@article{wu2017facial,

title={Facial landmark detection with tweaked convolutional neural networks},

author={Wu, Yue and Hassner, Tal and Kim, KangGeon and Medioni, Gerard and Natarajan, Prem},

journal={IEEE Transactions on Pattern Analysis and Machine Intelligence},

year={2017},

publisher={IEEE}

}

Implementation of the Vanilla CNN described in the paper Yue Wu and Tal Hassner, "Facial Landmark Detection with Tweaked Convolutional Neural Networks", arXiv preprint arXiv:1511.04031, 12 Nov. 2015. See project page for more information about this project. <http://www.openu.ac.il/home/hassner/projects/tcnn_landmarks/>

Written by Ishay Tubi : ishay2b [at] gmail [dot] com <https://www.linkedin.com/in/ishay2b>

This software is provided as is, without any warranty, with no legal constraints. The data is provided to allow easy deployment, is it not mine, a link to the original owners of the data is provided.

**=============================================== Prerequisites:**

Caffe, Python, Numpy, dlib

**=============================================== Environment variables**

To run the code on OSX (i.e. anaconda python), where ROOT means repository main folder export PYTHONPATH=$(ROOT):$PYTHONPATH export PYTHONHOME=/Applications/anaconda # To resolve issues running python layers from command line.

**=============================================== Paths needed in PYTHONPATH**

CAFFE\_ROOT the path to the caffe distribute folder. CAFFE\_ROOT+"/python" will be added to PYTHONPATH DLIB\_ROOT - Dlib’s python module - if not already in PYTHONPATH. ROOT is the git main path.

**=============================================== How to run this script? use mainLoop.py**

To run all steps assign STEPS with FULL\_STEPS: STEPS = FULL\_STEPS

Or run a partial script like this: STEPS = ['testset']

**The steps needed to run:**

1. Calculate train data mean matrix or load already calculated trainMean.png.
2. Calculate train data std matrix or load already calculated trainSTD.png.
3. Create train set hdf.
4. Create test set hdf.
5. Train from random initialization by running this command from ROOT path, dump both stdout and error to log.txt: caffe.bin train -caffeData/solver solver\_adam\_vanilla.prototxt >>log.txt 2>&1
6. Plot the error by parsing the log (from ROOT directory): python python/parseLog.py log.txt
7. Create benchmarks once using STEPS=['createAFLW\_TestSet', 'createAFW\_TestSet']
8. Run benchmarks test by: STEPS=['testAFW\_TestSet', 'testAFLW\_TestSet']

**=============================================== Main functions used**

BBox - generic box class with helpers. ErrorAcum - accumulates the error DataRow is a class with landmarks, image and parsed from CSV. Can accept bounding box and crop/scale to desired size. createDataRowsFromCSV - translates CSV file into a list of DataRow. Passing the CSV parser as a parameter for each format. Predictor - a wrapper for caffe network. Call predictor.preprocess() to get image subtracted by mean and divided by std image. Also returns the landmarks scaled -0.5..+0.5.