Development of Computer Vision tool for face recognition and identity authentication

Company supervisor: Christophe Lanternier
University supervisor: Anas Barakat

Outline

01 PRESENTATION OF THE PROBLEM

02 OUR SOLUTION

03 FINAL RESULT

Outline

01 **PRESENTATION OF THE PROBLEM**

02 OUR SOLUTION

03 FINAL RESULT

Presentation of the problem

Ubble.ai is an IT company working in the field of identity authentication through computer vision and machine learning systems.

In order to meet new customer demands, Ubble would like to add something new to its product: the face-authentication. This new brick would give the possibility to its customers, for example modern digital car rental companies with driver to ensure that a driver's account is correctly used by the same driver every time.

The main objective of this project is to build a face matching algorithm reliable enough to send automatic responses to their customers.

How should it work?

Ubble asks for a special procedure when a new customer is onboarding for the first time. He needs to ensure the authenticity of his identity: this step is done through confirming his identity though a video of his document and his face.

Thanks to this video, Ubble will be able to verify if the new face matches the one that was submitted in the first step.

If the comparison between these two videos gives a match between the driver's face and the account's face, the driver will be able to continue to drive.

Presentation of the Problem

Database research

In order to have scientific evidence of our algorithm performances, we need to perform a test on a big dataset, that includes faces in many different positions, situations, scenes. We want to simulate something that is as close as possible to a real case.

The characteristics we looked for:

- Video-oriented dataset
- High Scale
- Strong Heterogeneity
- Different facial expressions
- Different light conditions and environments

Presentation of the Problem

Related Work

The existing algorithms used in face recognition exploit Machine Learning function in combination with Computer Vision methods. We also analysed some unsupervised methods.

We took inspiration from some of them:

- Eigenfaces
- Self-organizing Maps for image clustering
- Local Binary Patterns Histograms
- DeepFace

Outline

01 PRESENTATION OF THE PROBLEM

02 OUR SOLUTION

03 FINAL RESULT

The Network

The network architecture used is based on ResNet-34 from the Deep Residual Learning for Image Recognition paper by He et al.

The technique used to compare encodings is called representation learning.

This kind of practice accepts a single input image and outputs a real-valued feature vector for that image (128-d).

The process of training a convolutional neural network to output face embeddings requires also a lot of computational power.

Architecture Stack

The baseline of net was mainly inspired by the philosophy of VGG nets. The convolutional layers mostly have *3×3* filters and follow a simple design rules: when the feature map size is halved, the number of filters doubles.

The network ends with a global average pooling layer and a 1000 fully-connected layer.

This model has fewer filters and it is less complex. The key changes in relation to VGG relate on the insertion of shortcut connections which turn the network into its residual version.

Architecture Stack

Using residuals, when the dimensions increase, we consider two options: either the shortcut still performs identity mapping with zero-padding, either we could perform a projection following the formula:

$$y = F(x, Wi) + Wsx.$$

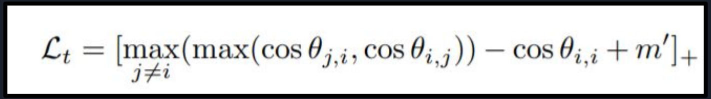
During training, scale augmentation is performed, resizing an image with its shorter side randomly sampled and their per-pixel mean is subtracted. Batch normalization right is applied and all residual nets are trained from scratch.

The optimizer used is SGD, with a mini-batch size of 256. The learning rate starts from 0.1 and is divided by 10 when the error does not decrease anymore. We use a weight decay of 0.0001 and a momentum of 0.9.

Architecture Stack

Anyway, the version used by the algorithm we selected is slightly different. Some layers are again removed.

The network train is then customized using a structured metric loss that tries to project all the identities into non-overlapping balls of radius 0.6. The loss is basically a type of pairwise hinge loss that runs over all pairs in a mini-batch and includes hard-negative mining at the mini-batch level:



$$\mathcal{L}_t = [\max_{j \neq i}(\max(\cos\theta_{j,i}, \cos\theta_{i,j})) - \cos\theta_{i,i} + m']_+$$

OUR SOLUTION

Face Crop Resolution filtering

The aim of this logic was removing background faces.

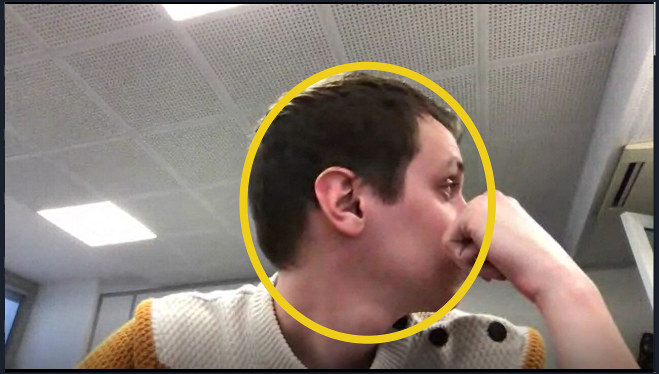
We computed the each face crop dimension and the original video one. The filtering values was selected from the dimension of the biggest face in the video. All the face boxes with a dimension lower than the 50% of the biggest one were discarded.

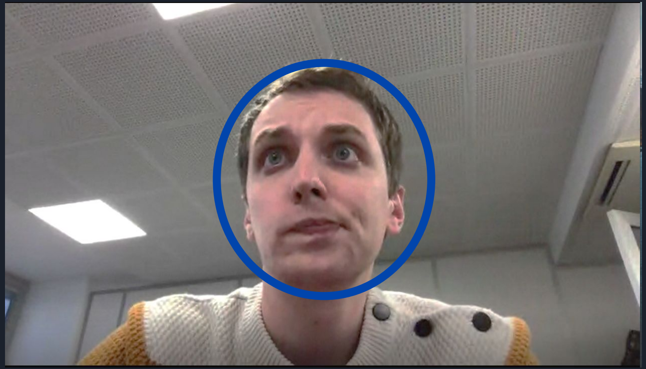
OUR SOLUTION

Front Facing picture filtering

The aim of this logic was detecting when a frame depicted a profile-side face.

By detecting landmarks - most peculiar face traits - we stated if the face depicted was looking at the camera or not. If a frame depicts a profile, we discarded the crop and informed the user to look at the camera.



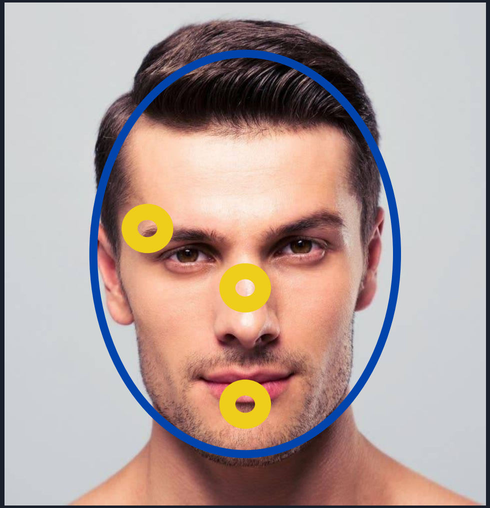


OUR SOLUTION

Main box and Landmark alignment

This logic helped in improving the robustness of the algorithm. The biggest face crop was extracted from a frame if and only if this main box had a landmark associated to it.

The point behind the smart strategy we developed is that every point inside the landmark dictionary should be inside the box coordinate, in order to state the landmarks appartenance to that block. Since landmarks are a quite big structure we just checked few determinant points: *nose_bridge*, *bottom_lip*, *right_eyebrow*.



OUR SOLUTION

Video confirmation logics

Here we want to compare the reference video (the one saved during the onboarding of the user) with a new one in order to decide if the two persons in the video are the same or not.

We have two different sets of encodings: the one related to the reference video (stored in mass memory) and the one that are generated directly from the input stream video after the filtering.

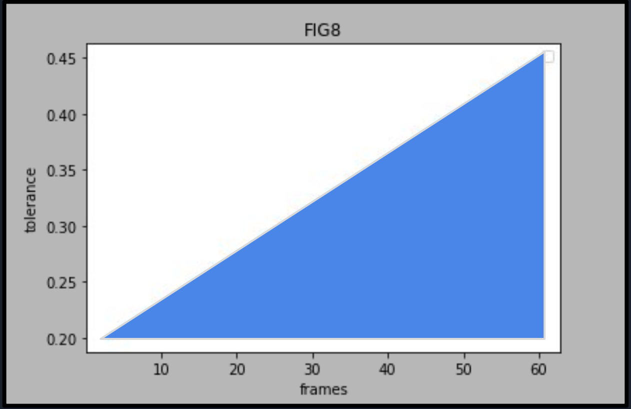
Since the encodings are here described by a 128-d vector we compute the ‘Euclidean Distance’ between them :

$$d\left(p,q\right)=\sqrt{\sum_{i=1}^{n}\left(q_i-p_i\right)^2}$$

Video confirmation logics - real time

For a real time application we need to give an answer before the end of the video. to do that we have two criteria:

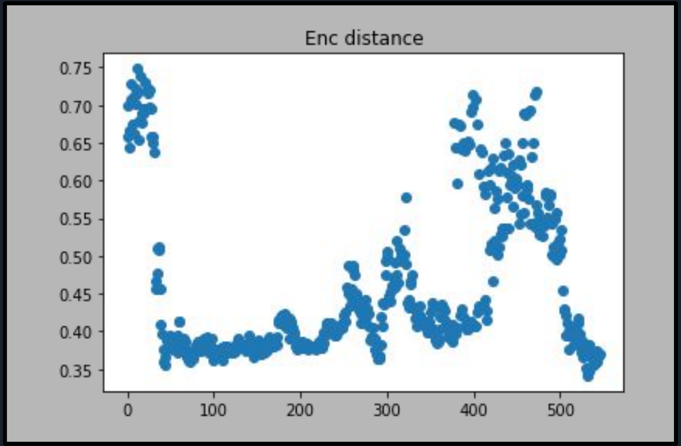
- Early stopping criteria for negative answer: triggered if after a certain number of frames, the number of distances lower than the tolerance is less than a fixed threshold.
- Early stopping criteria for positive answer: if the value of the average of the distances is in the area under the line, a positive answer is given.



OUR SOLUTION

Video confirmation logics - resolution

The quality of the images can increase through the time thanks to the automatic focus as well as a change of light exposure. The criterion described allows also videos with an increasing quality of the images through the time to be accepted.



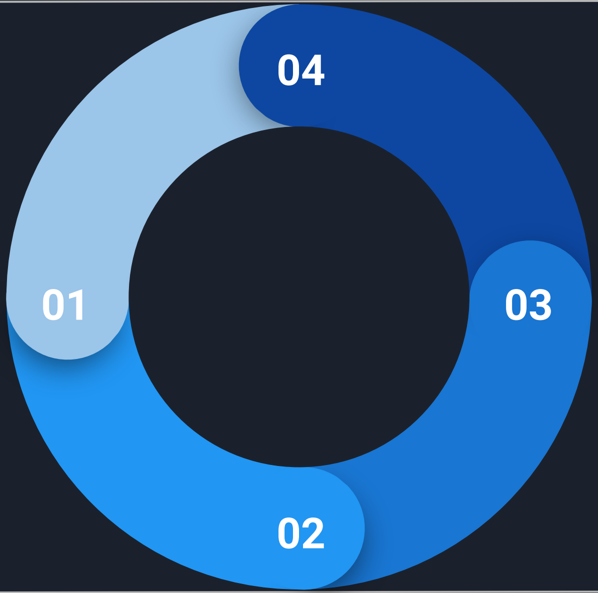
Outline

01 PRESENTATION OF THE PROBLEM

02 OUR SOLUTION

03 FINAL RESULT

General flow



Input images

We receive one frame at the time taken from the video

Filtering Logic

We retain only the frames that are able to pass the filtering

Final Answer

If we have enough frame the confirmation logic stop giving us a result

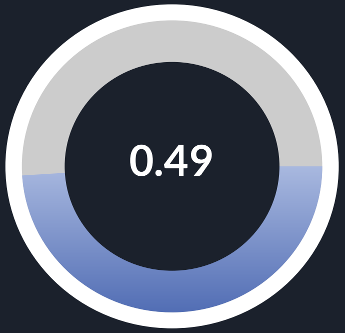
Confirmation Logic

We continuously collect frames and encoding and compute statistics on them

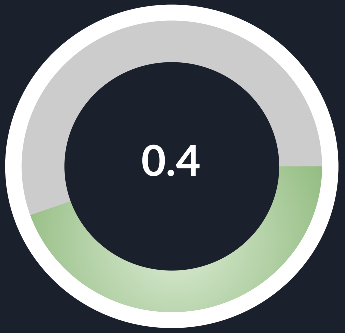
FINAL RESULT

Dataset for parameter tuning

We tuned our threshold and tolerance parameter in an efficient way, adapting it to the input videos that will be submitted by our users.



TOLERANCE



THRESHOLD

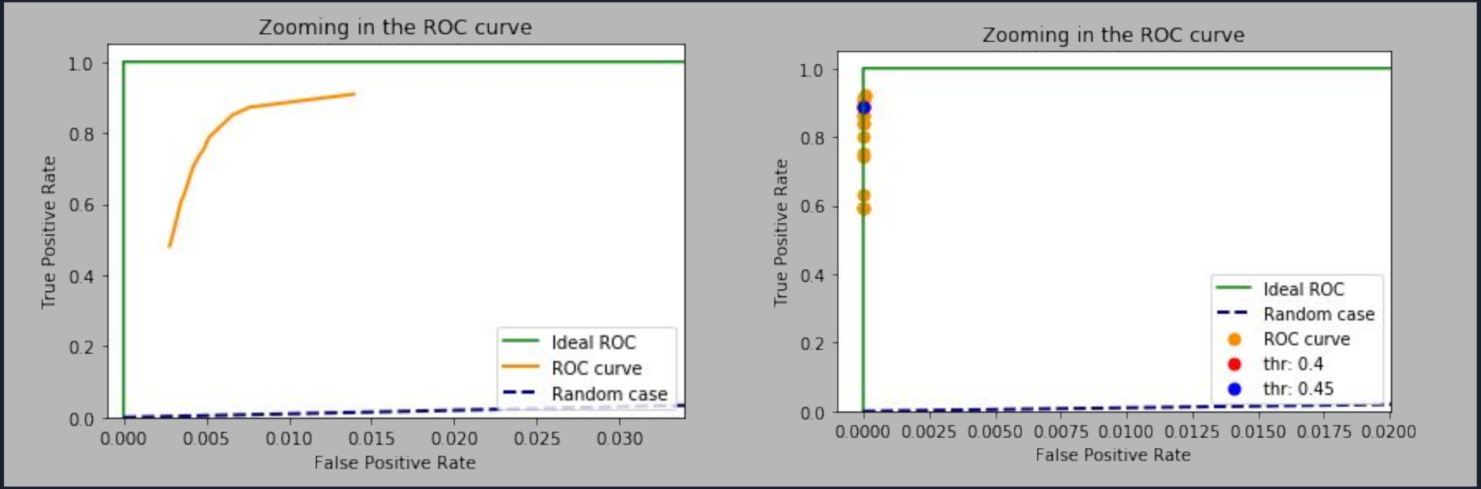
FINAL RESULT

ROC curve results comparison

ROC were produced moving the threshold. We show the evolution in ROC results, adding the solutions that we mentioned before, using the same dataset.

TAR 0.95

FAR 0.0009

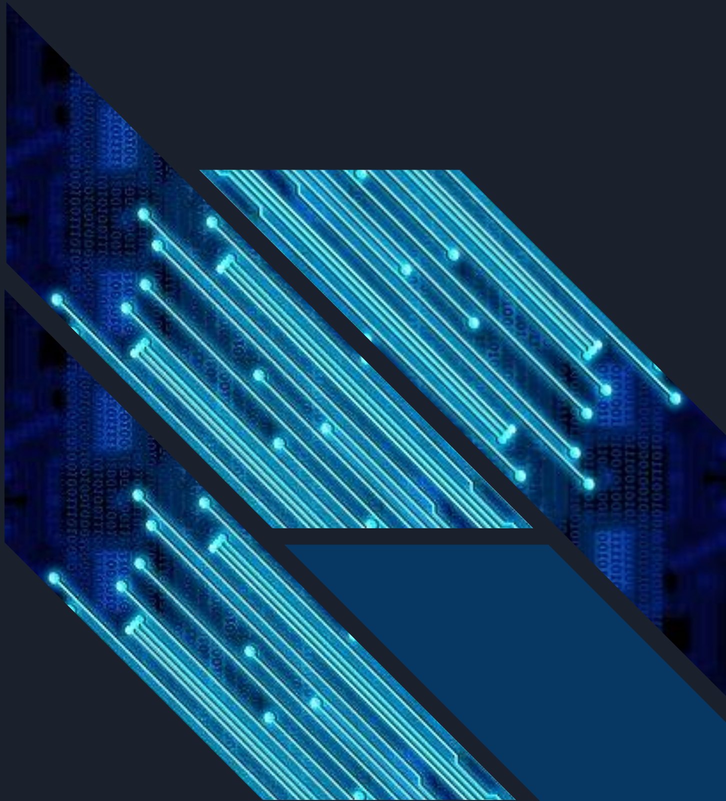


Code refactoring

The last step of the project was to rewrite the entire code following Ubble guidelines and best practices, in order to make the code directly accessible by the company and easy to integrate into their platform.

Two main class concepts:

- Encoding Class
- Authenticator Class



Future Improvements

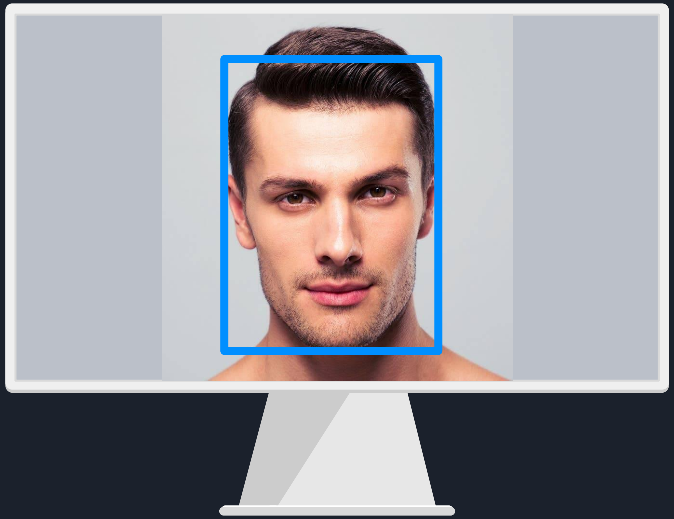
Beta testing to see if the mock-up experiments match with usage in real life.

Carrying out advanced experiments on twins and siblings.

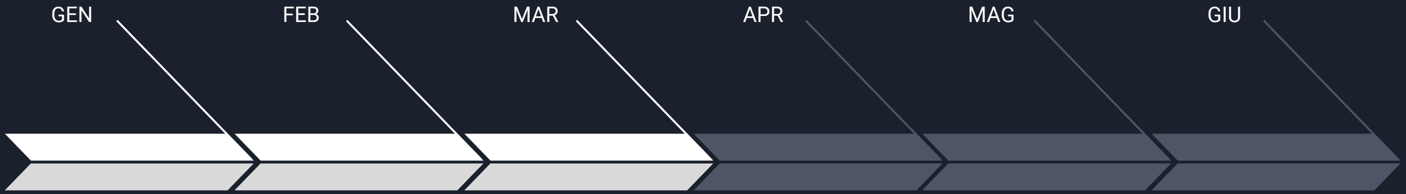
Further improvement could be related to the authentication through recognition of side-face pictures, offering our users a faster and more comfortable recognition.

Thank you!

Fabrizio Mazzone
Mattia Proietto



Tempistica del progetto



GEN

Inserisci qui il testo

Inserisci qui il testo
Inserisci qui il testo
Inserisci qui il testo.

FEB

Inserisci qui il testo

Inserisci qui il testo
Inserisci qui il testo
Inserisci qui il testo.

MAR

Inserisci qui il testo

Inserisci qui il testo
Inserisci qui il testo
Inserisci qui il testo.

APR

Inserisci qui il testo

Inserisci qui il testo
Inserisci qui il testo
Inserisci qui il testo.

MAG

Inserisci qui il testo

Inserisci qui il testo
Inserisci qui il testo
Inserisci qui il testo.

GIU

Inserisci qui il testo

Inserisci qui il testo
Inserisci qui il testo
Inserisci qui il testo.