MTCNN

--VX:shanqianyu1

目录

[1执行步骤 4](#_Toc6735812)

[1.0 数据集准备 4](#_Toc6735813)

[1.1 txt文件以及路径准备 4](#_Toc6735814)

[1.2训练步骤 4](#_Toc6735815)

[1.3 验证步骤 4](#_Toc6735816)

[one\_image\_test.py 5](#_Toc6735817)

[camera\_test.py 5](#_Toc6735818)

[2脚本文件 5](#_Toc6735819)

[3生成文件架构及命名 6](#_Toc6735820)

[4执行文件的思路 6](#_Toc6735821)

[4.1 PNP\_Pic\_E.py思路 6](#_Toc6735822)

[4.2 PNP\_Pic\_H.py思路 6](#_Toc6735823)

[4.3 Mark\_pic.py思路 8](#_Toc6735824)

[4.4 pnpm\_gather\_txt. py思路 8](#_Toc6735825)

[4.5 tfrecord.py思路 8](#_Toc6735826)

[4.6 train\_net.py思路 8](#_Toc6735827)

[总体步骤 8](#_Toc6735828)

[cls\_ohem 9](#_Toc6735829)

[box\_ohem 9](#_Toc6735830)

[mark\_ohem 9](#_Toc6735831)

[5文件中的重点API 9](#_Toc6735832)

[5.1 train\_net.py API 9](#_Toc6735833)

[tensorboard 部分 9](#_Toc6735834)

[model\_saver 9](#_Toc6735835)

[tfrecord读入数据和线程部分 10](#_Toc6735836)

[5.1.1 train\_lr\_op 10](#_Toc6735837)

[6 通过tensorboard查看图 10](#_Toc6735838)

[7梯度和动量法 10](#_Toc6735839)

[梯度 10](#_Toc6735840)

[动量法 11](#_Toc6735841)

[8训练结果 11](#_Toc6735842)

[Rnet 11](#_Toc6735843)

[Onet 12](#_Toc6735844)

[图片检测结果 14](#_Toc6735845)

[9缺点 16](#_Toc6735846)

1执行步骤

1. 数据集准备

wider\_face\_train:这个用于pos, neg, part三种照片的准备

lfw5590:这个用于landmark照片的准备

存放位置可以任意，只要和下面的1.1的路径对应起来即可

1.1 txt文件以及路径准备

将trainImageList.txt, wider\_face\_train\_bbox\_gt.txt, wider\_face\_train.txt这三个文件放进txt文件夹，用于训练数据的准备

python\_and\_config.py这个文件中定义两个edict，也就是字典。用这种方式存储路径和训练的时候用到的一些常量。

为了防止误解，特殊说明一下，每个路径的位置具体到哪一级目录。

path.WIDER\_FACE = '/media/q/deep/wider\_face/WIDER\_train/images'

path.LFW5590 = '/media/q/deep/mtcn\_landmark\_dataset\_github/train'

path.RESULTS\_ROOT = '/media/q/deep/mtcnn\_brief'

path.WIDER\_VAL='/media/q/deep/wider\_face/WIDER\_val/images'

下面的是重命名，这样使用起来简洁一些

path.pnp = path.WIDER\_FACE

path.mark = path.LFW5590

path.root = path.RESULTS\_ROOT

1.2训练步骤

PNP\_Pic\_E.py 获取图像大小为12 的pos,neg,part 图片文件夹以及txt 不变

Mark\_Pic.py 获取图像大小为12的mark图片文件夹以及txt 设置pic\_size = 12

pnpm\_gather.py 将图像大小为12的txt四个文件组合在一起 设置pic\_size = 12

tfrecord.py 将组合在一起的txt，生成一个tfrecord文件 设置pic\_size = 12

train\_net.py 训练P\_Net 设置pic\_size = 12

——————————————————————————————————————————

PNP\_Pic\_H.py 获取图像大小为24 的pos,neg,part 图片文件夹以及txt 设置pic\_size = 24

采用hard sample

Mark\_Pic.py 获取图像大小为24的mark图片文件夹以及txt 设置pic\_size = 24

pnpm\_gather.py 将图像大小为24的txt四个文件组合在一起 设置pic\_size = 24

tfrecord.py 将组合在一起的txt，生成一个tfrecord文件 设置pic\_size = 24

train\_net.py 训练R\_Net 设置pic\_size = 24

——————————————————————————————————————————

PNP\_Pic\_H.py 获取图像大小为48的pos,neg,part 图片文件夹以及txt 设置pic\_size = 48

采用hard sample

Mark\_Pic.py 获取图像大小为48mark图片文件夹以及txt 设置pic\_size = 48

pnpm\_gather.py 将图像大小为48的txt四个文件组合在一起 设置pic\_size = 48

tfrecord.py 将组合在一起的txt，生成一个tfrecord文件 设置pic\_size = 48

train\_net.py 训练O\_Net 设置pic\_size = 48

1.3 验证步骤

one\_image\_test.py

将几张照片放进一个文件夹，并且更改one\_image\_test.py中路径的位置就可以单张照片的人脸检测了。键盘上按任意键就可以显示下一张照片，显示完毕就退出程序。

camera\_test.py

将一个视频放进一个文件夹，并更改camera\_test.py中路径的位置，即可对视频进行人脸检测。如果手机上的视频有可能是反转90°的所以frame = rotate(fram,90)根据实际请款决定去留。

pic\_size = 24 表示执行Pnet，这个时候mark是none要去掉，

pic\_size = 48表示执行Pnet和RNet，这个时候mark是none要去掉，

pic\_size = 0 表示执行Pnet和Rnet，Onet

2脚本文件

readme.txt 简要描述

path\_and\_config.py 路径设置，训练的配置

PNP\_Pic\_E.py 获得pos,neg,part图片以及txt

PNP\_Pic\_H.py 获得pos,neg,part图片以及txt文件的hardsample方法

Mark\_Pic.py 获取mark图片以及txt

pnpm\_gather.py 将pos.txt,neg.txt,part.txt,mark.txt整合成一个txt文件

tfrecord.py 将整合的txt文件，转换成tfrecord文件

train\_net.py 执行训练的文件

txt 一些使用到的txt文件

trainImageList.txt mark的训练数据的txt格式

wider\_face\_train\_bbx\_gt.txt pnp数据的txt格式

wider\_face\_train.txt 和上面的一样，不过有什么饱和度之类的

assist 主目录文件夹，操作的辅助程序

assist\_train train\_net.py,操作的辅助程序

forward.py 前向传播

read\_tfrecord.py 读取tfrecord文件

train\_lr\_op.py 动量法以及学习率的分段取值

assist\_hard pnp\_pic\_h.py，操作的辅助程序

read\_txt.py 读取txt的文件的数据信息

iter\_data.py 将txt的文件读取的地址，读出图片，形成一个迭代器

mtcnndetect.py 执行的detect的整体步骤，不论是为哪个net准备的

singledetector.py 将图片单个的执行前向传播

batchdetector.py 将图片组合成一个批量来进行前向传播

convert\_to\_square.py 将前向传播的图片可能不是正放心，这里将他们做成正方形

iou.py 计算重叠度的

mark\_infoget\_imgchange.py mark\_pic.py,mark数据读取和一些图片格式的改变

threcord\_assist.py tfrecord.py辅助程序

program\_test 写程序的时候用来测试的文件可以不用注意

val\_test 用于验证训练模型和用于测试的文件

one\_image\_test.py 检测一张照片上的人脸以及mark点

camera\_test.py 检测一个视频文件上的人脸

3生成文件架构及命名

logs 用于存储训练用的tensorboard

mark\_12 关键点图片

mark\_24 关键点图片

mark\_48 关键点图片

pic\_12\_E pos,neg,part图片

pic\_24\_HE pos,neg,part hard sample

pic\_48\_HHE pos,neg,part hard sample

pic\_txt 有总的有分的

tfrecord tfrecord文件用于训练网路

pic\_12\_E0.tfrecord\_shuffle

pic\_24\_HE0.tfrecord\_shuffle

pic\_24\_HE1.tfrecord\_shuffle

pic\_24\_HE2.tfrecord\_shuffle

pic\_24\_HE3.tfrecord\_shuffle

pic\_48\_HHE0.tfrecord\_shuffle

pic\_48\_HHE1.tfrecord\_shuffle

pic\_48\_HHE2.tfrecord\_shuffle

pic\_48\_HHE3.tfrecord\_shuffle

train\_model 这个里面记录了所有的轮次的网络数据，一般记录最后5次就可以了。

pic\_12\_E

pic\_24\_HE

pic\_48\_HHE

4执行文件的思路

4.1 PNP\_Pic\_E.py思路

* 路径准备

pos,neg,part图片文件夹的准备

pos,neg,part.txt文件位置的准备

* 每张图片生成neg，执行50次
* 对于每个box，p1移动delt，来生成neg，执行5次
* 对于每个box，中心移动delt，来生成pos,part，执行20次

4.2 PNP\_Pic\_H.py思路

* 准备路径

使用模型的路径

pos，neg，part保存的路径

txt文件保存的的路径

* 数据读取

read\_wider\_face\_train\_bbx\_gt，读取出全部的数据data

TestLoader(data['allimages\_path'])迭代器，每使用一次，输出的是一个data地址处的图片

* 设置每个net的detector

如果使用Pnet产生尺寸为24的hard sample，使用SingleDetector

如果使用Rnet产生尺寸为48的hard sample，使用BatchDetector

如果使用Onet，使用BatchDetector,这个在训练的时候没有用，只有在整体的test 模式才使用

single 和batch的区别是，batch是将所有的图片组成一个批量放进网络进行输出

* 整体的MTCNN detector

通过Pnet为Rnet训练产生图片，通过Pnet和Rnet为Onet训练产生图片，通过P，R，O，三个net为整体测试产生结果

boxes, boxes\_c, mark = self.detect\_pnet(im)

对每张img进行scale

对每张scale图片进行pnet测试

slt\_cls\_clsgen\_box，通过cls的score来产生box

py\_nms,根据score由高到低剔除重叠率高的

再对每个scale后的结果进行py\_nms剔除

return 是

box[cls12x1,y1,x2,y2,score](x1是cls用12这个尺寸构造的)

box\_c[[x1,y1,x2,y2,score],scale次数个](x1是12\*box\_pred,score)

如果说pic\_size = 24,为Rnet准备训练图片，那么就不执行后面两个了

boxes,boxes\_c,mark = self.detect\_rnet(im, boxes\_c)

遍历每个产生的boxes\_c

resize(24,24,3)

读取box处的pic进行Rnet测试， 得出cls\_prob,box\_pred

根据cls\_prob得分,去掉对应Pnet 得到的box

根据cls\_prob得分,去掉Rnet测试出来的box\_pred

再py\_nms将Pnet得到的boxes剔除一遍

根据Pnet产生的boxes，和Rnet得到的box\_pred（这是个比例）来获得box\_c

return

boxes这个是Pnet 经过pred的框

boxes\_c这个是boxes经过Rnetpred的框

如果说pic\_size = 48,为Onet准备训练图片，那么就不执行后面一个了

boxes, boxes\_c,mark =self.detect\_onet(im, boxes\_c)

遍历每个产生的boxes\_c

resize(48,48,3)

读取box处的pic进行Onet测试， 得出cls\_prob,box\_pred,mark\_pred

根据cls\_prob得分,去掉对应Rnet 得到的box

根据cls\_prob得分,去掉Onet测试出来的box\_pred

根据cls\_prob得分,去掉Onet测试出来的mark\_pred

box\_c =box\*boxpred

mark\_c = box\*mark\_pred

return

Rnet 的py\_nms剔除后的boxes

box\_c

mark\_c

* 图片存储以及txt生成

根据pic\_size，选择detector执行的个数，然后保存成pickle

调用save\_hard\_example进行存储

载入pickle，

detections中每一个都是一个image的box结果

遍历detections——image

遍历image——box

将这个box和image txt/wider\_face\_train\_bbx\_gt.txt中所有的box IoU相比

最大值小于0.3

归结到neg，进行图片存储以及txt写入

最大值大于0.65

归结到pos，进行图片存储以及txt写入

这个检测的box和哪个txt中box的 IoU最大，就记录他们的相对位置offset

最大值0.4-0.65

归结到part，进行图片存储以及txt写入

这个检测的box和哪个txt中box的 IoU最大，就记录他们的相对位置offset

4.3 Mark\_pic.py思路

将txt文档中的marks相对于box的位置存进去

随机移动后的box，如果iou>0.65,添加进txt，再随机翻转，镜像添加进去，这一步执行10次

4.4 pnpm\_gather\_txt. py思路

将pos,neg,part,mark的txt放到一个文件夹内也没有打乱

4.5 tfrecord.py思路

* 初始化路径

txt的路径

tfrecord的路径

* 获取txt，并将txt分为1或则4份组成一个datasets列表，这样就可以组成1个或则多个tfrecord文件

将txt中的文件dir的机器代码，label，box，mark tf.train.Example

将label，box，mark tf.train.Example写进1或者4个tfrecord文件中

4.6 train\_net.py思路

总体步骤

graph中定义的几个操作

placeholder input\_image、 label、 box\_target、 mark\_target

定义前向传播的操作，以及cls\_ohem,box\_ohem,mark\_ohem,L2\_loss,----total\_loss

指定学习率和训练的方法，这里就是反馈，采用动量法

定义tensorboard的整体操作

tfrecord读取数据

设置文件队列

tfrecord reader

读取文件队列

解析读取的内容

tf.train.batch这个将读取的数据添加到内存队列。输出的时候就是一个批量的输出了。

coord设置一个线程协调器

tf.train.start\_queue\_runners(sess = sess, coord = coord)启动文件队列的入队

sess设置以及初始化

for 进行训练

执行动量法训练conv1,conv2,conv3,conv4\_1,conv4\_2,conv4\_3的权重，偏置，还有alpha

每训练100个批量进行损失、精度的打印

每训练一轮保存一下训练的网络数据的结果

cls\_ohem

损失就是：对于每个图片根据label决定它是否是人脸，是人脸，选择pi，不是人脸选择1-pi，然后对所有选择的概率求负对数，然后求平均，不过程序选择的是单个损失最大的前70%

box\_ohem

对一个例子的4个offset求平均，然后对一个批量的求平均。

mark\_ohem

对一个例子的10个offset求平均，然后对一个批量求平均。

5文件中的重点API

5.1 train\_net.py API

image\_color\_distort

tensorboard 部分

writer = tf.summary.FileWriter(logs\_folder, sess.graph)

projector\_config = projector.ProjectorConfig()

projector.visualize\_embeddings(writer, projector\_config)

summary\_op = tf.summary.merge\_all(),多条summary协议

summary = sess.run(summary\_op)

writer.add\_summary(summary, global\_step = step)#将多条summary协议添加进tensorboard

model\_saver

saver = tf.train.Saver(max\_to\_keep = 0)

保存模型

path\_prefix = saver.save(sess,model\_folder,global\_step = epoch\_count \* 2)

在sess会话中执行这个操作，路径是model\_folder-global\_step

tfrecord读入数据和线程部分

filename\_queue = tf.train.string\_input\_producer(tfrecord\_dir\_list,shuffle = true)

放进文件队列（此处需要一个开关）

reader= tf.TFRecordReader()

\_,serialized\_example = reader.read(filename\_queue)

读取文件队列中的一个例子（一个文件队列有好多文件，一个文件有好多例子）

image\_features = tf.parse\_single\_exxample()

image = tf.decode\_raw(image\_feature[‘image/encoded’],tf.uint8)

将得到的例子进行解析，转换成可以使用的数据类型

image, label,roi,mark = tf.train.batch(

[image,label,roi,mark],#一个内存队列，中一个位置的储存内容

batch\_size = batch\_size,#输出的image,label这些形状

num\_threads = 2,#读入的线程

capacity = 1\*batch\_size)#一个内存队列，最多可以放置多少组数据

构建内存队列，

coord = tf.train.Coordinator()

线程协调器

threads = tf.train.start\_queue\_runners(sess = sess,coord = coord)

加入线程协调器，启动前面的开关

except tf.errors.OutOfRangeError:

内存队列读取完后，会产生一个OutOfRangeError这样的一个异常，可以利用这个异常来打印一些信息。

finally: coord.request\_stop()

最终程序顺利执行完毕，就自己请求异常

coord.join(threads)

判断这些线程有没有停止

5.1.1 train\_lr\_op

global\_step = tf.Variable(0,trainable = False)

tf.train.piecewise\_constant(global\_step,boundaries, lr\_values)

optimizer = tf.train.MomentumOptimizer(lr\_op, 0.9, name = ‘Momentum’)

train\_op = optimizer.minimize(loss, global\_step)

6 通过tensorboard查看图

在这一步，通过tensorboard来查看图，看看scalar对标量记录的曲线，看看直方图对标量记录的曲线，看看graph整体的训练思路

7梯度和动量法

关于对训练的理解，关键的是要理解梯度，和动量法这个两个，这个比较重要

梯度

梯度就是输出的标量对所有参数的偏导数的集合。再梯度方向上取值的增加的，所以为了使损失函数取得最小值，那么我们就在梯度的反方向上就使取值减小，损失函数减低更新网络。

举例说明：

1、准备训练的数据input,和label

2、构建了一个网络（比如卷积网络或者全连接网络）f[weight,bias] weight和bias表示网络的参数

3、input执行进行训练也就是执行f这个函数，那么就会得到一个y\_pred

4、的损失函数loss = y\_pred – label = f(input) – label（这里的loss仅仅是举例，可能loss不仅仅是减法的

关系）

5、对loss求梯度label每一次都可能不同，这里就当作一个常数近似了。这里会产生一定额误差。

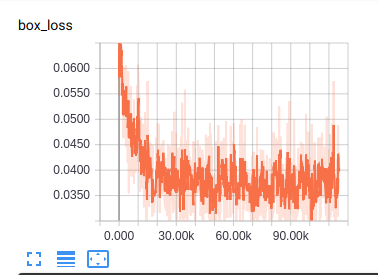
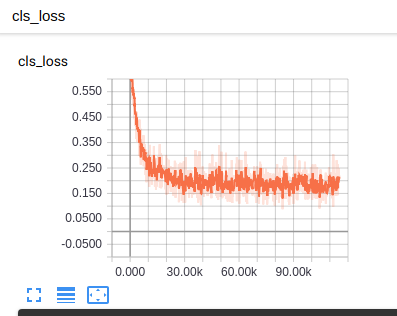
动量法

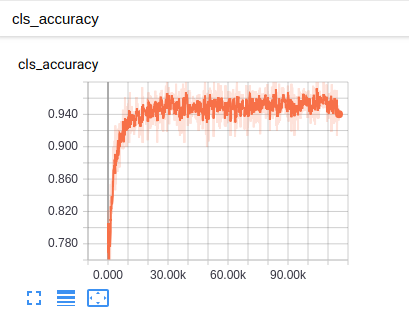
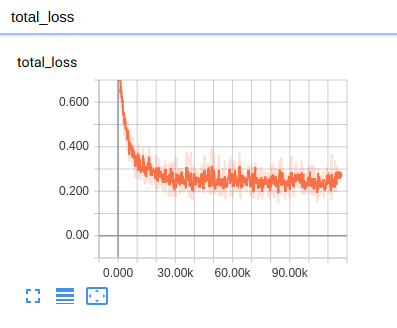
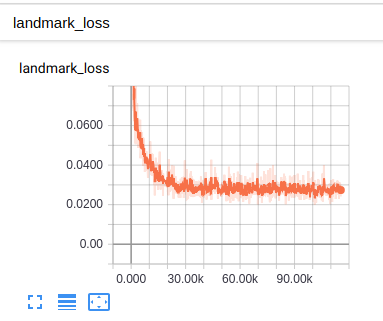
假如说W,B初始值都是0，或者随机的一个值，

动量法和SGD比的有点是，不会轻易停留在极小值处，他有动量，理科生应该明白，由于梯度的不断增加V，相当于能量的不断积累，从而动能增加，速度提高。如果遇到一个极小值也是不会停下来的，因为动量守恒，不会突然速度变为0 了

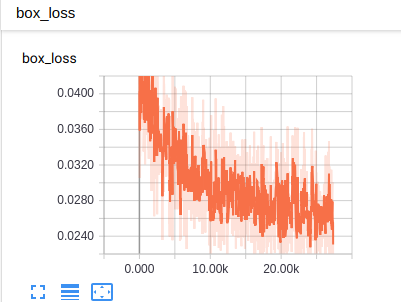
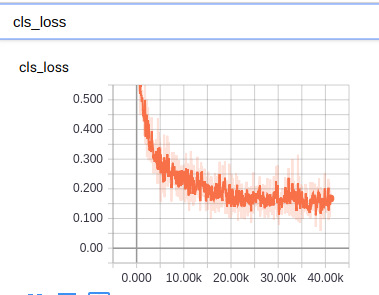
8训练结果

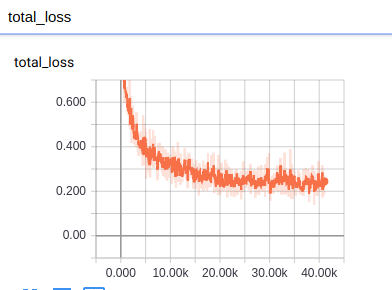
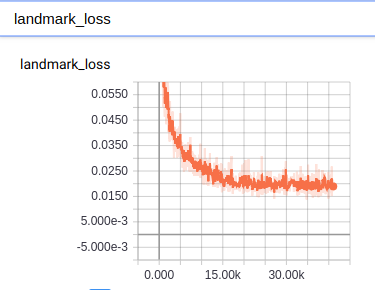
Rnet

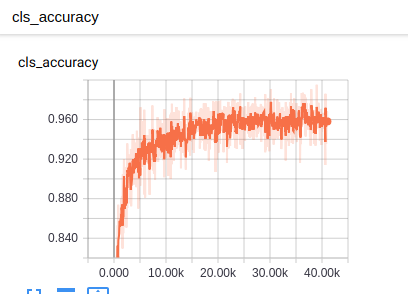




Onet







图片检测结果



9缺点

这个训练中使用人脸关键点的图片都几乎是正脸，所以对侧脸预测不准。镜头正对着检测更准确

如果有相关标记的数据集可以训练的更只能一些，比如一个侧脸一个后脑勺也能判断出这后面有一张人脸，并能预估出关键点的位置。

不过话分两头说，如果这样，这个就不能算是从人脸检测了，叫人头检测。

感谢<https://github.com/AITTSMD/MTCNN-Tensorflow>