or int(difficult)==1:
            continue
        cls_id = classes.index(cls)
        xmlbox = obj.find('bndbox')
        b = (float(xmlbox.find('xmin').text), float(xmlbox.find('xmax').text), float(xmlbox.find('ymin').text), float(xmlbox.find('ymax').text))
        bb = convert((w,h), b)
```

```

        out_file.write(str(cls_id) + " " + " ".join([str(a) for a in bb]) +
'\n')

wd = getcwd()

for year, image_set in sets:
    if not os.path.exists('myData/labels/'): # 改成自己建立的myData
        os.makedirs('myData/labels/')
    image_ids = open('myData/ImageSets/Main/%s.txt'%
(image_set)).read().strip().split()
    list_file = open('myData/%s_%s.txt'%(year, image_set), 'w')
    for image_id in image_ids:
        list_file.write('%s/myData/JPEGImages/%s.jpg\n'%(wd, image_id))
        convert_annotation(year, image_id)
    list_file.close()

```

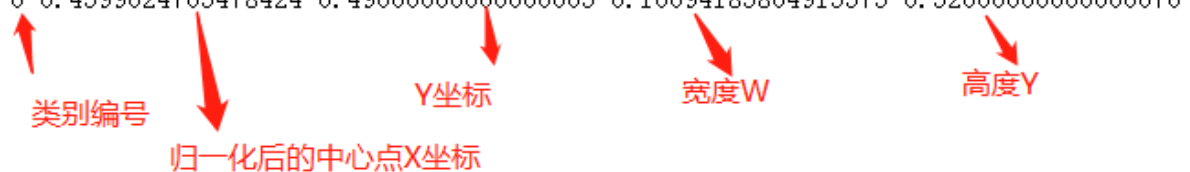
3.运行该脚本

```
python my_labels.py
```

会在./myData目录下生成一个labels文件夹一个txt文件(myData_train.txt)(内容是: 类别的编码和目标的相对位置)。

lables文件中的txt文件的含义为:

0 0.4399624765478424 0.49000000000000005 0.10694183864915573 0.52666666666666670



同理如果要生成训练数据 sets=[('myData', 'train')] 改为sets=[('myData', 'train'), ('myData', 'test')]

具体的每一个值的计算方式是这样的: 假设一个标注的boundingbox的左下角和右上角坐标分别为 (x1,y1) (x2,y2), 图像的宽和高分别为w,h

归一化的中心点x坐标计算公式: $((x2+x1) / 2.0) / w$

归一化的中心点y坐标计算公式: $((y2+y1) / 2.0) / h$

归一化的目标框宽度的计算公式: $(x2-x1) / w$

归一化的目标框高度计算公式: $(y2-y1) / h$

4.修改darknet/cfg下的voc.data和yolov3-voc.cfg文件

为了保险起见, 复制这两个文件, 并分别重命名为my_data.data和my_yolov3.cfg

my_data.data内容:

以下出现路径, 都需要改成自己真实使用路径

```
classes= 3 ##改为自己的分类个数
##下面都改为自己的路径
train = /home/XXX/darknet/myData/myData_train.txt
names = /home/XXX/darknet/myData/myData.names #稍后需要创建这个文件
backup = /home/XXX/darknet/myData/weights
```

my_yolov3.cfg的内容:

/yolo, 总共会搜出3个含有yolo的地方。

每个地方都必须要改2处, filters: $3 * (5 + \text{len}(\text{classes}))$;

其中: classes: $\text{len}(\text{classes}) = 3$, 这里以我的工程为例

filters = 24

classes = 3

可修改: random = 1: 原来是1, 显存小改为0。(是否要多尺度输出。)

```
[convolutional]
size=1
stride=1
pad=1
filters=24
activation=linear

[yolo]
mask = 6,7,8
anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326
classes=3
num=9
jitter=.3
ignore_thresh = .5
truth_thresh = 1
random=1

[route]
layers = -4

[convolutional]
batch_normalize=1
/yolo
```

5.可以指定训练批次和训练轮数

```
[net]
# Testing
batch=1
subdivisions=1
# Training
# batch=64
# subdivisions=16
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 50200
policy=steps
steps=40000,45000
scales=.1,.1
```

[net]

```

# Testing          ### 测试模式
# batch=1
# subdivisions=1
# Training        ### 训练模式，每次前向的图片数目 = batch/subdivisions
batch=64
subdivisions=16
width=416          ### 网络的输入宽、高、通道数
height=416
channels=3
momentum=0.9       ### 动量
decay=0.0005       ### 权重衰减
angle=0
saturation = 1.5    ### 饱和度
exposure = 1.5      ### 曝光度
hue=.1             ### 色调
learning_rate=0.001 ### 学习率
burn_in=1000        ### 学习率控制的参数
max_batches = 50200 ### 迭代次数
policy=steps        ### 学习率策略
steps=40000,45000   ### 学习率变动步长

```

因为是训练，所以注释Testing,打开Training，其中

batch=64 每batch个样本更新一次参数。

subdivisions=16 如果内存不够大，将batch分割为subdivisions个子batch，每个子batch的大小为batch/subdivisions。

6.在myData文件夹下新建myData.names文件

```

people
foot
car

```

7.下载预训练权重

```
wget https://pjreddie.com/media/files/darknet53.conv.74
```

8.开始训练

```
./darknet detector train cfg/my_data.data cfg/my_yolov3.cfg darknet53.conv.74
```

或者指定gpu训练，默认使用gpu0

```
./darknet detector train cfg/my_data.data cfg/my_yolov3.cfg darknet53.conv.74 -
gups 0,1,2,3
```

查看gpu信息


```

/tensorflow_container/darknet/myData$ nvidia-smi
Mon Sep 9 09:42:03 2019

```

NVIDIA-SMI 430.14 Driver Version: 430.14 CUDA Version: 10.2									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute	M.		
0	GeForce GTX 108...	Off	00000000:02:00.0	Off			N/A		
29%	47C	P2	59W / 250W	7293MiB / 11178MiB	0%		Default		
1	GeForce GTX 108...	Off	00000000:04:00.0	Off			N/A		
26%	55C	P2	275W / 250W	4657MiB / 11178MiB	93%		Default		
2	GeForce GTX 108...	Off	00000000:83:00.0	Off			N/A		
23%	41C	P2	157W / 250W	4657MiB / 11178MiB	90%		Default		
3	GeForce GTX 108...	Off	00000000:84:00.0	Off			N/A		
23%	42C	P2	214W / 250W	4657MiB / 11178MiB	90%		Default		

Processes:					GPU Memory
GPU	PID	Type	Process name		Usage
0	22050	C	./darknet		7283MiB
1	20327	C	./darknet		4647MiB
2	20327	C	./darknet		4647MiB
3	20327	C	./darknet		4647MiB

从停止处重新训练

```

./darknet detector train cfg/my_data.data cfg/my_yolov3.cfg darknet53.conv.74 -
gups 0,1,2,3 myData/weights/my_yolov3.backup -gpus 0,1,2,3

```

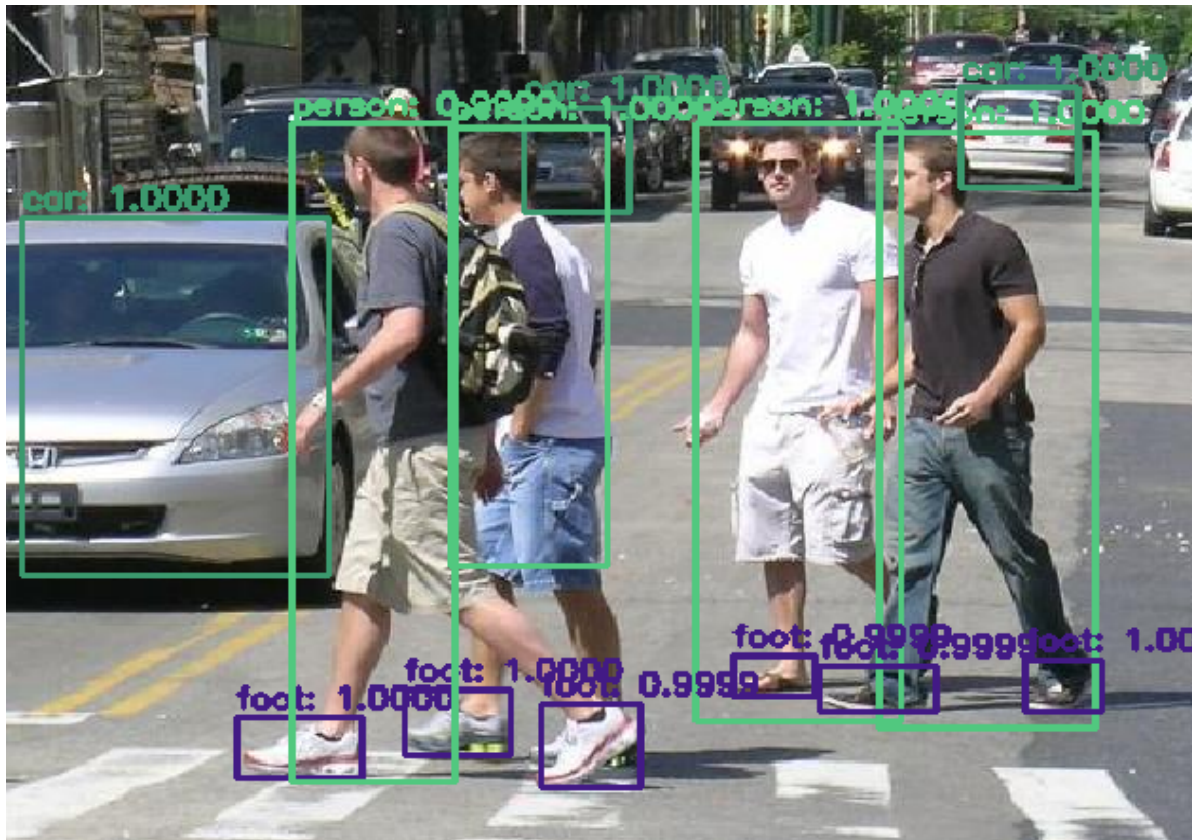
9.测试

```

./darknet detect cfg/my_yolov3.cfg weights/my_yolov3.weights 1.jpg

```

10.效果



11.关于opencv调用yolo模型, 参考<https://www.cnblogs.com/answerThe/p/11486090.html>

