## 使用tensorflow训练landmark模型

* 当前电脑python安装版本为Python 3.5.2 |Anaconda 4.2.0 (64-bit)，tensorflow版本为1.3.0，gpu。
* 整个训练代码都在E:\python\_vanilla文件夹下，也可以打开PyCharm里的工程python\_vanilla。

### 代码简介

#### model.py

建立模型类 OrdinaryCNNModel，该类会初始化神经网络模型。在调用时需要提供config, images, labels, is\_training参数，并且调用类对象的build()函数，才会完成初始化。参数的含义可以参考代码注释。

模型的结构可以参考类的build\_graph方法，当前使用的是8个卷积层接fc层。

#### dataset.py

该文件完成数据的处理，并用于训练过程中将图片，关键点坐标喂给网络。

具体来说，该文件使用了tensorflow提供的dataset接口，如果不了解可以参考<https://tensorflow.google.cn/programmers_guide/datasets>，简单来说，该类完成了数据从disk上的读取，预处理，并且可以并行来提高速度。

* get\_dataset\_from\_file(config=None)函数将用于返回dataset类，调用该函数可以直接返回用于训练和测试的dataset类，需要注意的是，如果你要增加或者减少数据，你需要改变该文件的path变量，它是所有数据存放的文件夹，也就是说你可以去查看这些文件夹来了解当前我用于训练的数据。
* 在get\_dataset\_from\_file中，dlib\_file\_list用于返回所有用dlib提取过人脸的图片的路径，它会对原有图片进行人脸提取，并将图片保存，记录landmark坐标，并返回新的路径和新的landmark坐标。需要注意的是，由于代码设计问题，我将所有提取好的路径和landmark使用pickle模块保存到了硬盘上，因此如果你需要更改人脸提取的算法，你需要删除保存的文件。或者，重新改写这部分代码。总体来说，dlib\_file\_list会返回一个列表，列表里每个元素都是由文件路径，landmark坐标列表组成的一个而元素列表。例如[[img\_path, [x1,y1,x2,y2…]], [img\_path, [x1,y1,x2,y2…]], …]
* get\_dataset\_from\_file函数中最重要的部分是\_f函数，该函数主要将python中的list，warp到了tensorflow的dataset中。整体流程类似于上面的网页里tensorflow文档中。这里需要注意的是，如果is\_train为正，那么会调用read\_py\_function来对图片进行读取，在这个函数中调用了random\_auc\_func对原图像进行augmentation。
* random\_auc\_func是通过random\_augmentation函数返回的一个函数，类似于函数指针，函数handle。简单来说random\_auc\_func会接受一个图片和landmark坐标组成的列表，返回随机处理后的图片和坐标，可以通过调用test\_aug()函数，来观察和测试它的功能。
* random\_augmentation调用该函数，并且对参数赋予不同的值便可以控制随机处理的程度。需要注意的是，随机翻转效果很差，随机加噪声也没有什么用处，所以请将它们置为None，具体的随机处理可以查看源码了解，也可以通过test\_aug()来观察。
* 整体来说，get\_dataset\_from\_file函数完成了图片的读入（真实读入发生在开始训练），图片，label的预处理，图片的resize，数据的batch等功能。

#### train.py

* 模型config的设置：train一开始进行了模型参数的设置，其中大多数可以通过参数名称了解含义。个别的，例如clip\_grad用于设置是否对梯度进行截断，在训练过程中并没有出现梯度爆炸的情况，因此我将它设置为False，max\_epoch设置最大训练的回合数，restore设置是否接着之前训练的结果继续finetune，它restore的目标是min\_save\_name设置的目标，所以如果你没有这个模型就设置为False，min\_save\_name设置了loss最小的保存的模型名称，bn设置了是否使用batch normalization，可以参考resnet来了解更多。
* 训练过程：整体的训练流程为，在训练集上训练一遍，在测试集（有人称为验证集）上测试一遍，并记录测试集的误差，保存当前误差最小的模型（类似于early stopping的想法）。
* tensorboard查看训练过程，在训练过程中，我将很多的向量写进了summary（如果不了解，可以查看tensorflow summary的文档），所有的内容都会写进E:\python\_vanilla\log\_dir文件夹，可以打开cmd运行 `tensorboard –logdir=E:\python\_vanilla\log\_dir` 并打开浏览器查看（默认localhost:6006），你也可以查看该文件夹，去了解我之前的部分训练结果。

#### 其他

* finetune.py 在试验过程中，遇到了定位不准的问题，在查看原数据的过程中发现，很多原图片的关键点标记不准确，因此我计划挑选出我认为准确的数据，并且用这些“准确”的数据去finetune训练好的模型。但是这一部分的工作更多的是在caffe下完成，你可以查看finetune.py代码，它和train.py没太大区别。你也可以查看这篇报告的第三部分，使用caffe训练以及finetune来了解我最新的试验结果。
* 文件夹validation\_dataset存放用于验证模型的图片
* 文件夹tensorpack\_implementation存放了一开始我想使用tensorpack写的代码，但是它对windows支持不好，bug频出，最后我放弃了，使用原生的tensorflow。
* 文件夹caffe存放了使用caffe训练需要的文件，所以你也可以用caffe训练

#### 服务器下

我在公司的186服务器下搭建了同样的训练环境，并且将代码移植到了服务器下，你可以查看路径/DATABASE/zcq/face\_landmark来了解，直接运行train.py就可以开始训练。但是，如果你将服务器下训练好的模型拷贝至当前这台电脑，会发现验证效果很差，原因我还没有找到。

## Windows下tensorflow编译

我在windows尝试编译tensorflow来使用它提供的c++接口，但是，没有成功。

我主要参考的是<https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/cmake>下readme文件，使用Visual Studio 2015和Python3.5进行编译。

你可以打开cmake查看我之间编译时进行的设置，在编译之前需要设置ENABLE\_GPU, BUILD\_SHARED\_LIB等，你可以打开E:\tensorflow-r1.3\tensorflow-r1.3\tensorflow\contrib\cmake\build 下的tensorflow.sln工程（请使用2015）来查看我使用cmake生产的工程，但是在build tensorflow过程中遇到了各种各样的bug。

## 使用caffe训练landmark模型以及finetune和验证

所有的代码都在E:\landmark\_cpp\landmark\_cpp\landmark\_cpp文件夹下。

你可以通过查看 README.md 查看大致流程。

### 代码流程

#### dataset

dataset主要用于图片数据和landmark数据的处理，并产生相关的txt文件，你可以查看E:\landmark\_cpp\landmark\_cpp\landmark\_cpp\dataset 文件夹下的txt文件来了解。

#### face\_detection

detection主要用于人脸的定位，运行README.md里对应的命令可以对人脸进行定位并保存，同样会产生txt文件。你可以查看E:\landmark\_cpp\landmark\_cpp\landmark\_cpp\face\_detection下的txt文件来了解。

#### train

在E:\landmark\_cpp\landmark\_cpp\landmark\_cpp\train文件夹下，主要分为两个部分，一是根据之前detection得到的结果，产生caffe训练需要的.h5文件，二是调用caffe来训练模型。

#### finetune

使用我挑出来的，关键点标记“准确”的样本进行finetune，主要是指E:\ce，E:\glasses两个文件夹下的样本。同样需要生成.h5文件，并且在调用caffe命令时，需要指定之前训练的模型的路径。

#### landmark\_detector

可以使用visual studio打开E:\landmark\_cpp\landmark\_cpp下的landmark\_cpp.sln工程，但是这个工程文件有点问题，导致一个我解决不了的问题，因此我在E:\ConsoleApplication1下又建立了一个工程文件，在里面的landmarkValidation.cpp源代码中，修改trained\_file为finetune好的模型文件，运行该项目即可将验证图片预测的关键点标记在E:\landmark\_cpp\ landmark\_cpp\landmark\_cpp\validation\_result\ 文件夹下（由于时间问题，我还没有跑到这里，所以可能会有bug）。

如果你有任何的疑问可以联系我 wechat:\*\*\*\*\*\*\*\* \*\*\*