PYIMAGESEA
RCH

FACE APPLICATIONS (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/FACES/)

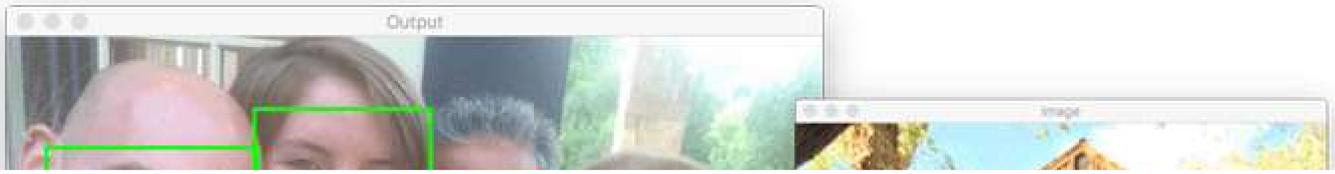
OPENCV TUTORIALS (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/OPENCV/)

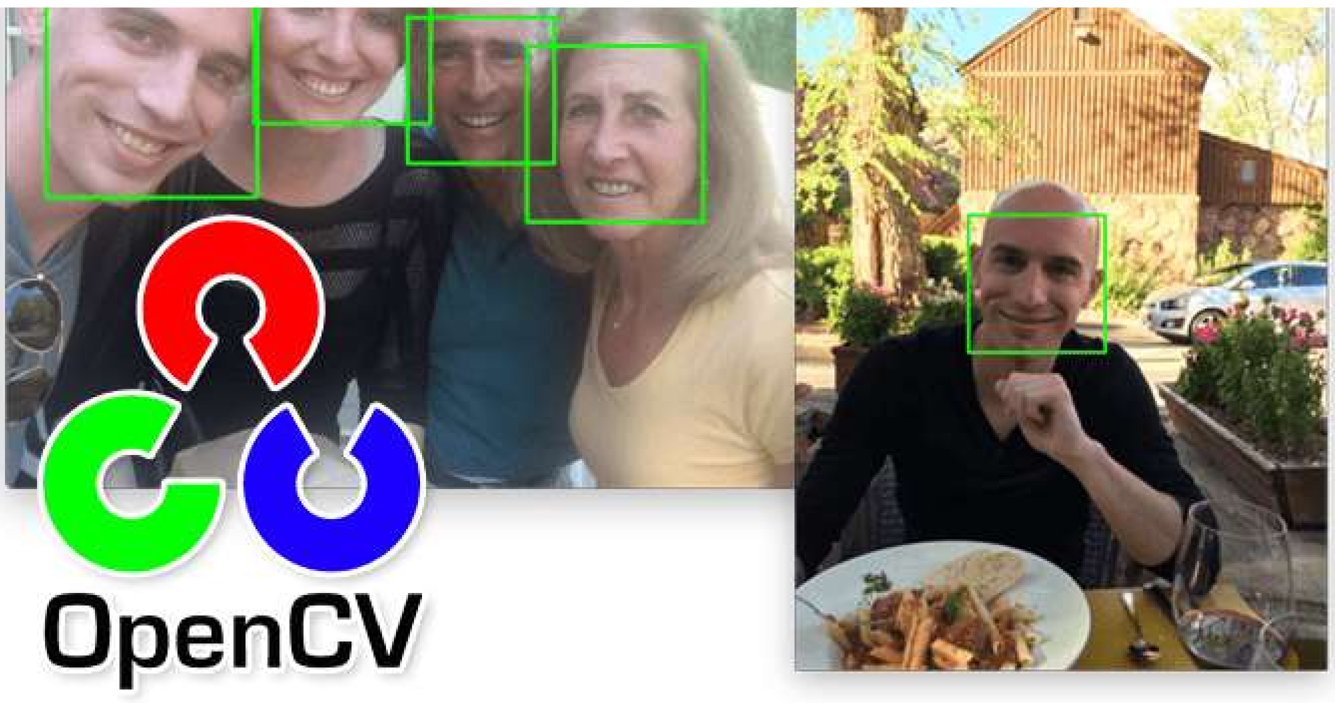
TUTORIALS (HTTPS://WWW.PYIMAGESEARCH.COM/CATEGORY/TUTORIALS/)

Face detection tips, suggestions, and best practices

by **Adrian Rosebrock (https://www.pyimagesearch.com/author/adrian/)** *on* April 26, 2021

e detection accuracy with OpenCV and dlib.





We’ve covered face detection four times on the PyImageSearch blog:

1. Face detection with OpenCV and Haar cascades (https://www.pyimagesearch.com/2021/04/05/opencv-face-detection-with-haar-cascades/)
2. Face detection with OpenCV and deep neural networks (DNNs) (https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/)
3. Face detection with dlib and the HOG + Linear SVM algorithm (https://www.pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/)
4. Face detection with dlib and the max-margin object detector (MMOD)

(https://www.pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/)

Note: *#3 and #4 link to the same tutorial as the guide covers* ***both*** *HOG + Linear SVM and the MMOD CNN face detector.*

Today we'll compare and contrast each of these methods, giving you a good idea of when you should be using each, allowing you to balance speed, accuracy, and efficiency.

To learn my face detection tips, suggestions, and best practices, *just keep reading.*

Face detection tips, suggestions, and best practices

In the first part of this tutorial, we'll recap the four primary face detectors you'll encounter when building your own computer vision pipelines, including:

1. OpenCV and Haar cascades
2. OpenCV's deep learning-based face detector
3. Dlib's HOG + Linear SVM implementation
4. Dlib's CNN face detector

We'll then compare and contrast each of these methods. Additionally, I'll give you the pros and cons for each, along with my personal recommendation on when you should be using a given face detector.

I'll wrap up this tutorial with my recommendation for a "default, all-purpose" face detector that should be your "first try" when building your own computer vision projects that require face detection.

4 popular face detection methods you'll often use in your computer vision projects

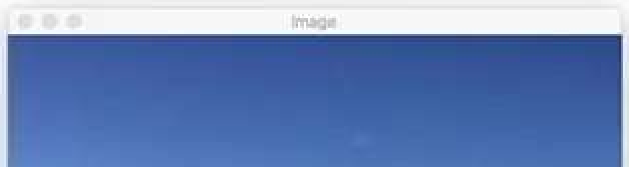
There are four primary face detection methods that we've covered on the PyImageSearch blog:

1. **OpenCV and Haar cascades (https://www.pyimagesearch.com/2021/04/05/opencv-face-detection-with-haar-cascades/)**
2. **OpenCV's deep learning-based face detector (https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/)**
3. **Dlib's HOG + Linear SVM implementation (https://www.pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/)**
4. **Dlib's CNN face detector (https://www.pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/)**

Note: *#3 and #4 link to the same tutorial as the guide covers* ***both*** *HOG + Linear SVM and the MMOD CNN face detector.*

Before continuing, I suggest you review each of those posts individually so you can better appreciate the compare/contrast we’re about to perform.

Pros and cons of OpenCV’s Haar cascade face detector



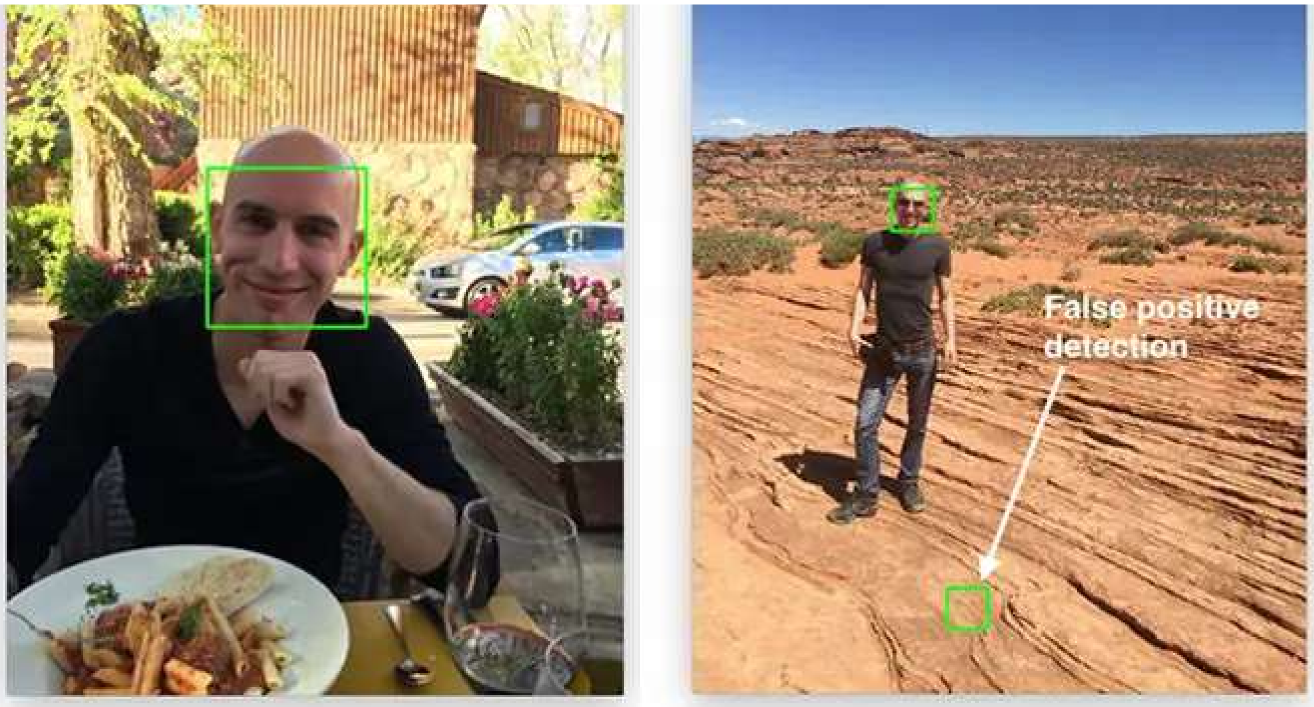


Figure 1: OpenCV's Haar cascade face detector is very fast but prone to false-positive detections.

OpenCV's Haar cascade face detector (https://www.pyimagesearch.com/2021/04/05/opencv-face-detection-with-haar-cascades/) is the original face detector that shipped with the library. It's also the face detector that is familiar to most everyone.

Pros:

- *Very fast*, capable of running in super real-time

- Low computational requirements — can *easily* be run on embedded, resource-constrained devices such as the Raspberry Pi (RPi), NVIDIA Jetson Nano, and Google Coral
- Small model size (just over 400KB; for reference, most deep neural networks will be anywhere between 20-200MB).

Cons:

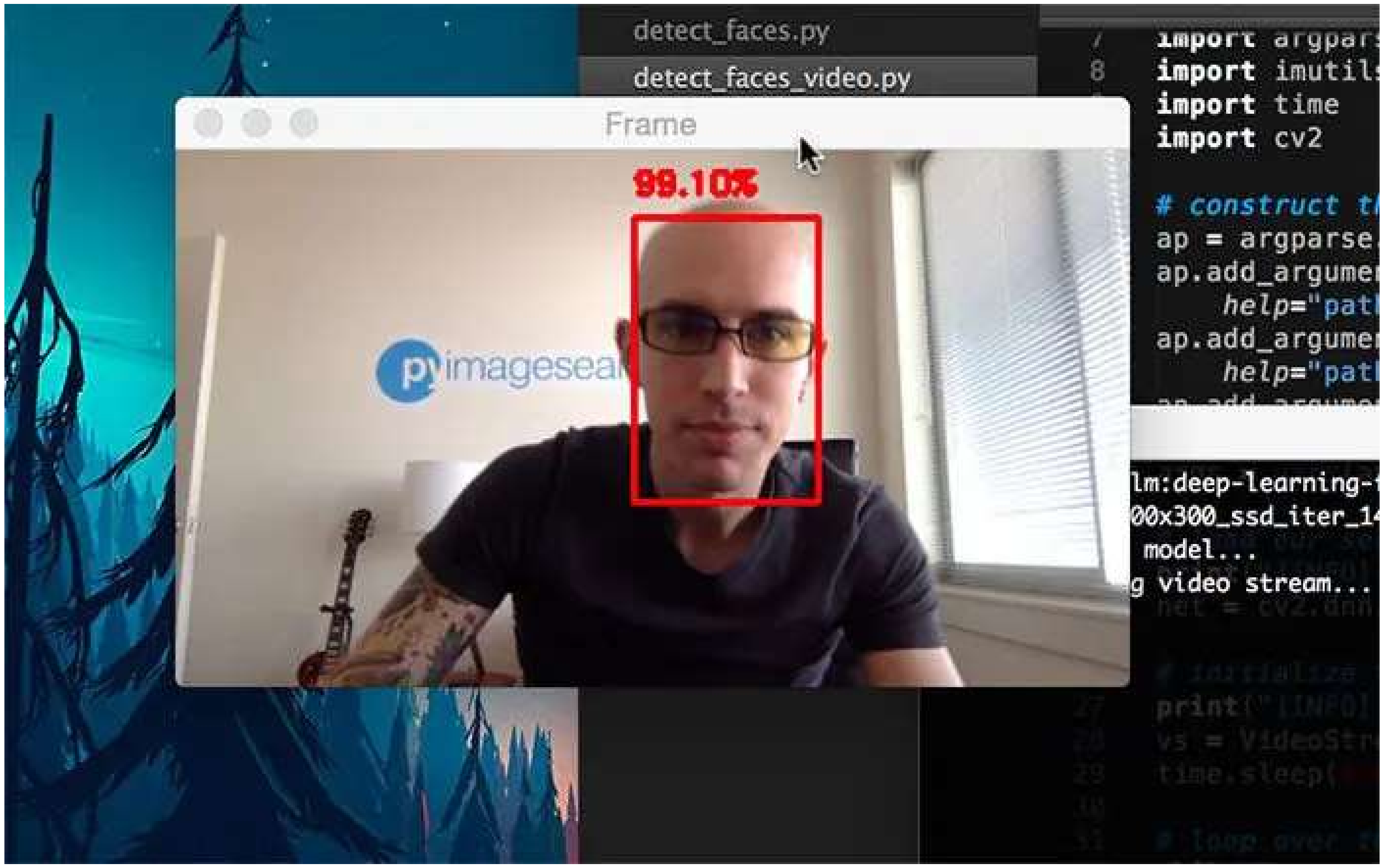
- Highly prone to false-positive detections
- Typically requires manual tuning to the `detectMultiScale` function
- Not anywhere near as accurate as its HOG + Linear SVM and deep learning-based face detection counterparts

My recommendation: Use Haar cascades when speed is your primary concern, and you're willing to sacrifice some accuracy to obtain real-time performance.

If you're working on an embedded device like the RPi, Jetson Nano, or Google Coral, consider:

- Using the Movidius Neural Compute Stick (NCS) on the RPi — that will allow you to run deep learning-based face detectors in real-time
- Reading the documentation associated with your device — the Nano and Coral have specialized inference engines that can run deep neural networks in real-time

Pros and cons of OpenCV's deep learning face detector



```
import argpar
import imutil
import time
import cv2

# construct t
ap = argparse
ap.add_argume
    help="pat
ap.add_argume
    help="pat
```

```
lm:deep-learning-
00x300_ssd_iter_14
 model...
g video stream...
```

Figure 2: OpenCV's deep learning SSD face detector is both fast and accurate, capable of running in real-time on modern laptop/desktop CPUs.

OpenCV's deep learning face detector (https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/) is based on a Single Shot Detector (SSD) with a small ResNet backbone, allowing it to be both ***accurate*** and ***fast.***

Pros:

- Accurate face detector
- Utilizes modern deep learning algorithms
- No parameter tuning required
- Can run in real-time on modern laptops and desktops
- Model is reasonably sized (just over 10MB)
- Relies on OpenCV's `cv2.dnn` module
- Can be made faster on embedded devices by using OpenVINO and the Movidius NCS

Cons:

- More accurate than Haar cascades and HOG + Linear SVM, but not as accurate as dlib's CNN MMOD face detector
- May have unconscious biases in the training set — may not detect darker-skinned people as accurately as lighter-skinned people

My recommendation: OpenCV's deep learning face detector is your best "all-around" detector. It's very simple to use, doesn't require additional libraries, and relies on OpenCV's `cv2.dnn` module, which is baked into the OpenCV library.

Furthermore, if you are using an embedded device, such as the Raspberry Pi, you can plug in a Movidius NCS and utilize OpenVINO to *easily* obtain real-time performance.

Perhaps the biggest downside of this model is that I've found that the face detections on darker-skinned people aren't as accurate as lighter-skinned people. That's not necessarily a problem with the model itself but rather the data it was trained on — to remedy that problem, I suggest training/fine-tune the face detector on a more diverse set of ethnicities.

Pros and cons of dlib's HOG + Linear SVM face detector

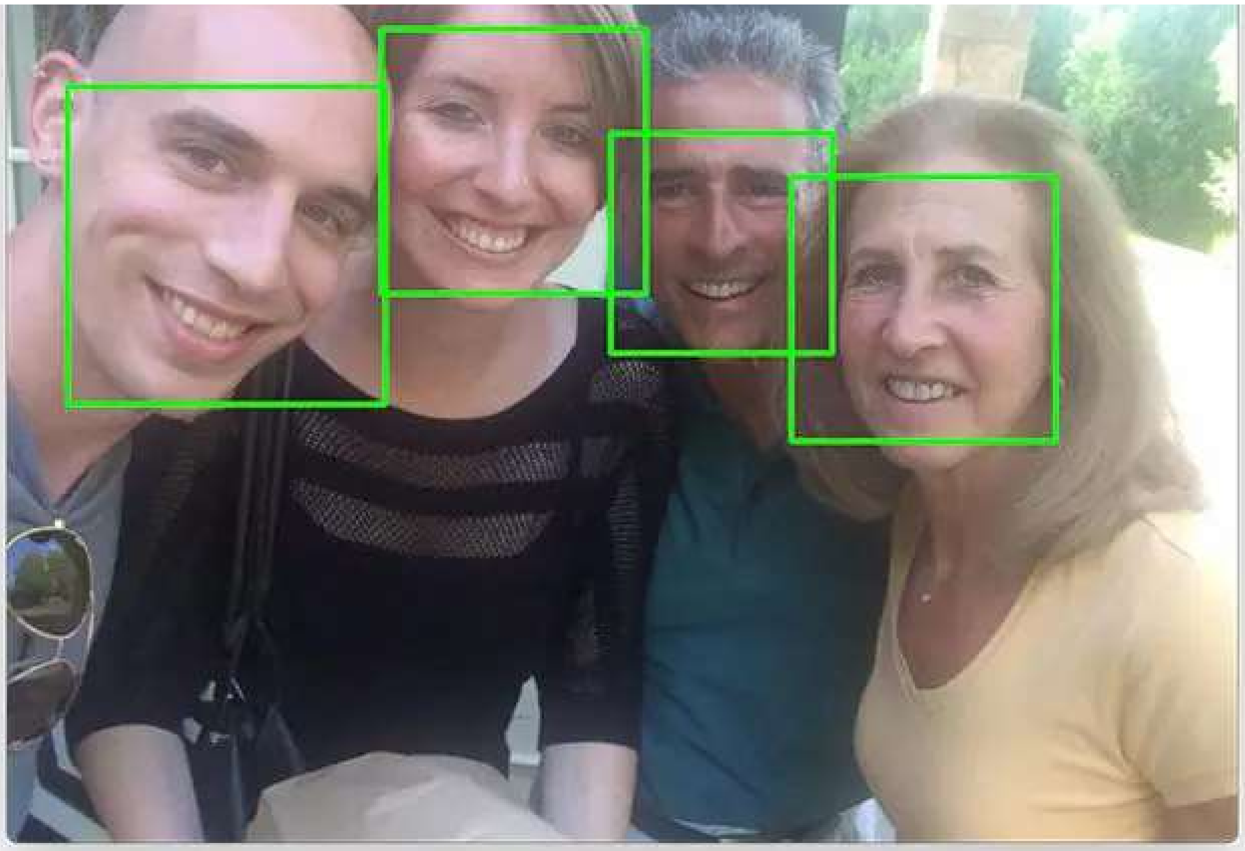


Figure 3: HOG + Linear SVM is a classic algorithm in the object detection/face detection literature. Use it when you need more accuracy than Haar cascades but cannot commit to the computational complexity of deep learning-based detectors.

The **HOG + Linear SVM algorithm (https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/)** was first introduced by Dalal and Triggs in their seminal 2005 work, ***Histograms of Oriented Gradients for Human Detection* (https://hal.inria.fr/inria-00548512/document)**.

Similar to Haar cascades, HOG + Linear SVM relies on image pyramids and sliding windows to detect objects/faces in an image.

The algorithm is a classic in computer vision literature and is still used today.

Pros:

- More accurate than Haar cascades
- More stable detection than Haar cascades (i.e., fewer parameters to tune)
- Expertly implemented by dlib creator and maintainer, Davis King
- *Extremely* well documented, both in terms of the dlib implementation and the HOG + Linear SVM framework in the computer vision literature

Cons:

- Only works on frontal views of the face — profile faces will *not* be detected as the HOG descriptor does not tolerate changes in rotation or viewing angle well
- Requires an additional library (dlib) be installed — not necessarily a problem per se, but if you're using *just* OpenCV, then you may find adding another library into the mix cumbersome
- Not as accurate as deep learning-based face detectors
- For the accuracy, it's actually quite computationally expensive due to image pyramid construction, sliding windows, and computing HOG features at every stop of the window

My recommendation: HOG + Linear SVM is a classic object detection algorithm that *every* computer vision practitioner should understand. That said, for the accuracy HOG + Linear SVM gives you, the algorithm itself is quite slow, especially when you compare it to OpenCV's SSD face detector.

I tend to use HOG + Linear SVM in places where Haar cascades aren't accurate enough, but I cannot commit to using OpenCV's deep learning face detector.

Pros and cons of dlib's CNN face detector

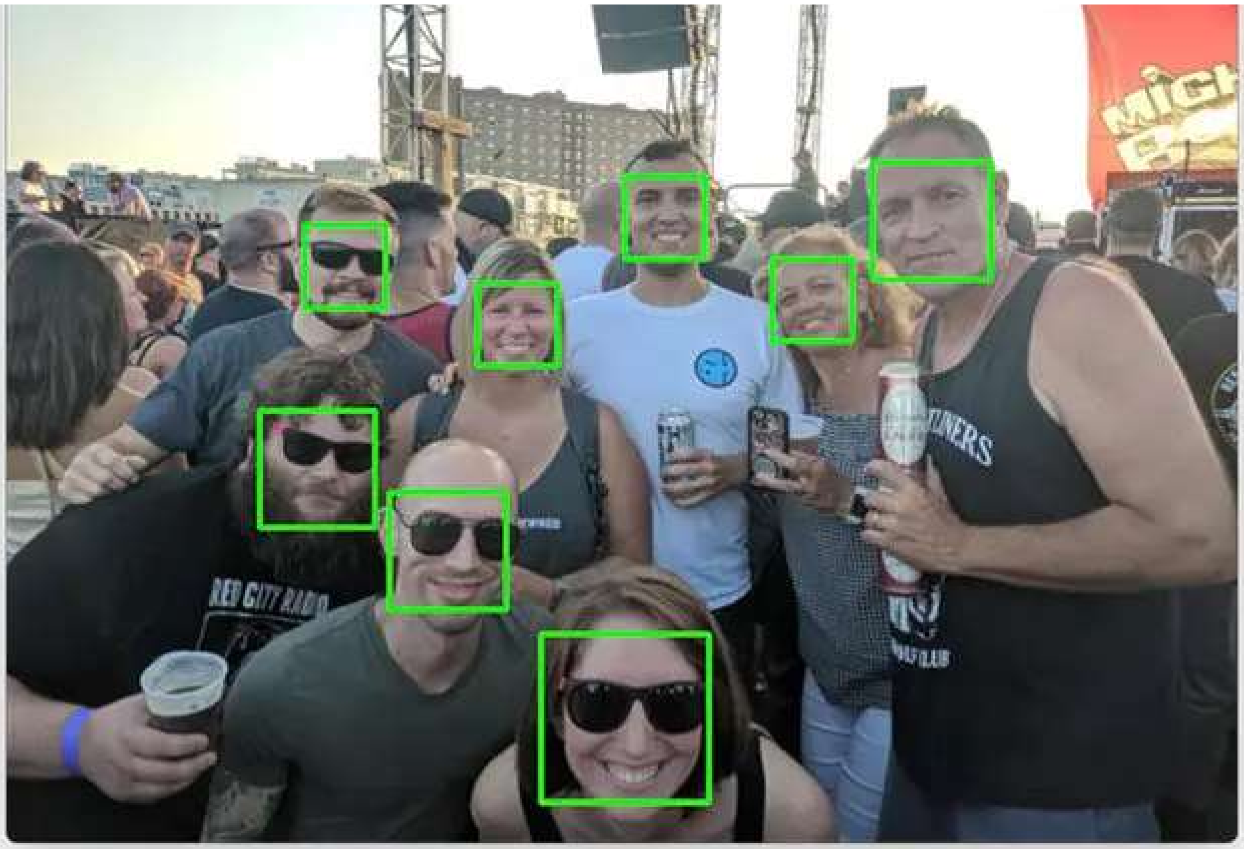


Figure 4: Dlib's CNN face detector is the most accurate of the bunch but is quite slow. Use it when you need accuracy above all else.

Davis King, the creator of dlib, trained a CNN face detector based on his work on **max-margin object detection (https://arxiv.org/abs/1502.00046)**. The method is *highly accurate*, thanks to the design of the algorithm itself, along with the care Davis took in curating the training set and training the model.

That said, without GPU acceleration, this model cannot realistically run in real-time.

Pros:

- ***Incredibly accurate*** face detector
- Small model size (under 1MB)
- Expertly implemented and documented

Cons:

- Requires an additional library (dlib) be installed
- Code is more verbose — end-user must take care to convert and trim bounding box coordinates if using OpenCV
- Cannot run in real-time without GPU acceleration
- Not out-of-the-box compatible for acceleration via OpenVINO, Movidius NCS, NVIDIA Jetson Nano, or Google Coral

My recommendation: I tend to use dlib's MMOD CNN face detector when batch processing face detection offline, meaning that I can set up my script and let it run in batch mode without worrying about real-time performance.

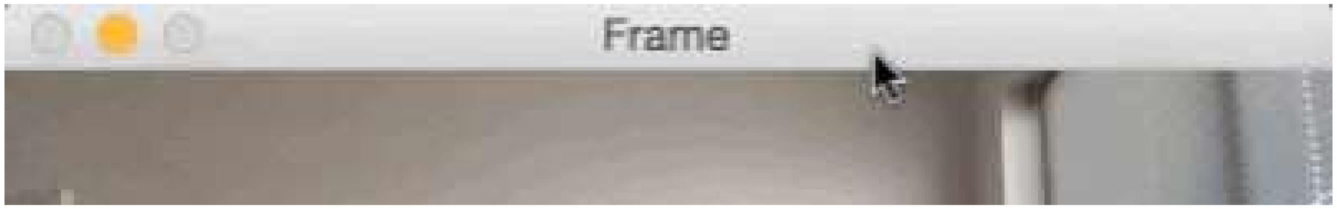
In fact, when I build **training sets for face recognition (https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/)**, I often use dlib's CNN face detector to detect faces before training the face recognizer itself. When I'm ready to deploy my face recognition model, I'll often swap out dlib's CNN face detector for a more computationally efficient one that can run in real-time (e.g., OpenCV's CNN face detector).

face detector for a more computationally efficient one that can run in real-time (e.g., OpenCV's CNN face detector).

The only place I tend *not* to use dlib's CNN face detector is when I'm using embedded devices. This model will not run in real-time on embedded devices, and it's out-of-the-box compatible with embedded device accelerators like the Movidius NCS.

That said, *you just cannot beat the face detection accuracy* of dlib's MMOD CNN, so if you need accurate face detections, *go with this model.*

My personal suggestions for face detection



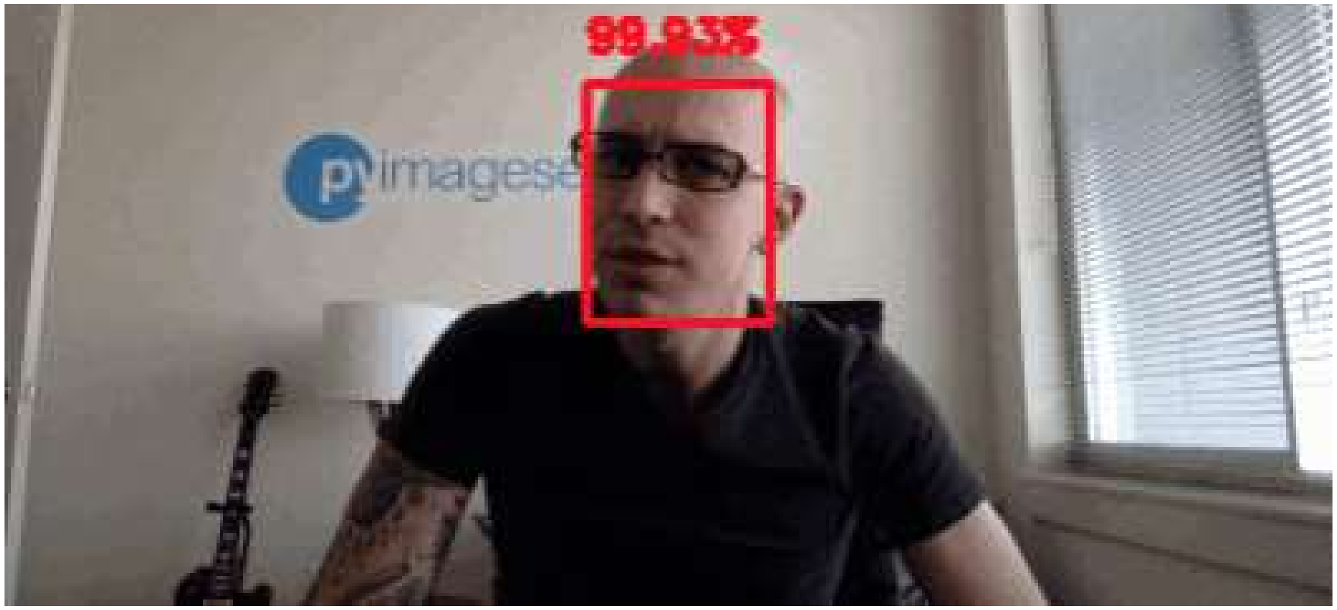


Figure 5: For a good all-around face detector, go with OpenCV's deep learning-based face detector. It's accurate and capable of running in real-time on modern laptops and desktops.

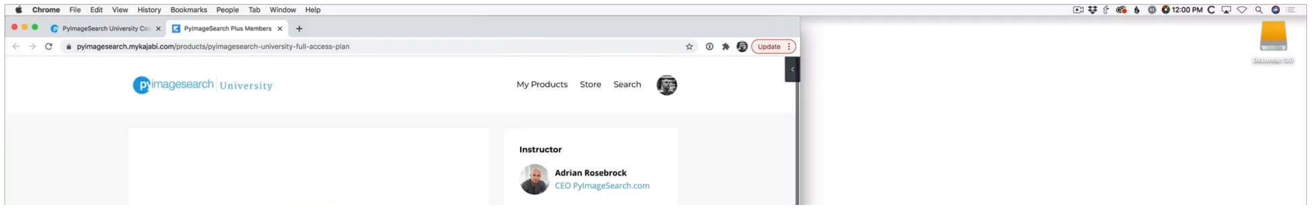
When it comes to a good, all-purpose face detector, I suggest using OpenCV's DNN face detector (https://www.pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/):

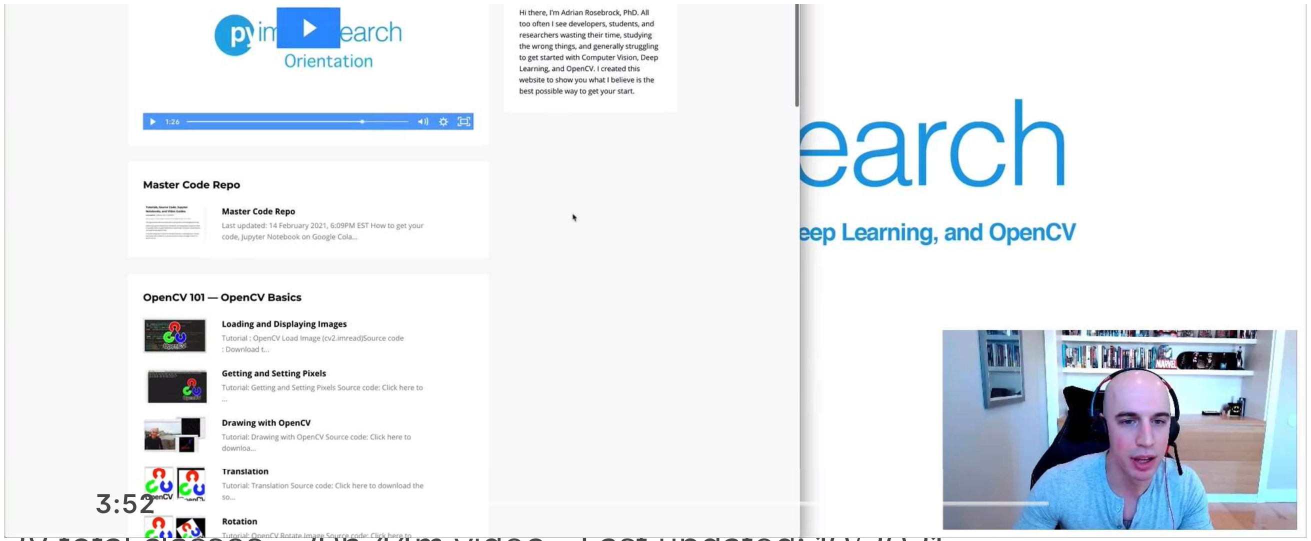
- It achieves a nice balance of *speed* and *accuracy*
- As a deep learning-based detector, it's *more accurate* than its Haar cascade and HOG + Linear SVM counterparts
- It's fast enough to run real-time on CPUs
- It can be further accelerated using USB devices such as the Movidius NCS
- No additional libraries/packages are required — support for the face detector is baked into OpenCV via the

- No additional libraries/packages are required — support for the face detector is baked into OpenCV via the `cv2.dnn` module

That said, there are times when you would want to use each of the face detectors mentioned above, so be sure to read through each of those sections *carefully*.

What's next? I recommend PyImageSearch University (https://www.pyimagesearch.com/pyimagesearch-university/?utm_source=blogPost&utm_medium=bottomBanner&utm_campaign=What%27s%20next%3F%20I%20recommend).


Chrome File Edit View History Bookmarks People Tab Window Help
PyImageSearch University
PyImageSearch Plus Members
pyimagesearch.mykajabi.com/products/pyimagesearch-university-full-access-plan
Update
pyimagesearch University
My Products Store Search
Instructor
Adrian Rosebrock
CEO PyImageSearch.com


pyimagesearch
Orientation
1:26
Master Code Repo
Master Code Repo
Last updated: 14 February 2021, 6:09PM EST How to get your code, Jupyter Notebook on Google Cola...
OpenCV 101 — OpenCV Basics
Loading and Displaying Images
Tutorial : OpenCV Load Image (cv2.imread)Source code : Download t...
Getting and Setting Pixels
Tutorial: Getting and Setting Pixels Source code: Click here to ...
Drawing with OpenCV
Tutorial: Drawing with OpenCV Source code: Click here to downloa...
Translation
Tutorial: Translation Source code: Click here to download the so...
Rotation
earch
eep Learning, and OpenCV
3:52

Hi there, I'm Adrian Rosebrock, PhD. All too often I see developers, students, and researchers wasting their time, studying the wrong things, and generally struggling to get started with Computer Vision, Deep Learning, and OpenCV. I created this website to show you what I believe is the best possible way to get your start.

28 total classes • 39h 44m video • Last updated: 10/2021

★★★★★ 4.84 (128 Ratings) • 3,000+ Students Enrolled

I strongly believe that if you had the right teacher you could *master* computer vision and deep learning.

Do you think learning computer vision and deep learning has to be time-consuming, overwhelming, and complicated? Or has to involve complex mathematics and equations? Or requires a degree in computer science?

That's *not* the case.

All you need to master computer vision and deep learning is for someone to explain things to you in *simple, intuitive* terms. *And that's exactly what I do*. My mission is to change education and how complex Artificial Intelligence topics are taught.

If you're serious about learning computer vision, your next stop should be PyImageSearch University, the most comprehensive computer vision, deep learning, and OpenCV course online today. Here you'll learn how to *successfully* and *confidently* apply computer vision to your work, research, and projects. Join me in computer vision mastery.

Inside PyImageSearch University you'll find:

✓ **28 courses** on essential computer vision, deep learning, and OpenCV topics

✓ 28 Certificates of Completion

✓ **39h 44m** on-demand video

✓ **Brand new courses released *every month***, ensuring you can keep up with state-of-the-art techniques

✓ **Pre-configured Jupyter Notebooks in Google Colab**

✓ Run all code examples in your web browser — works on Windows, macOS, and Linux (no dev environment configuration required!)

✓ Access to **centralized code repos for *all* 400+ tutorials** on PyImageSearch

✓ **Easy one-click downloads** for code, datasets, pre-trained models, etc.

✓ Access on mobile, laptop, desktop, etc.

CLICK HERE TO JOIN PYIMAGESEARCH UNIVERSITY (HTTPS://WWW.PYIMAGESEARCH.COM/PYIMAGESEARCH-UNIVERSITY/?UTM_SOURCE=BLOGPOST&UTM_MEDIUM=BOTTOMBANNER&UTM_CAMPAIGN=WHAT%27S%20NEXT%3F%20I%20RECOMMEND)

Summary

In this tutorial, you learned my tips, suggestions, and best practices for face detection.

In summary, they are:

1. **Use OpenCV's Haar cascades when speed is your primary concern (e.g., when you're using an embedded device like the Raspberry Pi).** Haar cascades aren't as accurate as their HOG + Linear SVM and deep learning-based counterparts, but they make up for it in raw speed. Just be aware there will *certainly* be some false-positive detections and parameter tuning required when calling `detectMultiScale`.

2. **Use dlib's HOG + Linear SVM detector when Haar cascades are not accurate enough, but you cannot commit to the computational requirements of a deep learning-based face detector.** The HOG + Linear SVM object detector is a classic algorithm in the computer vision literature that is still relevant today. The dlib library

does a *fantastic job* implementing it. Just be aware that running HOG + Linear SVM on a CPU will likely be too slow for your embedded device.

3. **Use dlib's CNN face detection when you need super-accurate face detections.** When it comes to face detection accuracy, dlib's MMOD CNN face detector is *incredibly accurate.* That said, there is a tradeoff — with higher accuracy comes slower run-time. This method *cannot* run in real-time on a laptop/desktop CPU, and even with GPU acceleration, you'll struggle to hit real-time performance. I typically use this face detector on offline batch processing where I'm less concerned about how long face detection takes (and instead, all I want is high accuracy).

4. **Use OpenCV's DNN face detector as a good balance.** As a deep learning-based face detector, this method is accurate — and since it's a shallow network with an SSD backbone, it's easily capable of running in real-time on a CPU. Furthermore, since you can use the model with OpenCV's `cv2.dnn` module, that *also* implies that (1) you can increase speed further by using a GPU or (2) utilizing the Movidius NCS on your embedded device.

In general, OpenCV's DNN face detector should be your "first stop" when applying face detection. You can try other methods based on the accuracy the OpenCV DNN face detector gives you.

To download the source code to this post (and be notified when future tutorials are published here on PyImageSearch), ***simply enter your email address in the form below!***

About the Author

Hi there, I’m Adrian Rosebrock, PhD. All too often I see developers, students, and researchers wasting their time, studying the wrong things, and generally struggling to get started with Computer Vision, Deep Learning, and OpenCV. I created this website to show you what I believe is the best possible way to get your start.

Previous Article:

Face detection with dlib (HOG and CNN)

(https://www.pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/)

Next Article:

OpenCV Morphological Operations

(https://www.pyimagesearch.com/2021/04/28/opencv-morphological-operations/)

Comment section

Hey, Adrian Rosebrock here, author and creator of PyImageSearch. While I love hearing from readers, a couple years ago I made the tough decision to no longer offer 1:1 help over blog post comments.

At the time I was receiving 200+ emails per day and another 100+ blog post comments. I simply did not have the time to moderate and respond to them all, and the sheer volume of requests was taking a toll on me.

Instead, my goal is to *do the most good* for the computer vision, deep learning, and OpenCV community at large by focusing my time on authoring high-quality blog posts, tutorials, and books/courses.

If you need help learning computer vision and deep learning, I suggest you refer to my full catalog of books

If you need help learning computer vision and deep learning, I suggest you refer to my full catalog of books and courses (https://www.pyimagesearch.com/books-and-courses/) — they have helped tens of thousands of developers, students, and researchers *just like yourself* learn Computer Vision, Deep Learning, and OpenCV.

Click here to browse my full catalog. (https://www.pyimagesearch.com/books-and-courses/)

Similar articles

LIBRARIES TUTORIALS

How to find functions by name in OpenCV

August 31, 2015

(https://www.pyimagesearch.com/2015/08/31/how-to-find-functions-by-name-in-opencv/) →

DEEP LEARNING KERAS AND TENSORFLOW TUTORIALS

Keras Tutorial: How to get started with Keras, Deep Learning, and Python

September 10, 2018

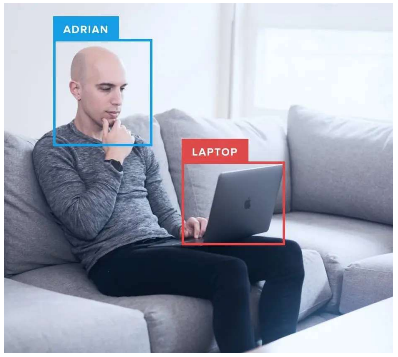
(https://www.pyimagesearch.com/2018/09/10/keras-tutorial-how-to-get-started-with-keras-deep-learning-and-python/) →

DEEP LEARNING TUTORIALS

Neural Style Transfer with OpenCV

August 27, 2018

(https://www.pyimagesearch.com/2018/08/27/neural-style-transfer-with-opencv/) →



You can learn Computer Vision, Deep Learning, and OpenCV.

Get your FREE 17 page Computer Vision, OpenCV, and Deep Learning Resource Guide PDF. Inside you'll find our hand-picked tutorials, books, courses, and libraries to help you master CV and DL.

Topics

Deep Learning (https://www.pyimagesearch.com/category/deep-learning-2/)

Machine Learning and Computer Vision (https://www.pyimagesearch.com/category/machine-learning-2/)

Medical Computer Vision (https://www.pyimagesearch.com/category/medical/)

Dlib Library (https://www.pyimagesearch.com/category/dlib/)

Embedded/IoT and Computer Vision (https://www.pyimagesearch.com/category/embedded/)

Face Applications (https://www.pyimagesearch.com/category/faces/)

Image Processing (https://www.pyimagesearch.com/category/image-processing/)

Interviews (https://www.pyimagesearch.com/category/interviews/)

Keras (https://www.pyimagesearch.com/category/keras/)

Optical Character Recognition (OCR) (https://www.pyimagesearch.com/category/optical-character-recognition-ocr/)

Object Detection (https://www.pyimagesearch.com/category/object-detection/)

Object Tracking (https://www.pyimagesearch.com/category/object-tracking/)

OpenCV Tutorials (https://www.pyimagesearch.com/category/opencv/)

Raspberry Pi (https://www.pyimagesearch.com/category/raspberry-pi/)

Books & Courses

FREE CV, DL, and OpenCV Crash Course (https://www.pyimagesearch.com/free-opencv-computer-vision-deep-learning-crash-course/)
Practical Python and OpenCV (https://www.pyimagesearch.com/practical-python-opencv/)

Deep Learning for Computer Vision with Python (https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/)

PyImageSearch Gurus Course (https://www.pyimagesearch.com/pyimagesearch-gurus/)

PyImageSearch

Get Started (https://www.pyimagesearch.com/start-here/)

OpenCV Install Guides (https://www.pyimagesearch.com/opencv-tutorials-resources-guides/)

About (https://www.pyimagesearch.com/about/)

FAQ (https://www.pyimagesearch.com/faqs/)

Blog (https://www.pyimagesearch.com/topics/)

Contact (https://www.pyimagesearch.com/contact/)

Raspberry Pi for Computer Vision (https://www.pyimagesearch.com/raspberry-pi-for-computer-vision/)

Privacy Policy (https://www.pyimagesearch.com/privacy-policy/)

(https://www.facebook.com/pyimagesearch) **(https://twitter.com/PyImageSearch)** **(http://www.linkedin.com/pub/adrian-rosebrock/2a/873/59b)** **(https://www.youtube.com/channel/UCoQK7OVcIVy-nV4m-SMCk_Q/videos)**