# 一 image classification

1. 训练

|  |
| --- |
| python train\_image\_classifier.py --train\_dir=satellite/train\_dir --dataset\_name=satellite --dataset\_split\_name=train --dataset\_dir=satellite/data --model\_name=inception\_v3 --checkpoint\_path=satellite/pretrained/inception\_v3.ckpt --checkpoint\_exclude\_scopes=InceptionV3/Logits,InceptionV3/AuxLogits --trainable\_scopes=InceptionV3/Logits,InceptionV3/AuxLogits --max\_number\_of\_steps=100000 --batch\_size=32 --learning\_rate=0.001 --learning\_rate\_decay\_type=fixed --save\_interval\_secs=300 --save\_summaries\_secs=2 --log\_every\_n\_steps=10 --optimizer=rmsprop --weight\_decay=0.00004 |

|  |
| --- |
| python eval\_image\_classifier.py --checkpoint\_path=satellite/train\_dir --eval\_dir=satellite/eval\_dir --dataset\_name=satellite --dataset\_split\_name=validation --dataset\_dir=satellite/data --model\_name=inception\_v3 |

# 二 object detection

第一步, 准备数据集.

下载:

|  |
| --- |
| wget http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval\_11-May-2012.tar |

用voc2012.整理成tfrecord模型,以便tf可以识别使用.

|  |
| --- |
| python create\_pascal\_tf\_record.py --data\_dir voc/VOCdevkit/ --year=VOC2012 --set=train --output\_path=voc/pascal\_train.record  python create\_pascal\_tf\_record.py --data\_dir voc/VOCdevkit/ --year=VOC2012 --set=val --output\_path=voc/pascal\_val.record |

拷贝label, 这个是human 可读的label描述标签.

|  |
| --- |
| cp data/pascal\_label\_map.pbtxt voc/ |

在voc中新建 pretrain, 构建pretrain模型.

|  |
| --- |
| wget http://download.tensorflow.org/models/object\_detection/faster\_rcnn\_inception\_resnet\_v2\_atrous\_coco\_11\_06\_2017.tar.gz |

第二步,训练

|  |
| --- |
| **export PYTHONPATH=$PYTHONPATH:`pwd`:`pwd`/slim**  chmod a+x ../research/bin/protoc  ../research/bin/protoc object\_detection/protos/\*.proto --python\_out=.  /home/julyedu\_433249/work/tf\_base/research/bin/protoc object\_detection/protos/\*.proto --python\_out=. |

|  |
| --- |
| python train.py --train\_dir voc/train\_dir/ --pipeline\_config\_path voc/voc.config |

第三步, 保存模型.

|  |
| --- |
| python3 export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path voc/voc.config --trained\_checkpoint\_prefix voc/train\_dir/model.ckpt-48 --output\_directory voc/export |

导出的模型是voc/export/frozen\_inference\_graph.pb 文件。

# 三 tensorflow训练过程中对变量值的打印的方法

目的是在训练过程中打印变量的值,不仅仅是打印tensor的情况.

一般使用print打印某个tensor时候.只能打印该tensor确定的shape信息,而且有些shape的维度和输入的图像有关系,会显示出?号的不确定含义.

如何能够在训练过程中实时打印某些变量的值呢?

第一步, 在根目录下构建一个文件tfprint.py.

|  |
| --- |
| # tf print using var  import tensorflow as tf  **tfp\_similarity\_matrix = tf.placeholder(tf.float32)**  # 需要定义一个tensor. |

第二步,在train函数中.

|  |
| --- |
| --- a/dl\_object\_detection/object\_detection/trainer.py  +++ b/dl\_object\_detection/object\_detection/trainer.py  @@ -31,7 +31,7 @@ from object\_detection.core import standard\_fields as fields  from object\_detection.utils import ops as util\_ops  from object\_detection.utils import variables\_helper  from deployment import model\_deploy  -  **+from object\_detection import tfprint**  slim = tf.contrib.slim  @@ -297,7 +297,7 @@ def train(create\_tensor\_dict\_fn, create\_model\_fn, train\_config, master, task,  slim.learning.train(  - train\_tensor,  **+ [train\_tensor,tfprint.tfp\_similarity\_matrix], # 添加训练时要操作的tensor对象.**  logdir=train\_dir,  master=master,  is\_chief=is\_chief, |

第三步,在待打印的变量的文件中.添加如下.

|  |
| --- |
| from object\_detection import tfprint  …  def \_match(self, similarity\_matrix):    **tfprint.tfp\_similarity\_matrix = tf.Print(similarity\_matrix,["argmax\_matcher's input : similarity\_matrix\n",similarity\_matrix],message="[trainning info]")** |

summarize=

是显示多少个单位的数据.默认是3个.

# 四 实战: 训练到tflite部署

## 4.1 预训练模型

<https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md>

里面有个***COCO-trained models***列表

wget <http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_coco_2018_03_29.tar.gz>

## 4.2 转换成tflite的

|  |
| --- |
| toco --graph\_def\_file=voc/export/frozen\_inference\_graph.pb --output\_file=voc/export/frozen\_inference\_graph.lite --input\_format=TENSORFLOW\_GRAPHDEF --output\_format=TFLITE --input\_shape=1,299,299,3 --input\_array=image\_tensor --output\_array=detection\_boxes,detection\_scores,detection\_classes,num\_detections --inference\_type=FLOAT --input\_data\_type=FLOAT  freeze\_graph --input\_graph=/tmp/mobilenet\_v1\_224.pb --input\_checkpoint=/tmp/checkpoints/mobilenet-10202.ckpt --input\_binary=true  --output\_graph=/tmp/frozen\_mobilenet\_v1\_224.pb  --output\_node\_names=MobileNet/Predictions/Reshape\_1            python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path voc/voc.config --trained\_checkpoint\_prefix voc/train\_dir/model.ckpt-1582 --output\_directory voc/export  python export\_inference\_graph.py --input\_type image\_tensor --pipeline\_config\_path voc/mob.config --trained\_checkpoint\_prefix voc/mob\_train\_dir/model.ckpt-0 --output\_directory voc/mob\_export |

网络的输入是: image\_tensor

输出是:

[\_export\_inference\_graph] output\_node\_names:%s **detection\_boxes,detection\_scores,detection\_classes,num\_detections**

Ssd

|  |
| --- |
| # for model in \  ssd\_mobilenet\_v1\_coco\_11\_06\_2017 \  ssd\_inception\_v2\_coco\_11\_06\_2017 \  rfcn\_resnet101\_coco\_11\_06\_2017 \  faster\_rcnn\_resnet101\_coco\_11\_06\_2017 \  faster\_rcnn\_inception\_resnet\_v2\_atrous\_coco\_11\_06\_2017  do \  curl -OL http://download.tensorflow.org/models/object\_detection/$model.tar.gz  tar -xzf $model.tar.gz $model/frozen\_inference\_graph.pb  cp -a $model /opt/graph\_def/ |

curl -OL <http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v1_coco_11_06_2017.tar.gz>

tar -xzf ssd\_mobilenet\_v1\_coco\_11\_06\_2017.tar.gz

python train.py --train\_dir voc/mob\_train\_dir/ --pipeline\_config\_path voc/mob.config

**objectdetection 本身就有一个ssd转tflite的工具.**

python export\_tflite\_ssd\_graph.py --pipeline\_config\_path=voc/mob.config --trained\_checkpoint\_prefix=voc/mob\_train\_dir/model.ckpt-0 --output\_directory=voc/mob\_export --add\_postprocessing\_op=true

python model\_main.py --model\_dir voc/mob\_train\_dir/ --pipeline\_config\_path voc/mob.config

python object\_detection/model\_main.py --model\_dir object\_detection/voc/train\_dir/ --pipeline\_config\_path object\_detection/voc/voc.config

有一个legecy的 train

python legacy/train.py --train\_dir vvv/trainout/ --pipeline\_config\_path vvv/vv.config

detection\_boxes

toco --graph\_def\_file=voc/export/frozen\_inference\_graph.pb --output\_file=voc/export/frozen\_inference\_graph.lite --input\_format=TENSORFLOW\_GRAPHDEF --output\_format=TFLITE --input\_shape=1,299,299,3 --input\_array=image\_tensor --output\_array= box\_encodings --inference\_type=FLOAT --input\_data\_type=FLOAT

可用的toco

|  |
| --- |
| toco --graph\_def\_file=vvv/export/tflite\_graph.pb--output\_file=vvv/litedir/li.lite --input\_format=TENSORFLOW\_GRAPHDEF --output\_format=TFLITE --input\_shape=1,299,299,3 --input\_array=image\_tensor --output\_array='TFLite\_Detection\_PostProcess','TFLite\_Detection\_PostProcess:1','TFLite\_Detection\_PostProcess:2','TFLite\_Detection\_PostProcess:3' --inference\_type=FLOAT --input\_data\_type=FLOAT --allow\_custom\_ops  ~~toco --graph\_def\_file=vvv/export/tflite\_graph.pb--output\_file=vvv/litedir/lili.lite -- --input\_shape=1,300,300,3 --input\_array= normalized\_input\_image\_tensor --output\_array='TFLite\_Detection\_PostProcess','TFLite\_Detection\_PostProcess:1','TFLite\_Detection\_PostProcess:2','TFLite\_Detection\_PostProcess:3' --inference\_type=FLOAT --allow\_custom\_ops~~  toco --graph\_def\_file=vvv/export/tflite\_graph.pb --output\_file=vvv/litedir/lili.lite --input\_shapes=1,**300,300**,3 --input\_arrays=**normalized\_input\_image\_tensor** --output\_arrays=**'TFLite\_Detection\_PostProcess','TFLite\_Detection\_PostProcess:1','TFLite\_Detection\_PostProcess:2','TFLite\_Detection\_PostProcess:3'** --inference\_type=FLOAT --allow\_custom\_ops  toco --input\_file=$OUTPUT\_DIR/tflite\_graph.pb --output\_file=$OUTPUT\_DIR/detect.tflite \  --input\_shapes=1,300,300,3 \  --input\_arrays=normalized\_input\_image\_tensor \  --output\_arrays='TFLite\_Detection\_PostProcess','TFLite\_Detection\_PostProcess:1','TFLite\_Detection\_PostProcess:2','TFLite\_Detection\_PostProcess:3' \  --inference\_type=FLOAT \  --allow\_custom\_ops |

参考:

<https://jefby.github.io/2018/08/20/%E5%B0%86mobilenet-ssd-tensorflow-pb%E8%BD%AC%E6%8D%A2%E4%B8%BAtflite%E7%9A%84%E8%AF%A6%E7%BB%86%E6%AD%A5%E9%AA%A4/>

关键是output\_array是怎么找到的?

## 4.3 训练ssd v2 模型.

第一, 下载v2 预训练模型.

|  |
| --- |
| wget <http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v2_coco_2018_03_29.tar.gz> |

第二, copy configs,并修改:

|  |
| --- |
| 9c9  < num\_classes: 90  ---  > num\_classes: 20  156c156  < fine\_tune\_checkpoint: "PATH\_TO\_BE\_CONFIGURED/model.ckpt"  ---  > fine\_tune\_checkpoint: "vvv/v2\_pretrain/model.ckpt"  175c175  < input\_path: "PATH\_TO\_BE\_CONFIGURED/mscoco\_train.record-?????-of-00100"  ---  > input\_path: "vvv/pascal\_train.record"  177c177  < label\_map\_path: "PATH\_TO\_BE\_CONFIGURED/mscoco\_label\_map.pbtxt"  ---  > label\_map\_path: "vvv/pascal\_label\_map.pbtxt"  189c189  < input\_path: "PATH\_TO\_BE\_CONFIGURED/mscoco\_val.record-?????-of-00010"  ---  > input\_path: "vvv/pascal\_val.record"  191c191  < label\_map\_path: "PATH\_TO\_BE\_CONFIGURED/mscoco\_label\_map.pbtxt"  ---  > label\_map\_path: "vvv/pascal\_label\_map.pbtxt" |

第三, 训练

|  |
| --- |
| python legacy/train.py --train\_dir vvv/trainout/ --pipeline\_config\_path vvv/vv.config |

第四, 转成tflite可用的pb文件.

|  |
| --- |
| python export\_tflite\_ssd\_graph.py --pipeline\_config\_path=voc/mob.config --trained\_checkpoint\_prefix=voc/mob\_train\_dir/model.ckpt-0 --output\_directory=voc/mob\_export --add\_postprocessing\_op=true |

第五, 把pb文件转成tflite文件

|  |
| --- |
| toco --graph\_def\_file=vvv/v2\_export/tflite\_graph.pb --output\_file=vvv/v2\_tflite/thedemo.tflite --input\_shapes=1,300,300,3 --input\_arrays=normalized\_input\_image\_tensor --output\_arrays='TFLite\_Detection\_PostProcess','TFLite\_Detection\_PostProcess:1','TFLite\_Detection\_PostProcess:2','TFLite\_Detection\_PostProcess:3' --inference\_type=FLOAT --allow\_custom\_ops |

4.5 Android端apk部署.

第一,下载最新tensorflow代码

|  |
| --- |
| Git clone https://github.com/tensorflow/tensorflow.git |

第二,找到如下目录的android工程.用as打开.

tensorflow/tensorflow/examples

# 五 解析ssd的多尺度features maps

## 5.1a 添加flags区分train和analysis

|  |
| --- |
| python legacy/train.py --train\_dir vvv/traindir/ --pipeline\_config\_path vvv/v2mob.config --analysising **true** |

## 5.1 ssd 采用的box predictor为:

|  |
| --- |
| [ConvolutionalBoxPredictor] \_min\_depth: 0  [ConvolutionalBoxPredictor] \_max\_depth: 0  [ConvolutionalBoxPredictor] \_conv\_hyperparams\_fn: <function build.<locals>.scope\_fn at 0x7f7cad310d90>  [ConvolutionalBoxPredictor] \_num\_layers\_before\_predictor: 0  [SSDMetaArch] \_box\_predictor: <object\_detection.predictors.convolutional\_box\_predictor.**ConvolutionalBoxPredictor** object at 0x7f7cad314630>  [ConvolutionalBoxPredictor.\_predict] sorted\_keys ['**box\_encodings**', 'class\_predictions\_with\_background']  [ConvolutionalBoxPredictor.\_predict] head\_name **box\_encodings**  [ConvolutionalBoxPredictor.\_predict] head\_obj <object\_detection.**predictors.heads.box\_head.ConvolutionalBoxHead** object at 0x7f31102d52b0>  [ConvolutionalBoxPredictor.\_predict] head\_name **class\_predictions\_with\_background**  [ConvolutionalBoxPredictor.\_predict] head\_obj <object\_detection.predictors.heads.class\_head.ConvolutionalClassHead object at 0x7f31102d52e8> |

Ssd的预测prediction,区分于分类和回归.



## 5.2 打印关心的变量

在ssd中第一个featuremap之后是一个尺度的卷积输出.想在此处添加一个roi.

那么这个roi的输入需要是**[24 19 19 576],**输出需要是两种,其一是回归**24 1083 1 4**,其二是分类**24 1083 21**,

|  |
| --- |
| if(idx==0):  ## add rfcn roi  tfprint.ssd\_fmap0 = tf.Print(image\_feature,["ssd\_fmap0",tf.shape(image\_feature)],summarize=64)  打印的结果是:  **[ssd\_fmap0][24 19 19 576]**  说明,给ssd的第一张feature map是24, 19x19, 576的.  **[batch\_size, height\_i, width\_i, channels\_i]**  看输出:  **[24 1083 21] 分类**  **[24 1083 1 4] 回归**  回归的是**24 1083 1 4**的  **[batch\_size, num\_anchors\_i, q, code\_size]**  这个是cls分类的输出.shape是 **[batch\_size, num\_anchors\_i, num\_classes + 1]**  对已ingde**[batch\_size, num\_anchors\_i, q, code\_size]**  **num\_predictions\_per\_location \* self.\_box\_code\_size == 1084??**  **self.\_box\_code\_size == 4**  无论分类还是回归,他们都需要num\_predictions\_per\_location\_list(描述对应于feature map的” spatial location, 空间位置”的框预测结果,box predictions,的个数.)  是个数?  还是框的回归坐标?  对于第一个feature map[24 19 19 576]. 需要经过一个roi.  输出同样的:  分类的 [24 1083 21]  回归的 [24 1083 1 4]  现在着手分析roi.  Roi是基于同个feature map. 对于分类和回归,经过不同的conv,输出chn不同.  然后在经ops.batch\_position\_sensitive\_crop\_regions做的roi算法.  输出  这个算法需要boxes.  其中boxes的格式如下:  [num\_boxes, 4] normalized coordinates `[y1, x1, y2, x2]   1. 需要弄明白的:   Rfcn的\_crop\_size的值.   1. 需要弄明白　tf.squeeze　dim和tf.unstack的关系. 2. 可能需要一个slim.conv 再折算一个1083的anchors. 3. 这个可能有些问题.   因为slim.conv的输出chn并不一定就是1083那个维度的.   1. Ssd的   tf.shape(predictions[BOX\_ENCODINGS][0]) 是什么?  [predictions[BOX\_ENCODINGS]][24 **1083** 1 4]  predictions[BOX\_ENCODINGS][1]应该是:[24,**600**,1,4]   1. Rfcn需要的box的size是:   [num\_boxes, 4]. Each box is specified in normalized coordinates [y1, x1, y2, x2] |

## 5.3 打印rfcn的roi的输入输出.

### 第一,先下载pretrained model.

|  |
| --- |
| wget http://download.tensorflow.org/models/object\_detection/rfcn\_resnet101\_coco\_2018\_01\_28.tar.gz |

### 第二,训练

python legacy/train.py --train\_dir vvv/rfcn\_traindir/ --pipeline\_config\_path vvv/rfcn.config --analysising false --logtostderr.

### 第三,分析

打印的结果:

|  |
| --- |
| [RfcnBoxPredictor] conv\_hyperparams\_fn: <function build.<locals>.scope\_fn at 0x7f03982ee378>  [RfcnBoxPredictor] num\_spatial\_bins: [3, 3]  [RfcnBoxPredictor] depth: 1024  [RfcnBoxPredictor] crop\_size: [18, 18]  [RfcnBoxPredictor] box\_code\_size: 4  [RfcnBoxPredictor] conv\_hyperparams\_fn: <function build.<locals>.scope\_fn at 0x7f03984c1598>  [RfcnBoxPredictor] num\_spatial\_bins: [3, 3]  [RfcnBoxPredictor] depth: 1024  [RfcnBoxPredictor] crop\_size: [18, 18]  [RfcnBoxPredictor] box\_code\_size: 4 |

分析

|  |
| --- |
| 输入:  [rfcn roi][1 38 50 189]  [batch\_size, height\_i, width\_i, channels\_i]  分类: [1 38 56 189] 分类的chn小  回归: [1 38 56 720] 回归的chn大些.  输出:  分类: [64 1 21]  是[batch\_size \* num\_boxes, 1, total\_classes]  回归: [64 1 20 4]  是[batch\_size \* num\_boxes, 1, self.num\_classes, self.\_box\_code\_size] |

# 六 在ssd中添加roi处理

## 6.1 需要完成的函数

Rfcn中roi的函数:

|  |
| --- |
| ops.batch\_position\_sensitive\_crop\_regions 将map计算出回归参数.  需要:  location\_feature\_map, 是Map, 在分类和回归时候使用不同的map.  boxes=proposal\_boxes, 是RPN的预测框  crop\_size=self.\_crop\_size,  num\_spatial\_bins=self.\_num\_spatial\_bins,  global\_pool=True |

回归的map:

|  |
| --- |
| net = slim.conv2d(net, self.\_depth, [1, 1], scope='reduce\_depth')  # Location predictions.  location\_feature\_map\_depth = (self.\_num\_spatial\_bins[0] \*  self.\_num\_spatial\_bins[1] \*  **self.num\_classes** \*  self.\_box\_code\_size)  location\_feature\_map = slim.conv2d(net, location\_feature\_map\_depth,  [1, 1], activation\_fn=None,  scope='refined\_locations') |

分类的map,和回归使用同一个net,只不过输出的chn要少很多:

|  |
| --- |
| class\_feature\_map\_depth = (self.\_num\_spatial\_bins[0] \*  self.\_num\_spatial\_bins[1] \*  total\_classes)  class\_feature\_map = slim.conv2d(net, class\_feature\_map\_depth, [1, 1],  activation\_fn=None,  scope='class\_predictions') |