DAS-FACE

Facial biometrics service

Revisión	Fecha	Descripción	Redactado	Revisado	Aprobado
1	22/01/2018	das-Face Documentation	SGP	EMR	EAL

1. das-Face microservice

das-Face verifies the similarity of two face images that can be extracted from any source:

- Veridas Capture SDK's: mobile and HTML
- Selfie image
- Printed image
- Image present in the NFC (if existing) from different documents: DNI 3.0, passports with ICAO normative (e-passports).
- Photograms of a video Stream.
- Batch of files

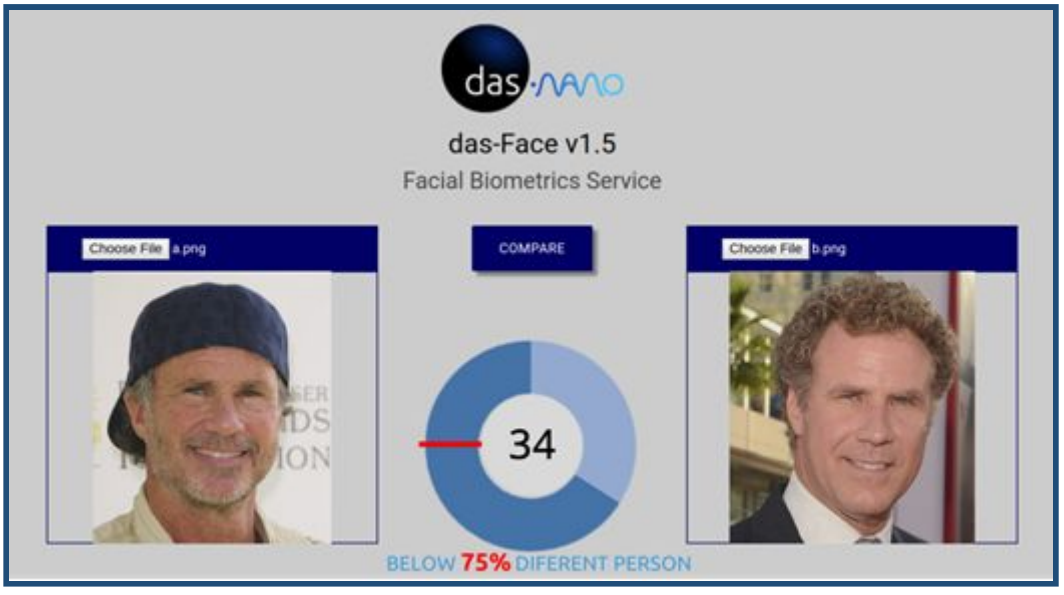
> das-Face is boosted by an Artificial Intelligence and Deep Learning software that is continuously improving its performance since it is currently in production, and analyzes a large variety of cases daily.

In the facial biometrics field,two situations are typically handled:

- **Verification:** The process of checking the identity of a person comparing two faces.
- **Identification:** The process of searching a person or a gallery of persons within a database of identities.

1.1. Verification

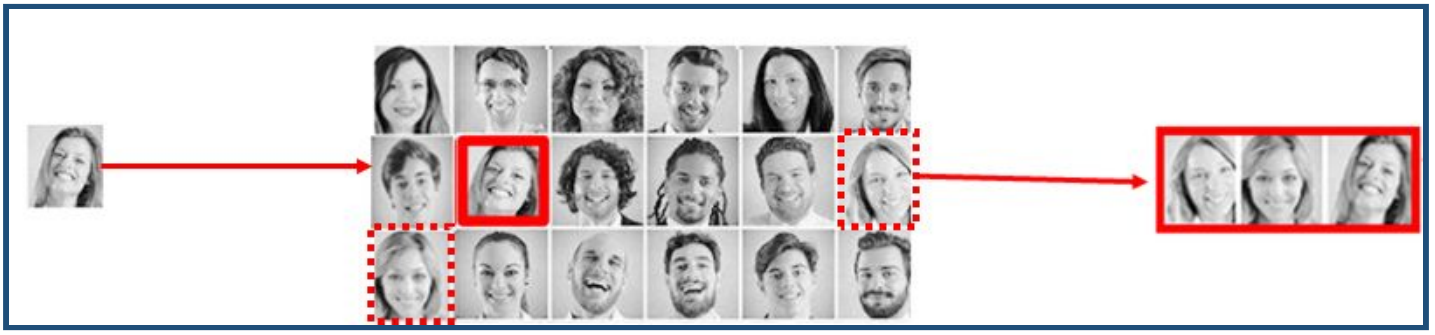
- **Facial Verification (1:1):** Given two faces, the system returns a score based on the similarity of both



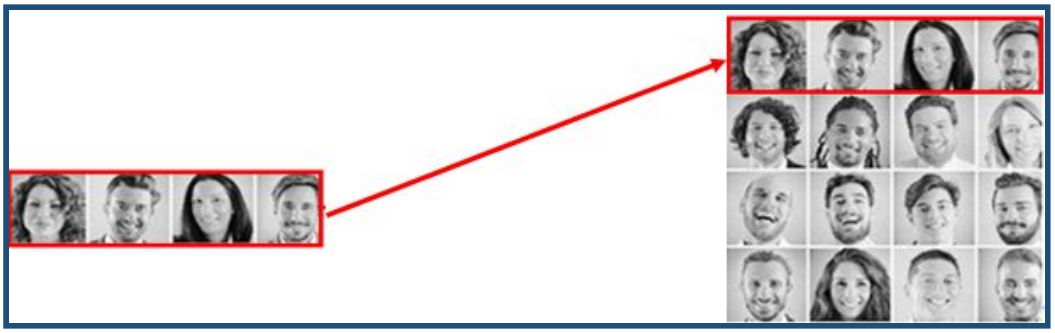
1.2. Identification

- **Facial Identification (1:N):** Given a gallery with multiple faces of people **(N)**, and a picture of a target person, the system returns the most similar **(1)** to the target inside the gallery .

- **Facial Identification (M:N)** Given a picture of a target person and a gallery with multiple faces of people **(N)**, the system returns the most similar faces **(M)** to the target person. This identification process is recommended when it has to be carried out within a huge number of faces in a short processing time.



- **Facial identification (X:N):** Given a gallery with multiple faces of people (N) and a specific gallery of photos that want to be identified (X), the system returns all photos of the gallery (X) that belong to the galley (N).

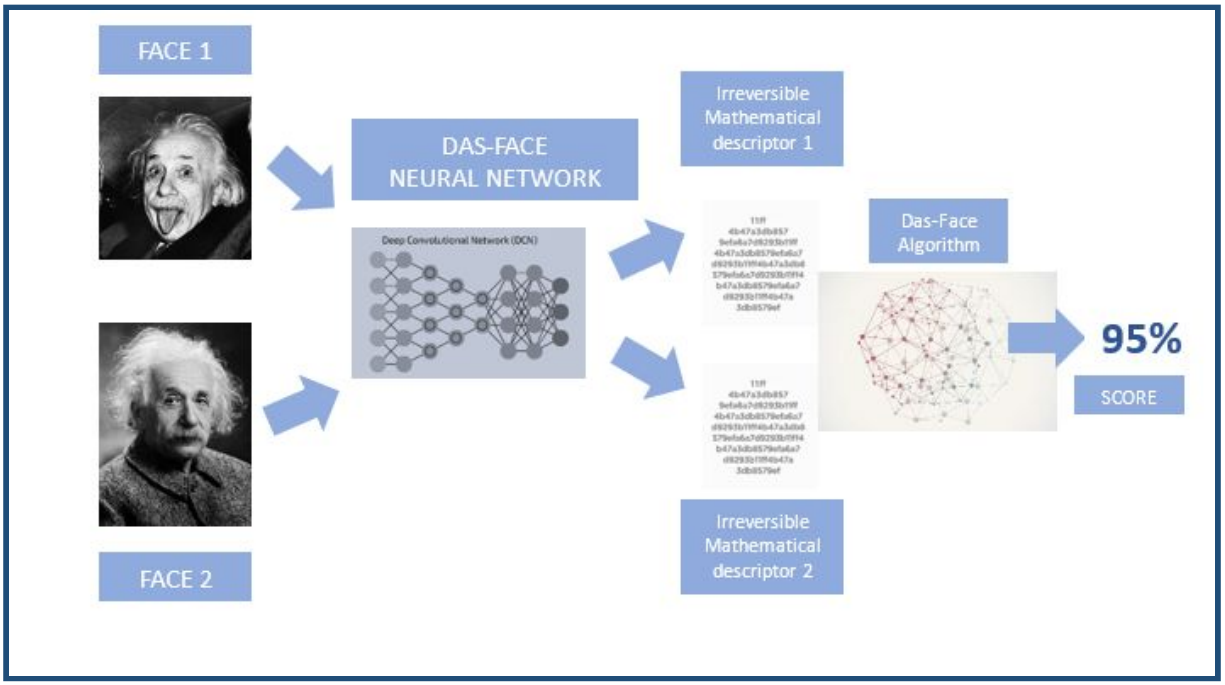


2. How it works?

2.1. Biometric engine

The microservice is offered as an API_REST format:

- Two pictures are sent to the server.
- The pictures are preprocessed.
- The pictures are converted into an irreversible mathematical descriptor (hash)
- Both hashes are compared using Veridas´ algorithms. The set-point can be controlled to control the FPR (false positive rate) and FNR (false negative rate)
- A Matching score is provided
- All the user information (faces' pictures) and descriptors are **immediately deleted**



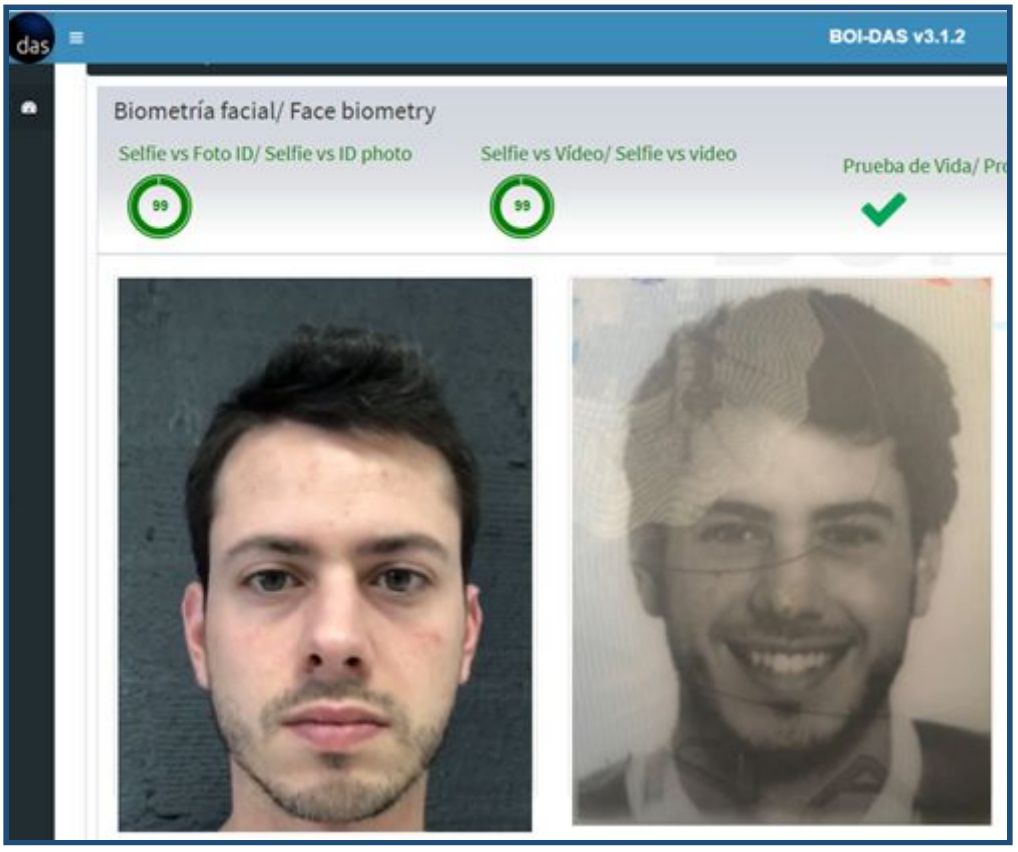
das-Face has overcome old school biometric techniques such as landmarks, in which distances between different facial features are taken into account to determine the identity of a person. These distances, in old school biometrics, are very dependent on aging, face pose, illumination, face gestures, face occlusions, and other situations which alter features taken directly from the image.

das-Face is backed by a strong artificial intelligence software that, using proprietary deep learning and neural network techniques, emulates the human brain performance.

2.1.1 Particular case: Selfie vs Document

The selfie vs document picture situation is a particular subcase of facial biometrics. Biometric systems typically compare two good-quality colour pictures. The case Selfie against document is a more complicated case, as normally document pictures are in black and white, printed with techniques to avoid duplication, they usually carry holograms on top of them. All these facts make the facial biometry a more complex scenario than the regular ones.

> Veridas has optimized its das-Face biometric engine for this specific case as the system is running in production for on-boarding processes for banks and insurance companies.



2.2. Antispoofing techniques (optional)

It can check that the person is real and alive using different techniques such as antispoofing.

das-Face uses three types of anti-spoofing techniques:

- Print attack (photo to photo).
- Replay attack (photo to screen).
- Duplicate attack (The compared photos are the same image)

3. System performance

Das-Face is the **proprietary** facial biometric engine developed by Veridas, using Deep Learning and AI technologies.
The system is in continuous improvement thanks to its intensive use in production environments.
Das-Face has been tested using the popular LFW dataset (13.223 pictures of 5.749 different people.
Under this test conditions, das-Face is compared with other popular biometric engines:

a. FaceNet + Picasa face alignment (Google) [1]: 99.63% +/- 0.09
b. **das-Face (Veridas) : 99.52% +/- 0.37**
c. FaceNet (Google) [1]: 98.87% +/- 0.15
d. Human : 97.53%
e. DeepFace (Facebook) [2]: 97.35% +/- 0.25
f. OpenFace: 92.92% +/- 1.34

[1] F. Schaff, D. Kalenichenko, J. Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering".
[2] Y. Taigman, M. Yang, M. Ranzato, L. Wolf. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification"

The computational times to **create** a mathematical descriptor vector are:

- If ran on CPU: 0,875 seconds
- If ran on GPU: 0.375 seconds

The computational times to **compare** several mathematical descriptor vectors are:

- If ran on CPU: 13.000 comparisons per second
- If ran on GPU: 50.000 comparisons per second

Annex 1: API operation

Requests to the server can be in application/json, and some endpoints accept as well multipart/form-data. Responses are always JSON, and in case of a failure, the server response is a JSON with the following fields:

Field	Required	Description
exception	yes	Error code, for example: `FormValidationError`, `FaceNotFoundError`, etc.
message	yes	A message providing more information about what went wrong.
errors	no	A list of field errors used in the case of the `FormValidationError` error.

Example:

```
{
      “exception”: “FaceNotFound”,
      “message”: “Unable to locate a face in the image”
}
```

1.1. API Endpoints

This microservice exposes an API with the following features:

Face similarity between two face images: The microservice receives a POST request with two image files containing one face per each, and returns a similarity value indicating how similar are both faces**POST** `/face_similarity_one2one`

Request:

Content-Type: application/json, multipart/form-data

Name	Req.	Type	Description
anchorImage	yes	file	As multipart/form-data, it should be a file, as application/json, the file content encoded in base64.
targetImage	yes	file	As multipart/form-data, it should be a file, as application/json, the file content encoded in base64.

Response: application/json

Returns the confidence (or similarity) between both faces, being more similar as much close this number is to one. The number is in range [0,1].

```
{
        "confidenceNumber": 0.4
}
```

Errors:

Code	HTTP Status	Message
`UnexpectedError`	400	An unknown error has occured while processing the request
`FaceNotFoundError`	400	No face was located on any of given images
`FaceAlignemntError`	400	Unable to locate face key points on any of located faces
`MoreThanOneFaceError`	400	More than one face was found on any of given images

- **Is alive**: The microservice receives a GET request with no params, and returns a 200 status code indicating that the server is up.

`GET /alive`

Errors:

Code	HTTP Status	Message
`UnexpectedError`	400	An unknown error has occured while processing the request

- **Duplicate attack check**: The microservice receives a POST request with two face images and returns the confidence number which represents the duplicate attack value. A duplicate attack consists on sending two images containing the same photo, for instance, one image could be a VISA photo and the other a photo of a passport issued using previous photo, therefore, both images contain the same photo.

POST `/check_duplicate_attack`

Request:
Content-Type: application/json, multipart/form-data

Name	Req.	Type	Description
anchorImage	yes	file	As multipart/form-data, it should be a file, as application/json, the file content encoded in base64.
targetImage	yes	file	As multipart/form-data, it should be a file, as application/json, the file content encoded in base64.

Response: application/json

Returns the confidence of both images being different, that is, the closer to 0, the more likely it is a duplicate attack; the closer to 1, the most likely it is not.

```
{
        "confidenceNumber": 0.8
}
```

Errors:

Code	HTTP Status	Message
`UnexpectedError`	400	An unknown error has occured while processing the request
`FaceNotFoundError`	400	No face was located on any of given images
`FaceAlignemntError`	400	Unable to locate face key points on any of located faces
`MoreThanOneFaceError`	400	More than one face was found on any of given images

- **Similarity between a face photo and a face video**: The microservice receives a POST request with a MP4 video and a face photo, and returns the similarity between the face on the video and the face on the image

POST `/video_face_similarity`

Request:
Content-Type: application/json, multipart/form-data

Name	Req.	Type	Description
anchorImage	yes	file	As multipart/form-data, it should be a file, as application/json, the file content encoded in base64.

videoTarget	yes	file	As multipart/form-data, it should be a file, as application/json, the file content encoded in base64.

Response: application/json

Returns the confidence of both faces being different, that is, the closer to 1, the most likely the face in the video is the same in the image.

```
{
        "confidenceNumber": 0.8
}
```

Errors:

Code	HTTP Status	Message
`UnexpectedError`	400	An unknown error has occured while processing the request
`FaceNotFoundError`	400	No face was located on any of given images
`FaceAlignemntError`	400	Unable to locate face key points on any of located faces
`MoreThanOneFaceError`	400	More than one face was found on any of given images
`ZeroLengthVideoError`	400	The video is empty

- **Authenticity of a selfie photo:** The microservice receives a POST request containing a selfie image and it the confidence of the system about the authenticity of the photo. That means, the closer the value to 0, the most likely it is a spoofing attempt.

POST `/photo_authenticity`

Request:

Content-Type: application/json, multipart/form-data

Name	Req.	Type	Description
videoTarget	yes	file	As multipart/form-data, it should be a file, as application/json, the file content encoded in base64.

Response: application/json

Returns the confidence of the given image to be authentic, that is, the closer to 0, the most likely it is a spoofing attempt.

```
{
```

“confidenceNumber”: 0.8

}

Errors:

Code	HTTP Status	Message
`UnexpectedError`	400	An unknown error has occured while processing the request
`FaceNotFoundError`	400	No face was located on any of given images
`FaceAlignemntError`	400	Unable to locate face key points on any of located faces
`MoreThanOneFaceError`	400	More than one face was found on any of given images