1、将训练类别修改为和自己一样的

SSD-Tensorflow-master—>datasets—>pascalvoc\_common.py

VOC\_LABELS = {

'none': (0, 'Background'),

'aeroplane': (1, 'Vehicle'),

'bicycle': (2, 'Vehicle'),

'bird': (3, 'Animal'),

'boat': (4, 'Vehicle'),

'bottle': (5, 'Indoor'),

'bus': (6, 'Vehicle'),

'car': (7, 'Vehicle'),

'cat': (8, 'Animal'),

'chair': (9, 'Indoor'),

'cow': (10, 'Animal'),

'diningtable': (11, 'Indoor'),

'dog': (12, 'Animal'),

'horse': (13, 'Animal'),

'motorbike': (14, 'Vehicle'),

'Person': (15, 'Person'),

'pottedplant': (16, 'Indoor'),

'sheep': (17, 'Animal'),

'sofa': (18, 'Indoor'),

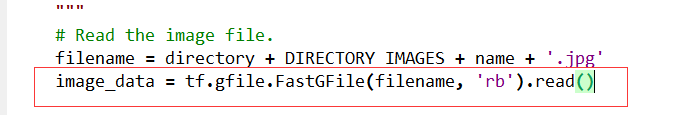
'train': (19, 'Vehicle'),

'tvmonitor': (20, 'Indoor'),

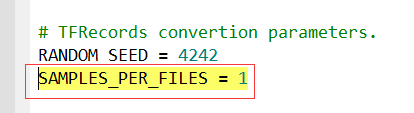
}

2、将图像数据转换为tfrecods格式

SSD-Tensorflow-master—>datasets—>pascalvoc\_to\_tfrecords.py 。。。然后更改文件的83行读取方式为’rb’）



修改如下行，可以修改几张图片转为一个tfrecords，如下图



3、训练模型

train\_ssd\_network.py修改第154行的最大训练步数，将None改为比如50000。

tf.app.flags.DEFINE\_integer('max\_number\_of\_steps', 50000,

'The maximum number of training steps.None is infinite')

（tf.contrib.slim.learning.training函数中max-step为None时训练会无限进行。）

train\_ssd\_network.py,网络参数配置，若需要改，再此文件中进行修改

修改如下参数的数字为500，可以改变训练多长时间保存一次模型

tf.app.flags.DEFINE\_integer(

'save\_summaries\_secs', 500,

'The frequency with which summaries are saved, in seconds.')

tf.app.flags.DEFINE\_integer(

'save\_interval\_secs', 500,

'The frequency with which the model is saved, in seconds.')

a. nets/ssd\_vgg\_300.py (因为使用此网络结构) ，修改87 和88行的类别

default\_params = SSDParams(

img\_shape=(300, 300),

num\_classes=21, #根据自己的数据修改为类别+1

no\_annotation\_label=21, #根据自己的数据修改为类别+1

feat\_layers=['block4', 'block7', 'block8', 'block9', 'block10', 'block11'],

feat\_shapes=[(38, 38), (19, 19), (10, 10), (5, 5), (3, 3), (1, 1)],

anchor\_size\_bounds=[0.15, 0.90],

# anchor\_size\_bounds=[0.20, 0.90],

b. train\_ssd\_network.py,修改类别120行，GPU占用量，学习率，batch\_size等

tf.app.flags.DEFINE\_integer(

'num\_classes', 21, 'Number of classes to use in the dataset.')

#根据自己的数据修改为类别+1

c. eval\_ssd\_network.py 修改类别，66行

tf.app.flags.DEFINE\_integer(

'num\_classes', 21, 'Number of classes to use in the dataset.')

#根据自己的数据修改为类别+1

d. datasets/pascalvoc\_2007.py 根据自己的训练数据修改整个文件

TRAIN\_STATISTICS = {

'none': (0, 0),

'aeroplane': (238, 306), #238图片书， 306目标总数

'bicycle': (243, 353),

'bird': (330, 486),

'boat': (181, 290),

'bottle': (244, 505),

'bus': (186, 229),

'car': (713, 1250),

'cat': (337, 376),

'chair': (445, 798),

'cow': (141, 259),

'diningtable': (200, 215),

'dog': (421, 510),

'horse': (287, 362),

'motorbike': (245, 339),

'person': (2008, 4690),

'pottedplant': (245, 514),

'sheep': (96, 257),

'sofa': (229, 248),

'train': (261, 297),

'tvmonitor': (256, 324),

'total': (5011, 12608), //5011 为训练的图片书，12608为目标总数

}

TEST\_STATISTICS = {

'none': (0, 0),

'aeroplane': (1, 1),

'bicycle': (1, 1),

'bird': (1, 1),

'boat': (1, 1),

'bottle': (1, 1),

'bus': (1, 1),

'car': (1, 1),

'cat': (1, 1),

'chair': (1, 1),

'cow': (1, 1),

'diningtable': (1, 1),

'dog': (1, 1),

'horse': (1, 1),

'motorbike': (1, 1),

'person': (1, 1),

'pottedplant': (1, 1),

'sheep': (1, 1),

'sofa': (1, 1),

'train': (1, 1),

'tvmonitor': (1, 1),

'total': (20, 20),

}

SPLITS\_TO\_SIZES = {

'train': 5011, #训练数据量

'test': 4952, #测试数据量

}

SPLITS\_TO\_STATISTICS = {

'train': TRAIN\_STATISTICS,

'test': TEST\_STATISTICS,

}

NUM\_CLASSES = 20 #类别，根据自己数据的实际类别修改（不包含背景）

方案1 从vgg开始训练其中某些层的参数

# 通过加载预训练好的vgg16模型，对“voc07trainval+voc2012”进行训练

# 通过checkpoint\_exclude\_scopes指定哪些层的参数不需要从vgg16模型里面加载进来

# 通过trainable\_scopes指定哪些层的参数是需要训练的，未指定的参数保持不变,<strong>若注释掉此命令，所有的参数均需要训练</strong>

DATASET\_DIR=/home/doctorimage/kindlehe/common/dataset/VOC0712/

TRAIN\_DIR=.././log\_files/log\_finetune/train\_voc0712\_20170816\_1654\_VGG16/

CHECKPOINT\_PATH=../checkpoints/vgg\_16.ckpt

python3 ../train\_ssd\_network.py \

--train\_dir=${TRAIN\_DIR} \ #训练生成模型的存放路径

--dataset\_dir=${DATASET\_DIR} \ #数据存放路径

--dataset\_name=pascalvoc\_2007 \ #数据名的前缀

--dataset\_split\_name=train \

--model\_name=ssd\_300\_vgg \ #加载的模型的名字

--checkpoint\_path=${CHECKPOINT\_PATH} \ #所加载模型的路径

--checkpoint\_model\_scope=vgg\_16 \ #所加载模型里面的作用域名

--checkpoint\_exclude\_scopes=ssd\_300\_vgg/conv6,ssd\_300\_vgg/conv7,ssd\_300\_vgg/block8,ssd\_300\_vgg/block9,ssd\_300\_vgg/block10,ssd\_300\_vgg/block11,ssd\_300\_vgg/block4\_box,ssd\_300\_vgg/block7\_box,ssd\_300\_vgg/block8\_box,ssd\_300\_vgg/block9\_box,ssd\_300\_vgg/block10\_box,ssd\_300\_vgg/block11\_box \

--trainable\_scopes=ssd\_300\_vgg/conv6,ssd\_300\_vgg/conv7,ssd\_300\_vgg/block8,ssd\_300\_vgg/block9,ssd\_300\_vgg/block10,ssd\_300\_vgg/block11,ssd\_300\_vgg/block4\_box,ssd\_300\_vgg/block7\_box,ssd\_300\_vgg/block8\_box,ssd\_300\_vgg/block9\_box,ssd\_300\_vgg/block10\_box,ssd\_300\_vgg/block11\_box \

--save\_summaries\_secs=60 \ #每60s保存一下日志

--save\_interval\_secs=600 \ #每600s保存一下模型

--weight\_decay=0.0005 \ #正则化的权值衰减的系数

--optimizer=adam \ #选取的最优化函数

--learning\_rate=0.001 \ #学习率

--learning\_rate\_decay\_factor=0.94 \ #学习率的衰减因子

--batch\_size=24 \

--gpu\_memory\_fraction=0.9 #指定占用gpu内存的百分比

方案2 ： 从自己预训练好的模型开始训练（依然可以指定要训练哪些层）

（当你的模型通过vgg训练的模型收敛到大概o.5mAP的时候，可以进行这一步的fine-tune）

# 通过加载预训练好的vgg16模型，对“voc07trainval+voc2012”进行训练

# 通过checkpoint\_exclude\_scopes指定哪些层的参数不需要从vgg16模型里面加载进来

# 通过trainable\_scopes指定哪些层的参数是需要训练的，未指定的参数保持不变

DATASET\_DIR=/home/doctorimage/kindlehe/common/dataset/VOC0712/

TRAIN\_DIR=.././log\_files/log\_finetune/train\_voc0712\_20170816\_1654\_VGG16/

CHECKPOINT\_PATH=./log\_files/log\_finetune/train\_voc0712\_20170712\_1741\_VGG16/model.ckpt-253287

python3 ../train\_ssd\_network.py \

--train\_dir=${TRAIN\_DIR} \ #训练生成模型的存放路径

--dataset\_dir=${DATASET\_DIR} \ #数据存放路径

--dataset\_name=pascalvoc\_2007 \ #数据名的前缀

--dataset\_split\_name=train \

--model\_name=ssd\_300\_vgg \ #加载的模型的名字

--checkpoint\_path=${CHECKPOINT\_PATH} \ #所加载模型的路径

--checkpoint\_model\_scope=vgg\_16 \ #所加载模型里面的作用域名

--checkpoint\_exclude\_scopes=ssd\_300\_vgg/conv6,ssd\_300\_vgg/conv7,ssd\_300\_vgg/block8,ssd\_300\_vgg/block9,ssd\_300\_vgg/block10,ssd\_300\_vgg/block11,ssd\_300\_vgg/block4\_box,ssd\_300\_vgg/block7\_box,ssd\_300\_vgg/block8\_box,ssd\_300\_vgg/block9\_box,ssd\_300\_vgg/block10\_box,ssd\_300\_vgg/block11\_box \

--trainable\_scopes=ssd\_300\_vgg/conv6,ssd\_300\_vgg/conv7,ssd\_300\_vgg/block8,ssd\_300\_vgg/block9,ssd\_300\_vgg/block10,ssd\_300\_vgg/block11,ssd\_300\_vgg/block4\_box,ssd\_300\_vgg/block7\_box,ssd\_300\_vgg/block8\_box,ssd\_300\_vgg/block9\_box,ssd\_300\_vgg/block10\_box,ssd\_300\_vgg/block11\_box \

--save\_summaries\_secs=60 \ #每60s保存一下日志

--save\_interval\_secs=600 \ #每600s保存一下模型

--weight\_decay=0.0005 \ #正则化的权值衰减的系数

--optimizer=adam \ #选取的最优化函数

--learning\_rate=0.001 \ #学习率

--learning\_rate\_decay\_factor=0.94 \ #学习率的衰减因子

--batch\_size=24 \

--gpu\_memory\_fraction=0.9 #指定占用gpu内存的百分比

从自己训练的ssd\_300\_vgg模型开始训练ssd\_512\_vgg的模型

因此ssd\_300\_vgg中没有block12,又因为block7,block8,block9,block10,block11,中的参数张量两个网络模型中不匹配，因此ssd\_512\_vgg中这几个模块的参数不从ssd\_300\_vgg模型中继承，因此使用checkpoint\_exclude\_scopes命令指出。因为所有的参数均需要训练，因此不使用命令--trainable\_scopes

#/bin/bash

DATASET\_DIR=/home/data/xxx/imagedata/xing\_tf/train\_tf/

TRAIN\_DIR=/home/data/xxx/model/xing300512\_model/

CHECKPOINT\_PATH=/home/data/xxx/model/xing300\_model/model.ckpt-60000 #加载的ssd\_300\_vgg模型

python3 ./train\_ssd\_network.py \

--train\_dir=${TRAIN\_DIR} \

--dataset\_dir=${DATASET\_DIR} \

--dataset\_name=pascalvoc\_2007 \

--dataset\_split\_name=train \

--model\_name=ssd\_512\_vgg \

--checkpoint\_path=${CHECKPOINT\_PATH} \

--checkpoint\_model\_scope=ssd\_300\_vgg \

--checkpoint\_exclude\_scopes=ssd\_512\_vgg/block7,ssd\_512\_vgg/block7\_box,ssd\_512\_vgg/block8,ssd\_512\_vgg/block8\_box, ssd\_512\_vgg/block9,ssd\_512\_vgg/block9\_box,ssd\_512\_vgg/block10,ssd\_512\_vgg/block10\_box,ssd\_512\_vgg/block11,ssd\_512\_vgg/b lock11\_box,ssd\_512\_vgg/block12,ssd\_512\_vgg/block12\_box \

#--trainable\_scopes=ssd\_300\_vgg/conv6,ssd\_300\_vgg/conv7,ssd\_300\_vgg/block8,ssd\_300\_vgg/block9,ssd\_300\_vgg/block1 0,ssd\_300\_vgg/block11,ssd\_300\_vgg/block4\_box,ssd\_300\_vgg/block7\_box,ssd\_300\_vgg/block8\_box,ssd\_300\_vgg/block9\_box,ssd\_3 00\_vgg/block10\_box,ssd\_300\_vgg/block11\_box \

--save\_summaries\_secs=28800 \

--save\_interval\_secs=28800 \

--weight\_decay=0.0005 \

--optimizer=adam \

--learning\_rate\_decay\_factor=0.94 \

--batch\_size=16 \

--num\_classes=4 \

-gpu\_memory\_fraction=0.8 \

方案3：从头开始训练自己的模型

# 注释掉CHECKPOINT\_PATH，不提供初始化模型,让模型自己随机初始化权重，从头训练

# 删除checkpoint\_exclude\_scopes和trainable\_scopes，因为是从头开始训练

#　CHECKPOINT\_PATH=./log\_files/log\_finetune/train\_voc0712\_20170712\_1741\_VGG16/model.ckpt-253287

python3 ../train\_ssd\_network.py \

--train\_dir=${TRAIN\_DIR} \ #训练生成模型的存放路径

--dataset\_dir=${DATASET\_DIR} \ #数据存放路径

--dataset\_name=pascalvoc\_2007 \ #数据名的前缀

--dataset\_split\_name=train \

--model\_name=ssd\_300\_vgg \ #加载的模型的名字

#--checkpoint\_path=${CHECKPOINT\_PATH} \ #所加载模型的路径,这里注释掉

#--checkpoint\_model\_scope=vgg\_16 \ #所加载模型里面的作用域名

--save\_summaries\_secs=60 \ #每60s保存一下日志

--save\_interval\_secs=600 \ #每600s保存一下模型

--weight\_decay=0.0005 \ #正则化的权值衰减的系数

--optimizer=adam \ #选取的最优化函数

--learning\_rate=0.00001 \ #学习率

--learning\_rate\_decay\_factor=0.94 \ #学习率的衰减因子

--batch\_size=32

8. 测试或验证

首先将测试数据转换为tfrecords

#!/bin/bash

#this is a shell script to convert pascal VOC datasets into tf-records only

#directory where the original dataset is stored

DATASET\_DIR=/home/xxx/imagedata/VOCdevkit/VOC2007/ #VOC数据保存的文件夹（VOC的目录格式未改变）

#output directory where to store TFRecords files

OUTPUT\_DIR=/home/xxx/imagedata/xingshizheng\_tf #自己建立的保存tfrecords数据的文件夹

python3 ./tf\_convert\_data.py \

--dataset\_name=pascalvoc \

--dataset\_dir=${DATASET\_DIR} \

--output\_name=voc\_2007\_test \ #注意修改为test

--output\_dir=${OUTPUT\_DIR}

在SSD-Tensorflow-master文件夹下建立一个sh文件

#!/bin/bash

# This is the eval script.

DATASET\_DIR=/home/xxx/voc2007\_test\_tfrecords/ #保存的转换为tfrcodes格式的数据

EVAL\_DIR=/home/xxx/ssd\_eval\_log/ # Directory where the results are saved to

CHECKPOINT\_PATH=/home/xxx/Downloads/SSD-Tensorflow-master/checkpoints/VGG\_VOC0712\_SSD\_300x300\_iter\_120000.ckpt #换为自己训练的模型

ython3 ./eval\_ssd\_network.py \

--eval\_dir=${EVAL\_DIR} \

--dataset\_dir=${DATASET\_DIR} \

--dataset\_name=pascalvoc\_2007 \

--dataset\_split\_name=test \

--model\_name=ssd\_300\_vgg \

--checkpoint\_path=${CHECKPOINT\_PATH} \

--batch\_size=1

–dataset\_name=pascalvoc\_2007 、–dataset\_split\_name=train、–model\_name=ssd\_300\_vgg这三个参数不要自己随便取，在代码里，这三个参数是if…else…语句，有固定的判断值，所以要根据实际情况取选择

TypeError: expected bytes, NoneType found SystemError: returned a result with an error set 这是由于CHECKPOINT\_PATH定义的时候不小心多了个＃号键，将输入给注释掉了，如果不想使用预训练的模型，需要将--checkpoint\_path=${CHECKPOINT\_PATH} \注释掉即可

SSD有在VOC07+12的训练集上一起训练的，用一个笨一点的办法： pascalvoc\_to\_tfrecords.py文件中，改变SAMPLES\_PER\_FILES,减少输出tfrecord文件的个数,再修改tf\_convert\_data.py的dataset参数，记得将前后两次的输出名改变一下，前后两次转换的tfrecords放在同一个文件夹下，然后手工重命名。（这里由于只是验证论文的训练方法是否有效，所以没必要写这些自动化代码实现合并，以后要用自己的数据集训练的时候就可以写一些自动化脚本）

有时候运行脚本会报错，可能是之前依次运行导致显存占满。

从pyCharm运行时，如果模型保存路径里之前的模型未删除，将会报错，必须保证该文件夹为空。

在TRAIN\_DIR路径下会产生四中文件： 1. checkpoint :文本文件，包含所有model.ckpt-xxxx,相当于是不同时间节点生成的所有ckpt文件的一个索引。 2. model.ckpt-2124.data-000000-of-000001：模型文件，保存模型的权重 3. model.ckpt-2124.meta：　图文件，保存模型的网络图 4. model.ckpt-2124.index : 这个没搞太清楚 5. graph.pbtxt：　用protobuf格式保存的模型的图