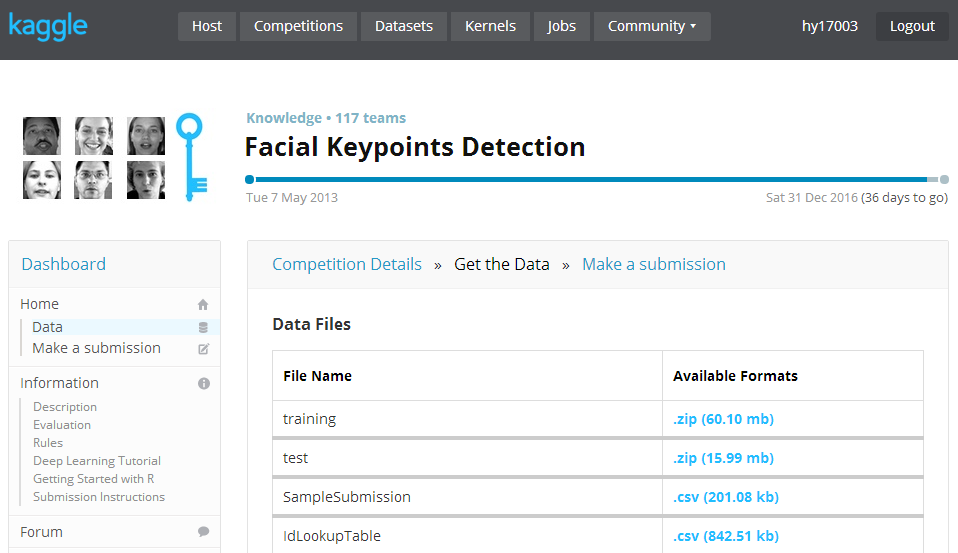
# 人脸特征点识别

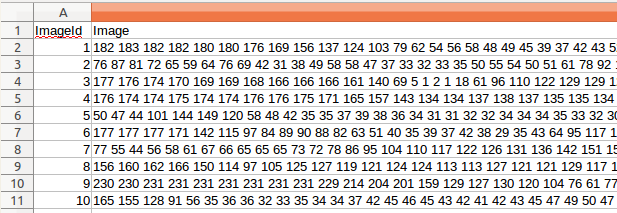
本实例主要演示了HDF5数据格式及内层数据层在深度学习上的使用。相比于LMDB数据格式只能输出一个标签，HDF5可以输出多个标签。相比于其它数据输入层从磁盘中读取数据，内存数据层直接从内存中读入数据。

## 准备数据

从上https://www.kaggle.com/c/facial-keypoints-detection/data下载训练和测试数据：



下载的数据是csv格式的数据，其格式如下：





每一张图像有15个坐标点，共30个标签输出，因此需要使用HDF5数据格式，将CSV格式转换到HDF5格式的代码如下：

import os

import numpy as np

from pandas.io.parsers import read\_csv

from sklearn.utils import shuffle

import h5py

TRAIN\_CSV = '/home/hy17003/caffe/examples/kaggle-face/data/csv/training.csv'

def csv\_to\_hd5():

dataframe = read\_csv(os.path.expanduser(TRAIN\_CSV))

dataframe['Image'] = dataframe['Image'].apply(lambda img:np.fromstring(img,sep=' '))

dataframe = dataframe.dropna()

data = np.vstack(dataframe['Image'].values)/255.

label = dataframe[dataframe.columns[:-1]].values

label = (label-48)/48

data, label = shuffle(data, label, random\_state = 0)

return data, label

if \_\_name\_\_ == '\_\_main\_\_':

data, label = csv\_to\_hd5()

data = data.reshape(-1,1,96,96)

data\_train = data[:-100,:,:,:]

data\_val = data[-100:,:,:,:]

label = label.reshape(-1,1,1,30)

label\_train = label[:-100,:,:,:]

label\_val = label[-100:,:,:,:]

#data\_train 的shape：(2040,1,96,96), label\_train 的shape：(2040,1,1,30)

fhandle = h5py.File('train.hd5','w')

fhandle.create\_dataset('data',data = data\_train, compression = 'gzip', compression\_opts = 4)

fhandle.create\_dataset('label',data = label\_train, compression = 'gzip', compression\_opts = 4)

fhandle.close()

fhandle = h5py.File('val.hd5','w')

fhandle.create\_dataset('data',data = data\_val, compression = 'gzip', compression\_opts = 4)

fhandle.create\_dataset('label',data = label\_val, compression = 'gzip', compression\_opts = 4)

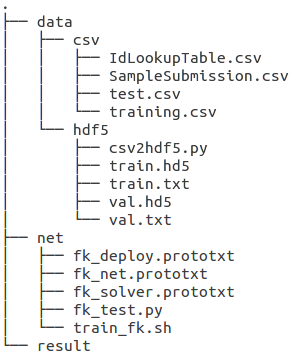
fhandle.close()

注意HDF5文件的生成方法，输入的是一个四维数组，另外在标签的处理上，将分布在96×96图像上的坐标归一化为[-1,1]。

为便于后面的测试演示，将test.scv文件用excel打开，删除大部分数据，只保留前10幅图像的数据。

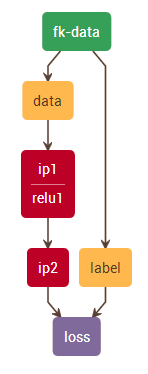
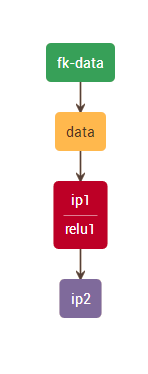
## 目录结构

由于代码中涉及到各种文件路径，将项目的结构进行简单说明，以便再现该实验。



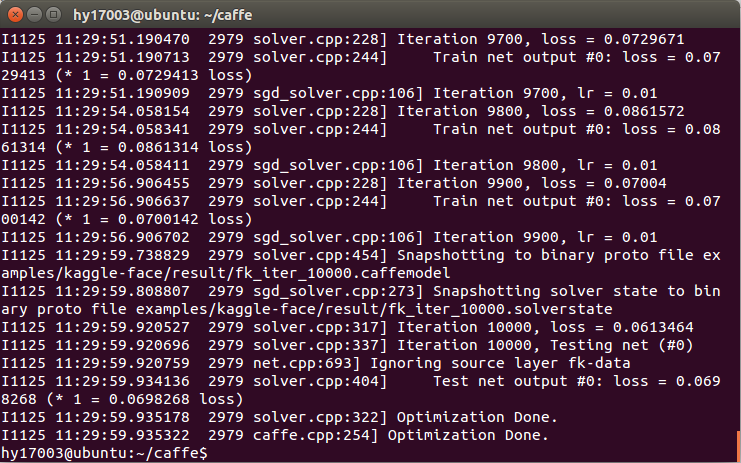
## 网络设计

使用两个全连接层，训练网络和测试网络结构图如下：

## 训练

执行train\_fk.sh脚本文件



## 预测

由于测试数据是csv文件，而测试网络使用的是内存数据层，因此将csv文件中的数据读取出来，使用Caffe的Net对象的set\_input\_arrays方法，将内层数据传输到网络中。另外，需要将输出的数据从[-1,1]区间转换到[0,96]。

import numpy as np

import pandas as pd

import caffe

MODEL\_FILE = '/home/hy17003/caffe/examples/kaggle-face/net/fk\_deploy.prototxt'

PRETRAINED = '/home/hy17003/caffe/examples/kaggle-face/result/fk\_iter\_10000.caffemodel'

dataframe = pd.read\_csv('/home/hy17003/caffe/examples/kaggle-face/data/csv/test.csv',header = 0)

dataframe['Image']=dataframe['Image'].apply(lambda im:np.fromstring(im,sep=' '))

data = np.vstack(dataframe['Image'].values)

data = data.reshape([-1,96,96])

data = data.astype(np.float32)

data = data/255.

data = data.reshape(-1,1,96,96)

net = caffe.Net(MODEL\_FILE,PRETRAINED,caffe.TEST)

caffe.set\_mode\_cpu()

total\_images = data.shape[0]

print 'Image to be predicted:', total\_images

dataL = np.zeros([total\_images,1,1,1],np.float32)

net.set\_input\_arrays(data.astype(np.float32),dataL.astype(np.float32))

pred=net.forward()

predicted=net.blobs['ip2'].data\*48+48

print 'Predicted:', predicted

print 'Predicted shape: ',predicted.shape

print 'Saving to csv..'

np.savetxt("fkp\_ouput.csv",predicted,delimiter=",")

