face-recognition 1.3.0

✓ Latest version

`pip install face-recognition`

Released: Feb 20, 2020

Recognize faces from Python or from the command line

Navigation

Project description

Release history

Download files

Project links

Homepage

Statistics

Project description

Face Recognition

Recognize and manipulate faces from Python or from the command line with
the world's simplest face recognition library.
Built using dlib's state-of-the-art face recognition
built with deep learning. The model has an accuracy of 99.38% on the
Labeled Faces in the Wild benchmark.
This also provides a simple `face_recognition` command line tool that lets
you do face recognition on a folder of images from the command line!

pypi v1.3.0

build passing

docs passing

Features

Forks: 10.292

Open issues/PRs: 541

View statistics for this project via Libraries.io, or by using our public dataset on Google BigQuery

Meta

License: MIT License (MIT license)

Author: Adam Geitgey

face_recognition

Maintainers

ageitgey

Classifiers

Development Status

- 4 - Beta

Intended Audience

- Developers

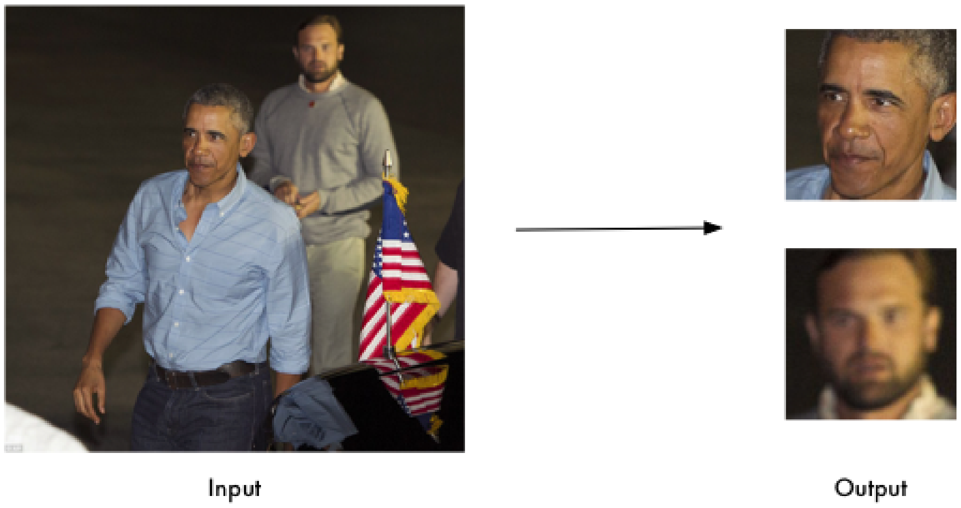
License

- OSI Approved :: MIT License

Natural Language

- English

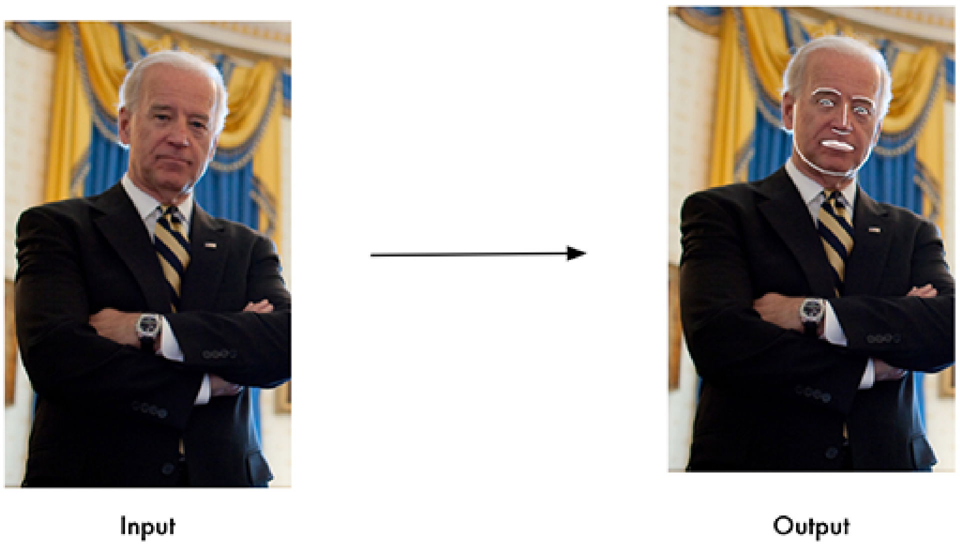
Find all the faces that appear in a picture:



```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg"
face_locations = face_recognition.face_locations(image)
```

Find and manipulate facial features in pictures

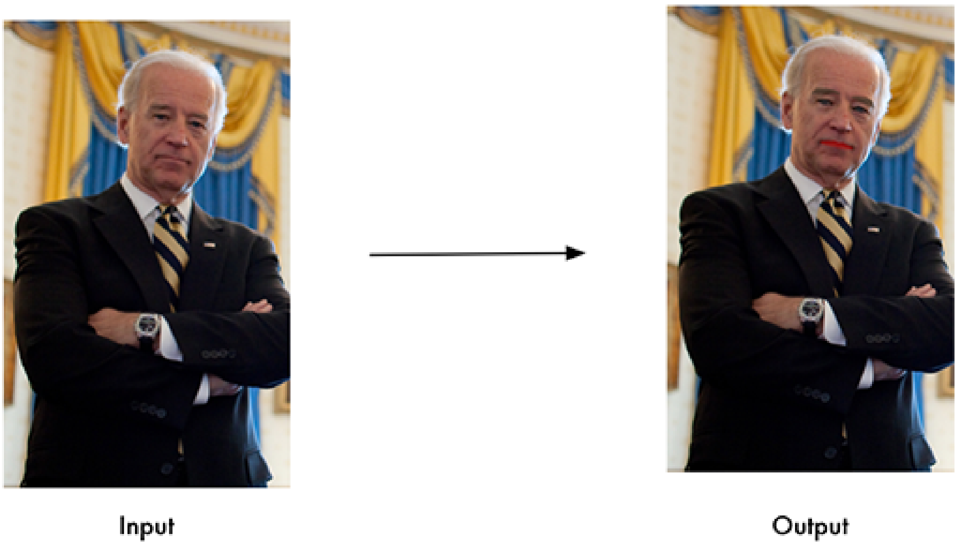
Get the locations and outlines of each person’s eyes, nose, mouth and chin.



- Python :: 2.7
- Python :: 3
- Python :: 3.5
- Python :: 3.6
- Python :: 3.7
- Python :: 3.8

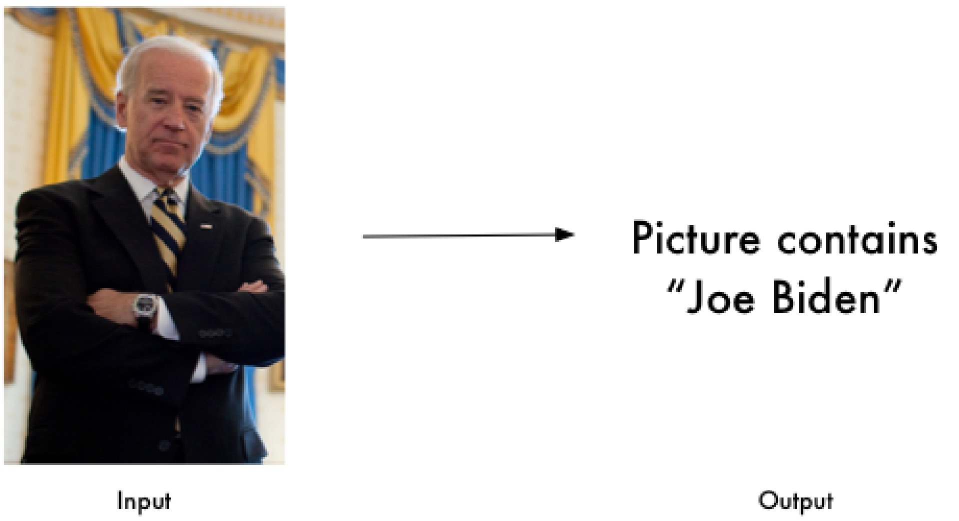
```
face_landmarks_list = face_recognition.face_landmarks(ima
```

Finding facial features is super useful for lots of important stuff. But you can also use for really stupid stuff
like applying digital make-up (think 'Meitu'):



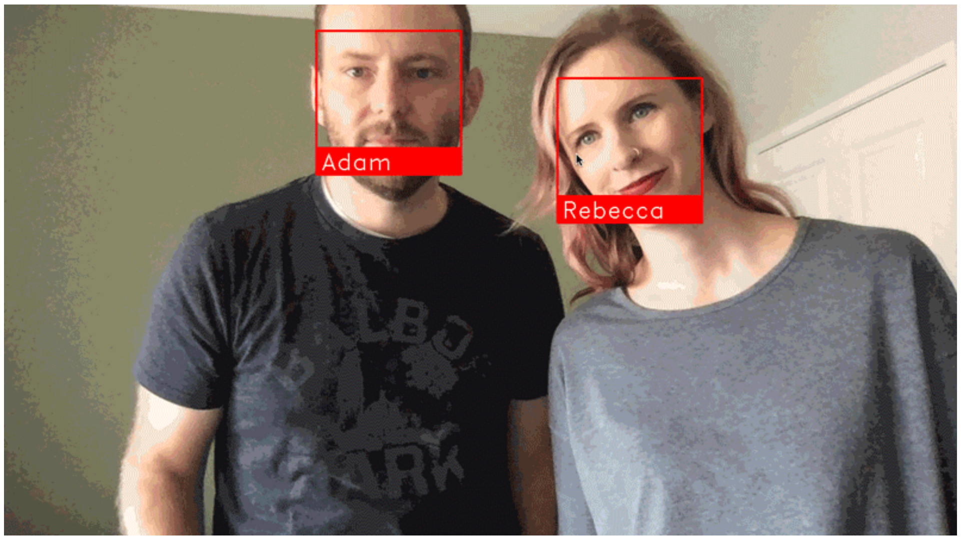
Identify faces in pictures

Recognize who appears in each photo.



```
import face_recognition
known_image = face_recognition.load_image_file("biden.jp
unknown_image = face_recognition.load_image_file("unknow
results = face_recognition.compare_faces([biden_encoding
```

You can even use this library with other Python libraries to do real-time face recognition:



See this example for the code.

Installation

Requirements

- Python 3.3+ or Python 2.7
- macOS or Linux (Windows not officially supported, but might work)

Installing on Mac or Linux

First, make sure you have dlib already installed with Python bindings:

- How to install dlib from source on macOS or Ubuntu

Then, install this module from pypi using `pip3` (or `pip2` for Python 2):

```
pip3 install face_recognition
```

Installing on Raspberry Pi 2+

- Raspberry Pi 2+ installation instructions

Installing on Windows

While Windows isn't officially supported, helpful users have posted instructions on how to install this library:

- @masoudr's Windows 10 installation guide (dlib + face_recognition)

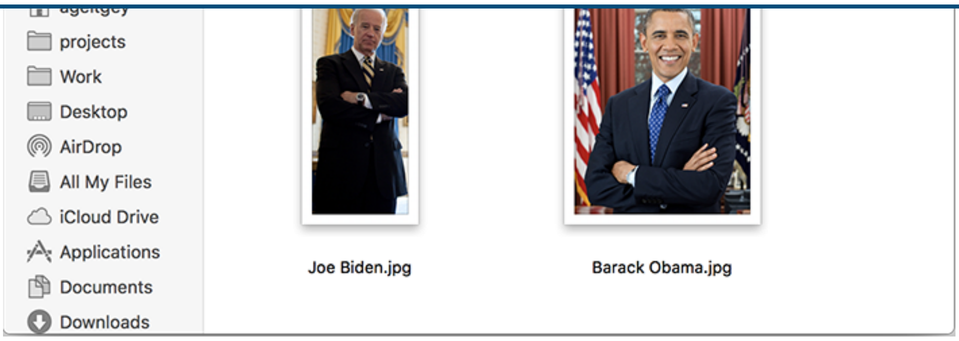
Installing a pre-configured Virtual Machine image

- Download the pre-configured VM image (for VMware Player or VirtualBox).

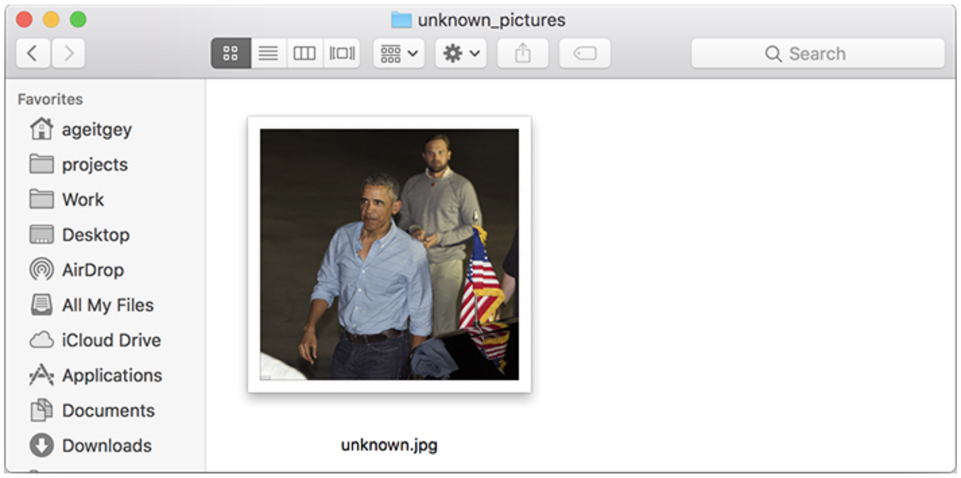
Usage

Command-Line Interface

When you install `face_recognition`, you get a simple command-line program
called `face_recognition` that you can use to recognize faces in a photograph or folder full for photographs.
First, you need to provide a folder with one picture of each person you already know. There should be one image file for each person with the files named according to who is in the picture:



Next, you need a second folder with the files you want to identify:



Then in you simply run the command `face_recognition`, passing in the folder of known people and the folder (or single image) with unknown
people and it tells you who is in each image:

```
$ face_recognition ./pictures_of_people_i_know/ ./unknowı

/unknown_pictures/unknown.jpg,Barack Obama
/face_recognition_test/unknown_pictures/unknown.jpg,unkn
```

There's one line in the output for each face. The data is comma-separated
with the filename and the name of the person found.
An `unknown_person` is a face in the image that didn't match anyone in your folder of known people.

ADJUSTING TOLERANCE / SENSITIVITY

is needed to make face comparisons more strict.
You can do that with the `--tolerance` parameter. The default tolerance
value is 0.6 and lower numbers make face comparisons more strict:

```
$ face_recognition --tolerance 0.54 ./pictures_of_people_

/unknown_pictures/unknown.jpg,Barack Obama
/face_recognition_test/unknown_pictures/unknown.jpg,unkn
```

If you want to see the face distance calculated for each match in order to adjust the tolerance setting, you can use `--show-distance true`:

```
$ face_recognition --show-distance true ./pictures_of_pe

/unknown_pictures/unknown.jpg,Barack Obama,0.37854229895
/face_recognition_test/unknown_pictures/unknown.jpg,unkn
```

MORE EXAMPLES

If you simply want to know the names of the people in each photograph but don't
care about file names, you could do this:

```
$ face_recognition ./pictures_of_people_i_know/ ./unknow

Barack Obama
unknown_person
```

SPEEDING UP FACE RECOGNITION

Face recognition can be done in parallel if you have a computer with multiple CPU cores. For example if your system has 4 CPU cores, you can all your CPU cores in parallel.

If you are using Python 3.4 or newer, pass in a `--cpus <number_of_cpu_cores_to_use>` parameter:

```
$ face_recognition --cpus 4 ./pictures_of_people_i_know/
```

You can also pass in `--cpus -1` to use all CPU cores in your system.

Python Module

You can import the `face_recognition` module and then easily manipulate
faces with just a couple of lines of code. It's super easy!

API Docs: https://face-recognition.readthedocs.io.

AUTOMATICALLY FIND ALL THE FACES IN AN IMAGE

```
import face_recognition

image = face_recognition.load_image_file("my_picture.jpg
face_locations = face_recognition.face_locations(image)

# face_locations is now an array listing the co-ordinate
```

See this example
to try it out.

You can also opt-in to a somewhat more accurate deep-learning-based face detection model.
Note: GPU acceleration (via nvidia's CUDA library) is required for good performance with this model. You'll also want to enable CUDA support when compliling `dlib`.

```
import face_recognition
```

```
# face_locations is now an array listing the co-ordinate
```

See this example
to try it out.
If you have a lot of images and a GPU, you can also
find faces in batches.

AUTOMATICALLY LOCATE THE FACIAL FEATURES OF A PERSON IN AN IMAGE

```
import face_recognition

image = face_recognition.load_image_file("my_picture.jpg
face_landmarks_list = face_recognition.face_landmarks(im

# face_landmarks_list is now an array with the locations
# face_landmarks_list[0]['left_eye'] would be the locati
```

See this example
to try it out.

RECOGNIZE FACES IN IMAGES AND IDENTIFY WHO THEY ARE

```
import face_recognition

picture_of_me = face_recognition.load_image_file("me.jpg
my_face_encoding = face_recognition.face_encodings(pictu

# my_face_encoding now contains a universal 'encoding' o

unknown_picture = face_recognition.load_image_file("unkn
unknown_face_encoding = face_recognition.face_encodings(

# Now we can see the two face encodings are of the same

results = face_recognition.compare_faces([my_face_encodi

if results[0] == True:
    print("It's a picture of me!")
```

See this example
to try it out.

Python Code Examples

All the examples are available here.

Face Detection

- Find faces in a photograph
- Find faces in a photograph (using deep learning)
- Find faces in batches of images w/ GPU (using deep learning)

Facial Features

- Identify specific facial features in a photograph
- Apply (horribly ugly) digital make-up

Facial Recognition

- Find and recognize unknown faces in a photograph based on photographs of known people
- Compare faces by numeric face distance instead of only True/False matches
- Recognize faces in live video using your webcam - Simple / Slower Version (Requires OpenCV to be installed)
- Recognize faces in live video using your webcam - Faster Version (Requires OpenCV to be installed)
- Recognize faces in a video file and write out new video file (Requires OpenCV to be installed)
- Recognize faces on a Raspberry Pi w/ camera
- Run a web service to recognize faces via HTTP (Requires Flask to be installed)
- Recognize faces with a K-nearest neighbors classifier

depending on a black box library, read my article.

Caveats

- The face recognition model is trained on adults and does not work very well on children. It tends to mix up children quite easy using the default comparison threshold of 0.6.

Deployment to Cloud Hosts (Heroku, AWS, etc)

Since `face_recognition` depends on `dlib` which is written in C++, it can be tricky to deploy an app
using it to a cloud hosting provider like Heroku or AWS.
To make things easier, there's an example Dockerfile in this repo that shows how to run an app built with
`face_recognition` in a Docker container. With that, you should be able to deploy
to any service that supports Docker images.

Common Issues

Issue: `Illegal instruction (core dumped)` when using face_recognition or running examples.
Solution: `dlib` is compiled with SSE4 or AVX support, but your CPU is too old and doesn't support that.
You'll need to recompile `dlib` after making the code change outlined here.

Issue: `RuntimeError: Unsupported image type, must be 8bit gray or RGB image.` when running the webcam examples.

Solution: Your webcam probably isn't set up correctly with OpenCV. Look here for more.

Issue: `MemoryError` when running `pip2 install face_recognition`
Solution: The face_recognition_models file is too big for your available pip cache memory. Instead,
try `pip2 --no-cache-dir install face_recognition` to avoid the issue.

Solution: The version of `dlib` you have installed is too old. You need version 19.7 or newer. Upgrade `dlib`.

Issue: `Attribute Error: 'Module' object has no attribute 'cnn_face_detection_model_v1'`

Solution: The version of `dlib` you have installed is too old. You need version 19.7 or newer. Upgrade `dlib`.

Issue: `TypeError: imread() got an unexpected keyword argument 'mode'`

Solution: The version of `scipy` you have installed is too old. You need version 0.17 or newer. Upgrade `scipy`.

Thanks

- Many, many thanks to Davis King (@nulhom) for creating dlib and for providing the trained facial feature detection and face encoding models used in this library. For more information on the ResNet that powers the face encodings, check out his blog post.
- Thanks to everyone who works on all the awesome Python data science libraries like numpy, scipy, scikit-image, pillow, etc, etc that makes this kind of stuff so easy and fun in Python.
- Thanks to Cookiecutter and the audreyr/cookiecutter-pypackage project template for making Python project packaging way more tolerable.

History

1.2.3 (2018-08-21)

- You can now pass model=”small” to face_landmarks() to use the 5-point face model instead of the 68-point model.
- Now officially supporting Python 3.7
- New example of using this library in a Jupyter Notebook

- Removed dependencies on scipy to make installation easier
- Cleaned up KNN example and fixed a bug with drawing fonts to label detected faces in the demo

1.2.1 (2018-02-01)

- Fixed version numbering inside of module code.

1.2.0 (2018-02-01)

- Fixed a bug where batch size parameter didn’t work correctly when doing batch face detections on GPU.
- Updated OpenCV examples to do proper BGR -> RGB conversion
- Updated webcam examples to avoid common mistakes and reduce support questions
- Added a KNN classification example
- Added an example of automatically blurring faces in images or videos
- Updated Dockerfile example to use dlib v19.9 which removes the boost dependency.

1.1.0 (2017-09-23)

- Will use dlib’s 5-point face pose estimator when possible for speed (instead of 68-point face pose esimator)
- dlib v19.7 is now the minimum required version
- face_recognition_models v0.3.0 is now the minimum required version

1.0.0 (2017-08-29)

- Added support for dlib’s CNN face detection model via model=”cnn” parameter on face detecion call
- Added support for GPU batched face detections using dlib’s CNN face detector model

- Added face_rec_from_video_file.py to examples
- dlib v19.5 is now the minimum required version
- face_recognition_models v0.2.0 is now the minimum required version

0.2.2 (2017-07-07)

- Added –show-distance to cli
- Fixed a bug where –tolerance was ignored in cli if testing a single image
- Added benchmark.py to examples

0.2.1 (2017-07-03)

- Added –tolerance to cli

0.2.0 (2017-06-03)

- The CLI can now take advantage of multiple CPUs. Just pass in the -cpus X parameter where X is the number of CPUs to use.
- Added face_distance.py example
- Improved CLI tests to actually test the CLI functionality
- Updated facerec_on_raspberry_pi.py to capture in rgb (not bgr) format.

0.1.14 (2017-04-22)

- Fixed a ValueError crash when using the CLI on Python 2.7

0.1.13 (2017-04-20)

- Raspberry Pi support.

0.1.12 (2017-04-13)

- Fixed: Face landmarks wasn’t returning all chin points.

0.1.10 (2017-03-21)

- Minor pref improvements with face comparisons.
- Test updates.

0.1.9 (2017-03-16)

- Fix minimum scipy version required.

0.1.8 (2017-03-16)

- Fix missing Pillow dependency.

0.1.7 (2017-03-13)

- First working release.

Join the official 2020 Python Developers Survey: Start the survey!

Status: All Systems Operational

Developed and maintained by the Python community, for the Python community.
Donate today!

Site map

Switch to desktop version

English español français 日本語 português (Brasil) українська Ελληνικά Deutsch 中文 (简体) русский
עברית

Pingdom Monitoring

Google Object Storage and Download Analytics

Sentry Error logging

AWS Cloud computing

DataDog Monitoring

Fastly CDN

DigiCert EV certificate

StatusPage Status page