Face detector

**DNN Face Detector in OpenCV**

It is a Caffe model which is based on the Single Shot-Multibox Detector (SSD) and uses ResNet-10 architecture as its backbone. It was introduced post OpenCV 3.3 in its deep neural network module. There is also a quantized Tensorflow version that can be used but we will use the Caffe Model.

pip install opencv-python

Load the network using cv2.dnn.readNetFromCaffe and pass the model's layers and weights as its arguments. To achieve the best accuracy run the model on BGR images resized to 300x300 applying mean subtraction of values (104, 177, 123)for each blue, green and red channels correspondingly.

Finally, a 4-D array is returned which contains the confidence and coordinates scaled down to the range of 0 to 1 such that by multiplying them by the original width and height to predictions for the original image can be obtained as opposed to of the 300x300 on which the model predicted.



Model’s layers and weights:

https://github.com/opencv/opencv\_3rdparty/raw/dnn\_samples\_face\_detector\_20170830/res10\_300x300\_ssd\_iter\_140000.caffemodel

<https://github.com/opencv/opencv/blob/master/samples/dnn/face_detector/deploy.prototxt>

The .prototxt file(s) which define the model architecture (i.e., the layers themselves)

The .caffemodel file which contains the weights for the actual layers

The face detection model of the DNN module of OpenCV works well but if the size of the image is very large then it can cause problems. Generally, we don’t work with such 3000x3000 images so it should not be a problem.

For general computer vision problems, OpenCV’s Caffe model of the DNN module is the best. It works well with occlusion, quick head movements, and can identify side faces as well. Moreover, it also gave the quickest fps among all. You can perform *fast, accurate* face detection with OpenCV using a pre-trained deep learning face detector model shipped with the library.

Back in August 2017, OpenCV 3.3 was officially released, bringing it with it a highly improved “deep neural networks” (dnn ) module. This module supports a number of deep learning frameworks, including Caffe, TensorFlow, and Torch/PyTorch. OpenCV’s deep learning face detector is based on the Single Shot Detector (SSD) framework with a ResNet base network (unlike other OpenCV SSDs that you may have seen which typically use MobileNet as the base network).

The dnn.blobFromImage takes care of pre-processing which includes setting the blob dimensions and normalization.

Object detectors are not perfect so you are bound to see some false-positives. The SSD algorithm works (at a very simplistic level) by dividing your image into boxes and classifying each of them, class-wise. Since your face most of the frame being close up to the camera, there are likely a large number of boxes that contain face-like regions. This would imply why you may see a detection adjacent to the real one.

“The model was trained in Caffe framework on some huge and available online dataset.” I asked him and it’s the WIDER face dataset. He blurred small <30px faces.

Does this algorithm do non-max suppression as well? Yes, the algorithm is internally doing NMS.

You can use the cv2.imwrite function to write images to disk and cv2.VideoWriter to write videos to disk.

(104.0, 177.0, 123.0) They are the mean RGB values across all pixels in the training set. We use them to perform mean subtraction.

output of net.forward(): The documentation says it’s a “blob for first output of specified layer,” (if I am reading things correctly), however, it’s clearly different than the blob I give it. The shape of the output is something like [1, 1, 200, 7] where the third value seems to be the only thing that changes depending on the image — I think that’s the number of “detected faces” based on your tutorial. My input blob, on the other hand is [num\_images, num\_channels, width, height].

The detections.shape[2] is the number of detected objects.

The file “res10\_300x300\_ssd\_iter\_140000.caffemodel” is a raw binary file that is generated by Caffe and then read by OpenCV.

Detection array has shape (1, 1, 108, 7). You have already explained definition of last dimension. What first three dimensions signify? Can you please point me to definition of this structure? The 7 values returned are:

1. Batch ID

2. Class ID

3. Confidence

4-7. Left, top, right, bottom

Unfortunately OpenCV does not yet support NVIDIA GPUs for their “dnn” module. They should hopefully be supporting it soon but currently they do not.

Facial landmarks detector

http://dlib.net/files/shape\_predictor\_68\_face\_landmarks.dat.bz2"

Face detection

<https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c>

<https://github.com/opencv/opencv/tree/master/samples/dnn>

<https://pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/>

<https://pyimagesearch.com/2021/04/26/face-detection-tips-suggestions-and-best-practices/>

<https://pyimagesearch.com/2017/11/06/deep-learning-opencvs-blobfromimage-works/>

<https://github.com/opencv/opencv/blob/master/samples/dnn/face_detector/how_to_train_face_detector.txt>

Face recognition

<https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/>

<https://pyimagesearch.com/2018/09/24/opencv-face-recognition/>

<https://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf>

<https://cmusatyalab.github.io/openface/models-and-accuracies/>

<https://pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/>

Face landmarks

<https://yinguobing.com/facial-landmark-localization-by-deep-learning-background/>

<https://github.com/yinguobing/cnn-facial-landmark>

<https://towardsdatascience.com/robust-facial-landmarks-for-occluded-angled-faces-925e465cbf2e>

<https://github.com/yinguobing/head-pose-estimation/tree/master/assets/pose_model>

<https://github.com/faust690226/cnn-facial-landmark-tutorial>

<https://www.academia.edu/35372127/Approaching_human_level_facial_landmark_localization_by_deep_learning>

<https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/>

Head pose

<https://github.com/yinguobing/head-pose-estimation>

<https://medium.com/analytics-vidhya/real-time-head-pose-estimation-with-opencv-and-dlib-e8dc10d62078>

<https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/#disqus_thread>

<https://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html>

<https://github.com/niconielsen32/ComputerVision/blob/master/headPoseEstimation.py>

<https://www.youtube.com/watch?v=-toNMaS4SeQ>

People detector

<https://pyimagesearch.com/2018/08/13/opencv-people-counter/>

<https://viso.ai/deep-learning/object-detection/>

<https://towardsdatascience.com/picking-fast-people-detector-working-with-opencv-on-cpu-in-2021-ff8d752088af>

<https://vidishmehta204.medium.com/object-detection-using-ssd-mobilenet-v2-7ff3543d738d>

<https://arxiv.org/abs/1512.02325>

<https://blog.roboflow.com/training-a-tensorflow-object-detection-model-with-a-custom-dataset/>

<https://github.com/chuanqi305/MobileNet-SSD>

<https://github.com/weiliu89/caffe/tree/ssd>

<https://pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/>

Eyes tracking

<https://medium.com/@stepanfilonov/tracking-your-eyes-with-python-3952e66194a6>

<https://stackoverflow.com/questions/61016954/controlling-contrast-and-brightness-of-video-stream-in-opencv-and-python>

<https://github.com/antoinelame/GazeTracking>

<https://stackoverflow.com/questions/3490727/what-are-some-methods-to-analyze-image-brightness-using-python>

Liveness

<https://pyimagesearch.com/2019/03/11/liveness-detection-with-opencv/>

<https://medium.com/codex/fake-webcam-for-your-online-meetings-with-python-755556d7667b>

<https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/>