LAPORAN UAS
PENGOLAHAN CITRA DIGITAL

Nama	:	Umi Fifa Latifah
Kelas	:	6E
NPM	:	2010631170036

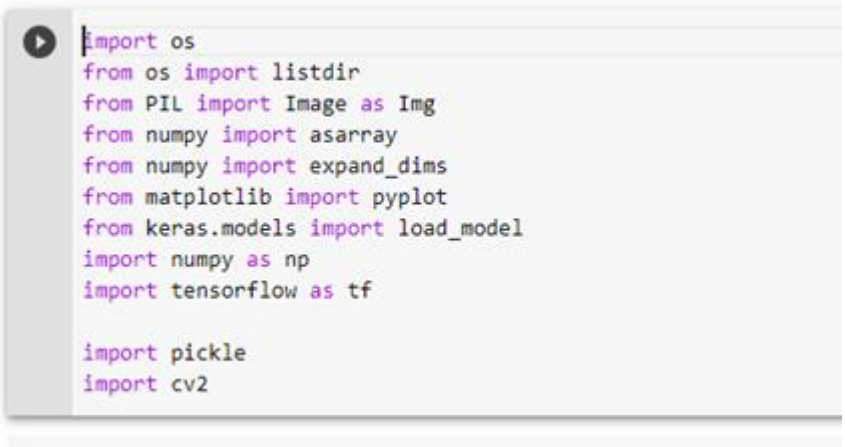
PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SINGAPERBANGSA KARAWANG

2023

Langkah pertama yaitu mengimpor librari yang akan digunakan dalam pemrosesan gambar dan pengenalan wajah



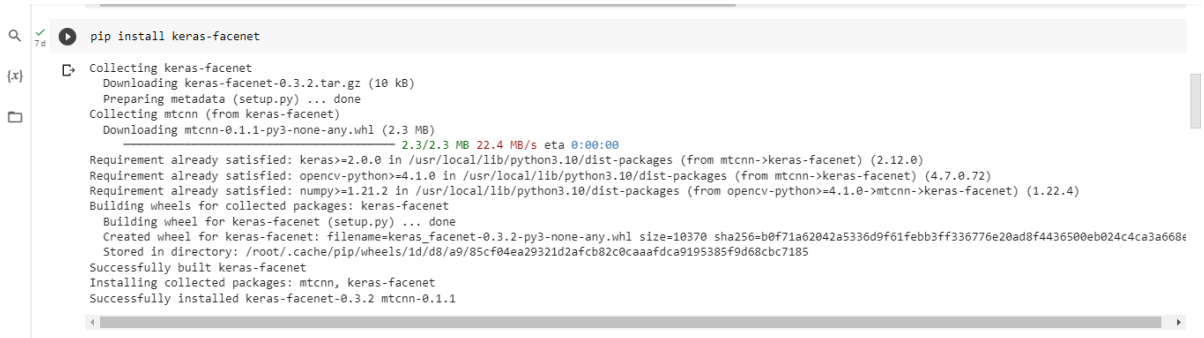
```
import os
from os import listdir
from PIL import Image as Img
from numpy import asarray
from numpy import expand_dims
from matplotlib import pyplot
from keras.models import load_model
import numpy as np
import tensorflow as tf

import pickle
import cv2
```

Selanjutnya download HaarCascade, yaitu untuk membuat objek detector wajah

```
HaarCascade = cv2.CascadeClassifier(cv2.samples.findFile(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'))
```

Install terlebih dahulu Keras-Facenet menggunakan pip! Untuk melakukan pengenalan wajah

```
!wget "https://drive.google.com/uc?export=download&id=1PZ_6Zsy1Vb0s0JmjEmVd8FS99zoMCiN1"

--2023-06-21 07:22:23--  https://drive.google.com/uc?export=download&id=1PZ_6Zsy1Vb0s0JmjEmVd8FS99zoMCiN1
Resolving drive.google.com (drive.google.com)... 74.125.142.138, 74.125.142.139, 74.125.142.101, ...
Connecting to drive.google.com (drive.google.com)|74.125.142.138|:443... connected.
HTTP request sent, awaiting response... 303 See Other
Location: https://doc-04-4s-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/i9ni7vn3l1vb9s6q4s4t8qk0dthmajs1/1687332075000/093792228482
Warning: wildcards not supported in HTTP.
--2023-06-21 07:22:26--  https://doc-04-4s-docs.googleusercontent.com/docs/securesc/ha0ro937gcuc7l7deffksulhg5h7mbp1/i9ni7vn3l1vb9s6q4s4t8qk0dthmajs1/168733207506
Resolving doc-04-4s-docs.googleusercontent.com (doc-04-4s-docs.googleusercontent.com)... 74.125.195.132, 2607:f8b0:400e:c09::84
Connecting to doc-04-4s-docs.googleusercontent.com (doc-04-4s-docs.googleusercontent.com)|74.125.195.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 92397640 (88M) [application/octet-stream]
Saving to: 'uc?export=download&id=1PZ_6Zsy1Vb0s0JmjEmVd8FS99zoMCiN1'

uc?export=download& 100%[===================>]  88.12M   139MB/s    in 0.6s

2023-06-21 07:22:28 (139 MB/s) - 'uc?export=download&id=1PZ_6Zsy1Vb0s0JmjEmVd8FS99zoMCiN1' saved [92397640/92397640]
```

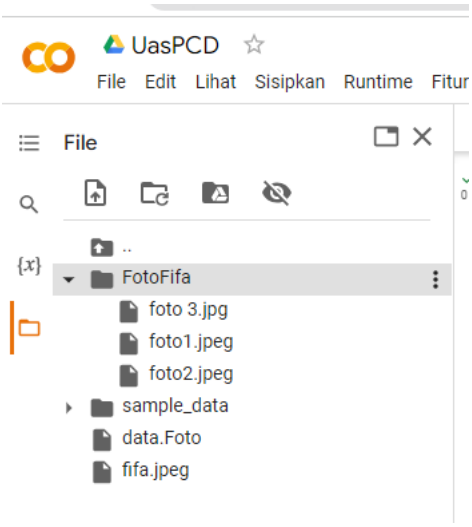


```
pip install keras-facenet

Collecting keras-facenet
  Downloading keras-facenet-0.3.2.tar.gz (10 kB)
  Preparing metadata (setup.py) ... done
Collecting mtcnn (from keras-facenet)
  Downloading mtcnn-0.1.1-py3-none-any.whl (2.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 22.4 MB/s eta 0:00:00
Requirement already satisfied: keras>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from mtcnn->keras-facenet) (2.12.0)
Requirement already satisfied: opencv-python>=4.1.0 in /usr/local/lib/python3.10/dist-packages (from mtcnn->keras-facenet) (4.7.0.72)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python>=4.1.0->mtcnn->keras-facenet) (1.22.4)
Building wheels for collected packages: keras-facenet
  Building wheel for keras-facenet (setup.py) ... done
  Created wheel for keras-facenet: filename=keras_facenet-0.3.2-py3-none-any.whl size=10370 sha256=b0f71a62042a5336d9f61febb3ff336776e20ad8f4436500eb024c4ca3a668
  Stored in directory: /root/.cache/pip/wheels/1d/d8/a9/85cf04ea29321d2afcb82c0caaafdca9195385f9d68cbc7185
Successfully built keras-facenet
Installing collected packages: mtcnn, keras-facenet
Successfully installed keras-facenet-0.3.2 mtcnn-0.1.1
```

Langkah selanjutnya yaitu membuat objek MyFaceNet yang merupakan instance dari kelas FaceNet. FaceNet adalah implementasi model jaringan saraf yang digunakan untuk ekstraksi fitur wajah. Objek MyFaceNet akan digunakan untuk melakukan pengenalan wajah dan ekstraksi fitur pada gambar wajah yang diberikan.

```
[5]  from keras_facenet import FaceNet

MyFaceNet = FaceNet()
```

Selanjutnya untuk Langkah kelima yaitu melakukan loop loop melalui semua file dalam folder yang ditentukan (FotoFifa/), sebelumnya membuat folder terlebih dahulu,isi folder tersebut yaitu foto-foto yang akan di cantumkan.

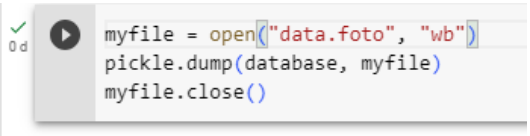



```
folder='FotoFifa/'
database = {}

for filename in listdir(folder):

    path = folder + filename
    gbr1 = cv2.imread(folder + filename)

    wajah = HaarCascade.detectMultiScale(gbr1,1.1,4)

    if len(wajah)>0:
        x1, y1, width, height = wajah[0]
    else:
        x1, y1, width, height = 1, 1, 10, 10

    x1, y1 = abs(x1), abs(y1)
    x2, y2 = x1 + width, y1 + height

    gbr = cv2.cvtColor(gbr1, cv2.COLOR_BGR2RGB)
    gbr = Img.fromarray(gbr)                  # konversi dari OpenCV ke PIL
    gbr_array = asarray(gbr)

    face = gbr_array[y1:y2, x1:x2]

    face = Img.fromarray(face)
    face = face.resize((160,160))
```

```
[27]    face = face.resize((160,160))
        face = asarray(face)



        face = expand_dims(face, axis=0)
        signature = MyFaceNet.embeddings(face)

        database[os.path.splitext(filename)[0]]=signature

1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 135ms/step
1/1 [==============================] - 0s 142ms/step
```

Langkah ini membuka file bernama "data.Foto" dalam mode penulisan biner ("wb"). Kemudian, menggunakan modul pickle, objek database yang berisi vektor fitur wajah dari gambar-gambar yang diproses sebelumnya diserialisasi dan disimpan dalam file tersebut.

Setelah selesai, file ditutup dengan menggunakan metode close() pada objek file yang dibuka. Dengan demikian, vektor fitur wajah yang disimpan dalam database akan tersimpan dalam file "data.Foto".



```
myfile = open("data.foto", "wb")
pickle.dump(database, myfile)
myfile.close()
```

Selanjutnya yaitu membuka file bernama "data.Foto" dalam mode pembacaan biner ("rb"). Menggunakan modul pickle, data yang terserialisasi dalam file tersebut di-deserialisasi dan dimuat ke dalam variabel database. Setelah proses pembacaan selesai, file ditutup menggunakan metode close() pada objek file yang dibuka. Dengan demikian, data yang telah tersimpan dalam file "data.Foto" akan dimuat kembali ke dalam variabel database untuk digunakan dalam program.

```
myfile = open("data.foto", "rb")
database = pickle.load(myfile)
myfile.close()
```

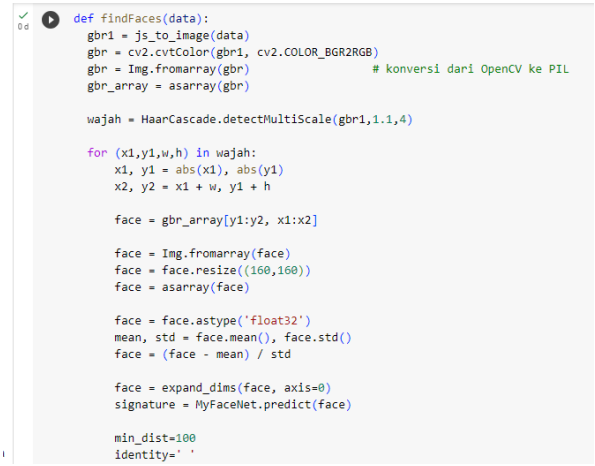
Pada Langkah ini yaitu mengonversi representasi gambar dalam format JavaScript (base64) menjadi objek gambar dalam Python

```
def js_to_image(js_reply):
  image_bytes = b64decode(js_reply.split(',')[1])
  jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
  img = cv2.imdecode(jpg_as_np, flags=1)
  return img
```

Selanjutnya untuk “FindFaces” mengambil data gambar dalam format JavaScript, mendeteksi wajah dalam gambar tersebut, dan memberikan penanda pada setiap wajah yang terdeteksi dengan label identitas.

- Mengonversi data gambar JavaScript menjadi objek gambar dalam format OpenCV menggunakan fungsi js_to_image().
- Mengonversi data gambar JavaScript menjadi objek gambar dalam format OpenCV menggunakan fungsi js_to_image().
- Mengonversi gambar dari format BGR ke RGB menggunakan cv2.cvtColor().
- Mengonversi gambar menjadi objek PIL menggunakan Img.fromarray().
- Mengubah objek PIL menjadi array numpy menggunakan asarray().
- Mendeteksi wajah dalam gambar menggunakan metode Haar Cascade dengan HaarCascade.detectMultiScale().
- Untuk setiap wajah yang terdeteksi, mengambil wilayah wajah dari array gambar.
- Mengubah ukuran wajah menjadi 160x160 piksel menggunakan resize().

- Mengnormalisasi wajah dengan mengurangi rata-rata dan membaginya dengan standar deviasi.
- Menambahkan dimensi tambahan pada array wajah menggunakan expand_dims().
- Menghasilkan vektor fitur wajah menggunakan metode embeddings() dari objek MyFaceNet.
- Membandingkan vektor fitur wajah dengan vektor fitur wajah dalam database untuk mencari identitas yang paling dekat menggunakan jarak Euclidean.
- Menandai setiap wajah dengan label identitas menggunakan cv2.putText() dan cv2.rectangle().
- Menyimpan gambar yang telah ditandai dalam file "photo.jpg" menggunakan cv2.imwrite().
- Mengembalikan nama file gambar yang telah disimpan



```
def findFaces(data):
    gbr1 = js_to_image(data)
    gbr = cv2.cvtColor(gbr1, cv2.COLOR_BGR2RGB)
    gbr = Img.fromarray(gbr)                  # konversi dari OpenCV ke PIL
    gbr_array = asarray(gbr)

    wajah = HaarCascade.detectMultiScale(gbr1,1.1,4)

    for (x1,y1,w,h) in wajah:
        x1, y1 = abs(x1), abs(y1)
        x2, y2 = x1 + w, y1 + h

        face = gbr_array[y1:y2, x1:x2]

        face = Img.fromarray(face)
        face = face.resize((160,160))
        face = asarray(face)

        face = face.astype('float32')
        mean, std = face.mean(), face.std()
        face = (face - mean) / std

        face = expand_dims(face, axis=0)
        signature = MyFaceNet.predict(face)

        min_dist=100
        identity=' '
```

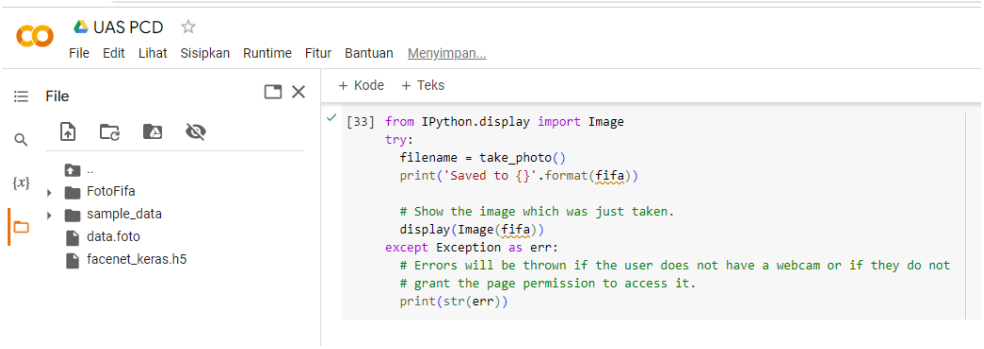
Pada Kode ini adalah untuk mengambil foto menggunakan kamera pada Google Colab dan mendeteksi wajah dalam foto yang diambil.

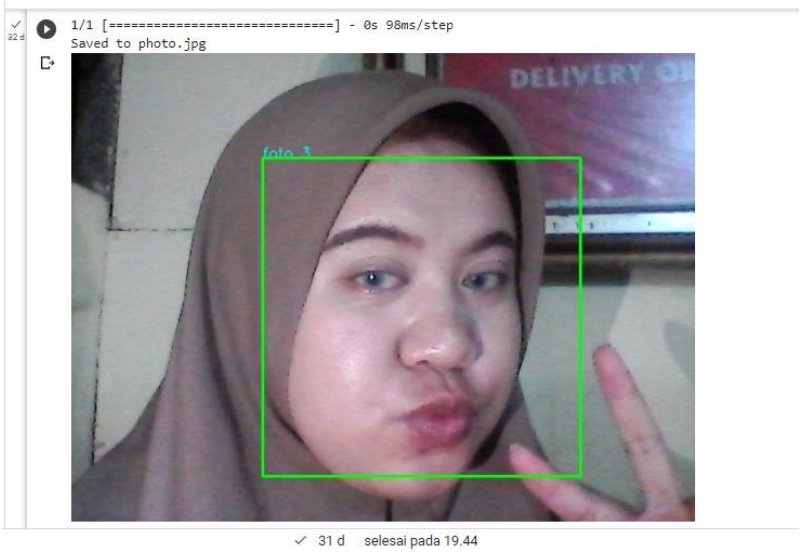
- Mengimpor modul dan fungsi yang diperlukan, seperti display, Javascript, eval_js, dan b64decode.
- Mendefinisikan fungsi take_photo yang akan digunakan untuk mengambil foto.
- Membuat skrip JavaScript yang akan dijalankan di dalam Colab menggunakan objek Javascript. Skrip ini berfungsi untuk menampilkan tampilan kamera, mengambil foto, dan mengonversi foto menjadi format base64.
- Menampilkan skrip JavaScript menggunakan fungsi display(js).
- Menggunakan eval_js untuk mengeksekusi skrip JavaScript yang telah ditampilkan dan mengambil data foto dalam format base64.
- Memanggil fungsi findFaces untuk mendeteksi wajah dalam foto yang diambil dan menyimpan foto yang telah ditandai.

- Mengembalikan nama file foto yang telah ditandai.

```
        const video = document.createElement('video');
        video.style.display = 'block';
        const stream = await navigator.mediaDevices.getUserMedia({video: true});

        document.body.appendChild(div);
        div.appendChild(video);
        video.srcObject = stream;
        await video.play();

        // Resize the output to fit the video element.
        google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

        // Wait for Capture to be clicked.
        await new Promise((resolve) => capture.onclick = resolve);

        const canvas = document.createElement('canvas');
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
        canvas.getContext('2d').drawImage(video, 0, 0);
        stream.getVideoTracks()[0].stop();
        div.remove();
        return canvas.toDataURL('image/jpeg', quality);
      }
      ''')
    display(js)
    data = eval_js('takePhoto({})'.format(quality))
```

Langkah yang terakhir yaitu menampilkan antar muka yang memungkinkan pengguna untuk mengambil foto menggunakan kamera dan menampilkan foto yang akan diambil





https://colab.research.google.com/drive/1M7uMax6Pdj_A7CRORvZCCYALxwcYaa_y?usp=sharing