Face Recognition

このファイルは 英語（オリジナル） in English、 中国語 简体中文版 、 韓国語 한국어で読むこともできます。

世界で最もシンプルな顔認識ライブラリを使って、Pythonやコマンドラインで顔を認識・操作することができるライブラリです。

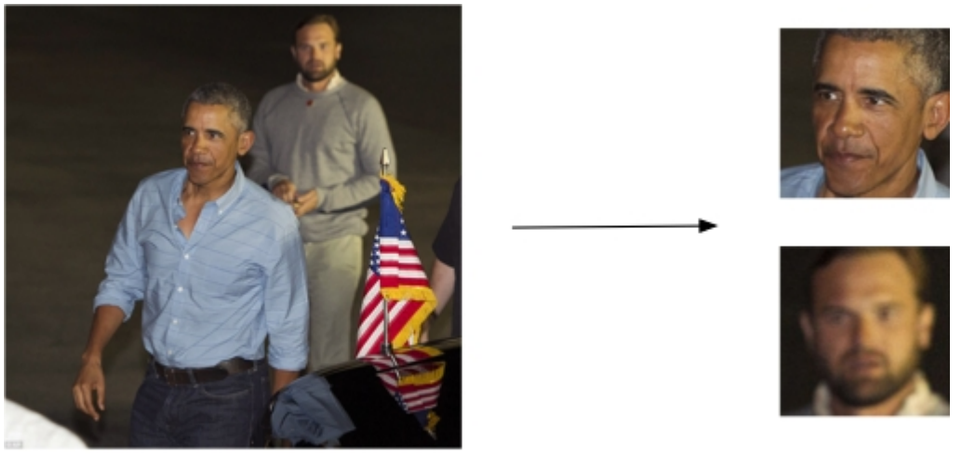
dlibのディープラーニングを用いた最先端の顔認識を使用して構築されており、このモデルはLabeled Faces in the Wildベンチマークにて99.38%の正解率を記録しています。

シンプルな`face_recognition`コマンドラインツールも用意しており、コマンドラインでフォルダ内の画像を顔認識することもできます。

特徴

画像から顔を探す

画像に写っているすべての顔を探します。

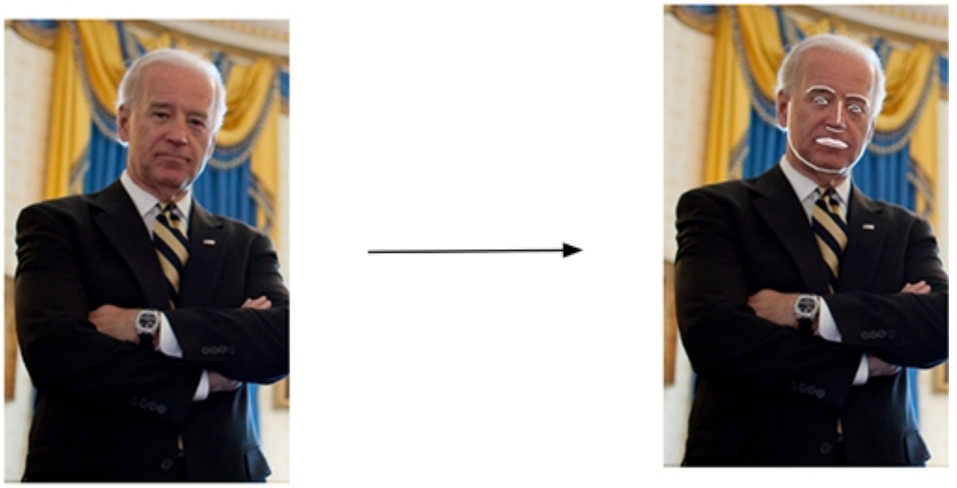


Input Output

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

画像から顔の特徴を取得する

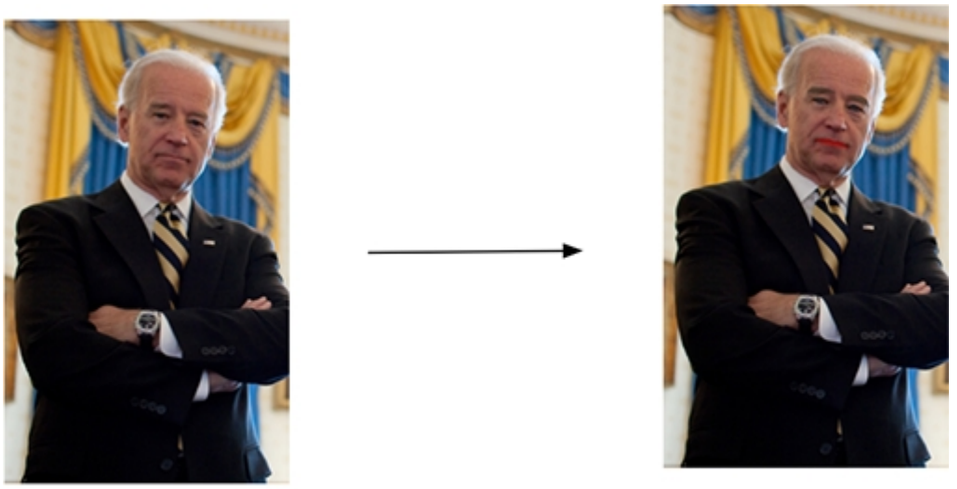
画像の中の顔から目、鼻、口、あごの場所と輪郭を得ることができます。



Input Output

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

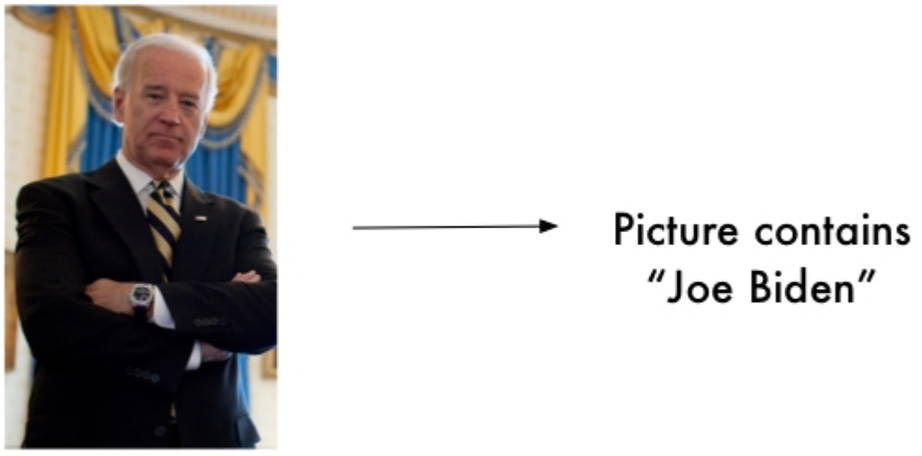
顔の特徴を見つけることは多くの重要なことに役立ちますが、[デジタルメイクアップ]() のようにさほど重要ではないことにも使うことができます。



Input Output

画像の中の顔を特定する

それぞれの画像に写っている人物を認識します。



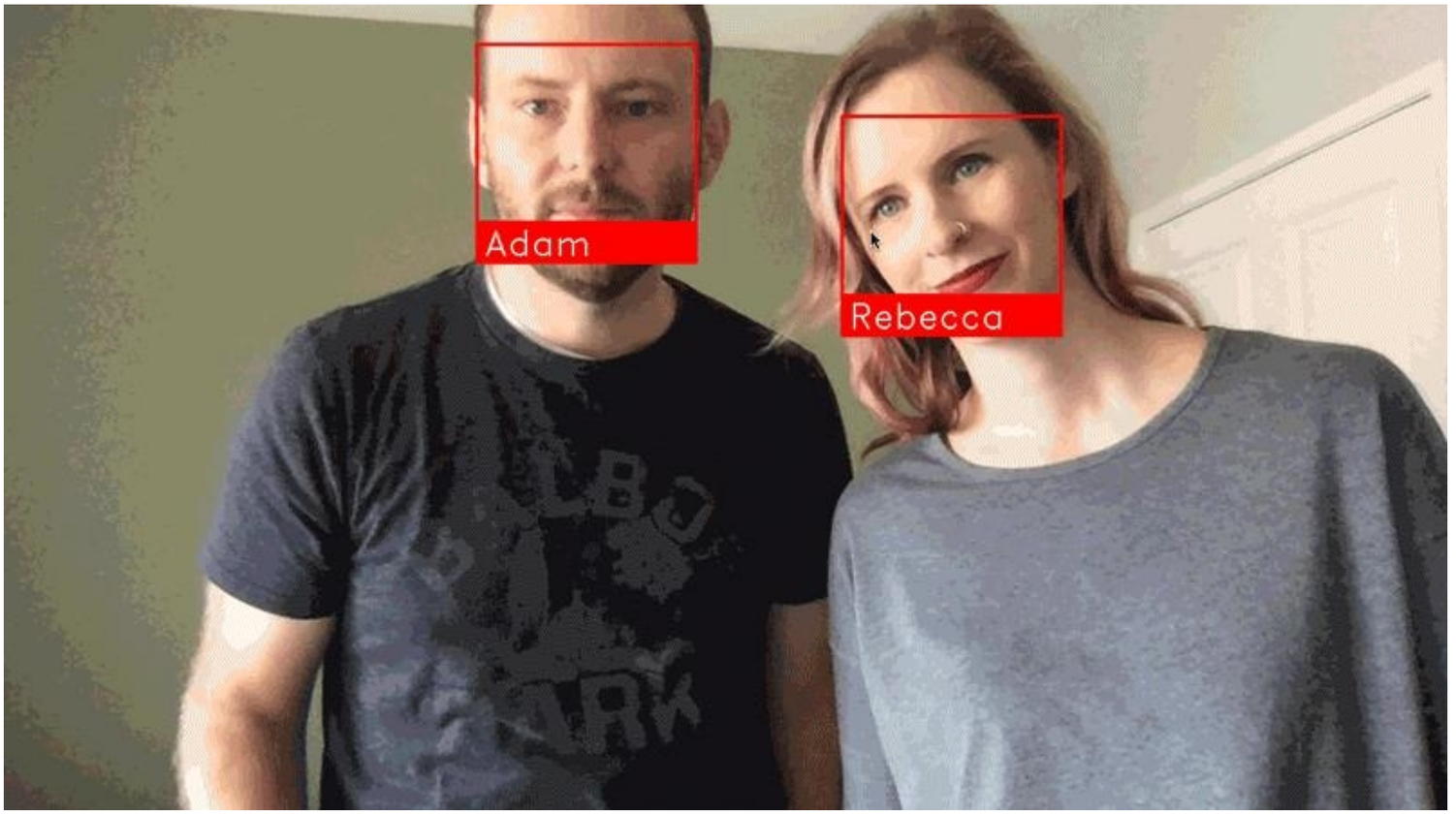
Input Output

```
import face_recognition
known_image = face_recognition.load_image_file("biden.jpg")
unknown_image = face_recognition.load_image_file("unknown.jpg")

biden_encoding = face_recognition.face_encodings(known_image)[0]
unknown_encoding = face_recognition.face_encodings(unknown_image)[0]

results = face_recognition.compare_faces([biden_encoding], unknown_encoding)
```

他のPythonライブラリと一緒に用いてリアルタイムに顔認識することも可能です。



試す場合はこちらのサンプルコード を参照してください。

デモ

ユーザーがコントリビュートした共有のJupyter notebookのデモがあります。（公式なサポートはありません）

Try in a Jupyter notebook

インストール

必要なもの

- Python 3.3+ もしくは Python 2.7
- macOS もしくは Linux (Windowsは公式にはサポートしていませんが動くかもしれません)

インストールオプション：

Macもしくは**Linux**にインストール

はじめに、dlibをインストールします。（Pythonの拡張機能も有効にします）

- macOSもしくはUbuntuにdlibをソースコードからインストールする方法

次に、このモジュールをpypiから`pip3`（Python2の場合は`pip2`）を使ってインストールします。

```
pip3 install face_recognition
```

あるいは、Dockerでこのライブラリを試すこともできます。詳しくは こちらのセクション を参照してください。

もし、インストールが上手くいかない場合は、すでに用意されているVMイメージで試すこともできます。詳しくは事前構成済みのVM を参照してください。(VMware Player もしくは VirtualBoxが対象)

Nvidia Jetson Nanoボードにインストール

- Jetson Nanoインストール手順
  - この記事の手順通りにインストールを行ってください。現在、Jetson NanoのCUDAライブラリにはバグがあり、記事の手順通りにdlibの一行をコメントアウトし再コンパイルしないと失敗する恐れがあります。

Raspberry Pi 2+にインストール

- Raspberry Pi 2+インストール手順

Windowsにインストール

Windowsは公式サポートされていませんが、役立つインストール手順が投稿されています。

- @masoudr's Windows 10 インストールガイド (dlib + face_recognition)

使用方法

コマンドライン

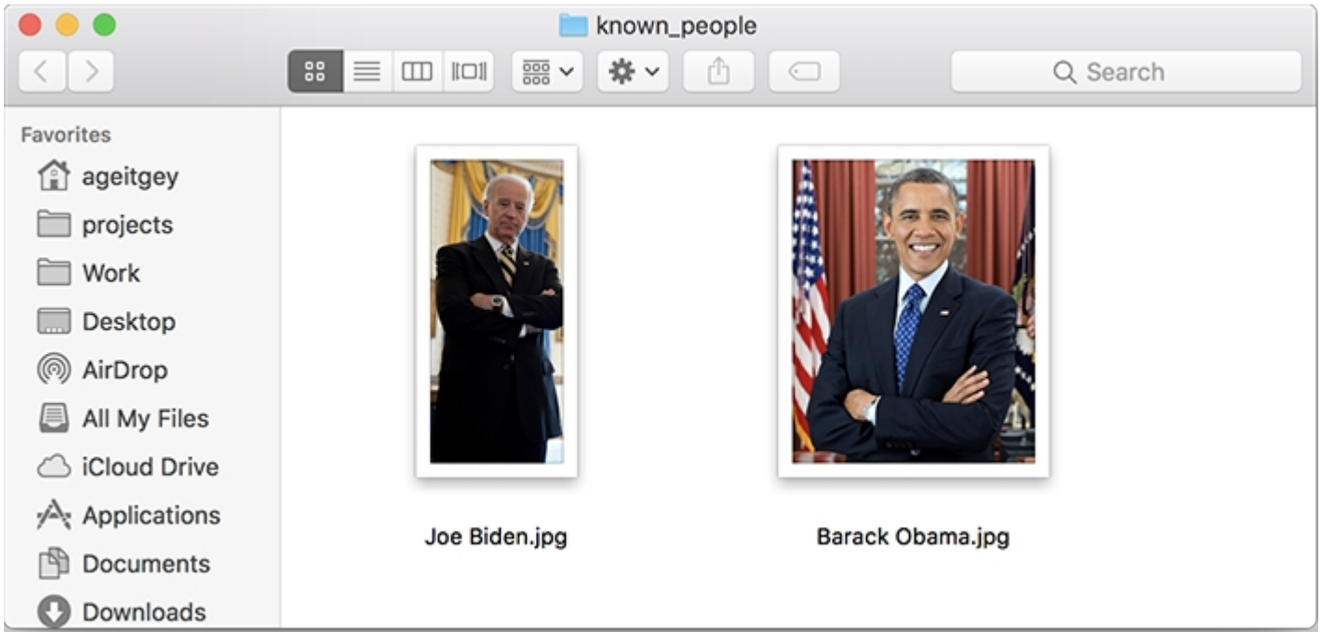
`face_recognition`をインストールすると、2つのシンプルなコマンドラインがついてきます。

- `face_recognition` - 画像もしくはフォルダの中の複数の画像から顔を認識します
- `face_detection` - 画像もしくはフォルダの中の複数の画像から顔を検出します

`face_recognition` コマンドラインツール

`face_recognition` コマンドによって、画像もしくはフォルダの中の複数の画像から顔を認識することができます。

まずは、フォルダに知っている人たちの画像を一枚ずつ入れます。一人につき１枚の画像ファイルを用意し、画像のファイル名はその画像に写っている人物の名前にします。



次に、2つ目のフォルダに特定したい画像を入れます。

そして、face_recognitionコマンドを実行し、知っている人の画像を入れたフォルダのパスと特定したい画像のフォルダ（もしくは画像ファイル）のパスを渡すと、それぞれの画像に誰がいるのかが分かります。

```
$ face_recognition ./pictures_of_people_i_know/ ./unknown_pictures/

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

一つの顔につき一行が出力され、ファイル名と特定した人物の名前がカンマ区切りで表示されます。

unknown_personは知っている人の画像の中の誰ともマッチしなかった顔です。

face_detection コマンドラインツール

face_detection コマンドによって、画像の中にある顔の位置（ピクセル座標）を検出することができます。

face_detection コマンドを実行し、顔を検出したい画像を入れたフォルダ（もしくは画像ファイル）のパスを渡してあげるだけです。

```
$ face_detection  ./folder_with_pictures/

examples/image1.jpg,65,215,169,112
examples/image2.jpg,62,394,211,244
examples/image2.jpg,95,941,244,792
```

検出された顔一つにつき一行が出力され、顔の上・右・下・左の座標（ピクセル単位）が表示されます。

許容誤差の調整 / 感度

もし同一人物に対して複数の一致があった場合、画像の中に写っている人たちの顔が非常に似ている可能性があるので、顔の比較をより厳しくするために許容誤差の値を下げる必要があります。

--tolerance コマンドによってそれが可能になります。デフォルトの許容誤差の値（tolerance value）を0.6よりも低くすると、より厳密に顔の比較をすることができます。

```
$ face_recognition --tolerance 0.54 ./pictures_of_people_i_know/ ./unknown_pictures/

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

もし許容誤差の設定を調整するために一致した顔の距離値（face distance）を確認したい場合は --show-distance true を使ってください。

```
$ face_recognition --show-distance true ./pictures_of_people_i_know/ ./unknown_pictures/

/unknown_pictures/unknown.jpg,Barack Obama,0.378542298956785
/face_recognition_test/unknown_pictures/unknown.jpg,unknown_person,None
```

その他の例

ファイル名は出力せずに人物の名前だけを表示することもできます。

```
$ face_recognition ./pictures_of_people_i_know/ ./unknown_pictures/ | cut -d ',' -f2

Barack Obama
unknown_person
```

Face Recognition の高速化

マルチコア搭載コンピューターの場合は並列で実行することも可能です。例えば4CPUコアの場合、同じ時間で約4倍の画像を処理することができます。

Python 3.4 以上を使っている場合は`--cpus <number_of_cpu_cores_to_use>` パラメータを渡します。

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

`--cpus -1` のパラメータを渡すことで、システムのすべてのCPUコアを使うことも可能です。

Pythonモジュール

`face_recognition` モジュールをインポートすると、数行のコードでとても簡単に操作を行うことができます。

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

自動的に画像の中のすべての顔を見つける

```
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-ordinates of each face!
```

試す場合はこちらのサンプルコードを参照してください。

さらに正確でディープラーニングをもとにした顔検出モデルを選択することも可能です。

注意：このモデルで良いパフォーマンスを出すにはGPUアクセラレーション（NVidiaのCUDAライブラリ経由）が必要です。また、`dlib` をコンパイルする際にCUDAサポートを有効にする必要あります。

```
import face_recognition

image = face_recognition.load_image_file("my_picture.jpg")
face_locations = face_recognition.face_locations(image, model="cnn")

# face_locations is now an array listing the co-ordinates of each face!
```

試す場合はこちらのサンプルコードを参照してください。

大量の画像をGPUを使って処理する場合は、こちらのサンプルコードのようにバッチ処理することも可能です。

自動的に画像の中の顔特徴を見つける

```
import face_recognition

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

# face_landmarks_list is now an array with the locations of each facial feature in each face.
# face_landmarks_list[0]['left_eye'] would be the location and outline of the first person's left eye.
```

試す場合はこちらのサンプルコードを参照してください。

画像の中の顔を認識し、その人物を特定する

```
import face_recognition

picture_of_me = face_recognition.load_image_file("me.jpg")
my_face_encoding = face_recognition.face_encodings(picture_of_me)[0]

# my_face_encoding now contains a universal 'encoding' of my facial features that can be compared to any other picture of a face!

unknown_picture = face_recognition.load_image_file("unknown.jpg")
unknown_face_encoding = face_recognition.face_encodings(unknown_picture)[0]

# Now we can see the two face encodings are of the same person with `compare_faces`!

results = face_recognition.compare_faces([my_face_encoding], unknown_face_encoding)

if results[0] == True:
    print("It's a picture of me!")
else:
    print("It's not a picture of me!")
```

試す場合はこちらのサンプルコードを参照してください。

Pythonコードのサンプル

すべてのサンプルはこちらで見ることができます。

顔検出

- 画像から顔を見つける
- 画像から顔を見つける（ディープラーニングを使用する）
- 大量の画像からGPUを用いて顔を見つける（ディープラーニングを使用する）
- WEBカメラによるライブ動画のすべての顔をぼかす(OpenCVのインストールが必要)

顔の特徴

- 画像から顔の特徴を特定する
- デジタルメイクアップを施す

顔認識

- 知っている人の画像をもとに画像の中の知らない顔を発見する
- 画像の中の顔を四角で囲む
- 顔の距離値（face distance）によって比較する
- WEBカメラによるライブ動画で顔認識する シンプル／低速バージョン (OpenCVのインストールが必要)
- WEBカメラによるライブ動画で顔認識する - 高速バージョン (OpenCVのインストールが必要)
- 動画ファイルを顔認識して新しいファイルに書き出す (OpenCVのインストールが必要)
- カメラ付きのRaspberry Piによって顔認識する
- 顔認識ウェブサービスをHTTP経由で実行する(Flaskのインストールが必要)
- k近傍法で顔認識する
- 人物ごとに複数の画像をトレーニングし、SVM（サポートベクターマシン）を用いて顔認識する

スタンドアロンの実行ファイルの作成

`python` や `face_recognition`のインストールをせずに実行することができるスタンドアロンの実行ファイルを作る場合は、PyInstallerを使います。しかし、このライブラリを使用するにはカスタム設定が必要です。

`face_recognition`をカバーする記事とガイド

- 顔認識の仕組みについての記事: ディープラーニングによる最新の顔認識
    - アルゴリズムとそれらがどのように動くかを取り上げています。

- Adrian Rosebrock氏の OpenCV、Python、ディープラーニングによる顔認識
  - 実際に顔認識を使用する方法について取り上げています。
- Adrian Rosebrock氏の Raspberry Pi 顔認識
  - Raspberry Piで使用する方法について取り上げています。
- Adrian Rosebrock氏の Pythonによる顔のクラスタリング
  - それぞれの画像に出現する人物に基づき、教師なし学習を用いて自動的に画像をクラスター化する方法について取り上げています。

顔認識の仕組み

ブラックボックスライブラリに依存せず、顔の位置や認識の仕組みを知りたい方はこちらの記事を読んでください。

注意事項

- この顔認識モデルは大人でトレーニングされており、子どもではあまり上手く機能しません。比較する閾値をデフォルト（0.6）のままで使用すると子どもを混同しやすくなります。

- 精度は民族グループによって異なる可能性があります。詳しくはこちらのwikiページを参照してください。

クラウドにデプロイ (Heroku, AWSなど)

`face_recognition`はC++で書かれた`dlib`に依存しているため、HerokuやAWSのようなクラウドサーバにこれらを使ったアプリをデプロイするのは難しい場合があります。

それを簡単にするために、このレポジトリにはDockerコンテナ内で`face_recognition`のビルドされたアプリを実行する方法を示したサンプルDockerfileがあります。これによって、Dockerイメージをサポートしているすべてのサービスにデプロイできるようになるはずです。

コマンドを実行し、ローカルでDockerイメージを試すことができます。: `docker-compose up --build`

GPU (drivers >= 384.81) および Nvidia-Docker がインストールされているLinuxユーザーはGPUでサンプルを実行することができます。docker-compose.yml を開き、`dockerfile: Dockerfile.gpu`と`runtime: nvidia`の行をコメントアウトしてください。

なにか問題が発生したら

もし問題が発生した場合はGitHubにIssueをあげる前に、まずはwikiのよくあるエラーをお読みください

謝意

- dlibを作り、このライブラリで使っているトレーニングされた顔の特徴検出とフェイスエンコーディングモデルを提供してくれたDavis King (@nulhom)、本当にありがとうございます。 フェイスエンコーディングを動かしているResNetについての情報は彼のブログを見てください。

- このようなライブラリがPythonで簡単に楽しくできるためのnumpy, scipy, scikit-image, pillow など全ての素晴らしいPythonデータサイエンスライブラリに取り組んでいる人たちに感謝しています。

- PythonプロジェクトのパッケージングをよりEasyにするCookiecutterとaudreyr/cookiecutter-pypackageに感謝しています。