# 一 环境及数据集介绍

## 1.1 代码环境

以<https://github.com/BobLiu20/mtcnn_tf>为base版本.

## 1.2 训练数据集

### 1.2.1 WIDER train数据集.

WIDER face dataset数据集.( [**WIDER FACE: A Face Detection Benchmark**)](http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/)

特点:

1. Face detection benchmark datasets, 图片均为公共图片
2. 含有32203张图, 有393703个人脸(scale, pose,occlusion(遮盖)等多变性图片).
3. 含有61分类事件. 40%/10%/50%的train,val,test占比.
4. 对于test图片,并不release对应的boundingbox ground truth.用户需要提交预测文件.由WIDER来做预测.



1.2.2 landmark 数据集

对于人脸关键点数据集采用的是:[CNN\_FacePoint](http://mmlab.ie.cuhk.edu.hk/archive/CNN_FacePoint.htm)其特点是:

1. 含有5590个LFW图像和7876个网络下载的图像.
2. trainImageList.txt和testImageList.txt分别保存有train,val数据集.

其格式是: 图片名字+矩形框+landmark的5个关键点(five facial points).

# 二 训练MTCNN

2.1 训练命令

|  |
| --- |
| #!/bin/bash  **set** -e  ### All of your tmp data will be saved in ./tmp folder  **echo** "Hello! I will prepare training data and starting to training step by step."  # 1. checking dataset if OK  **if** **[** **!** **-d** "./dataset/WIDER\_train/images" **];** **then**  **echo** "Error: The WIDER\_train/images is not exist. Read dataset/README.md to get useful info."  **exit**  **fi**  **if** **[** **!** **-d** "./dataset/lfw\_5590" **];** **then**  **echo** "Error: The lfw\_5590 is not exist. Read dataset/README.md to get useful info."  **exit**  **fi**  **echo** "Checking dataset pass."  **if** **[** **-d** "./tmp" **];** **then**  **echo** "Warning: The tmp folder is not empty. A good idea is to run ./clearAll.sh to clear it before training."  **fi**  # 2. stage: P-Net  ### generate training data(Face Detection Part) for PNet  **echo** "Preparing P-Net training data: bbox"  **python** prepare\_data**/**gen\_hard\_bbox\_pnet.py  ### generate training data(Face Landmark Detection Part) for PNet  **echo** "Preparing P-Net training data: landmark"  **python** prepare\_data**/**gen\_landmark\_aug.py **--**stage**=**pnet  ### generate tfrecord file for tf training  **echo** "Preparing P-Net tfrecord file"  **python** prepare\_data**/**gen\_tfrecords.py **--**stage**=**pnet  ### start to training P-Net  **echo** "Start to training P-Net"  **python** training**/**train.py **--**stage**=**pnet  # 3. stage: R-Net  ### generate training data(Face Detection Part) for RNet  **echo** "Preparing R-Net training data: bbox"  **python** prepare\_data**/**gen\_hard\_bbox\_rnet\_onet.py **--**stage**=**rnet  ### generate training data(Face Landmark Detection Part) for RNet  **echo** "Preparing R-Net training data: landmark"  **python** prepare\_data**/**gen\_landmark\_aug.py **--**stage**=**rnet  ### generate tfrecord file for tf training  **echo** "Preparing R-Net tfrecord file"  **python** prepare\_data**/**gen\_tfrecords.py **--**stage**=**rnet  ### start to training R-Net  **echo** "Start to training R-Net"  **python** training**/**train.py **--**stage**=**rnet  # 4. stage: O-Net  ### generate training data(Face Detection Part) for ONet  **echo** "Preparing O-Net training data: bbox"  **python** prepare\_data**/**gen\_hard\_bbox\_rnet\_onet.py **--**stage**=**onet  ### generate training data(Face Landmark Detection Part) for ONet  **echo** "Preparing O-Net training data: landmark"  **python** prepare\_data**/**gen\_landmark\_aug.py **--**stage**=**onet  ### generate tfrecord file for tf training  **echo** "Preparing O-Net tfrecord file"  **python** prepare\_data**/**gen\_tfrecords.py **--**stage**=**onet  ### start to training O-Net  **echo** "Start to training O-Net"  **python** training**/**train.py **--**stage**=**onet  # 5. Done  **echo** "Congratulation! All stages had been done. Now you can going to testing and hope you enjoy your result."  **echo** "haha...bye bye" |

## 2.1 生成Pnet训练数据

### 2.1.1 生成Neg,pos等boxes

prepare\_data**/**gen\_hard\_bbox\_pnet.py

|  |
| --- |
| **def** gen\_hard\_bbox\_pnet**(**srcDataSet**,** srcAnnotations**):**  srcDataSet **=** os**.**path**.**join**(**rootPath**,** srcDataSet**)**  srcAnnotations **=** os**.**path**.**join**(**rootPath**,** srcAnnotations**)**  saveFolder **=** os**.**path**.**join**(**rootPath**,** "tmp/data/pnet/"**)**  **print(**">>>>>> Gen hard samples for pnet..."**)**  **## 根据gtbox,原始图片,裁剪生成positive, negative,not care的矩形数据库.因为mtcnn输入是12x12的图,原图太大了. 所以要做一些crop.**  **同时保证正负样本均衡分布.**  typeName **=** **[**"pos"**,** "neg"**,** "part"**]**  saveFiles **=** **{}**  **for** tp **in** typeName**:**  \_saveFolder **=** os**.**path**.**join**(**saveFolder**,** tp**)**  **if** **not** os**.**path**.**isdir**(**\_saveFolder**):**  os**.**makedirs**(**\_saveFolder**)**  saveFiles**[**tp**]** **=** open**(**os**.**path**.**join**(**saveFolder**,** "{}.txt"**.**format**(**tp**)),** 'w'**)**  annotationsFile **=** open**(**srcAnnotations**,** "r"**)**  pIdx **=** 0 # positive  nIdx **=** 0 # negative  dIdx **=** 0 # dont care  idx **=** 0  **for** annotation **in** annotationsFile**:**  annotation **=** annotation**.**strip**().**split**(**' '**)**  # image path  imPath **=** annotation**[**0**]**  # boxed change to float type  bbox **=** map**(**float**,** annotation**[**1**:])**  # gt. each row mean bounding box  boxes **=** np**.**array**(**bbox**,** dtype**=**np**.**float32**).**reshape**(-**1**,** 4**)**  #load image  img **=** cv2**.**imread**(**os**.**path**.**join**(**srcDataSet**,** imPath **+** '.jpg'**))**  idx **+=** 1  height**,** width**,** channel **=** img**.**shape  # 1. NEG: random to crop negative sample image  **## 生成规定格式的负样本(iou<0.3).**  **Region大小是变化的.后来在把它们同意resize到12x12(pnet输入是12x12)的.**    **上图蓝色框是随机的neg框.生成过程是:**   1. **每次选一个size, 在0到虚线间找一个点做为左上角点(这个虚线到末尾的距离是size,同时蓝框是正方形,其变长也是size).** 2. **这个size是可以变化的,如右图所示变大了.** 3. **对于每个size,都只有一个篮框.** 4. **蓝框描述的Neg样本,其与gtbox的iou不能大于0.3.**      1. **这些不同尺寸的负样本框,需要resize到12x12(pnet输入shape).**   negNum **=** 0  **while** negNum **<** 50**:**  size **=** np**.**random**.**randint**(**12**,** min**(**width**,** height**)** **/** 2**)**  # top\_left  nx **=** np**.**random**.**randint**(**0**,** width **-** size**)**  ny **=** np**.**random**.**randint**(**0**,** height **-** size**)**  # random crop  **## 生成矩形框,为了先计算一下iou,如果iou合适(小于0.3)才会把图像的数据保存下来.**  crop\_box **=** np**.**array**([**nx**,** ny**,** nx **+** size**,** ny **+** size**])**  # cal iou and iou must below 0.3 for neg sample  iou **=** IoU**(**crop\_box**,** boxes**)**  **if** np**.**max**(**iou**)** **>=** 0.3**:**  **continue ## 负样本不满足要求就重新random,此时不消耗negNum**  # crop sample image  cropped\_im **=** img**[**ny **:** ny **+** size**,** nx **:** nx **+** size**,** **:]**  resized\_im **=** cv2**.**resize**(**cropped\_im**,** **(**12**,** 12**),** interpolation**=**cv2**.**INTER\_LINEAR**)**  # now to save it  save\_file **=** os**.**path**.**join**(**saveFolder**,** "neg"**,** "%s.jpg"**%**nIdx**)**  **saveFiles['neg'].write(save\_file + ' 0\n')**  cv2**.**imwrite**(**save\_file**,** resized\_im**)**  nIdx **+=** 1  negNum **+=** 1  **for** box **in** boxes**:**  # box (x\_left, y\_top, x\_right, y\_bottom)  x1**,** y1**,** x2**,** y2 **=** box  #bbox's width and height  w**,** h **=** x2 **-** x1 **+** 1**,** y2 **-** y1 **+** 1  # ignore small faces  # in case the ground truth boxes of small faces are not accurate  **## gtbox中可能有些人脸太小了,mtcnn对这部分支持不好.去掉它们**  **if** max**(**w**,** h**)** **<** 40 **or** x1 **<** 0 **or** y1 **<** 0**:**  **continue**  # 2. NEG: random to crop sample image in bbox inside  **## 此处创造一些和gtbox相交的负样本.**     1. **红色虚线框是负样本左上角可能落在的位置(是四个红色矩阵框)** 2. **它们的特点是蓝色矩形框都能和gt box相交.** 3. **Size是从12到w,h的最小值的一半.** 4. **和gtbox的Iou大于0.3的不需要.**   **for** i **in** range**(**5**):**  size **=** np**.**random**.**randint**(**12**,** min**(**width**,** height**)** **/** 2**)**  # delta\_x and delta\_y are offsets of (x1, y1)  delta\_x **=** np**.**random**.**randint**(**max**(-**size**,** **-**x1**),** w**)**  delta\_y **=** np**.**random**.**randint**(**max**(-**size**,** **-**y1**),** h**)**  nx1 **=** int**(**max**(**0**,** x1 **+** delta\_x**))**  ny1 **=** int**(**max**(**0**,** y1 **+** delta\_y**))**  **if** nx1 **+** size **>** width **or** ny1 **+** size **>** height**:**  **continue**  crop\_box **=** np**.**array**([**nx1**,** ny1**,** nx1 **+** size**,** ny1 **+** size**])**  Iou **=** IoU**(**crop\_box**,** boxes**)**  **if** np**.**max**(**iou**)** **>=** 0.3**:**  **continue**  cropped\_im **=** img**[**ny1**:** ny1 **+** size**,** nx1**:** nx1 **+** size**,** **:]**  resized\_im **=** cv2**.**resize**(**cropped\_im**,** **(**12**,** 12**),** interpolation**=**cv2**.**INTER\_LINEAR**)**  save\_file **=** os**.**path**.**join**(**saveFolder**,** "neg"**,** "%s.jpg"**%**nIdx**)**  saveFiles**[**'neg'**].**write**(**save\_file **+** ' 0\n'**)**  cv2**.**imwrite**(**save\_file**,** resized\_im**)**  nIdx **+=** 1  # 3. POS and PART  **## 生成POS和PART框.**   1. **根据左上角,找到中心点.**   x1 **+** w **/** 2   1. **在中心点左右gtbox宽1/5内随机.**   delta\_x **=** np**.**random**.**randint**(-**w **\*** 0.2**,** w **\*** 0.2**)**   1. **根据这个锚点,找到POS框的左上角**   **-** size **/** 2   1. **然后找到右下角**   **相当于POS矩形端点到中心点的x差,y差都是size/2.**  **进而POS矩形窗口的变长为size.**   1. **并且需要保证IOU在0.65以上(含)**     **for** i **in** range**(**20**):**  # pos and part face size [minsize\*0.8,maxsize\*1.25]  size **=** np**.**random**.**randint**(**int**(**min**(**w**,** h**)** **\*** 0.8**),** np**.**ceil**(**1.25 **\*** max**(**w**,** h**)))**  # delta here is the offset of box center  delta\_x **=** np**.**random**.**randint**(-**w **\*** 0.2**,** w **\*** 0.2**)**  delta\_y **=** np**.**random**.**randint**(-**h **\*** 0.2**,** h **\*** 0.2**)**  #show this way: nx1 = max(x1+w/2-size/2+delta\_x)  nx1 **=** max**(**x1 **+** w **/** 2 **+** delta\_x **-** size **/** 2**,** 0**)**  #show this way: ny1 = max(y1+h/2-size/2+delta\_y)  ny1 **=** max**(**y1 **+** h **/** 2 **+** delta\_y **-** size **/** 2**,** 0**)**  nx2 **=** nx1 **+** size  ny2 **=** ny1 **+** size  **if** nx2 **>** width **or** ny2 **>** height**:**  **continue**  crop\_box **=** np**.**array**([**nx1**,** ny1**,** nx2**,** ny2**])**  #yu gt de offset  offset\_x1 **=** **(**x1 **-** nx1**)** **/** float**(**size**)**  offset\_y1 **=** **(**y1 **-** ny1**)** **/** float**(**size**)**  offset\_x2 **=** **(**x2 **-** nx2**)** **/** float**(**size**)**  offset\_y2 **=** **(**y2 **-** ny2**)** **/** float**(**size**)**  #crop  cropped\_im **=** img**[**int**(**ny1**)** **:** int**(**ny2**),** int**(**nx1**)** **:** int**(**nx2**),** **:]**  #resize  resized\_im **=** cv2**.**resize**(**cropped\_im**,** **(**12**,** 12**),** interpolation**=**cv2**.**INTER\_LINEAR**)**  box\_ **=** box**.**reshape**(**1**,** **-**1**)**  **if** IoU**(**crop\_box**,** box\_**)** **>=** 0.65**:**  save\_file **=** os**.**path**.**join**(**saveFolder**,** "pos"**,** "%s.jpg"**%**pIdx**)**  saveFiles**[**'pos'**].**write**(**save\_file **+** ' **1** %.2f %.2f %.2f %.2f\n'**%(**offset\_x1**,** offset\_y1**,** offset\_x2**,** offset\_y2**))**  cv2**.**imwrite**(**save\_file**,** resized\_im**)**  pIdx **+=** 1  **elif** IoU**(**crop\_box**,** box\_**)** **>=** 0.4**:**  save\_file **=** os**.**path**.**join**(**saveFolder**,** "part"**,** "%s.jpg"**%**dIdx**)**  saveFiles**[**'part'**].**write**(**save\_file **+** ' **-1** %.2f %.2f %.2f %.2f\n'**%(**offset\_x1**,** offset\_y1**,** offset\_x2**,** offset\_y2**))**  cv2**.**imwrite**(**save\_file**,** resized\_im**)**  dIdx **+=** 1  printStr **=** "\r[{}] pos: {} neg: {} part:{}"**.**format**(**idx**,** pIdx**,** nIdx**,** dIdx**)**  sys**.**stdout**.**write**(**printStr**)**  sys**.**stdout**.**flush**()**  **for** f **in** saveFiles**.**values**():**  f**.**close**()**  **print** '\n' |

### 2.1.2 生成关键点

prepare\_data/gen\_landmark\_aug.py

|  |
| --- |
| **def** gen\_landmark\_data**(**srcTxt**,** net**,** augment**=False):**  '''  srcTxt: each line is: 0=path, 1-4=bbox, 5-14=landmark 5points  net: PNet or RNet or ONet  augment: if enable data augmentation  **## srcTxt是个label文件.**  **格式: path, 4个gtbox数值, 10个landmark坐标(5对,分x,y方向).**  '''  **print(**">>>>>> Start landmark data create...Stage: %s"**%(**net**))**  srcTxt **=** os**.**path**.**join**(**rootPath**,** srcTxt**)**  saveFolder **=** os**.**path**.**join**(**rootPath**,** "tmp/data/%s/"**%(**net**))**  saveImagesFolder **=** os**.**path**.**join**(**saveFolder**,** "landmark"**)**  sizeOfNet **=** **{**"pnet"**:** 12**,** "rnet"**:** 24**,** "onet"**:** 48**}**  **if** net **not** **in** sizeOfNet**:**  **raise** Exception**(**"The net type error!"**)**  **if** **not** os**.**path**.**isdir**(**saveImagesFolder**):**  os**.**makedirs**(**saveImagesFolder**)**  saveF **=** open**(**join**(**saveFolder**,** "landmark.txt"**),** 'w'**)**  imageCnt **=** 0  # image\_path bbox landmark(5\*2)  **for** **(**imgPath**,** bbox**,** landmarkGt**)** **in** getBboxLandmarkFromTxt**(**srcTxt**): ## 取出label配置.**  **F\_imgs = [] ## 是个list, 存imgs**  **F\_landmarks = []**  **## 是list, 存landmarks**  img **=** cv2**.**imread**(**imgPath**)**  **assert(**img **is** **not** **None)**  img\_h**,** img\_w**,** img\_c **=** img**.**shape  gt\_box **=** np**.**array**([**bbox**.**left**,** bbox**.**top**,** bbox**.**right**,** bbox**.**bottom**])**  f\_face **=** img**[**bbox**.**top**:** bbox**.**bottom**+**1**,** bbox**.**left**:** bbox**.**right**+**1**]**  f\_face **=** cv2**.**resize**(**f\_face**,** **(**sizeOfNet**[**net**],** sizeOfNet**[**net**])) ## 把每张图缩小,pnet输入要求12x12的.**  landmark **=** np**.**zeros**((**5**,** 2**)) ## landmark数组.**  #normalize  **for** index**,** one **in** enumerate**(**landmarkGt**):**  **## 因为上面图像缩小了.**  **Landmark也需要同比例缩小到12x12的图上.**  **Landmark在gtbox中比例位置是不变的.**  rv **=** **((**one**[**0**]-**gt\_box**[**0**])/(**gt\_box**[**2**]-**gt\_box**[**0**]),** **(**one**[**1**]-**gt\_box**[**1**])/(**gt\_box**[**3**]-**gt\_box**[**1**]))**  landmark**[**index**]** **=** rv  **F\_imgs.append(f\_face) ##保存face图.**  **F\_landmarks.append(landmark.reshape(10)) ## 保存landmark**  landmark **=** np**.**zeros**((**5**,** 2**))**  **if** augment**:**  x1**,** y1**,** x2**,** y2 **=** gt\_box  #gt's width  gt\_w **=** x2 **-** x1 **+** 1  #gt's height  gt\_h **=** y2 **-** y1 **+** 1  **if** max**(**gt\_w**,** gt\_h**)** **<** 40 **or** x1 **<** 0 **or** y1 **<** 0**:**  **continue**  **## 随机平移:**     1. **在crop层面,缩小0.8到扩大1.25之间.缩放** 2. **平移,以x轴为例,左右偏移1/5w.y轴上是上下偏移1/5y.**   #random shift  **for** i **in** range**(**10**):**  bbox\_size **=** np**.**random**.**randint**(**int**(**min**(**gt\_w**,** gt\_h**)** **\*** 0.8**),** np**.**ceil**(**1.25 **\*** max**(**gt\_w**,** gt\_h**)))**  delta\_x **=** np**.**random**.**randint**(-**gt\_w **\*** 0.2**,** gt\_w **\*** 0.2**)**  delta\_y **=** np**.**random**.**randint**(-**gt\_h **\*** 0.2**,** gt\_h **\*** 0.2**)**  nx1 **=** max**(**x1**+**gt\_w**/**2**-**bbox\_size**/**2**+**delta\_x**,**0**)**  ny1 **=** max**(**y1**+**gt\_h**/**2**-**bbox\_size**/**2**+**delta\_y**,**0**)**    nx2 **=** nx1 **+** bbox\_size  ny2 **=** ny1 **+** bbox\_size  **if** nx2 **>** img\_w **or** ny2 **>** img\_h**:**  **continue**  crop\_box **=** np**.**array**([**nx1**,**ny1**,**nx2**,**ny2**])**  cropped\_im **=** img**[**ny1**:**ny2**+**1**,**nx1**:**nx2**+**1**,:]**  resized\_im **=** cv2**.**resize**(**cropped\_im**,** **(**sizeOfNet**[**net**],** sizeOfNet**[**net**]))**  #cal iou  iou **=** IoU**(**crop\_box**,** np**.**expand\_dims**(**gt\_box**,**0**))**  **if** iou **<=** 0.65**:**  **continue**  F\_imgs**.**append**(**resized\_im**)**  #normalize  **for** index**,** one **in** enumerate**(**landmarkGt**):**  rv **=** **((**one**[**0**]-**nx1**)/**bbox\_size**,** **(**one**[**1**]-**ny1**)/**bbox\_size**)**  landmark**[**index**]** **=** rv  F\_landmarks**.**append**(**landmark**.**reshape**(**10**))**  landmark **=** np**.**zeros**((**5**,** 2**))**  landmark\_ **=** F\_landmarks**[-**1**].**reshape**(-**1**,**2**)**  bbox **=** BBox**([**nx1**,**ny1**,**nx2**,**ny2**])**  **## 在”shift”基础上做mirror.**   1. **有概率(50%)的去做mirror.** 2. **利用flip()实现镜像.位于tools\landmark\_utils.py.实现的是x轴方向的镜像.**   **相应的要把landmark也给镜像了(左眼和右眼互换,左右嘴角互换.)**  #mirror  **if** random**.**choice**([**0**,**1**])** **>** 0**:**  face\_flipped**,** landmark\_flipped **=** flip**(**resized\_im**,** landmark\_**)**  face\_flipped **=** cv2**.**resize**(**face\_flipped**,** **(**sizeOfNet**[**net**],** sizeOfNet**[**net**]))**  #c\*h\*w  F\_imgs**.**append**(**face\_flipped**)**  F\_landmarks**.**append**(**landmark\_flipped**.**reshape**(**10**))**  #rotate  **if** random**.**choice**([**0**,**1**])** **>** 0**:**  **## 也是概率的去做旋转.**   1. **先逆时针旋转5度.** 2. **然后偏移,flip** 3. **再转回来(顺时针5度).** 4. **旋转使用cv2. getRotationMatrix2D函数.** 5. **它需要待旋转的centerX,CenterY.** 6. **旋转角度** 7. **缩放比例** 8. **输出一个rot\_mat矩阵.矩阵如下:**   **对应位置相乘,然后加上最后一列.**    face\_rotated\_by\_alpha**,** landmark\_rotated **=** rotate**(**img**,** bbox**,** \  bbox**.**reprojectLandmark**(**landmark\_**),** 5**)**#逆时针旋转  #landmark\_offset  landmark\_rotated **=** bbox**.**projectLandmark**(**landmark\_rotated**)**  face\_rotated\_by\_alpha **=** cv2**.**resize**(**face\_rotated\_by\_alpha**,** **(**sizeOfNet**[**net**],** sizeOfNet**[**net**]))**  F\_imgs**.**append**(**face\_rotated\_by\_alpha**)**  F\_landmarks**.**append**(**landmark\_rotated**.**reshape**(**10**))**    #flip  face\_flipped**,** landmark\_flipped **=** flip**(**face\_rotated\_by\_alpha**,** landmark\_rotated**)**  face\_flipped **=** cv2**.**resize**(**face\_flipped**,** **(**sizeOfNet**[**net**],** sizeOfNet**[**net**]))**  F\_imgs**.**append**(**face\_flipped**)**  F\_landmarks**.**append**(**landmark\_flipped**.**reshape**(**10**))**    #inverse clockwise rotation  **if** random**.**choice**([**0**,**1**])** **>** 0**:**  face\_rotated\_by\_alpha**,** landmark\_rotated **=** rotate**(**img**,** bbox**,** \  bbox**.**reprojectLandmark**(**landmark\_**),** **-**5**)**#顺时针旋转  landmark\_rotated **=** bbox**.**projectLandmark**(**landmark\_rotated**)**  face\_rotated\_by\_alpha **=** cv2**.**resize**(**face\_rotated\_by\_alpha**,** **(**sizeOfNet**[**net**],** sizeOfNet**[**net**]))**  F\_imgs**.**append**(**face\_rotated\_by\_alpha**)**  F\_landmarks**.**append**(**landmark\_rotated**.**reshape**(**10**))**    face\_flipped**,** landmark\_flipped **=** flip**(**face\_rotated\_by\_alpha**,** landmark\_rotated**)**  face\_flipped **=** cv2**.**resize**(**face\_flipped**,** **(**sizeOfNet**[**net**],** sizeOfNet**[**net**]))**  F\_imgs**.**append**(**face\_flipped**)**  F\_landmarks**.**append**(**landmark\_flipped**.**reshape**(**10**))**  F\_imgs**,** F\_landmarks **=** np**.**asarray**(**F\_imgs**),** np**.**asarray**(**F\_landmarks**)**  **for** i **in** range**(**len**(**F\_imgs**)): ## 保存**  path **=** os**.**path**.**join**(**saveImagesFolder**,** "%d.jpg"**%(**imageCnt**))**  cv2**.**imwrite**(**path**,** F\_imgs**[**i**])**  landmarks **=** map**(**str**,** list**(**F\_landmarks**[**i**]))**  saveF**.**write**(**path **+** " -2 " **+** " "**.**join**(**landmarks**)+**"\n"**)**  imageCnt **+=** 1  printStr **=** "\rCount: {}"**.**format**(**imageCnt**)**  sys**.**stdout**.**write**(**printStr**)**  sys**.**stdout**.**flush**()**  saveF**.**close**()**  **print** "\nLandmark create done!" |

2.1.3 训练Pnet

training/train.pyt

Pnet的训练参数以及Loss组成占比.

|  |
| --- |
| **if** net **==** 'pnet'**:**  image\_size **=** 12  ratio\_cls\_loss**,** ratio\_bbox\_loss**,** ratio\_landmark\_loss **=** 1.0**,** 0.5**,** 0.5 |

Mtcnn训练的配置

|  |
| --- |
| config**.**BATCH\_SIZE **=** 384 ## 每次处理384张图  config**.**CLS\_OHEM **=** **True**  config**.**CLS\_OHEM\_RATIO **=** 0.7 ## Iou阈值(nms)  config**.**BBOX\_OHEM **=** **False**  config**.**BBOX\_OHEM\_RATIO **=** 0.7  config**.**EPS **=** 1e-14  config**.**LR\_EPOCH **=** **[**6**,** 14**,** 20**]** |

|  |
| --- |
| #define placeholder  **##　输入的图和label.**  input\_image **=** tf**.**placeholder**(**tf**.**float32**,** shape**=[**config**.**BATCH\_SIZE**,** image\_size**,** image\_size**,** 3**],** name**=**'input\_image'**)**  label **=** tf**.**placeholder**(**tf**.**float32**,** shape**=[**config**.**BATCH\_SIZE**],** name**=**'label'**)**  bbox\_target **=** tf**.**placeholder**(**tf**.**float32**,** shape**=[**config**.**BATCH\_SIZE**,** 4**],** name**=**'bbox\_target'**)**  landmark\_target **=** tf**.**placeholder**(**tf**.**float32**,**shape**=[**config**.**BATCH\_SIZE**,**10**],**name**=**'landmark\_target'**)**  #class,regression  **## 这个netFactory指的是PNET,RNET,ONET的model.**  cls\_loss\_op**,**bbox\_loss\_op**,**landmark\_loss\_op**,**L2\_loss\_op**,**accuracy\_op **=** **netFactory(**input\_image**,** label**,** bbox\_target**,**landmark\_target**,**training**=True)**  #train,update learning rate(3 loss)  train\_op**,** lr\_op **=** train\_model**(**baseLr**,** ratio\_cls\_loss**\***cls\_loss\_op **+** ratio\_bbox\_loss**\***bbox\_loss\_op **+** ratio\_landmark\_loss**\***landmark\_loss\_op **+** L2\_loss\_op**,** total\_num**)**  # init  init **=** tf**.**global\_variables\_initializer**()**  gpu\_options **=** tf**.**GPUOptions**(**allow\_growth**=True)**  sess **=** tf**.**Session**(**config**=**tf**.**ConfigProto**(**gpu\_options**=**gpu\_options**))**  #save model  saver **=** tf**.**train**.**Saver**(**max\_to\_keep**=**0**)**  sess**.**run**(**init**)**  #visualize some variables  tf**.**summary**.**scalar**(**"cls\_loss"**,**cls\_loss\_op**)**#cls\_loss  tf**.**summary**.**scalar**(**"bbox\_loss"**,**bbox\_loss\_op**)**#bbox\_loss  tf**.**summary**.**scalar**(**"landmark\_loss"**,**landmark\_loss\_op**)**#landmark\_loss  tf**.**summary**.**scalar**(**"cls\_accuracy"**,**accuracy\_op**)**#cls\_acc  summary\_op **=** tf**.**summary**.**merge\_all**()**  logs\_dir **=** os**.**path**.**join**(**rootPath**,** "tmp"**,** "logs"**,** net**)**  **if** os**.**path**.**exists**(**logs\_dir**)** **==** **False:**  os**.**makedirs**(**logs\_dir**)**  writer **=** tf**.**summary**.**FileWriter**(**logs\_dir**,** sess**.**graph**)**  #begin  coord **=** tf**.**train**.**Coordinator**()**  #begin enqueue thread  threads **=** tf**.**train**.**start\_queue\_runners**(**sess**=**sess**,** coord**=**coord**)**  i **=** 0  #total steps  MAX\_STEP **=** int**(**total\_num **/** config**.**BATCH\_SIZE **+** 1**)** **\*** endEpoch  **print** "\n\nTotal step: "**,** MAX\_STEP  epoch **=** 0  sess**.**graph**.**finalize**()**  **try:**  **for** step **in** range**(**MAX\_STEP**):**  i **=** i **+** 1  **if** coord**.**should\_stop**():**  **break**  image\_batch\_array**,** label\_batch\_array**,** bbox\_batch\_array**,**landmark\_batch\_array **=** sess**.**run**([**image\_batch**,** label\_batch**,** bbox\_batch**,**landmark\_batch**])**  #random flip  image\_batch\_array**,**landmark\_batch\_array **=** random\_flip\_images**(**image\_batch\_array**,**label\_batch\_array**,**landmark\_batch\_array**)**  '''  print image\_batch\_array.shape  print label\_batch\_array.shape  print bbox\_batch\_array.shape  print landmark\_batch\_array.shape  print label\_batch\_array[0]  print bbox\_batch\_array[0]  print landmark\_batch\_array[0]  '''  \_**,**\_**,**summary **=** sess**.**run**([**train\_op**,** lr\_op **,**summary\_op**],** feed\_dict**={**input\_image**:** image\_batch\_array**,** label**:** label\_batch\_array**,** bbox\_target**:** bbox\_batch\_array**,**landmark\_target**:**landmark\_batch\_array**})**    **if** **(**step**+**1**)** **%** display **==** 0**:**  #acc = accuracy(cls\_pred, labels\_batch)  cls\_loss**,** bbox\_loss**,**landmark\_loss**,**L2\_loss**,**lr**,**acc **=** sess**.**run**([**cls\_loss\_op**,** bbox\_loss\_op**,**landmark\_loss\_op**,**L2\_loss\_op**,**lr\_op**,**accuracy\_op**],**  feed\_dict**={**input\_image**:** image\_batch\_array**,** label**:** label\_batch\_array**,** bbox\_target**:** bbox\_batch\_array**,** landmark\_target**:** landmark\_batch\_array**})**  **print(**"%s [%s] Step: %d, accuracy: %3f, cls loss: %4f, bbox loss: %4f, landmark loss: %4f,L2 loss: %4f,lr:%f " **%** **(**  datetime**.**now**(),** net**,** step**+**1**,** acc**,** cls\_loss**,** bbox\_loss**,** landmark\_loss**,** L2\_loss**,** lr**))**  #save every two epochs  **if** i **\*** config**.**BATCH\_SIZE **>** total\_num**\***2**:**  epoch **=** epoch **+** 1  i **=** 0  saver**.**save**(**sess**,** modelPrefix**,** global\_step**=**epoch**\***2**)**  writer**.**add\_summary**(**summary**,**global\_step**=**step**)**  **except** tf**.**errors**.**OutOfRangeError**:**  **print(**"Done!"**)**  **finally:**  coord**.**request\_stop**()**  writer**.**close**()**  coord**.**join**(**threads**)**  sess**.**close**()** |

#### 2.1.2.1 何为netFactory之P\_Net.

在training/mtcnn\_model.py中

|  |
| --- |
| #construct Pnet  #label:batch  **def** P\_Net**(**inputs**,** label**=None,** bbox\_target**=None,** landmark\_target**=None,** training**=True):**  #define common param  **with** slim**.**arg\_scope**([**slim**.**conv2d**],**  activation\_fn**=**prelu**,## prelu激活函数**  weights\_initializer**=**slim**.**xavier\_initializer**(),**  biases\_initializer**=**tf**.**zeros\_initializer**(),**  weights\_regularizer**=**slim**.**l2\_regularizer**(**0.0005**),**  padding**=**'valid'**):**  net **=** slim**.**conv2d**(**inputs**,** 10**,** 3**,** stride**=**1**,**scope**=**'conv1'**)**  net **=** slim**.**max\_pool2d**(**net**,** kernel\_size**=[**2**,**2**],** stride**=**2**,** scope**=**'pool1'**)**  net **=** slim**.**conv2d**(**net**,** num\_outputs**=**16**,** kernel\_size**=[**3**,**3**],** stride**=**1**,** scope**=**'conv2'**)**  net **=** slim**.**conv2d**(**net**,** num\_outputs**=**32**,** kernel\_size**=[**3**,**3**],** stride**=**1**,** scope**=**'conv3'**)**  **# 三个任务输出.**   1. **2分类, 是否人脸** 2. **Boundingbox回归, 4个尺寸.** 3. **Landmark回归, 5对10个尺寸.**     #batch\*H\*W\*2  conv4\_1 **=** slim**.**conv2d**(**net**,** num\_outputs**=**2**,** kernel\_size**=[**1**,**1**],** stride**=**1**,** scope**=**'conv4\_1'**,** activation\_fn**=**tf**.**nn**.**softmax**)**  #batch\*H\*W\*4  bbox\_pred **=** slim**.**conv2d**(**net**,** num\_outputs**=**4**,** kernel\_size**=[**1**,**1**],** stride**=**1**,** scope**=**'conv4\_2'**,** activation\_fn**=None)**  #batch\*H\*W\*10  landmark\_pred **=** slim**.**conv2d**(**net**,** num\_outputs**=**10**,** kernel\_size**=[**1**,**1**],** stride**=**1**,** scope**=**'conv4\_3'**,** activation\_fn**=None)**  **if** training**:**  #batch\*2  cls\_prob **=** tf**.**squeeze**(**conv4\_1**,** **[**1**,**2**],** name**=**'cls\_prob'**)**  cls\_loss **=** **cls\_ohem(**cls\_prob**,** label**)**  #batch  bbox\_pred **=** tf**.**squeeze**(**bbox\_pred**,** **[**1**,** 2**],** name**=**'bbox\_pred'**)**  bbox\_loss **=** bbox\_ohem**(**bbox\_pred**,** bbox\_target**,** label**)**  #batch\*10  landmark\_pred **=** tf**.**squeeze**(**landmark\_pred**,** **[**1**,** 2**],** name**=**"landmark\_pred"**)**  landmark\_loss **=** landmark\_ohem**(**landmark\_pred**,** landmark\_target**,** label**)**  accuracy **=** cal\_accuracy**(**cls\_prob**,** label**)**  L2\_loss **=** tf**.**add\_n**(**slim**.**losses**.**get\_regularization\_losses**())**  **return** cls\_loss**,** bbox\_loss**,** landmark\_loss**,** L2\_loss**,** accuracy  **else:** # testing  #when test, batch\_size = 1  cls\_pro\_test **=** tf**.**squeeze**(**conv4\_1**,** axis**=**0**)**  bbox\_pred\_test **=** tf**.**squeeze**(**bbox\_pred**,** axis**=**0**)**  landmark\_pred\_test **=** tf**.**squeeze**(**landmark\_pred**,** axis**=**0**)**  **return** cls\_pro\_test**,** bbox\_pred\_test**,** landmark\_pred\_test |

何为cls\_ohem.

为了产生cls的Loss.

1. 输入有Pnet的预测, 以及label的ground truth.

|  |
| --- |
| **def** cls\_ohem**(**cls\_prob**,** label**):**  zeros **=** tf**.**zeros\_like**(**label**)**  #label=-1 --> label=0net\_factory  **## label值负数重置为0值.**  **考虑到会有Neg的样本,**  label\_filter\_invalid **=** tf**.**where**(**tf**.**less**(**label**,**0**),** zeros**,** label**)**  num\_cls\_prob **=** **tf.size(**cls\_prob**)**  **## tf.size会把cls\_prob这个tensor整理成0-d的格式.**  **然后再计算有多少个元素.**  cls\_prob\_reshape **=** tf**.**reshape**(**cls\_prob**,[**num\_cls\_prob**,-**1**])**  label\_int **=** tf**.**cast**(**label\_filter\_invalid**,**tf**.**int32**)**  num\_row **=** tf**.**to\_int32**(**cls\_prob**.**get\_shape**()[**0**])**  row **=** tf**.**range**(**num\_row**)\***2  **indices\_ = row + label\_int**  label\_prob **=** tf**.**squeeze**(**tf**.**gather**(**cls\_prob\_reshape**,** indices\_**))**  loss **=** **-**tf**.**log**(**label\_prob**+**1e-10**)**  zeros **=** tf**.**zeros\_like**(**label\_prob**,** dtype**=**tf**.**float32**)**  ones **=** tf**.**ones\_like**(**label\_prob**,**dtype**=**tf**.**float32**)**  valid\_inds **=** tf**.**where**(**label **<** zeros**,**zeros**,**ones**)**  num\_valid **=** tf**.**reduce\_sum**(**valid\_inds**)**  keep\_num **=** tf**.**cast**(**num\_valid**\***num\_keep\_radio**,**dtype**=**tf**.**int32**)**  #set 0 to invalid sample  loss **=** loss **\*** valid\_inds  loss**,**\_ **=** tf**.**nn**.**top\_k**(**loss**,** k**=**keep\_num**)**  **return** tf**.**reduce\_mean**(**loss**)** |