

dn-5. 顔情報処理 (Dlib, InsightFace を使用)

(ディープラーニング入門演習)

URL: <https://www.kkaneko.jp/ai/dn/index.html>

金子邦彦



顔検出

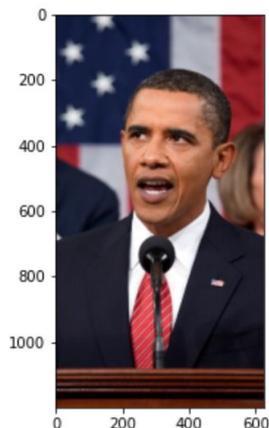
顔検出

写真やビデオの中から、
顔を識別すること

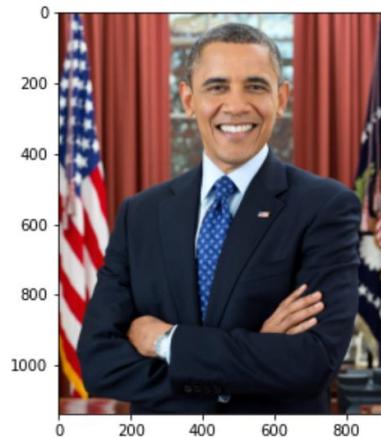
- ・ **顔の場所と領域**を検出
- ・ 複数ある場合にはすべて識別



requirement already satisfied: status=0
[True]



requirement already satisfied: status=10.7
[False]



• 顔検証 (face verification)

2つの別の写真あるいはビデオを照合し、同一人物であるかを判定すること

同一人物である 同一人物でない

顔情報処理の用途

- 顔による本人確認
- 集団行動解析、人流解析、人数推定
- 顔情報処理（表情判定、3次元化、顔面切り抜きなど）の前処理

顔のランドマーク

顔のランドマーク

まゆげ、目、鼻、くち、
顔の輪郭など、**顔の目
印**となるポイント

顔特徴ベクトル

顔を数値ベクトル化したもの



顔の 68 ランドマーク

```
-0.0512202  
0.0150111  
0.0642803  
-0.0438789  
-0.0485441  
-0.0107222  
-0.0653341  
-0.144676  
0.156388  
-0.12738  
0.137432  
-0.059849  
-0.1569  
-0.0690487  
-0.0250859  
0.215287  
-0.134682  
-0.212719  
-0.0921698  
0.019872  
-0.0154232  
0.0199377  
-0.0035686  
-0.0199529
```

Dlib

Dlib

- Dlib は、数多くの機能を持つソフトウェア。
- ソースコードはオープン
- Python, C++ のプログラムから使うためのインタフェースを持つ。
- 機能: 機械学習, 数値計算, グラフィカルモデル推論, 画像処理, スレッド, 通信, GUI, データ圧縮・一貫性, テスト, さまざまなユーティリティ

URL: <http://dlib.net/>

Dlib の顔情報処理

Dlib には、顔情報処理に関して、次の機能がある

- 顔検出
- 顔ランドマークの検出
- 顔のアラインメント
- 顔のコード化

※ ディープニューラルネットワークの学習済みモデルも配布されている

<https://github.com/davisking/dlib-models>

https://github.com/ageitgey/face_recognition/blob/master/examples/face_recognition_knn.py

Dlib での顔のコード化

- 人工知能 (Resnet) を使用. すでに多数の顔画像により学習済み
- **顔のコード化**の, さらなる精度向上には, 1000万をこえる新規画像が必要になる可能性あり
- **顔認証**での利用では, 精度向上のため, 既知の顔画像1つでなく, **複数の顔画像のコードと照合**することを考慮する

https://github.com/ageitgey/face_recognition/blob/master/examples/face_recognition_knn.py

顔認証は：顔画像から個体を識別すること

普通のアプリとライブラリの比較



• 普通のアプリ



「アプリ」というときは、
ひとつおりの機能がそろった
ソフトウェア。
そこで仕事を済ませる。

• ライブラリ



「ライブラリ」というときは、
基本機能が完備。
その上で、プログラムを作り
アプリを完成させる。

顔検出

- 写真やビデオの中から、顔を検出



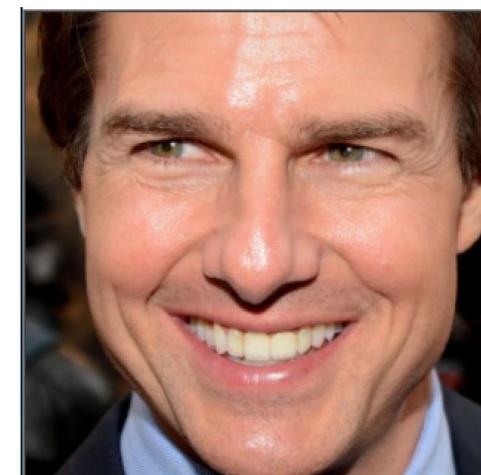
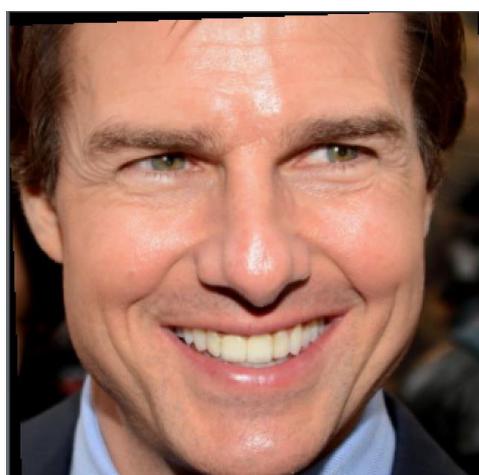
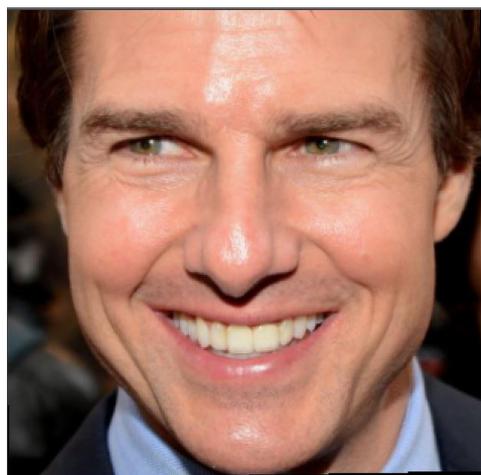
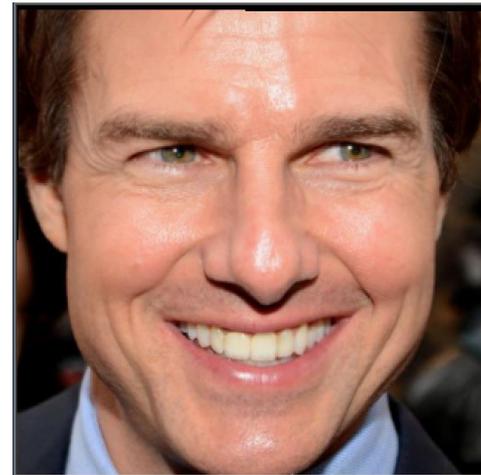
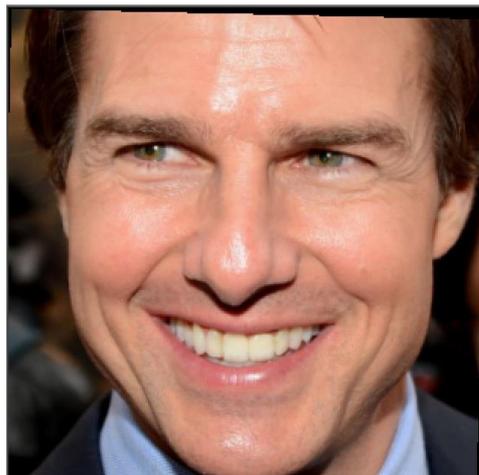
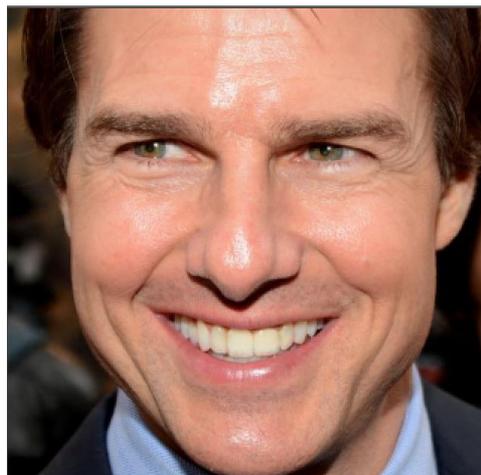
顔のアライメント

- 個々の顔の傾きを自動調整



顔の増量

- 元画像から，少しずつ変換した画像を多数生成



5ランドマーク

- 右目, 左目, 鼻の5つのランドマーク



68ランドマーク

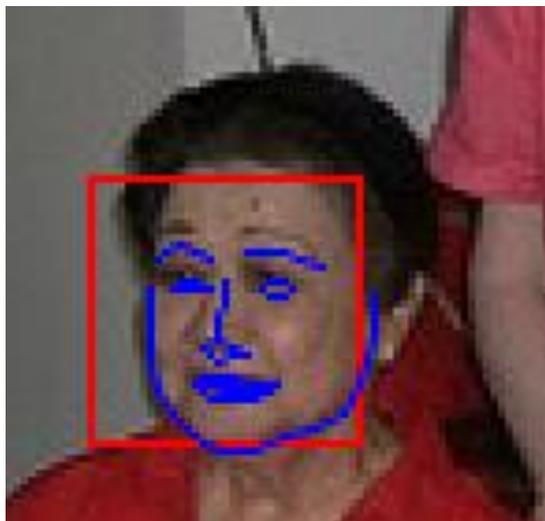
- 顔の所定の箇所に設けられた68個のランドマーク



特徴ベクトル



- Dlibでは、顔のアライメントののち、ランドマークを用いて特徴ベクトルが算出される

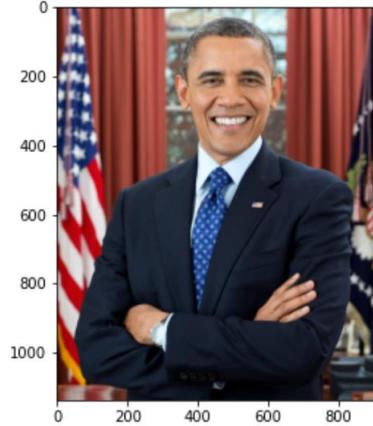


```
0. 0512202  
0. 0150111  
0. 0642803  
-0. 0438789  
-0. 0485441  
-0. 0107222  
-0. 0653341  
-0. 144676  
0. 156388  
-0. 12738  
0. 137432  
-0. 059849  
-0. 1569  
-0. 0690487  
-0. 0250859  
0. 215287  
-0. 134682  
-0. 212719  
-0. 0921698  
0. 019872  
-0. 0154232  
0. 0199377  
-0. 0035686  
-0. 0199529  
0. 118588
```

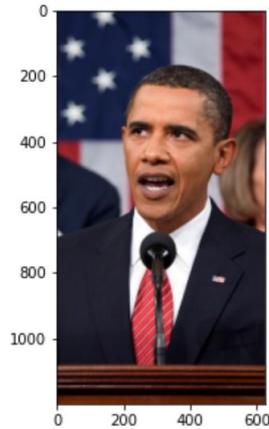
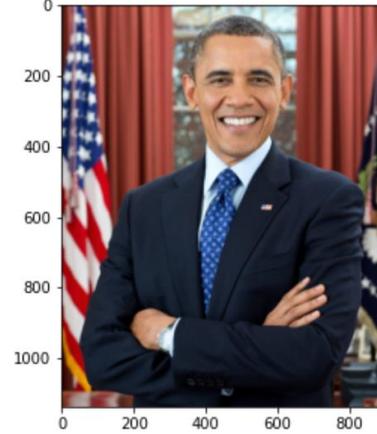
顔検証 (face verification)

- 顔検証により, 同一人物かを識別

requirements already satisfied: error=0
[True]



requirements already satisfied: error=10.7
[False]

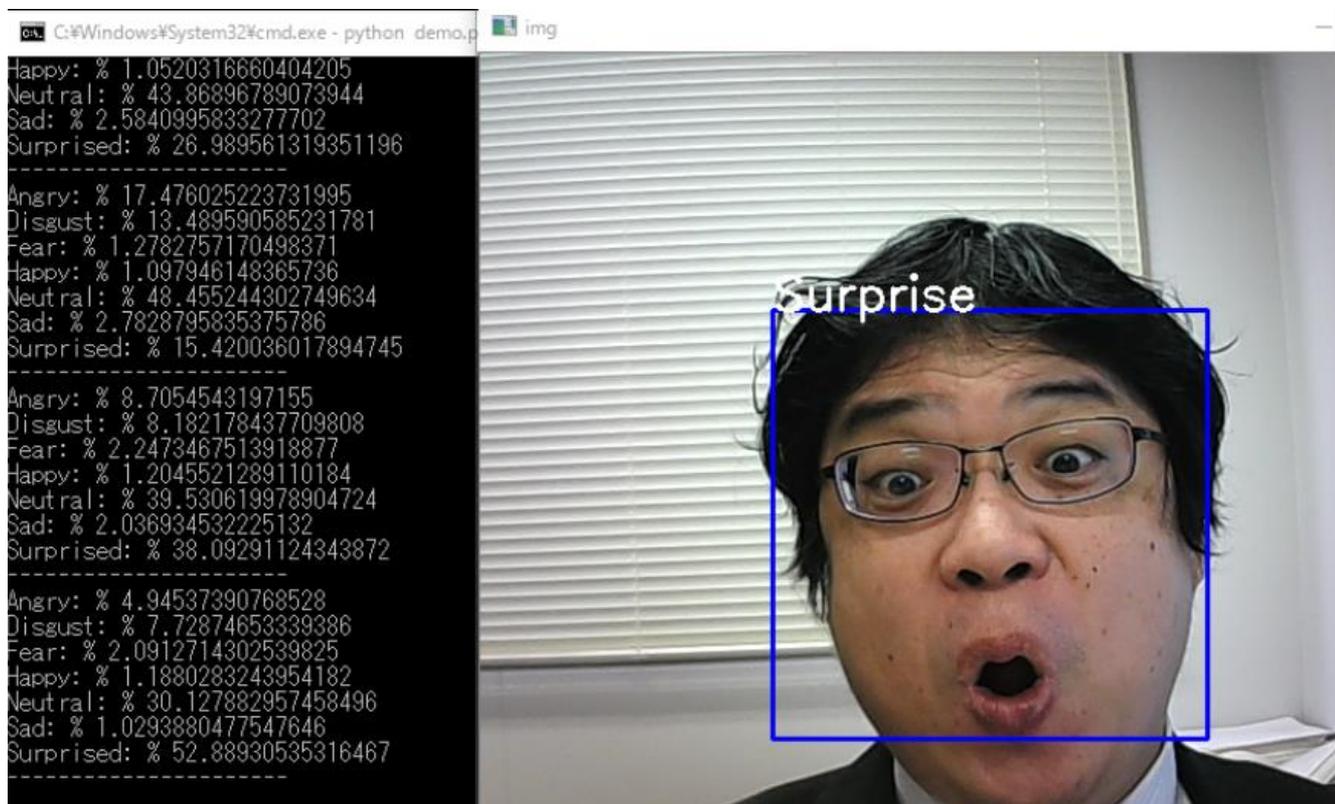


同一人物である 同一人物でない

Dlib の応用例 表情判定



- 7種の表情 Angry, Disgust, Fear, Happy, Neutral, Sad, Surprised のそれぞれの確率を判定
- <https://github.com/ezgiakcora/Facial-Expression-Keras> で公開されている成果物



顔検出，顔認識などを
行ってみる

Colaboratory ノートブックの WEB ページ



コードセル

テキストセル

コードセル

- オンラインで公開可能
- WEBブラウザでアクセス
- コードセルは Python プログラム。各自の Google アカウントでログインすれば、変更、再実行可能

実行結果

```
[ ] files = ['a.png', 'b.png', 'c.png', '126.png', '127.png']
```

6. 顔検出

顔検出は、写真やビデオの中の顔を検出すること。顔とそれ以外のオブジェクトを区別することも行う。顔検出の結果は、バウンディングボックスで得られるのが普通である。

次のプログラムは、Dlib を用いて、画像からの顔検出を行う。

- 「dets = cnn_face_detector(img, 6)」・・・顔検出の実行
- 「cv2.rectangle(disp, (d.rect.left(), d.rect.top()), (d.rect.right(), d.rect.bottom()), (255, 0, 0), 1)」・・・顔検出の結果を四角形で表示

結果は、赤い四角で表示される。1, 3, 4, 5 番目の画像 (a.png, c.png, 126.png, 127.png) からは、顔が検出される。2 番目の画像 (b.png, 手で顔を覆い隠したもの) からは顔が検出されない。少し隠れていたり、顔が傾いていても顔検出ができるが、大きく隠れていると顔検出できない。

実行結果が長いので、スクロールして全体を確認すること。

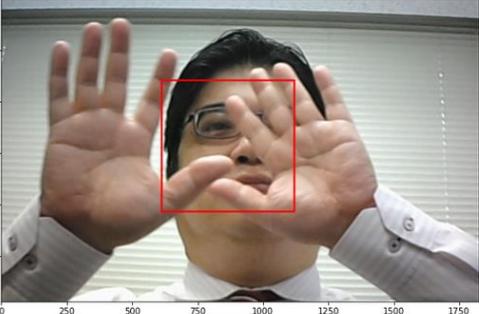
謝辞: この Python プログラムは、Dlib に付属の cnn_face_detector.py を書き換えて使用している

```
import sys
import dlib
import os
import urllib.request
import cv2
import matplotlib.pyplot as plt

cnn_face_detector = dlib.cnn_face_detection_model_v1('mmod_human_face_detector.dat')

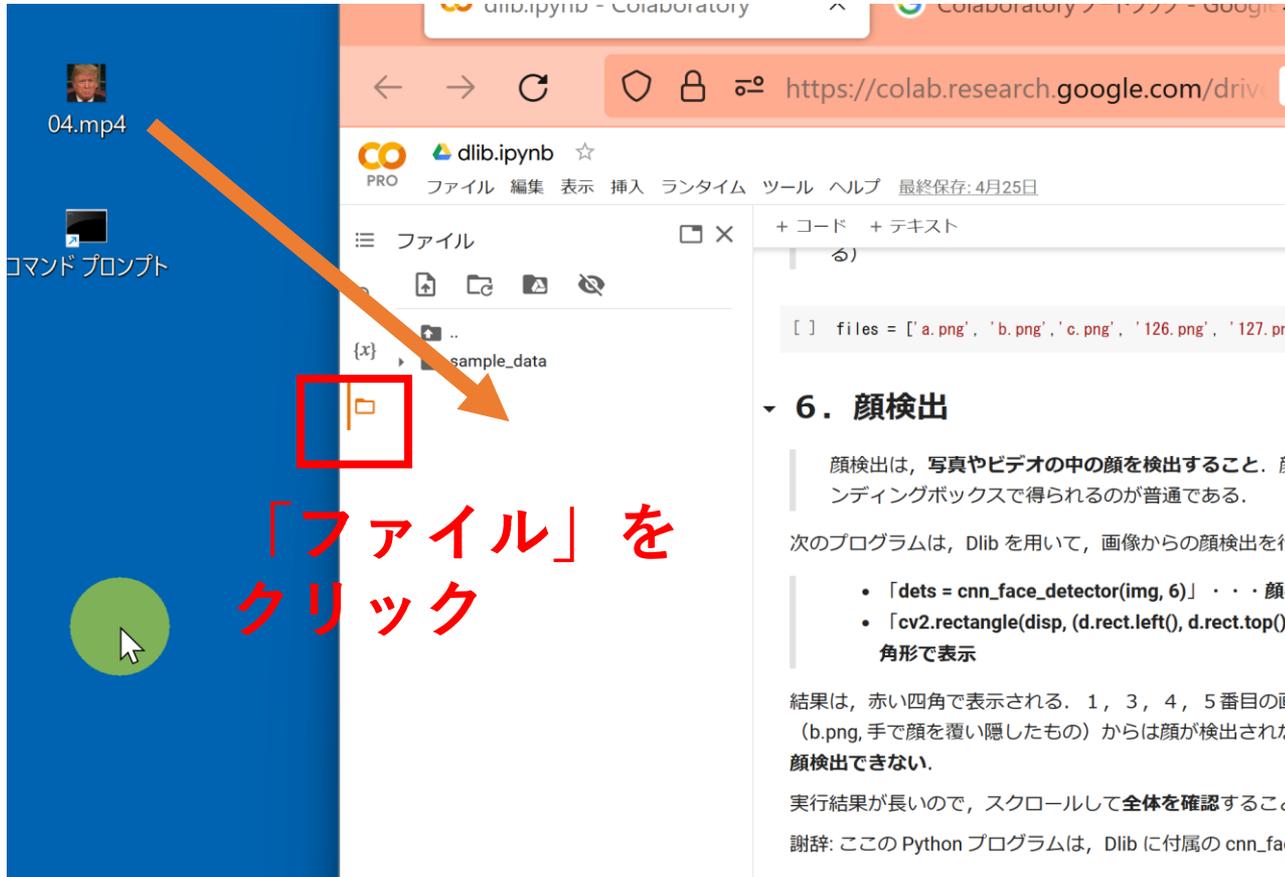
for f in files:
    print("### file: {} {}".format(f))
    img = dlib.load_rgb_image(f)
    dets = cnn_face_detector(img, 1)
    print("Number of faces detected: {}".format(len(dets)))
    for i, d in enumerate(dets):
        print("Detection {}: Left: {} Top: {} Right: {} Bottom: {} Confidence: {}".format(
            i, d.rect.left(), d.rect.top(), d.rect.right(), d.rect.bottom(), d.confidence))
        disp = img.copy()
        for i, d in enumerate(dets):
            cv2.rectangle(disp, (d.rect.left(), d.rect.top()), (d.rect.right(), d.rect.bottom()), (255, 0, 0), 6)
    plt.figure(figsize=(10,10))
    plt.imshow(disp)
    plt.show()
```

*** file: a.png ***
Number of faces detected: 1
Detection 0: Left: 614 Top: 319 Right: 1121 Bottom: 827 Confidence: 0.20801100134849548



*** file: b.png ***
Number of faces detected: 0

Colaboratory ノートブックでの ファイルアップロード



The image shows a split-screen view. On the left is a blue sidebar with a user profile, a file named '04.mp4', and a terminal icon labeled 'コマンドプロンプト'. A red box highlights a file upload icon in the sidebar, with an orange arrow pointing to it from the text '「ファイル」をクリック' (Click 'File'). On the right is a web browser window showing a Colaboratory notebook. The browser address bar is 'https://colab.research.google.com/drive'. The notebook title is 'dlib.ipynb'. The file manager shows a folder named 'sample_data'. The code editor contains a Python snippet:

```
[ ] files = ['a.png', 'b.png', 'c.png', '126.png', '127.png']
```

 Below the code is a section titled '6. 顔検出' (6. Face Detection). The text explains that face detection is used to find faces in photos or videos. It mentions that the provided program uses Dlib for face detection. A list of code snippets is shown:

- 「dets = cnn_face_detector(img, 6)」・・・顔
- 「cv2.rectangle(dis, (d.rect.left(), d.rect.top())

 The text continues: '角形で表示' (display as a quadrangle). It notes that the results are shown as red quadrangles, and that faces in images like 'b.png' (where the face is covered by a hand) cannot be detected. It concludes by saying the execution result is long and suggests scrolling to check the entire output, and that the Python program uses Dlib's 'cnn_face_detector'.

Google アカウントの取得



(プログラムの変更, 再実行したい, ファイルをアップロードしたい) ときに Google アカウントでのログインが必要)

- 次のページを使用

<https://accounts.google.com/SignUp>

- 次の情報を登録する

氏名

自分が希望するメールアドレス

<ユーザー名> [@gmail.com](mailto:kanekokunihiko@gmail.com)

パスワード

生年月日, 性別

Google

Google アカウントの作成

姓 名

ユーザー名

半角英字、数字、ピリオドを使用できます。

選択可能なユーザー名:

[bangyanjinzi6](#) [jinzibangyan6](#) [kanekokunihiko72](#)

代わりに現在のメールアドレスを使用

パスワード 確認

半角英字、数字、記号を組み合わせて 8 文字以上で入力してください

[代わりにログイン](#)

[次へ](#)

前準備



顔がうつったファイルを2つ以上準備.

- ・ **ファイル名は、数字と英字のみ**を使用
- ・ 各自で準備してください。説明では次の画像を使用



1.png



2.png

Google Colaboratory の WEBページは 2 つ

URL は次の通り

実験 1

<https://colab.research.google.com/drive/1S55yEFiQpdIRdjWbdH0zzEYD5VAfklHd?usp=sharing>

実験 2

https://colab.research.google.com/drive/13fXJ4f2dF-53YI_6i_rAJl17cuYuLE91?usp=sharing

(URL は YouTube 動画の概要欄, 大学のゼッソの「レポート」内に記載)

プログラムを実行するときの注意点



- ① コードセルの左の実行ボタンで実行
- ② 次のような**エラー**が出て、**進めない**ことがある

```
↳ RuntimeError                                Traceback (most recent call last)
<ipython-input-4-7262e66b8792> in <module>()
    10 urllib.request.urlretrieve('https://www.kkaneko.jp/sample/dlib/2008_001009.jpg', '2008_001009.jpg')
    11
----> 12 cnn_face_detector = dlib.cnn_face_detection_model_v1('mmod_human_face_detector.dat')
    13
    14 files = ['2007_007763.jpg', '2008_001009.jpg']
```

`RuntimeError: Error while calling cudaGetDevice(&the_device_id) in file /tmp/pip-wheel-66glv9rf/dlib/dlib/cuda`

- メニューで「ランタイム」を選び
- 「ランタイムのタイプを変更」。
- 新しく出てきた画面で、ハードウェアアクセラレータのところを「GPU」にして、「保存」と操作
- コードセルの実行を最初からやり直す

