**FaceRecognition项目总结**

项目github地址：https://github.com/lyf524951805/MTCNN/tree/master/FaceRecognition

项目总结：

一.mtcnn人脸侦测器: 之前已训练，直接使用即可，在mtcnn\_detector目录下。

mtcnn检测器已优化，检测一张图片时间在0.06秒(平均)左右：

1）将逻辑没有先后顺序的所有for循环改为矩阵操作。

2）测试发现numpy比pytorch快，将相关pytorch操作改为numpy。

3）cv2.resize比PIL.Image的resize快，图像金字塔那里需要多次调用resize函数，因此image先转换为numpy。

二.特征提取器: 在extractor目录下

1.生成样本数据集: extractor/sample/gen\_sample.py

1）选择数据集

自己先网上爬虫了一些图片，但是爬的数据并不好，要自己手动清理，太费时。

最后决定直接使用CASIA-FaceV5数据集(500\*5=2500张)进行训练。

CASIA-FaceV5数据集(百度网盘)：

链接：https://pan.baidu.com/s/1LKw\_ZzlT9fLeuw8RI7uUpQ

提取码：cfbx

2）样本生成

使用mtcnn对CASIA-FaceV5数据集进行人脸检测并生成样本(训练集:测试集 =4:1)。

2.自定义人脸数据集: extractor/data.py

优化操作：

1）测试发现cv2读取图片速度比PIL块，因此使用cv2来读取图片。

cv2\_time: 0.0006249809265136719秒/张(平均)

pil\_time:0.0010937166213989258秒/张(平均)

1. 将transforms.ToTensor()放到\_\_init\_\_()中，这样不同每次都创建该对象。

3.模型: extractor/nets.py

同时定义DenseNet121(迁移)、ResNet50(迁移)、CenterLossNet、ArcFaceNet四种网 络模型，可对不同的模型、不同的损失函数的训练效果进行对比。

4.训练：extractor/train.py

1）数据和参数相同，DenseNet121与ResNet50作对比，DenseNet121的测试平均 余弦相似度高达0.81，而ResNet50只有0.65。DenseNet121比ResNet50效果更好， 因此后面的训练都选用DenseNet121模型。

2）DenseNet121+ArcFace条件下，设置不同的超参得到不同的效果：

第一次训练：feature\_dim=512, s=10, m=0.1; theta = torch.acos(cosine / 10)

训练平均余弦相似度能到0.99，测试平均余弦相似度0.82左右

第二次训练：feature\_dim=512, s=10, m=0.1; theta = torch.acos(cosine)

训练平均余弦相似度能到0.99，测试平均余弦相似度0.81左右

第三次训练：feature\_dim=1024, s=30, m=0.5, theta = torch.acos(cosine)

训练平均余弦相似度能到0.79，测试平均余弦相似度0.51左右，分类 accuracy为0.87。

第四次训练：feature\_dim=512, s=30, m=0.5, theta = torch.acos(cosine)

训练平均余弦相似度能到0.79，测试平均余弦相似度0.47左右，分类 accuracy为0.82。

总结：第一次和第二次效果一般，尽管余弦相似度很高，但是经常识别错误(分 类错误)的情况。第三次和第四次效果要更好一些，识别分类精度更好，由于 训练余弦相似度不高，故设置余弦相似度阈值=0.7。

5.特征提取器: extractor/feature\_extractor.py

操作：

1）设置模型为evaluate模式(模型中使用了Batch Normalization)。

2）调用with torch.no\_grad():, 包含的块中参数不进行梯度计算，节省cuda显存。

三.数据库: 在database目录下

1.人脸检测框图片存储目录: database/img

2.特征向量数据存储目录:

1）简化版: database/table，直接将特征向量数据转换为json格式保存到文件中。

2）Milvus版: 使用相似度检索引擎Milvus数据库。

简化版是直接将向量数据以json格式保存到文件中，但是由于频繁进行文件的读 写操作，特征识别对比时间太长。因此后期进行优化，直接使用相似向量检索引擎 Milvus。

相似向量检索引擎Milvus安装，百度网盘:

链接：https://pan.baidu.com/s/1w-pRpnVCSOdtnHISfzOv0Q

提取码：1yqs

四.相似向量搜索引擎: search\_engine/milvus\_engine.py

主要功能：

1）创建客户端(连接Milvus服务端)

2）保存向量：创建集合同时创建向量的索引，并将向量插入集合中。

3）向量相似度检索

4）查询数据库中的所有集合名称列表

5）清空数据库中的所有集合

五.人脸识别器: 在recognizer目录下

主要功能包括:人脸注册(特征入库)、人脸识别。

并且同时支持从视频源或图片源的相关操作。

简化版：recognizer/recognize\_simple.py

Milvus版: recognizer/recognize\_milvus.py

简单版与Milvus版的时间对比:

1）人脸注册(mtcnn检测时间+特征入库时间):

简单版: 0.06916962669353292秒/张(平均)

Milvus版: 0.0709135023632435秒/张(平均)

2）人脸识别(mtcnn检测时间+特征对比时间):

简单版: 0.366640928752227秒/张(平均)

Milvus版: 0.08610883278209888秒/张(平均)

对比结果分析:

1）对于人脸注册，简单版和Milvus版的速度基本一致，达到14帧/秒。

2）对于人脸识别，简单版速度只有3帧/秒；Milvus版向量相似检索速度很快，可 以保持12帧/秒。

------

参考资料：

1.https://github.com/deepinsight/insightface/tree/master/recognition/ArcFace