登录 | 注册

luuuyi的博客

计算机视觉小硕

目录视图 摘要视图 RSS 订阅

个人资料

luuuyi

访问： 45443次

积分： 699

等级： BLOG 3

排名： 千里之外

原创： 22篇 转载： 0篇

译文： 0篇 评论： 12条

文章搜索

Ubuntu上用caffe的SSD方法训练umdfaces数据集

标签： caffe 深度学习 ssd ubuntu

2017-03-13 22:42 1866人阅读 评论

分类： caffe（4） SSD（2）

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?) [+]

关闭

实验目的

继前一段时间用SSD训练过VOC数据集以后，这一次使用SSD+K80服务器来训练自己的人脸识别应用，选择的数据集还是之前下载的umdfaces，总共36w张人脸图像。

实验环境

训练平台：NVIDIA K80

预测平台：NVIDIA TX1

语言　　：C++，Python

框架　　：caffe

方法　　：SSD

数据集　：umdfaces

实验准备

文章分类

caffe	(5)
Zbar	(1)
感悟	(1)
Python	(2)
SSD	(3)
C++	(9)
机器学习	(3)
练习	(9)

文章存档

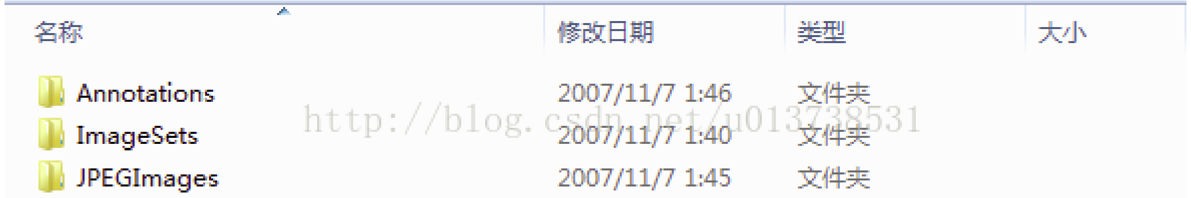
2017年07月	(2)
2017年06月	(4)
2017年05月	(3)
2017年04月	(1)
2017年03月	(4)

展开

阅读排行

Ubuntu上配置caffe+SSD	(7371)
Zbar算法流程介绍	(5545)
使用caffe框架利用faster-	(4530)
Ubuntu上用caffe的SSD方	(3704)
Windows7下Python安装:	(3229)
用C++实现定积分运算	(2733)
Ubuntu上用caffe的SSD方	(1864)

其实训练的步骤在上一篇博客中写得已经很清楚了，这一次主要关注的是数据集的处理，最方便的方法就是将自己的数据集也做成VOC数据集格式，也即是三个文件夹路径的结构：



名称	修改日期	类型	大小
Annotations	2007/11/7 1:46	文件夹	
ImageSets	2007/11/7 1:40	文件夹	
JPEGImages	2007/11/7 1:45	文件夹	

关于数据集的处理，在这边博文中写得很清楚：

使用faster rcnn训练umdfaces数据集

数据集处理好之后，和上一篇博客不同的主要是训练脚本（Python）的写法，这里贴出我的文件，在后面有

地方都是需要自己去配置的地方：

[python]

```python
#!/usr/bin/python

from __future__ import print_function
import caffe
from caffe.model_libs import *
from google.protobuf import text_format

import math
import os
import shutil
import stat
import subprocess
import sys

# Add extra layers on top of a "base" network (e.g. VGGNet or Inception).
def AddExtraLayers(net, use_batchnorm=True, lr_mult=1):
    use_relu = True

    # Add additional convolutional layers.
    # 19 x 19
    from_layer = net.keys()[-1]

```

关闭

使用faster rcnn训练umdf	(1783)
LeetCode周练Contest-3	(1702)
机器学习算法逻辑回归（	(1399)

评论排行

Zbar算法流程介绍	(5)
Ubuntu上配置caffe+SSD	(3)
Ubuntu上用caffe的SSD方	(2)
Windows7下Python安装	(1)
使用faster rcnn训练umdf	(1)
用C++实现定积分运算	(0)
Ubuntu上用caffe的SSD方	(0)
使用Python分析移动语	(0)
2016年总结	(0)
LeetCode周练Contest-3	(0)

推荐文章

- * CSDN新版博客feed流内测用户征集令
- * Android检查更新下载安装
- * 动手打造史上最简单的 Recycleview 侧滑菜单
- * TCP网络通讯如何解决分包粘包问题
- * SDCC 2017之大数据技术实战线上峰会
- * 快速集成一个视频直播功能

最新评论

```
    # TODO(weiliu89): Construct the name using the last layer to avoid duplication.
    # 10 x 10
    out_layer = "conv6_1"
    ConvBNLayer(net, from_layer, out_layer, use_batchnorm, use_relu, 256, 1, 0, 1,
      lr_mult=lr_mult)

    from_layer = out_layer
    out_layer = "conv6_2"
    ConvBNLayer(net, from_layer, out_layer, use_batchnorm, use_relu, 512, 3, 1, 2,
      lr_mult=lr_mult)

    # 5 x 5
    from_layer = out_layer
    out_layer = "conv7_1"
    ConvBNLayer(net, from_layer, out_layer, use_batchnorm, use_relu, 128, 1, 0, 1,
     lr_mult=lr_mult)

    from_layer = out_layer
    out_layer = "conv7_2"
    ConvBNLayer(net, from_layer, out_layer, use_batchnorm, use_relu, 256, 3, 1, 2,
     lr_mult=lr_mult)

    # 3 x 3
    from_layer = out_layer
    out_layer = "conv8_1"
    ConvBNLayer(net, from_layer, out_layer, use_batchnorm, us _    ,    ,  ,  ,  ,
     lr_mult=lr_mult)

    from_layer = out_layer
    out_layer = "conv8_2"
    ConvBNLayer(net, from_layer, out_layer, use_batchnorm, use_relu, 256, 3, 0, 1,
     lr_mult=lr_mult)

    # 1 x 1
    from_layer = out_layer
    out_layer = "conv9_1"
    ConvBNLayer(net, from_layer, out_layer, use_batchnorm, use_relu, 128, 1, 0, 1,
     lr_mult=lr_mult)

```

关闭

Zbar算法流程介绍
XXDK141: 大神有源码吗，我的是zbarimg.c 是基于imageMagick版本的，opencv版本的哪里可...

Windows7下Python安装最新xgb
echorma: 楼主太棒了！奇怪的是，楼主分享的东西，只有预编译的xgboost能下载成功，其他都下不了...

Ubuntu上用caffe的SSD方法训练
yx2017: @erfenxing:你好，我也是在用ssd训练widerface，可否交流下？1343102499

Ubuntu上用caffe的SSD方法训练
erfenxing: 您好，我用SSD训练widerface数据集，但是loss值从20左右下降到大约5的时候就不再下降了...

Ubuntu上配置caffe+SSD及demc
luuuyi: @Reezea:435977170

Ubuntu上配置caffe+SSD及demc
Reezea: 能否加个好友请教一下，最近一直没有安装成功caffe，总是各种问题，希望能够指点一二，万分感激了。系...

Zbar算法流程介绍
windwolf2004: 大好人，绝对原创

Zbar算法流程介绍
ATUXLI: 楼主想请教一下，在提取明暗宽度流之前会有density，其定义是：int density = CFG...

Zbar算法流程介绍
luuuyi: @qq_34344335:Linux上的，不过clone下来自己改一改，可以移植到Windows上

Zbar算法流程介绍
qq_34344335: 请问这个是什么环境下的源码

```
    from_layer = out_layer
    out_layer = "conv9_2"
    ConvBNLayer(net, from_layer, out_layer, use_batchnorm, use_relu, 256, 3, 0, 1,
      lr_mult=lr_mult)

    return net


### Modify the following parameters accordingly ###
# The directory which contains the caffe code.
# We assume you are running the script at the CAFFE_ROOT.
caffe_root = os.getcwd()

# Set true if you want to start training right after generating all files.
run_soon = True
# Set true if you want to load from most recently saved snapshot.
# Otherwise, we will load from the pretrain_model defined below.
resume_training = True
# If true, Remove old model files.
remove_old_models = False

# The database file for training data. Created by data/VOC0712/create_data.sh
train_data = "/media/scs4450/hard/VOCdevkit/FACE_LUYI/lmdb/FACE_LUYI_trainval_lmdb"
# The database file for testing data. Created by data/VOC0712/create_data.sh
test_data = "/media/scs4450/hard/VOCdevkit/FACE_LUYI/lmdb/FACE_LUYI_test_lmdb"
# Specify the batch sampler.
resize_width = 300
resize_height = 300
resize = "{}x{}".format(resize_width, resize_height)
batch_sampler = [
        {
                'sampler': {
                        },
                'max_trials': 1,
                'max_sample': 1,
        },
        {
                'sampler': {
                        'min_scale': 0.3,
```

关闭

```
                        'max_scale': 1.0,
                        'min_aspect_ratio': 0.5,
                        'max_aspect_ratio': 2.0,
                        },
                'sample_constraint': {
                        'min_jaccard_overlap': 0.1,
                        },
                'max_trials': 50,
                'max_sample': 1,
        },
        {
                'sampler': {
                        'min_scale': 0.3,
                        'max_scale': 1.0,
                        'min_aspect_ratio': 0.5,
                        'max_aspect_ratio': 2.0,
                        },
                'sample_constraint': {
                        'min_jaccard_overlap': 0.3,
                        },
                'max_trials': 50,
                'max_sample': 1,
        },
        {
                'sampler': {
                        'min_scale': 0.3,
                        'max_scale': 1.0,
                        'min_aspect_ratio': 0.5,
                        'max_aspect_ratio': 2.0,
                        },
                'sample_constraint': {
                        'min_jaccard_overlap': 0.5,
                        },
                'max_trials': 50,
                'max_sample': 1,
        },
        {
                'sampler': {
                        'min_scale': 0.3,
```

关闭

```
                        'max_scale': 1.0,
                        'min_aspect_ratio': 0.5,
                        'max_aspect_ratio': 2.0,
                        },
                'sample_constraint': {
                        'min_jaccard_overlap': 0.7,
                        },
                'max_trials': 50,
                'max_sample': 1,
        },
        {
                'sampler': {
                        'min_scale': 0.3,
                        'max_scale': 1.0,
                        'min_aspect_ratio': 0.5,
                        'max_aspect_ratio': 2.0,
                        },
                'sample_constraint': {
                        'min_jaccard_overlap': 0.9,
                        },
                'max_trials': 50,
                'max_sample': 1,
        },
        {
                'sampler': {
                        'min_scale': 0.3,
                        'max_scale': 1.0,
                        'min_aspect_ratio': 0.5,
                        'max_aspect_ratio': 2.0,
                        },
                'sample_constraint': {
                        'max_jaccard_overlap': 1.0,
                        },
                'max_trials': 50,
                'max_sample': 1,
        },
        ]
train_transform_param = {
        'mirror': True,
        'mean_value': [104, 117, 123],
        'resize_param': {
                'prob': 1,
                'resize_mode': P.Resize.WARP,
                'height': resize_height,
                'width': resize_width,
                'interp_mode': [
                        P.Resize.LINEAR,
                        P.Resize.AREA,
                        P.Resize.NEAREST,
                        P.Resize.CUBIC,
                        P.Resize.LANCZOS4,
                        ],
                },
        'distort_param': {
                'brightness_prob': 0.5,
                'brightness_delta': 32,
                'contrast_prob': 0.5,
                'contrast_lower': 0.5,
                'contrast_upper': 1.5,
                'hue_prob': 0.5,
                'hue_delta': 18,
                'saturation_prob': 0.5,
                'saturation_lower': 0.5,
                'saturation_upper': 1.5,
                'random_order_prob': 0.0,
                },
        'expand_param': {
                'prob': 0.5,
                'max_expand_ratio': 4.0,
                },
        'emit_constraint': {
            'emit_type': caffe_pb2.EmitConstraint.CENTER,
            }
        }
test_transform_param = {
        'mean_value': [104, 117, 123],
        'resize_param': {
                'prob': 1,
```

关闭

```
                'resize_mode': P.Resize.WARP,
                'height': resize_height,
                'width': resize_width,
                'interp_mode': [P.Resize.LINEAR],
                },
        }

# If true, use batch norm for all newly added layers.
# Currently only the non batch norm version has been tested.
use_batchnorm = False
lr_mult = 1
# Use different initial learning rate.
if use_batchnorm:
    base_lr = 0.0004
else:
    # A learning rate for batch_size = 1, num_gpus = 1.
    base_lr = 0.00004

# Modify the job name if you want.
job_name = "SSD_{}".format(resize)
# The name of the model. Modify it if you want.
model_name = "VGG_FACE_LUYI_{}".format(job_name)

# Directory which stores the model .prototxt file.
save_dir = "/home/ly/ssd/caffe/models/VGGNet/FACE_LUYI/{}".format(job_name)    #luyi
# Directory which stores the snapshot of models.
snapshot_dir = "/home/ly/ssd/caffe/models/VGGNet/FACE_LUYI/{}".format(job_name) #luyi
# Directory which stores the job script and log file.
job_dir = "/home/ly/ssd/caffe/jobs/VGGNet/FACE_LUYI/{}".format(job_name)    #luyi
# Directory which stores the detection results.
output_result_dir = "{}/data/VOCdevkit/results/FACE_LUYI/{}/Main".format(os.environ['CAFFE_ROOT'], job_name)  #luyi

# model definition files.
train_net_file = "{}/train.prototxt".format(save_dir)
test_net_file = "{}/test.prototxt".format(save_dir)
deploy_net_file = "{}/deploy.prototxt".format(save_dir)
solver_file = "{}/solver.prototxt".format(save_dir)
# snapshot prefix.
```

```
snapshot_prefix = "{}/{}".format(snapshot_dir, model_name)
# job script path.
job_file = "{}/{}.sh".format(job_dir, model_name)

# Stores the test image names and sizes. Created by data/VOC0712/create_list.sh
name_size_file = "/home/ly/ssd/caffe/data/FACE_LUYI/test_name_size.txt"
# The pretrained model. We use the Fully convolutional reduced (atrous) VGGNet.
pretrain_model = "/home/ly/ssd/caffe/models/VGGNet/VGG_ILSVRC_16_layers_fc_reduced.caffemodel"
# Stores LabelMapItem.
label_map_file = "/home/ly/ssd/caffe/data/FACE_LUYI/labelmap_voc.prototxt"

# MultiBoxLoss parameters.
num_classes = 2                          #luyi
share_location = True
background_label_id=0
train_on_diff_gt = True
normalization_mode = P.Loss.VALID
code_type = P.PriorBox.CENTER_SIZE
ignore_cross_boundary_bbox = False
mining_type = P.MultiBoxLoss.MAX_NEGATIVE
neg_pos_ratio = 3.
loc_weight = (neg_pos_ratio + 1.) / 4.
multibox_loss_param = {
    'loc_loss_type': P.MultiBoxLoss.SMOOTH_L1,
    'conf_loss_type': P.MultiBoxLoss.SOFTMAX,
    'loc_weight': loc_weight,
    'num_classes': num_classes,
    'share_location': share_location,
    'match_type': P.MultiBoxLoss.PER_PREDICTION,
    'overlap_threshold': 0.5,
    'use_prior_for_matching': True,
    'background_label_id': background_label_id,
    'use_difficult_gt': train_on_diff_gt,
    'mining_type': mining_type,
    'neg_pos_ratio': neg_pos_ratio,
    'neg_overlap': 0.5,
    'code_type': code_type,
    'ignore_cross_boundary_bbox': ignore_cross_boundary_bbox,
    }
```

```
loss_param = {
    'normalization': normalization_mode,
    }

# parameters for generating priors.
# minimum dimension of input image
min_dim = 300
# conv4_3 ==> 38 x 38
# fc7 ==> 19 x 19
# conv6_2 ==> 10 x 10
# conv7_2 ==> 5 x 5
# conv8_2 ==> 3 x 3
# conv9_2 ==> 1 x 1
mbox_source_layers = ['conv4_3', 'fc7', 'conv6_2', 'conv7_2', 'conv8_2', 'conv9_2']
# in percent %
min_ratio = 20
max_ratio = 90
step = int(math.floor((max_ratio - min_ratio) / (len(mbox_source_layers) - 2)))
min_sizes = []
max_sizes = []
for ratio in xrange(min_ratio, max_ratio + 1, step):
  min_sizes.append(min_dim * ratio / 100.)
  max_sizes.append(min_dim * (ratio + step) / 100.)
min_sizes = [min_dim * 10 / 100.] + min_sizes
max_sizes = [min_dim * 20 / 100.] + max_sizes
steps = [8, 16, 32, 64, 100, 300]
aspect_ratios = [[2], [2, 3], [2, 3], [2, 3], [2], [2]]
# L2 normalize conv4_3.
normalizations = [20, -1, -1, -1, -1, -1]
# variance used to encode/decode prior bboxes.
if code_type == P.PriorBox.CENTER_SIZE:
  prior_variance = [0.1, 0.1, 0.2, 0.2]
else:
  prior_variance = [0.1]
flip = True
clip = False

# Solver parameters.
# Defining which GPUs to use.
```

关闭

```
gpus = "1"                                                      #luyi
gpulist = gpus.split(",")
num_gpus = len(gpulist)

# Divide the mini-batch to different GPUs.
batch_size = 32
accum_batch_size = 32
iter_size = accum_batch_size / batch_size
solver_mode = P.Solver.CPU
device_id = 0
batch_size_per_device = batch_size
if num_gpus > 0:
  batch_size_per_device = int(math.ceil(float(batch_size) / num_gpus))
  iter_size = int(math.ceil(float(accum_batch_size) / (batch_size_per_device * num_
  solver_mode = P.Solver.GPU
  device_id = int(gpulist[0])

if normalization_mode == P.Loss.NONE:
  base_lr /= batch_size_per_device
elif normalization_mode == P.Loss.VALID:
  base_lr *= 25. / loc_weight
elif normalization_mode == P.Loss.FULL:
  # Roughly there are 2000 prior bboxes per image.
  # TODO(weiliu89): Estimate the exact # of priors.
  base_lr *= 2000.

# Evaluate on whole test set.
num_test_image = 4952                       #luyi
test_batch_size = 8
# Ideally test_batch_size should be divisible by num_test_image,
# otherwise mAP will be slightly off the true value.
test_iter = int(math.ceil(float(num_test_image) / test_batch_size))

solver_param = {
    # Train parameters
    'base_lr': base_lr,
    'weight_decay': 0.0005,
    'lr_policy': "multistep",
    'stepvalue': [80000, 100000, 120000],
```

关闭

```
    'gamma': 0.1,
    'momentum': 0.9,
    'iter_size': iter_size,
    'max_iter': 120000,
    'snapshot': 80000,
    'display': 10,
    'average_loss': 10,
    'type': "SGD",
    'solver_mode': solver_mode,
    'device_id': device_id,
    'debug_info': False,
    'snapshot_after_train': True,
    # Test parameters
    'test_iter': [test_iter],
    'test_interval': 10000,
    'eval_type': "detection",
    'ap_version': "11point",
    'test_initialization': False,
    }

# parameters for generating detection output.
det_out_param = {
    'num_classes': num_classes,
    'share_location': share_location,
    'background_label_id': background_label_id,
    'nms_param': {'nms_threshold': 0.45, 'top_k': 400},
    'save_output_param': {
        'output_directory': output_result_dir,
        'output_name_prefix': "comp4_det_test_",
        'output_format': "VOC",
        'label_map_file': label_map_file,
        'name_size_file': name_size_file,
        'num_test_image': num_test_image,
        },
    'keep_top_k': 200,
    'confidence_threshold': 0.01,
    'code_type': code_type,
    }

```

关闭

```
# parameters for evaluating detection results.
det_eval_param = {
    'num_classes': num_classes,
    'background_label_id': background_label_id,
    'overlap_threshold': 0.5,
    'evaluate_difficult_gt': False,
    'name_size_file': name_size_file,
    }

### Hopefully you don't need to change the following ###
# Check file.
check_if_exist(train_data)
check_if_exist(test_data)
check_if_exist(label_map_file)
check_if_exist(pretrain_model)
make_if_not_exist(save_dir)
make_if_not_exist(job_dir)
make_if_not_exist(snapshot_dir)

# Create train net.
net = caffe.NetSpec()
net.data, net.label = CreateAnnotatedDataLayer(train_data, batch_size=batch_size_per
        train=True, output_label=True, label_map_file=label_map_file,
        transform_param=train_transform_param, batch_sampler=batch_sampler)

VGGNetBody(net, from_layer='data', fully_conv=True, reduced=T    ,              ,
    dropout=False)

AddExtraLayers(net, use_batchnorm, lr_mult=lr_mult)

mbox_layers = CreateMultiBoxHead(net, data_layer='data', from_layers=mbox_source_layers,
        use_batchnorm=use_batchnorm, min_sizes=min_sizes, max_sizes=max_sizes,
        aspect_ratios=aspect_ratios, steps=steps, normalizations=normalizations,
        num_classes=num_classes, share_location=share_location, flip=flip, clip=clip,
        prior_variance=prior_variance, kernel_size=3, pad=1, lr_mult=lr_mult)

# Create the MultiBoxLossLayer.
name = "mbox_loss"
mbox_layers.append(net.label)
```

关闭

```
net[name] = L.MultiBoxLoss(*mbox_layers, multibox_loss_param=multibox_loss_param,
        loss_param=loss_param, include=dict(phase=caffe_pb2.Phase.Value('TRAIN')),
        propagate_down=[True, True, False, False])

with open(train_net_file, 'w') as f:
    print('name: "{}_train"'.format(model_name), file=f)
    print(net.to_proto(), file=f)
shutil.copy(train_net_file, job_dir)

# Create test net.
net = caffe.NetSpec()
net.data, net.label = CreateAnnotatedDataLayer(test_data, batch_size=test_batch_size,
        train=False, output_label=True, label_map_file=label_map_file,
        transform_param=test_transform_param)

VGGNetBody(net, from_layer='data', fully_conv=True, reduced=True, dilated=True,
    dropout=False)

AddExtraLayers(net, use_batchnorm, lr_mult=lr_mult)

mbox_layers = CreateMultiBoxHead(net, data_layer='data', from_layers=mbox_source_layers,
        use_batchnorm=use_batchnorm, min_sizes=min_sizes, max_sizes=max_sizes,
        aspect_ratios=aspect_ratios, steps=steps, normalizations=normalizations,
        num_classes=num_classes, share_location=share_location, flip=flip, clip=clip
        prior_variance=prior_variance, kernel_size=3, pad=1, lr_mult=lr_mult)

conf_name = "mbox_conf"
if multibox_loss_param["conf_loss_type"] == P.MultiBoxLoss.SOFTMAX:
  reshape_name = "{}_reshape".format(conf_name)
  net[reshape_name] = L.Reshape(net[conf_name], shape=dict(dim=[0, -1, num_classes]))
  softmax_name = "{}_softmax".format(conf_name)
  net[softmax_name] = L.Softmax(net[reshape_name], axis=2)
  flatten_name = "{}_flatten".format(conf_name)
  net[flatten_name] = L.Flatten(net[softmax_name], axis=1)
  mbox_layers[1] = net[flatten_name]
elif multibox_loss_param["conf_loss_type"] == P.MultiBoxLoss.LOGISTIC:
  sigmoid_name = "{}_sigmoid".format(conf_name)
  net[sigmoid_name] = L.Sigmoid(net[conf_name])
  mbox_layers[1] = net[sigmoid_name]
```

关闭

```

net.detection_out = L.DetectionOutput(*mbox_layers,
    detection_output_param=det_out_param,
    include=dict(phase=caffe_pb2.Phase.Value('TEST')))
net.detection_eval = L.DetectionEvaluate(net.detection_out, net.label,
    detection_evaluate_param=det_eval_param,
    include=dict(phase=caffe_pb2.Phase.Value('TEST')))

with open(test_net_file, 'w') as f:
    print('name: "{}_test"'.format(model_name), file=f)
    print(net.to_proto(), file=f)
shutil.copy(test_net_file, job_dir)

# Create deploy net.
# Remove the first and last layer from test net.
deploy_net = net
with open(deploy_net_file, 'w') as f:
    net_param = deploy_net.to_proto()
    # Remove the first (AnnotatedData) and last (DetectionEvaluate) layer from test net.
    del net_param.layer[0]
    del net_param.layer[-1]
    net_param.name = '{}_deploy'.format(model_name)
    net_param.input.extend(['data'])
    net_param.input_shape.extend([
        caffe_pb2.BlobShape(dim=[1, 3, resize_height, resize_width])])
    print(net_param, file=f)
shutil.copy(deploy_net_file, job_dir)

# Create solver.
solver = caffe_pb2.SolverParameter(
        train_net=train_net_file,
        test_net=[test_net_file],
        snapshot_prefix=snapshot_prefix,
        **solver_param)

with open(solver_file, 'w') as f:
    print(solver, file=f)
shutil.copy(solver_file, job_dir)

```

关闭

```
max_iter = 0
# Find most recent snapshot.
for file in os.listdir(snapshot_dir):
  if file.endswith(".solverstate"):
    basename = os.path.splitext(file)[0]
    iter = int(basename.split("{}_iter_".format(model_name))[1])
    if iter > max_iter:
      max_iter = iter

train_src_param = '--weights="{}" \\\n'.format(pretrain_model)
if resume_training:
  if max_iter > 0:
    train_src_param = '--snapshot="{}_iter_{}.solverstate" \\\n'.format(snapshot_prefix, max_iter)

if remove_old_models:
  # Remove any snapshots smaller than max_iter.
  for file in os.listdir(snapshot_dir):
    if file.endswith(".solverstate"):
      basename = os.path.splitext(file)[0]
      iter = int(basename.split("{}_iter_".format(model_name))[1])
      if max_iter > iter:
        os.remove("{}/{}".format(snapshot_dir, file))
    if file.endswith(".caffemodel"):
      basename = os.path.splitext(file)[0]
      iter = int(basename.split("{}_iter_".format(model_name))[1])
      if max_iter > iter:
        os.remove("{}/{}".format(snapshot_dir, file))

# Create job file.
with open(job_file, 'w') as f:
  f.write('cd {}\n'.format(caffe_root))
  f.write('/home/ly/ssd/caffe/build/tools/caffe train \\\n')           #luyi
  f.write('--solver="{}" \\\n'.format(solver_file))
  f.write(train_src_param)
  if solver_param['solver_mode'] == P.Solver.GPU:
    f.write('--gpu {} 2>&1 | tee {}/{}.log\n'.format(gpus, job_dir, model_name))
  else:
    f.write('2>&1 | tee {}/{}.log\n'.format(job_dir, model_name))
```

关闭

```

# Copy the python script to job_dir.
py_file = os.path.abspath(__file__)
shutil.copy(py_file, job_dir)

# Run the job.
os.chmod(job_file, stat.S_IRWXU)
if run_soon:
  subprocess.call(job_file, shell=True)
```

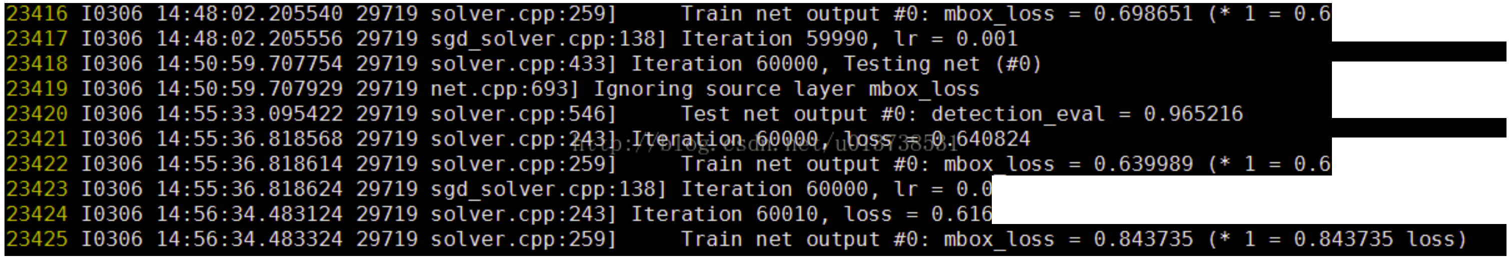
实验结果

计划训练的迭代次数是12w次，但是在K80上只开了一个核来进行计算，差不多一天可以迭代1w+次吧，跑了

次，打断来测试，在K80上，检测单张人脸图片，分辨率在300X300左右，速度为40ms左右，也就是说帧率

25fps，速度还是很快的。至于准确度，在log文件里面，每1w次迭代之后会计算一个mAP，第6w次的时候i

mAP（作者自定义过，跟mAP比较像）为0.965：



```
I0306 14:48:02.205540 29719 solver.cpp:259]     Train net output #0: mbox_loss = 0.698651 (* 1 = 0.6
I0306 14:48:02.205556 29719 sgd_solver.cpp:138] Iteration 59990, lr = 0.001
I0306 14:50:59.707754 29719 solver.cpp:433] Iteration 60000, Testing net (#0)
I0306 14:50:59.707929 29719 net.cpp:693] Ignoring source layer mbox_loss
I0306 14:55:33.095422 29719 solver.cpp:546]     Test net output #0: detection_eval = 0.965216
I0306 14:55:36.818568 29719 solver.cpp:243] Iteration 60000, loss = 0.640824
I0306 14:55:36.818614 29719 solver.cpp:259]     Train net output #0: mbox_loss = 0.639989 (* 1 = 0.6
I0306 14:55:36.818624 29719 sgd_solver.cpp:138] Iteration 60000, lr = 0.0
I0306 14:56:34.483124 29719 solver.cpp:243] Iteration 60010, loss = 0.616
I0306 14:56:34.483324 29719 solver.cpp:259]     Train net output #0: mbox_loss = 0.843735 (* 1 = 0.843735 loss)
```

关闭

总体来说，效果不错，不过目标是在嵌入式平台上做到实时，还需要再继续努力。

欢迎各位来交流：435977170（Q&Q）

顶
0

踩
0

上一篇 用C++实现定积分运算

下一篇 机器学习KNN方法介绍及实现并实践（约会满意度统计）

相关文章推荐

- Ubuntu上配置caffe+SSD及demo演示（附带问题...
- Presto的服务治理与架构在京东的实践与应用--王...
- 【用Python学习Caffe】2. 使用Caffe完成图像目标...
- 深入掌握Kubernetes应用实践--王渊命
- Ubuntu上用caffe的SSD方法训练Pascal VOC数据集
- Python基础知识汇总
- RCNN系列实验的PASCAL VOC数据集格式设置
- Android核心技术详解
- SSD框架训练自己的数据集
- Retrofit 从入门封装到源码解析
- 论文笔记 SSD: Single Shot MultiBox Detector
- 自然语言处理工具Word2Vec
- SSD框架训练自己的数据集
- SSD源码解读1~~~~~~~~~~ssd_pascal.py
- SSD 安装、训练、测试（ubuntu14.04+cuda7.5+
- 使用faster rcnn训练umdfaces数据集

查看评论

关闭

1楼 erfenxing 2017-06-05 17:13发表

您好，我用SSD训练widerface数据集，但是loss值从20左右下降到大约5的时候就不再下降了，请问会是数据集本身的问题吗？请问用create_list.sh 和create_data.sh生成lmdb的时候，有对文件做修改吗？谢谢！

Re: yx2017 2017-07-12 11:20发表

回复erfenxing：你好，我也是在用ssd训练widerface，可否交流下？1343102499

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场