The API

The model

For the recognition model a ResNet-34 network like architecture have been implemented as a face descriptor computer. The net was trained (by *Davis E. King*) from a dataset of about 3 million faces derived from a certainly datasets. The resulting model obtains a mean error of 0.993833 with a standard deviation of 0.00272732 on the LFW (Labeled Faces in the Wild) benchmark especially to face recognition.

The client side

All the client side features must be provided and developed by the user of this solution. On the devlopment of this library, the *Postman* software has been used as client for testing the server, and another *html+javascript* client exemple has been done for *face detecting* (not recognition). The example is acessible at s3 website .

The server side

Call all methods to the base endpoint, adding the ***/[SERVICE]*** to the *URI* from the server. Except if in the service there is an different instruction (in this documentation the name of the service will be all capitals). Example: **localhost:3000/Create** to create a person or **localhost:3000/100** with *DELETE* http method to delete the person with id *100*.

Into ./Server is the webserver api to use folowing service methods:

- ***Create***

  It must be passed as ***POST*** the person to be stored. The string param is personid and more 3 params with fotos containing a unique face each one param.

- ***Recognize***

  Finds into the stored records all especified persons (***persons*** param), the best match of the second parameter ***face***, that is a image with one face only. The persons must be passed as array-like (example: *['1','2','3','4','5','6','7']*).

  - The ***updatethreshold*** param is a float number to determine if the stored profile of person should be updated with this new face information. Ex: if the param is *0.6* and one person is recognized with proximity of 0.7, the new data of the image face is set into the person record, but if the proximity is 0.55, the person record stay as it is. The use of this feature can deteriorate (increase the time) the performance (in my test aproximately **400ms higher**).
  - The return is a JSON with the *PersonId* and the *proximity* that is a float with how close the person from the image sent is compared to the registred before.
  - The *updated* field indicate if the registry has been updated using this new image if it pass to the threshold value.
  - The *prediciton* field contain the estimated expression, estimated gender, and estimated age from the image queried. It is especified as param on the request body with the key ***predict*** and e the possible values are *true*, *false* or the param is not sent what makes the default behavior as *true*.

- ***All***

  Return an array-like of some stored data from all persons.

- ***GET***

  It must be called as ***GET*** http method, direcly to the base endpoint. In the endpoint must be added the ***PersonID*** value to be deleted as a param. It just return some stored information about a single Person.

- ***DELETE***

  It must be called as ***DELETE*** http method, direcly to the base endpoint. In the endpoint must be added the ***PersonID*** value, as a param, of the person to be deleted.

Test deployed at Heroku

An test version is avaiable at ***https://pstfacetools.herokuapp.com/[SERVICE]***

Some useful Commands

```
git push heroku main
heroku logs --tail
heroku ps --app pstfacetools
heroku ps:scale web=0 --app pstfacetools
```

Heroku fature

As free plan the dyno will be automatically put to sleep after *30s* without request. If the dyno instance is sleeping, any request will start the wakeup process (it will be done after around *20s*).

Warnings:

1. All work must be done with a single face in the image.
2. Web formats such as JPEG, GIF, and PNG generally are accepteds, since it works with canvas.
3. All requisiton to the server must pass the ***X-API-KEY*** key-value param in the header.
4. Send all params and values in the body of your request as a *form-data* format.
5. The server can take few minutes to start as it has to load the model and library.
6. No aditional logging is provided in the server side. You have to deal with default logs of your server or cloud provider.
7. The client example show some manipulations, never brings the include files to you project. This action can perform a regration in speed and architecture broke. It is a best practices to use CDN to delivery web massive content.