FaceDetector app documentation

This is the application that allows one to test performance of open source face detectors implemented in Python using only CPU computational capacity.

1. Installation

Run next commands in terminal or command line interpreter.

```
cd ~
git clone https://github.com/Slavastas119/face-detectors-comparison
cd face-detectors-comparison
pip install -r requirements.txt
```

It is assumed that you have **pip** already installed. If not, please install **pip** separately.

2. Application parameters

2.1. Video stream

Choose a video stream where faces will be detected.

2.1.1. default video

Video file which comes with the app.

2.1.2. webcam

Use webcam as a source of video.

2.1.3. open new file

Choose a different file to run face detection on. These could be images or videos.

2.2. Model type

High level approaches to face detection task that are implemented in the app.

2.2.1. Cascades

A group of face detection models provided by OpenCV.

2.2.2. Dlib

Open source library that provides two face detection algorithms.

2.2.3. Caffe

Pre-trained deep learning model implemented in Caffe framework.

2.3. Model

Separate face detection models for each approach.

2.3.1. Cascades

2.3.1.1. Haar cascades

The Viola–Jones object detection framework. There are four pre-trained models that run under this algorithm.

2.3.1.1.1. haarcascade_frontalface_default

A default model. Will use it as a baseline.

2.3.1.1.2. haarcascade_frontalface_alt

About 2 times faster than the default model.

2.3.1.1.3. haarcascade_frontalface_alt2

About 2 times faster than the default model.

2.3.1.1.4. haarcascade_frontalface_alt_tree

About 2 times faster than the default model.

2.3.1.2. LBP cascades

Local binary patterns (LBP) visual descriptor used for classification in computer vision.
Usually works few times faster than Haar cascades, but with less accuracy. However, that may be overcomed via proper parameter tuning.

2.3.1.2.1. lbpcascade_frontalface

About 8-10 times faster than the default haar model.

2.3.1.2.2. lbpcascade_frontalface_improved

About 4-5 times faster than the default haar model.

2.3.2. Dlib

2.3.2.1. hog

This face detector is made using the now classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme.

2.3.2.2. cnn

Convolutional Neural Network based face detector using dlib library. The CNN model is much more accurate than the HOG based model, but takes much more computational power to run, and is meant to be executed on a GPU to attain reasonable speed.

2.3.3. Caffe

2.3.3.1. res10_300x300_ssd_iter_140000

The model was created with SSD framework using ResNet-10 like architecture as a backbone. Channels count in ResNet-10 convolution layers was significantly dropped (2x- or 4x- fewer channels).
The model was trained in Caffe framework on some huge and available online dataset.

2.4. Parameters

2.4.1. Cascades

2.4.1.1. Scale factor

Parameter specifying how much the image size is reduced at each image scale. Basically the scale factor is used to create your scale pyramid. More explanation can be found here.
Lower values allow to detect more faces, but decrease the speed.
Higher values may miss some faces, but perform much faster.

2.4.1.2. Minimum number of neighbors

Parameter specifying how many neighbors each candidate rectangle should have to retain it.
This parameter will affect the quality of the detected faces.
Lower values result in more detections but with lower quality.
Higher values result in less detections but with higher quality.
3~6 is a good value for it, and 5 is recommended.

This parameter does not affect the speed significantly.

2.4.1.3. Minimum size of a face (x axis)

Minimum possible object size by x axis. Works together with the next parameter. Objects smaller than that are ignored.
This parameter determine how small size you want to detect.
Lower values allow to detect all (of all sizes) faces, but perform slower.
Higher values ignore small faces, but speed up the process.

2.4.1.4. Minimum size of a face (y axis)

Minimum possible object size by y axis. Works together with the previous parameter. Objects smaller than that are ignored.
This parameter determine how small size you want to detect.
Lower values allow to detect all (of all sizes) faces, but perform slower.
Higher values ignore small faces, but speed up the process.

2.4.2. Dlib

2.4.2.1. Number of times to unsample

Indicates how many times to upsample the image looking for faces.
Lower values may miss small faces, but perform much faster.
Higher numbers find smaller faces, but significantly slower.

2.4.3. Caffe

2.4.3.1. Threshold

Faces with lower confidence than threshold will not be reported.
This parameter will affect the quality of the detected faces.
Lower values result in more detections but with lower quality.
Higher values result in less detections but with higher quality.
This parameter does not affect the speed at all.

2.5. Average time per frame

Average time (in milliseconds) spent to process one frame during the last run.
Important: this time includes only the process of face detection algorithm. Resulted video stream with bounding boxes may look slower, because of the time to draw those boxes.

2.6. Average FPS number

Average number of frames per second that can be processed by the algorithm based on **Average time per frame**.

3. Usage

3.1. Go to directory where **face_detectors_comparison.py** file is located in terminal or command line interpreter.

3.2. Run **face_detectors_comparison.py** file.

3.2.1. MacOS/Linux/Ubuntu

In terminal execute

```
python face_detectors_comparison.py
```

3.2.2. Windows

In command line interpreter execute

```
face_detectors_comparison.py
```

3.3. Choose desirable parameters and run face detector by clicking on the respective button on the bottom of the application window.