Master thesis : Facial recognition using deep neural networks.

Auteur : Dubois, Antoine
Promoteur(s) : Wehenkel, Louis; Van Droogenbroeck, Marc
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en informatique, à finalité spécialisée en "intelligent systems"
Année académique : 2017-2018
URI/URL : http://hdl.handle.net/2268.2/4650

University of Liège

Faculty of Applied Sciences

Departement of Electrical Engineering and Computer Science

Automatic Face Detection and Recognition using Neural Networks

Graduation Studies conducted for obtaining the Master's degree in Computer Engineering by Antoine Dubois

Author:
Antoine Dubois

Promoters:
Pr. Louis Wehenkel
Pr. Marc Van Droogenbroeck

Academic Year 2017-2018

Abstract

Innovation or the introduction of the new. Innovation has always been the motor of civilization by introducing new ideas to solve the countless problems that such a complex system encounters. It has spurred economic growth and brought the comfort we live in today. In my opinion, in Belgium, particularly in the south, we have often rested on our laurels and forgot to innovate. This observation fostered my choice to undertake this thesis in the context of the RAGI project. This innovative project, relying on academic research to produce a commercial product, is developing an "intelligent system for recognition, welcoming, and guidance". Innovation in this project comes partly from the use of a machine learning technique called automatic face recognition to be able to identify people and guide them in a building. The goal of this thesis is to study this concept to help the team working on RAGI take appropriate decisions. To achieve this goal, I search for and analyze state-of-the-algorithms in both face detection and face recognition. The research for algorithms goes through the analysis of recent benchmarks, two of which (i.e. WIDER FACE [111] and MegaFace [47]) are also used for evaluating those algorithms. Simultaneously, this work points out the difficulties to perform such a research and testing process. The results on these benchmarks allows to determine which algorithms perform better, that is to say SSH [65] for detection and both Dlib-R [20] and ArcFace [21] for recognition. For detection, the influence of facial attributes such as pose, size or blur is explored. Finally, to have more relatable results with regards to the RAGI project, we designed a specific dataset on which the same algorithms are tested. Composed of 494 frames with 3561 annotated faces from 13 different identities, it allowed us to study other parameters while confirming the results obtained on the publicly available datasets. All those tests are performed with algorithm efficiency in mind and computation time measurements show that the best techniques tend to work slower but that they can achieve practical execution times.

Acknowledgments

First, I would like to thank Professors Louis Wehenkel and Marc Van Droogenbroeck for accepting to be my supervisors for this thesis and Pierre Geurts and François Van Lishout for being part of my thesis jury. I am especially grateful to mister Wehenkel who followed me through this work and gave me useful advice.

I am also thankful to the RAGI team members, François Van Lishout, Linda Wang and Tom Ewbank for lending me one of their computer as well as the constructive exchange we had on the project.

Then, I want to warmly acknowledge the group of volunteers that accepted to collaborate in the construction of the RAGI dataset. In addition to the RAGI team members, I thank Denis Bourguignon, Michael Di Bartelomeo, Laura Ferire, Marc Frédéric, Damien Gérard, Samuel Hiard, Benoît Mattheus, Olivier Suplis and Nicolas Vecoven.

Finally, I would like to give thanks to my family members for supporting me throughout this work.

Antoine Dubois

Contents

Chapter 1

Introduction

This master thesis was undertaken in the context of the RAGI project. This project, financed by the Walloon Region, is currently conducted by a team of 3 people: François Van Lishout (project manager), Linda Wang and Tom Ewbank and under the supervision of Damien Ernst, Marc Van Droogenbroeck and Louis Wehenkel, Professors at the University of Liège.

The goal of the RAGI project is to develop "an intelligent system for recognition, welcoming and guidance" (which is translated from the French "un système de Reconnaissance, Accueil et Guidance Intelligent" or RAGI) of users of a building. This system will allow people working in a building to better collaborate (via collective intelligence augmentation through artificial intelligence) while also providing a personal service to visitors. In terms of schedule, a complete operational beta-version of the system should be installed in the Montefiore building for the start of 2019, before entering a commercialization phase.

François Van Lishout gives the following description[1] of the technical goal of the system:

> "The RAGI system is made up of welcome stations, allowing users to search for people or premises while also providing explanations on the internal workings of the system. When a user comes in the field of view of the system, RAGI uses face recognition to identify him/her. If this person is unknown to the system, he/she can register either with his/her real identity or as a temporary avatar. RAGI has two possibilities to guide a user: via robots or via digital screens. On the one hand, Loomo robots[2], programmed by the RAGI team, will be able to guide users from any welcome station to any place or registered staff member. The guidance via screens, on the other hand, will rely on digital screens or tablets installed at crossroads of the building. Once a guided user approaches one of these crossroads, he/she will be identified and a direction indication will be displayed on the screen. In addition to that, RAGI has the ability to localize, in real time, all the staff members that would have given their prior approval. This allows RAGI to effectively guide a user to the person he/she is searching for. Staff members will have access to an application that allows them to deal with their private information (e.g. choose not to be localized anymore)."

As can be inferred from this description, automatic face recognition is a cornerstone of the project. The face recognition system should be both robust and efficient. The robustness is

[1] This description is translated from French.

[2] Segway Robotics website: `https://www.segwayrobotics.com/`

needed to ensure that the right person is being guided while efficiency is required to make the system interactive. Face recognition corresponds to a pipeline whose two main components are face detection followed by the face recognition per-say. Currently, face detection is done in RAGI by using the OpenCV[3] implementation of the Viola&Jones algorithm [94], while face recognition is based on the OpenFace library[4].

The goal of this thesis was to search for and evaluate state-of-the-art algorithms in both face detection and recognition in order to allow the RAGI team to take informed decisions regarding the implementation choices of their computer vision system. To reach this aim, I decided to work in a scientific way so that the conclusions of this report can be directly exploited and if need be, that my researches and experiments can be reiterated to test new algorithms.

The structure of this report follows the systematic procedure that I used.

- Chapter 2 follows this introduction by laying an informative ground on useful notions while also highlighting some practical decisions.
- The aforementioned procedure starts really with Chapter 3: Benchmark Analysis. Benchmarks are at the same time a means of evaluation and a source of consolidation of algorithms' capabilities, constituting an ideal research base for selecting algorithms to test.
- The selection of the algorithms to focus on is performed in Chapter 4 and the selected algorithms are then described in Chapter 5.
- Two benchmarks, one for face detection and one for face recognition, chosen in Chapter 3, are used in Chapter 6 to evaluate the selected algorithms.

  It is noteworthy that each of these four chapters (Chapters 3-6) is divided in two parts: face detection and face recognition.
- This division continues in the penultimate chapter: Application to RAGI. The natural step after testing algorithms on publicly available datasets was to test them in practical conditions by building a dataset specific to the RAGI system. Chapter 7 describes how this dataset was built, its characteristics and how face detection and recognition algorithms perform on it.
- I conclude this thesis by summarizing the contributions and findings and providing directions for further work.

The results of this work have also been summarized in a scientific abstract "Integrating facial detection and recognition algorithms into real-life application" [92] by F. Van Lishout, A. Dubois, M. L. Wang, T. Ewbank and L. Wehenkel presented at the "Workshop on the Architecture of Smart Cameras" (WASC) 2018 occurring on 27-28 June in Coimbra, Portugal.

[3]OpenCV: `https://opencv.org/`
[4]OpenFace library: `http://cmusatyalab.github.io/openface/`

Chapter 2

Background

This chapter aims at providing an overview of the concepts underlying this thesis while also providing some information about practical choices that were made. The two first sections introduce face detection and recognition through their definitions, evaluation methods and brief reviews of methods that were deployed to solve those problems. Section 2.3 and 2.4 highlight particularities of the context in which these two concepts are studied. I then explain how the overwhelming presence of deep learning in those domains has influenced the realization of this thesis and I finish by some considerations about computational efficiency in section 2.6.

2.1 Face detection

2.1.1 Definition

As presented in [111], "given an arbitrary image, the goal of face detection is to determine whether or not there are any faces in the image and, if present, return the image location and extent of each face [109]". Practically, face detection algorithms will aim at generating bounding boxes (often rectangular or elliptical) around all the faces in the image and only around faces.

2.1.2 Evaluation

With regard to this definition, a perfect detector should correctly detect all and only faces in an image. To evaluate algorithms, we, therefore, have to define two aspects. First, we must define what is the meaning of a 'correct detection'. Second, we must use evaluation metrics that reflects correctly both the 'all' aspect (i.e. assuring there are no false negatives, and the 'only' aspect (i.e. assuring there are no false positives) of face detection, so as to choose a sought compromise between those two types of errors.

To define when we consider a detection to be correct, we must beforehand decide how a detection is represented. As reported in [43] and illustrated in Figure 2.1, there is no clear consensus and "representations vary from image regions such as rectangular and elliptic regions or patches of arbitrary shape to locations of facial landmarks. Combinations of those two can also be found in the literature with additional features such as head pose [120]". Most of the recent datasets (e.g. [111], [24], [120], [106], [108], [49]) use rectangular bounding boxes as ground-truth which are usually reported using the pixel coordinates of their upper-left corner, their height and their width. This representation will be used throughout this master thesis.

Figure 2.1: Illustrative figure from [43]. The red ellipses are annotations while the squares correspond to the outputs of two different detectors. We can see that except for the one false positive, the detections are correct but are not represented in the same way by the two algorithms and are also very different from the annotations.

Choosing to represent detection with rectangular boxes still leaves open a large choice in terms of dimensions of these boxes, the main question being how much of the face should be included. There is no clear consensus on this question either. To confirm that a detection is correct, most benchmarks rely on the bounding box overlap ratio or intersection over union measure, or IoU for short. As defined in [24], "the overlap ratio between a predicted bounding box B_p and ground truth bounding box B_{gt} is given by

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{2.1}$$

where $B_p \cap B_{gt}$ denotes the intersection of the predicted and ground truth bounding boxes and $B_p \cup B_{gt}$ their union", and where the area of a region is measured by the number of pixels it contains. For all the benchmarks analyzed in this thesis (see Section 3.1), a detection is considered 'correct' when this value is greater than 50%. This choice was generally made following the PASCAL VOC [24] protocol. It is noteworthy that this protocol is initially aimed at object recognition, rather than face recognition specifically. Other objects than faces must thus be detected and the authors mention that this 50% threshold was "set deliberately low to account for inaccuracies in bounding boxes in the ground truth data, for example defining the bounding box for a highly non-convex object, e.g. a person with arms and legs spread, is somewhat subjective." In the case of face detection, this low value could possibly be contested, as faces are, oppositely, highly convex objects, but it seems like most benchmarks decided to use this value which is why I am also using it.

Once we have defined what we consider to be a correct detection, we can estimate how well a face detection algorithm performs by counting the number of true positives (i.e. detections considered as correct, noted TP), false positives (i.e. detections considered as incorrect, noted FP) and false negative (i.e. faces that were not detected, noted FN). The concept of true negative (i.e. regions that were correctly detected as non-faces, noted TN) does not apply in this context as there is a virtually infinite number of these.

Using these numbers, a perfect detector would be one that would simultaneously have 0 false negatives (correctly detecting all faces) and 0 false positives (not having any incorrect detection). This is nearly never the case and one has to make a compromise between those two

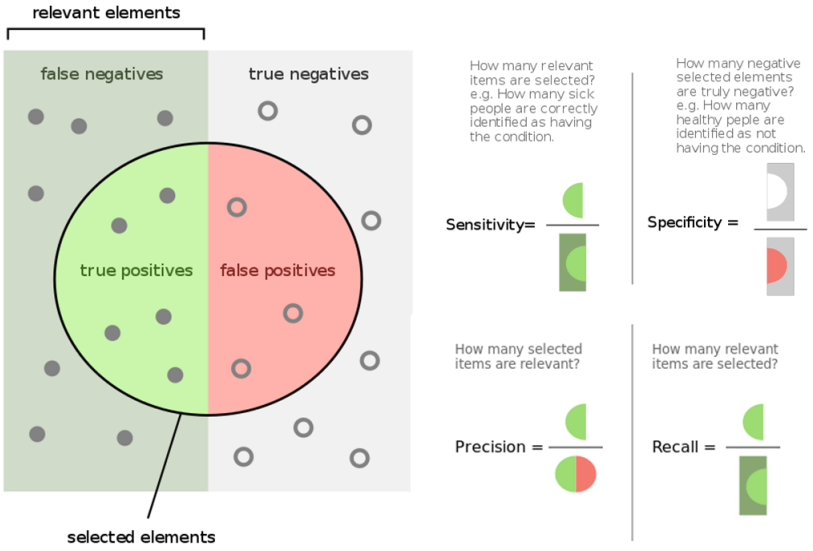


Figure 2.2: Combination of images coming from the Wikipedia pages 'Sensitivity and Specificity' and 'Precision and Recall' showing how the different binary classification measures are obtained.

values. To quantify this compromise, the benchmarks propose generally two approaches.

The first one is to use adapted Receiver Operating Characteristic (ROC) curves. A ROC curve is generally created by plotting the True Positive Rate, also called sensitivity or recall and computed as TP/(TP+FN), against the False Positive Rate, also called specificity and computed as FP/(FP+TN), as shown in Figure 2.2. However, as mentioned before, the TN does not make sense in our context. Therefore, benchmarks like [43] or [108] use on the x-axis the total number of False Positives or the number of False Positives per image. This curve can also be summarized using the area under the curve, or AUC, which is simply the integral of that curve.

The second approach is based on precision-recall curves where precision is computed as TP/(TP+FP). Contrarily to the previous case, this curve does not need to be modified as it does not make use of TN. It can also be summarized using the Average Precision or AP defined in [24]. It is defined as the set of mean precision at a set of eleven equally spaced recall levels {0, 0.1, ..., 1}:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_{interp}(r). \tag{2.2}$$

The precision at each recall level r is *interpolated* by taking the maximum precision measured for a method for which the corresponding recall exceeds r: $p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$ where $p(\tilde{r})$ is the precision at recall $\tilde{r}$.

I had no clear assumptions on which of those two approaches was more appropriate and I wanted to choose one not to overwhelm this thesis with too many results. However, the authors of [24] mention that based on previous experience, using precision-recall curves was used "to improve the sensitivity of the metric [...], to improve interpretability (especially for image retrieval applications), to give increased visibility to performance at low recall" and they showed that using either of the two the "ranking of participants was generally in agreement but

that the AP measure highlighted differences between methods to a greater extent". The choice of precision-recall curves for evaluating face detection seems, therefore, more appropriate.

2.1.3 Brief literature review

Anyone who has some knowledge in the field of face detection would have heard of the Viola&Jones Haar-cascade algorithm [94] which is often considered as the first practical face detector. As mentioned in [44], "the use of hand-designed features methods continued with for instance SURF [6], LBP [4] or HOG [19]. These features were combined with Deformable Parts Model [26] to produce significant advances." However, the real burst in performance came with the renewed use of neural networks based on deep architectures. According to [44], "CNN's had already been applied to face detection as far back as 1994 in [91]" but since 2012, deep architectures started being used such as in [29] or [54]. I will go into more details about the former architecture in section 5.1.3.

These techniques are designed to be applied to 2D images. In parallel to this, research on 3D or 2D+3D face detection has also emerged. As mentioned in [51], 2D and 3D approaches are complementary in the sense that "3D data compensates for the lack of depth information in a 2D image while it is also relatively insensitive to pose and illumination variations". It is therefore not surprising to see that numerous researchers have tried to use 3D information either alone such as in [18] by doing curvature analysis or by combining the two such as in [51]. [75] takes even another path where they apply 2D detections technique to 3D data which is preprocessed via orthogonal projection. These techniques could emerge due to the arrival of affordable 3D acquisition systems. However, the main burden remains "the intrinsic complexity in representing and processing 3D data" [34]. This complexity comes with a need for large amounts of data, which does not seem to be tackled for now. Due to this, I decided to focus only on 2D face detection techniques

2.2 Face recognition

2.2.1 Definition

Face recognition is generally divided into two sub-categories. On the one side, face verification (or 1:1 face recognition) consists in checking if a face corresponds to a given identity. On the other side, face identification (or 1:N face recognition) consists in finding the identity corresponding to a given face. Face recognition can also be divided in terms of evaluation protocol. "Either algorithms are tested under the closed-set protocol or under the open-set protocol. In the former, the testing identities are the same as the training ones and face recognition can then be assimilated to a classification problem. In the latter case, the testing identities are usually disjoint from the training ones. The problem becomes more one of encoding faces into a discriminative feature space." [57]

In the case of RAGI, we are typically trying to solve a face identification problem under the open-set protocol. Indeed, the goal of the project is to identify any registered person that comes in front of the welcome station without asking him to identify himself. Moreover, there is no plan to retrain the face recognition algorithm each time a new person registers in the system and it is thus more appropriate to consider that the testing identities will be mostly

different from the training ones.

2.2.2 Evaluation

There are several ways to evaluate the quality of face recognition algorithms but we need to go through some terminology and methodology before going further into this. First, the faces for which we want to obtain the identity are generally called the 'probes', 'probe faces' or 'probe set'. The goal is to compare these probes to our database of faces, generally called 'gallery' or 'gallery set', and find for each of those faces at least one face of the same identity in the gallery, if there is such a face. To achieve this, face recognition algorithms produce for each face, both in the probe set and in the gallery set, a vector of features. Then using a distance measure, one can rank all the faces in the gallery from closest to furthest for each given probe.

To evaluate the performances of an algorithm, one then typically looks at each rank in this ordering if the true identity of the person was found before this rank or not. This leads to a series of measures called respectively rank-1, rank-2, ..., rank-N identification rates (or performance). More specifically, the rank-N performance is equal to the percentage of probes for which a face from the gallery corresponding to the right identity was found in at least one of the N first ranks. These measures can be combined into a Cumulative Matching Characteristic (CMC) curve that shows identification rates for each possible rank (1 to the size of the gallery). Another aspect that can be analyzed is the performances of the algorithm with regards to the size of the gallery. Indeed, the more identities it contains the more difficult it is to discriminate between different identities in the feature space. This evaluation scenario is the one proposed in the MegaFace challenge [47].

A more detailed approach of the quantification of the performances of face recognition algorithms is given in the Face Recognition Vendor Test (FRVT) [31]. It starts by referring to the two types of errors a face identification algorithm can encounter:

- **False alarms** or **Type 1 errors**: When the algorithm identifies someone who has never been seen before
- **Misses** or **Type 2 errors**: When the algorithm returns an incorrect identity

We can notice that rank performances and CMC take into account the second type of error but not the first one. FRVT proposes therefore to introduce two measures defined based on the following setting:

- "The search [for the identity of a given face] is conducted into an enrolled population of N identities
- The algorithm returns the L (generally equal to N) closest candidate identities which are ranked by their closeness in descending order.
- The human analyst can choose to analyze those top L candidates or just the top R ones of this list, or the ones that are below a distance threshold T."

From there, they define two error measures; "False positive identification rate (FPIR) for false alarms and the False Negative Identification Rate (FNIR) for misses which are computed as:

$$\text{FPIR(N, T, L)} = \frac{\begin{array}{c}\text{\# nonmate searches where at least one enrolled candidate is} \\ \text{returned at or above threshold T}\end{array}}{\text{\# nonmate searches attempted}} \tag{2.3}$$

$$\text{FNIR(N, R, T, L)} = \frac{\begin{array}{c}\text{\# mate searches with enrolled mate found outside top R} \\ \text{ranks or score below threshold, T}\end{array}}{\text{\# mate searches attempted}} \tag{2.4}$$

where a nonmate search corresponds to a trial to identify a face that is not in the database and mate search corresponds to trying to identify a registered person."

These two measures are often combined to obtain a Detection Error Tradeoff (DET) curve. One can use the "True Positive Identification Rate instead of FNIR (TPIR = 1 - FNIR) if we want to characterize the hit rate rather than the miss rate". We can note here that rank-N performance is actually equivalent to TPIR(N, R, 0, L) and that the CMC is given by TPIR(N, R, 0, L) for all values of R.

In the context of RAGI, the two type of errors are important to consider. Indeed, one does not want to identify someone who is not registered in the system while at the same time identifying correctly registered people. To deal with the first error type, one needs to fix a threshold on the distance measure under which the identification is considered as not reliable. The second error is a bit more complex to deal with because one needs to take into account the rank. Indeed, depending on the way we select the identity that will be displayed, one might need to focus on different ranks. For instance, if we only use the identity at rank 1 to make the prediction, then one needs to maximize rank 1 performance. However, if we use identities at the N first rank and use, for example, the most current identity inside this selection as prediction then we need to maximize all the rank-i performances from rank-1 to rank-N.

2.2.3 Brief literature review

The history of face recognition techniques is quite similar to the one of face detection going from hand-crafted features to features generated by CNN's while also going through the use of other machine learning techniques. As mentioned in [5], "face recognition research can be characterized into feature-based and holistic approaches. The earliest work in face recognition was feature-based and sought to explicitly define a low-dimensional face representation based on ratios of distances, areas, and angles [45]. An explicitly defined face representation is desirable for an intuitive feature space and technique. However, in practice, explicitly defined representations are not accurate. Later work sought to use holistic approaches stemming from statistics and Artificial Intelligence (AI) that learn from and perform well on a dataset of face images. Statistical techniques such as Principal Component Analysis (PCA) [37] represent faces as a combination of eigenvectors [80]. Eigenfaces [90] and fisherfaces [7] are landmark techniques in PCA-based face recognition. Lawrence et al. [52] present an AI technique that uses convolutional neural networks to classify an image of a face."

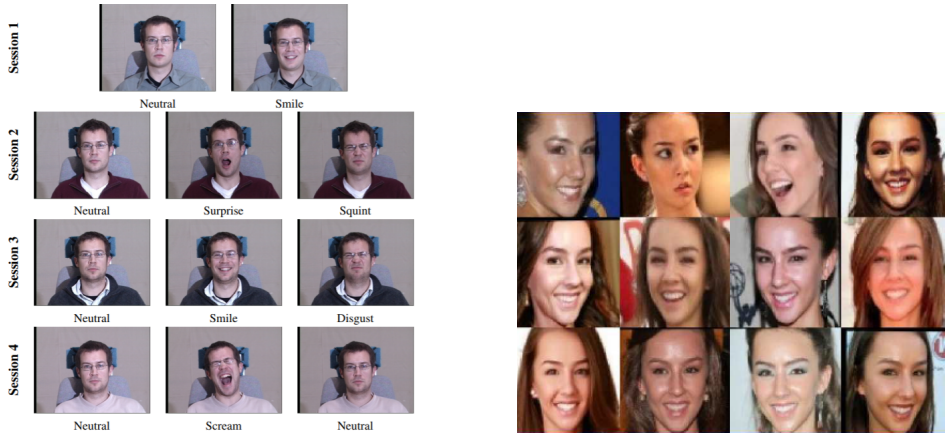
This last technique was presented in 1997 but once again the use of CNN's has massively increased in recent years due to their good performances in such tasks. One of the most recent and best-known such approaches is Google's FaceNet [74] but new networks have been developed as explained in section 5.2.

Finally, as for face detection, research has also turned to 3D imagery. Techniques also vary between combining 2D and 3D ([51], [81]), transforming 3D data to 2D data ([34]) or extracting features directly from 3D data ([69], [64], [104]). These three last techniques are actually initially designed for object recognition and based on neural networks. They propose interesting ways on how to use neural networks on 3D data. However, due to lack of data and

also because there seems to be much more research in 2D face recognition, I decided to not take into account 3D face recognition techniques.

2.3 The in-the-wild concept

To support research in the fields of face detection and face recognition, a series of datasets have been released over the years. However, until 2007 and the arrival of the 'Labeled Faces in the Wild' (LFW) face recognition dataset, "most of these databases had been created under controlled conditions to facilitate the study of specific parameters on the face recognition problem." [42]. This database was one of the first to be aimed at "studying the unconstrained, face recognition problem" which is also referred to as 'in-the-wild' face recognition. More specifically, "the database represents an initial attempt to provide a set of labeled face photographs spanning the range of conditions typically encountered by people in their everyday lives". This last sentence directly shows the interest of such database in evaluating the performances of algorithms designed to be used in practical environments. Another obvious advantage of these datasets is that they contain much more images. Following LFW, a lot of benchmarks, both in face detection and recognition, claimed to be in-the-wild. As this thesis is occurring in the context of a practical project, I will, therefore, focus on those datasets.



(a) Multi-PIE (b) FaceScrub

Figure 2.3: Sample images coming two face recognition datasets: (a) the constrained dataset Multi-PIE [30] and (b) the in-the-wild dataset FaceScrub [67].

2.4 End-to-end pipeline

In the previous sections, I have focused on face detection and face recognition. However, the full face recognition pipeline is generally considered to be composed of four main steps:

1. Detection
2. Preprocessing
3. Feature extraction
4. Classification

In this thesis, I consider that face recognition encompasses the three last steps even though in the literature, face recognition algorithms usually refer to and try to solve the feature extraction step. Indeed, the following classification step is simply performed using a generic nearest neighbor technique while more particular algorithms are developed for the preprocessing step. This step is generally constituted of cropping, rescaling, and alignment of the detected faces. While cropping and rescaling do not require complicated techniques, alignment is not an easily-solved problem. Practically, alignment consists in detecting a series of facial landmark (nose, eyes, ...) and then transforming the face photo so that the position of those landmarks is constant. Figure 2.4 displays an example of the alignment process used in [87]

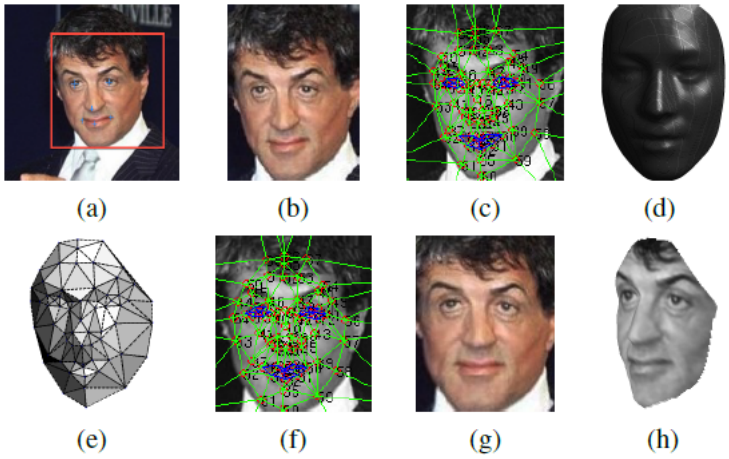


Figure 2.4: Illustrative figure from [87]: "**Alignment pipeline.** (a) The detected face, with 6 initial fiducial points. (b) The induced 2D-aligned crop. (c) 67 fiducial points on the 2D-aligned crop with their corresponding Delaunay triangulation, we added triangles on the contour to avoid discontinuities. (d) The reference 3D shape transformed to the 2D-aligned crop image-plane. (e) Triangle visibility w.r.t. to the fitted 3D-2D camera; darker triangles are less visible. (f) The 67 fiducial points induced by the 3D models that are used to direct the piece-wise affine wrapping. (g) The final frontalized crop. (h) A new view generated by the 3D model."

While I could have studied face alignment separately of face recognition by comparing the performances of different face alignment methods, I did not think it was useful. Indeed, the problem is that in the context of face recognition, we do not use face alignment on its own but as a preprocessing step of feature extraction. However, as with any algorithm trained on data that is preprocessed, better performances are obtained if the test data goes through the same preprocessing phase making the algorithm linked to a specific preprocessing algorithm. As I did not train myself any face recognition algorithm as explained in the following section, I decided to simply evaluate each tested face recognition algorithm with, or without, the alignment technique that was applied to the faces used to train that algorithm.

2.5 Using artificial neural networks

The use of artificial neural networks in this thesis relies on the reality of techniques used in face detection and recognition nowadays. As expressed briefly in the review sections, algorithms have switched mainly from hand-crafted features to deep-generated features. Most researchers justify the use of CNN's based on their state-of-the-art performances on other image-based

problems. Their choice is further justified by the realization of state-of-the-art performances in both face detection and recognition using these same techniques.

The omnipresence of neural networks has two main implications for this thesis. The multiplication of deep neural network research has been accompanied by the appearance of multiple deep learning frameworks. To cite only the ones used by algorithms mentioned later in this thesis, we have at least Caffe[1], TensorFlow[2], Torch[3], MXNET[4] and Theano[5]. These frameworks rely on platforms such as CUDA[6] and libraries like cuDNN[7]. Finally, those frameworks can be used with a series of programming languages such as C++, Python or Matlab. To wrap things up, algorithms generally rely on a specific version of each of these different parts. This myriad of components and all the compatibility issues that come with it had the major effect that I did not achieve to test many of the algorithms that I identified as testable as shown and explained in sections 4 and 5.

The second impact it had on my thesis is linked to the two pillars of deep learning: massive data and massive computing power. As I had access to neither of those and that I had a limited amount of time, I decided not to train any neural network by myself but to use already pretrained networks. This decision was also motivated by the various frameworks in which each algorithm is written. As I only have some notions in TensorFlow and Theano, I decided that trying to train the networks by myself would have been a waste of time.

2.6 Time matters

I covered the evaluation metrics for face detection and recognition algorithms. However, in a practical context, the computational efficiency of the system matters at least as much as its robustness. In this thesis, I will therefore also analyze the computing times of various algorithms. The exact description of which parts of algorithms will be timed will be explained later in this document. However, I will here briefly describe the characteristics of the computer on which the algorithms were run.

The processor's model is an Intel® Core™ i7-7820X[8]. This model is characterized by 8 cores, each possessing 2 threads, a base frequency of 3.6 GHz and 11 MB of L3 cache. The computer has 15,705 MB of physical memory with an additional 16,054MB of swap memory. Finally, it possesses two GPUs. The first one is a GeForce Gt 1030[9] with 2,001 MB of dedicated memory. However, this GPU was not used for computation purposes. The one that I used to make my test is a GeForce GTX 1080 Ti[10] with 11,177 MB of dedicated memory.

[1]Caffe - Deep Leaning Framework: `caffe.berkeleyvision.org/`

[2]TensorFlow: `https://www.tensorflow.org/`

[3]Torch - Scientific computing for LuaJIT: `torch.ch/`

[4]MXNet - A Scalable Deep Learning Framework: `https://mxnet.apache.org/`

[5]Theano: `www.deeplearning.net/software/theano/`

[6]About CUDA: `https://developer.nvidia.com/about-cuda`

[7]NVIDIA cuDNN: `https://developer.nvidia.com/cudnn`

[8]Intel® Core™ i7-7820X X-series Processor: `https://ark.intel.com/products/123767/Intel-Core-i7-7820X-X-series-Processor-11M-Cache-up-to-4_30-GHz`

[9]GeForce Gt 1030: `https://www.geforce.com/hardware/desktop-gpus/geforce-gt-1030/specifications`

[10]GeForce GTX 1080 Ti: `https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-1080-ti/specifications`

Chapter 3

Benchmark Anaysis

In this section, I make a tour of benchmarks for evaluating face detection and face recognition algorithms. The goal of this listing is two-fold. Firstly, to compare algorithms together, I need at least one benchmark for face detection and one for face recognition. As more performing algorithms have appeared, more and more difficult benchmarks were required to be able to discriminate between algorithms. With these lists, I try to understand how the benchmarks evolved and what makes the new benchmarks better, if it is the case. Secondly, to find algorithms to test, benchmarks are a first-choice source as authors of algorithms use them to prove the efficiency of their work.

3.1 Face detection

To benchmark face detection algorithms, a series of datasets have been publicly deployed. Those datasets are numerous as can attest the list on the following website: `www.face-rec.org/databases/`. Of course, a lot of those databases are irrelevant to our goal and we could probably scan them all to identify the most interesting ones. However, I choose to take a different path by trying to identify the most-frequently referenced databases in recently published algorithms or referenced by other benchmarks that I had already identified. In addition, as a reminder, I focused so-called 'in-the-wild' benchmarks. The following sections describe this list of benchmarks in publication order.

3.1.1 FDDB

This dataset was released in 2010 with the paper "FDDB: A Benchmark for Face Detection in Unconstrained Settings" [43]. The goal of this paper was to provide a new dataset for the problem of face detection in response to issues of earlier datasets (MIT+CMU [73], GENKI [1], Kodak [60], UCD [78], VT-AAST [3]) while proposing a protocol for comparing face detectors. To achieve this aim, the dataset contains "2845 images (gray-scale or colored) with a total of 5,171 faces which were all annotated using elliptical regions." The authors specify that the dataset includes "a wide range of difficulties including occlusions, difficult poses, and low resolution and out-of-focus faces" without giving a quantitative or statistic analysis of those characteristics.

FDDB is based on the images of Berg et al's dataset [10] collected from Yahoo news website[1]. The problem with this dataset is that it contains near-duplicate images (coming from

[1]Yahoo: `http://new.yahoo.com`

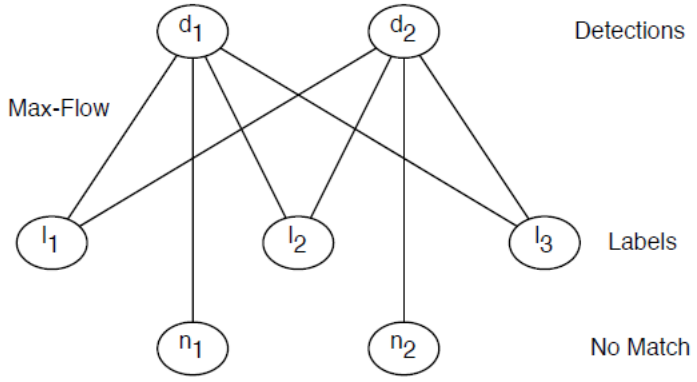


Figure 3.1: Illustrative figure from [43]. Description: "*Maximum weight matching in a bipartite graph.* We make an injective (one-to-one) mapping from the set of detected image regions d_i to the set of image regions l_i annotated as face regions. The property of the resulting mapping is that it maximizes the cumulative similarity score for all the detected image regions."

the fact that different media sources use the same image while modifying them slightly). To solve this problem, 103 clusters of similar images where computed and systematically replaced by a single image from each cluster. The dataset was then preliminary annotated by drawing a rectangular bounding box around each face and discarding face regions with height or width less than 20 pixels. In this context, the authors mention the problem of labeling certain image regions as 'face' or 'non-face' due to factors such as low resolution, occlusion, and head-pose. To solve this ambiguity, they resorted to the use of human judgment. All the images were provided to multiple human annotators with some guidelines through a web interface. Finally, the remaining face regions were annotated "using an ellipse parametrized by the location of its center, the lengths of its major and minor axes, and its orientation." The authors use this annotations scheme because they believe that "the resulting representation of a face region as an ellipse provides a more accurate specification than a bounding box without introducing any additional parameters."

In terms of evaluation, IoU (which is called ratio of intersected areas to joined areas in this case and denoted by S) is used to quantify the correspondence between the elliptical face regions. Moreover, the problem of matching detections with annotations is addressed as a maximum weighted matching in a bipartite graph as exemplified in Figure 3.1. Once this matching is done, for each pair of detection d_i and matched annotation v_i, we can compute a discrete or continuous score y_i:

- Discrete Score (DS): $y_i = \delta_{S(d_i,v_i)>0.5}$
- Continuous Score (CS): $y_i = S(d_i, v_i)$

The individual scores are then accumulated into a ROC curve. The database is provided at `http://vis-www.cs.umass.edu/fddb/`. Moreover, results are already presented for a large series of algorithms at `http://vis-www.cs.umass.edu/fddb/results.html`.

3.1.2 AFLW

Released in 2011, the Annotated Facial Landmarks in the Wild (AFLW) [49] dataset has the particularity not to be aimed at face detection specifically but oriented towards face alignment,

considered as a crucial step for face recognition. It contains 21,997 Flickr[2] images with 25,993 faces annotated with facial landmarks. Images "exhibit a large variety in face appearance (e.g pose, expression, ethnicity, age, gender)" but the only quantitative information that is given is that 56% of the faces are tagged as female and 44% as male and that 66% of faces are non-frontal. An important aspect of this database is that most of the images only contain one face (average of 1.179 faces per image).

The images were gathered from the photo-sharing platform Flickr "using a wide range of face relevant tags (e.g. face, mugshot, profile face)" which might explain the relatively low number of faces per image. The images were also manually scanned for faces. In terms of annotations, rectangular and elliptical face regions are provided. Ellipse outline "the 3D ellipsoid capturing the front of the head". The paper also explains its procedure for landmark annotation.

For evaluation, a series of tools including a SQLite Database, a Label Gui, and some programming tools are provided. The evaluation scheme follows the one of FDDB [43]. The database and tools are available at `https://www.tugraz.at/institute/icg/research/team-bischof/lrs/downloads/aflw/`.

3.1.3 AFW

As its name indicates, the "Face Detection, Pose Estimation and Landmark Localization in the Wild" [120] paper aims at more than just face detection but one part of this work is to provide a "new 'in the wild' annotated dataset" called "Annotated Face in-the-Wild" or AFW. However, the main goal of the paper is to present a new algorithm for simultaneous face detection, pose estimation and landmark localization based on tree-structured models and therefore a single paragraph is dedicated to the database.

The 205 images with 468 faces of the dataset were collected from Flickr and "contain cluttered backgrounds with large variations in both face viewpoint, and appearance" (with no quantitative measure of those attributes).

The faces are labeled with a rectangular "bounding box, 6 landmarks and a discretized viewpoint (°90° to 90° every 15°) along with pitch and yaw directions and (left, center, right) viewpoint along the roll direction.

The testing protocol is the same as the one proposed in the PASCAL VOC [24] paper and a Matlab implementation is available at `https://www.ics.uci.edu/~xzhu/face/`.

3.1.4 Pascal-Faces

As for AFW, the Pascal-Faces dataset is only part of the contributions of the paper "Face Detection by structural models" [106] issued in 2014 and little information is provided about this dataset. It contains 851 images with 1335 faces with large appearance variations (even though not quantified). The images are "collected from the test set of Pascal person layout dataset, which is a subset from Pascal VOC [24]" the annotation and evaluation processes are similar to the one of PASCAL VOC. The main problem with this dataset is that I could not found it online.

[2]Flickr: `https://www.flickr.com/`

3.1.5 MALF

The Multi-Attribute Labeled Faces (MALF) dataset was presented in "Fine-grained evaluation on face detection in the wild" [108] (2015) as one of the first face detection database that "makes it possible for fine-grained performance analysis" due to "annotation of multiple facial attributes". The paper also provides reasons why preexisting face detection benchmarks are biased (i.e. no fine-grained analysis, no reflection of the 'true' real world and no reporting of the 'true' state-of-the-art results) and how their dataset is solving those problems. In short, the dataset contains 5,250 high-resolution images and 11,931 faces annotated with square bounding boxes and attributes including "gender (male, female, unknown), pose deformation level of yaw, pitch and roll (small, medium, large), occluded (true/false), wearingGlasses (true/false), exaggeratedExpression (true/false)."

The collection process was the following. About 2000 images were gathered from Flickr and around 30,000 "using the similar image search service provided by Baidu Inc[3].". The images were then "manually examined by two persons to pick out images that are included in the dataset". All "recognizable faces" are then labeled by two persons with a square bounding box containing "the eyebrow, the chin, and the cheek, while keeping the nose located approximately at the box center" and examined by one. A boolean 'ignore' flag is also provided with each face and set to 'true' if the face is "very difficult to recognize due to very large occlusion, blurring and other extreme deformations, or the size of the bounding box is below 20." This 'ignore' flag is only annotated by one person. Finally, each face with 'false' ignore flag is labeled with the set of attributes listed above.

The paper also provides some interesting statistics.

- Average image size: 753x638
- Images contain on average 2.27 faces (1 face: 46.97%, 2-4 faces: 43,41%, 5-9 faces: 8,30% and 10+: 1.31%)
- Mean size of faces: 83x83
- Median size of faces: 64x64

The evaluation methodology follows the one of the PASCAL VOC Challenge [24] with the same threshold for IoU. However, the algorithms are benchmarked using a ROC curve of True Positive Rate against False Positive Per Image in log scale. They justify that by the fact that what they "care about is how the algorithm performs (the recall rate) at a high precision level (low false alarm)". In addition to the differentiated evaluation by attribute, the benchmark proposes a division between 'easy' (i.e. faces larger than 60x60, no large pose, occluded or exaggerated expression) and 'hard' (i.e. face larger than 60x60 with one of the extreme conditions) subsets.

The dataset is available at `http://www.cbsr.ia.ac.cn/faceevaluation/`. However, the problem with this dataset is that among all the images it contained only 250 are provided with annotations, the rest being used only for internal tests. Moreover, the website does not provide any evaluation tools but it provides results for benchmarked algorithms (up to 2015).

[3]Baidu: `https://www.baidu.com/`

3.1.6 WIDER FACE

Having similar purposes to MALF [108], the WIDER FACE dataset is an effort to trigger progress in face detection in unconstrained scenarios (i.e. extreme pose, exaggerated expressions, large portions of occlusion) by providing labels for fine-grained evaluation. Released in 2016 with the paper "WIDER FACE: A Face Detection Benchmark" [111], it is composed of 32,203 images with 393,703 labeled faces.

The WIDER FACE dataset is actually a subset of the WIDER dataset [105] which is an event recognition benchmark, organized in 60 event classes. To construct WIDER, the event categories were first defined and chosen following the Large Scale Ontology for Multimedia (LSCOM) [66] and according to students propositions. Then, using search engines like Bing[4] or Google[5], between 1000-3000 images were collected for each category.

To obtain WIDER FACE from this dataset, the data was manually cleaned by removing images without human faces and, within each category, similar images were removed to ensure large diversity in facial appearance. The annotation process was done by labeling each recognizable face with a tight rectangular bounding box containing the forehead, chin, and cheeks. For faces that are very difficult to recognize due to low resolution or small scale (10 pixels or less), an 'ignore' flag is provided. Moreover, each face is annotated with its pose (typical or atypical) and occlusion level (none, partial, heavy). For occlusion, faces are considered as partially occluded if 1-30% of the total face area is occluded and as heavily occluded if more than 30% is occluded. For the pose, faces are denoted as atypical under two conditions: either the roll or the pitch degree is larger than 30°, or the yaw is larger than 90°. It is not mentioned in the paper but the toolbox also provides labels for blur, expression, and illumination. However, these labels are annotated using numbers accompanied by one or two words of description without further details. Each annotation is labeled by one annotator and cross-checked by two different people.

The evaluation protocol follows exactly the PASCAL VOC protocol [24]. The positive aspect of this dataset is that evaluation can be divided across different features. As mentioned before, the dataset can be first divided by category which is interesting as the authors have classified these categories according to their difficulty. For each of this event classes, data is randomly split into training, validation and testing sets according to the following proportions 40%/10%/50%. Ground truth is provided trough rectangular bounding boxes for training and validation sets but not for testing sets. Apart from that, all the faces are also divided into three levels of difficulty: 'easy', 'medium' and 'hard' based on the face detection performance of the algorithm EdgeBox [121]. In the paper, they also specify that the database can be subdivided according to face height in pixels between small (10-50), medium (50-300) and large (>300). However, this division is not provided in the evaluation script but it can easily be generated from the ground truth. Finally, the results can be also divided across the different traits that were mentioned above.

The dataset, evaluation toolbox and results (for algorithms as recent as March 2018) can be found at `http://mmlab.ie.cuhk.edu.hk/projects/WIDERFACE/`.

[4]Bing: `https://www.bing.com/`
[5]Google: `https://www.google.com/`

3.1.7 IJB-C

The IARPA Janus Benchmark C (IJB-C) is the 2017 new extended version of the IARPA Janus Benchmark A (IJB-A) which was introduced in 2015 by the paper "Pushing the Frontiers of Unconstrained Face Detection and Recognition: IARPA Janus Benchmark A" [48] and released by the U.S. National Institute of Standards and Technology[6] (NIST). The IJB-C dataset is composed of 21,294 images with 117,542 faces as well as 11,779 videos and is aimed both at the development of face detection and face recognition. The IJB-C is the third version of the IARPA Janus Benchmark following IJB-A and IJB-B. Unfortunately, no paper similar to [48] or [101] for IJB-B has been released for IJB-C. I will, therefore, refer to those two papers to describe the two first versions of the benchmark and infer that IJB-C is similar to them.

The original version of the IARPA Janus Benchmark was composed of 5,712 images and 2,085 videos of 500 subjects. This dataset was released to overcome the limitations of many in-the-wild face recognition dataset whose faces were detected using commodity face detectors such as the Viola&Jones face detector [94]. To overcome this problem, IJB-A was fully manually annotated using the following procedure. A series of 500 subjects with various geographical origins were chosen and images and videos were collected "by performing internet searches on Creative Commons licensed imagery". Then with the help of at least 5 annotators of the Amazon Mechanical Turk[7] (AMT) workforce, bounding boxes were annotated in each image/frame for both the subjects of interest and all other visible faces. The bounding boxes corresponding to subjects of interest were clearly identified and further annotated with locations of eyes and nose, skin color, gender, pose, occlusion (eyes, mouth/nose, forehead), environment (indoor or outdoor) and facial hear. In total, this lead to the annotation of 67,183 faces and to an average number of 11.4 images and 4.2 videos per subject.

To evaluate face detection, "the imagery in the dataset is partitioned into 10 randomly sampled training and testing sets, with two-thirds of the imagery made available for training detector". More precisely, each training split is built by randomly sampling 333 subjects. It is important to notice that only faces larger than 36 pixels were kept. The performances of algorithms are then reported using ROC curves associating the true positive rate to false positive rate per image (i.e. the number of false positives divided by the number of images/frames tested).

This initial dataset was then augmented to produce its second version, IJB-B. To obtain this new version, 1345 subjects were added which lead to a total of 21,798 images (11,754 with faces and 10,044 without faces) and 55,206 frames from 7,011 videos. To select those new subjects, "10,000 names were first scrapped from Freebase[8]" across 10 different geographic regions. For each of these candidates, "the number of available Creative Commons (CC)-licensed videos was determined using the YouTube search API[9]". Then 20 CC images were downloaded for each candidate via Google and Yahoo images and annotated through AMT. If at least two of these images contained the subject of interests, all the YouTube CC videos were collected and annotated. This process leads to a total of 125,474 annotated faces with an average of 6 images and 4 videos per subject.

[6] NIST website: `https://www.nist.gov/`

[7] Amazon Mechanical Turk: `https://www.mturk.com/`

[8] "Since the time of collection and the submission of the paper, Freebase has transitioned to Wikimedia Commons." [101]

[9] YouTube search API: `https://developers.google.com/youtube/v3/docs/search`

With this new version came also 10 new testing protocols, 7 of which are described in [101]. These protocols cover evaluation for face detection, verification, identification, and clustering. The protocol for face detection is basically the same than for IJB-A. The main difference comes from the presence of non-face imagery (i.e. images not containing any faces) in IJB-B. The authors argue that making tests on images without faces is relevant because, in real-applications, most of the images will be of this type.

As mentioned earlier, I could not find a clear documentation on how IJB-C differed from IJB-B. The only source of information is the face challenge page of NIST[10]. According to this page, IJB-C contains "138,000 face images, 11,000 face videos and 10,000 non-face images". By looking at the Challenge Documentation file provided on this page, it is more clearly stated that the dataset is actually composed of 21,294 still face images and 117,542 face frames coming from 11,779 videos. The total number of annotated faces is however not mentioned. However, as there are 138,836 images/frames containing at least one face, this number must be greater than 138,836.

It is on this same website that a link to download the dataset is provided. However, when arriving on the download page, the first sentence is a note specifying that the complete dataset is approximately 325 GB in size. I tried to find a way to download only part of this dataset but I could not find one. Moreover, there is no description of what kind of evaluation code is provided

3.1.8 Summary and choice of testing benchmark

Tables 3.1 and 3.2 summarize the information about each dataset through a series of elements. The size of the database is described in terms of the number of images and faces to detect. The origin of the images and the format of the bounding box are given more as a way to compare the way the dataset were obtained and annotated. Finally, the second table lists the protocols that each benchmark uses for evaluation with the level of precision at which the dataset was annotated (i.e. the relative level at which each bounding box was annotated with information such as pose, blur, etc.) and what type of code is provided. Based on that information, we can make a choice for face detection evaluation benchmark.

This choice must be aligned with the goals of the thesis. As declared previously, we want to build a robust system for the RAGI project. However, the project contains two very different face detection context. The first one is the welcome stations where people we want to recognize will generally be in front of the screen at a constant distance. Then, as the people are guided through the corridors, this distance will vary and there is less guarantee that he/she will face the camera. Moreover, in both scenarios, the number of people to detect can be very variable, going from a single person to a group of 20-30 people, maybe more. In addition, while the cameras will be a priori fixed, the variability in illumination conditions can be very large. All these conditions imply that we need a dataset that allows testing how each algorithm perform in these different settings. Among the ones that we have presented, the best suited along this idea are MALF, WIDER FACE, and IJB-C which provide a high level of annotation granularity The problem with the first one is that it contains only 250 testing images while for IJB-C, the size of the download is prohibitively large. Concerning WIDER FACE, it is manageable in

[10]NIST Face Challenges: `https://www.nist.gov/programs-projects/face-challenges`

terms of size and has the advantage of furnishing evaluation tools. I decided, therefore, to focus on this benchmark to evaluate face detection algorithms in section 6.1.

Name	Nb images	Nb faces	Origin	Bbx format
FDDB	2,845	5,171	Yahoo news website	Elliptical
AFLW	21,997	25,993	Flickr	Rectangular/Elliptical
AFW	205	468	Flickr	Rectangular
Pascal-Faces	851	1,335	PASCAL-VOC	Rectangular
MALF	5,250	11,931	Flickr, Baidu Inc.	Square
WIDER FACE	32,303	393,703	Bing, Google search engines	Rectangular
IJB-C	138,836	>138,836	CC, Google, Yahoo, YouTube	Rectangular

Table 3.1: Summary of the characteristics of the face detection benchmarks including the total number of images in the dataset, number of annotated faces, origin of the images and bounding box format.

Name	Protocol	Annotation Precision	Code
FDDB	ROC, specific matching	Low	None
AFLW	Same as FDDB	Low	SQlite Database, Label GUI
AFW	PASCAL VOC	Medium	Training and Testing
Pascal-Faces	PASCAL VOC	Low	None
MALF	PASCAL VOC, ROC	High	None
WIDER FACE	PASCAL VOC	High	Testing
IJB-C	ROC	High	Unknown

Table 3.2: Summary of the characteristics of the face detection benchmarks including evaluation protocol used, level of annotation precision and which type of code is provided.

3.2 Face recognition

To identify interesting benchmarks, I proceeded similarly than in the previous section by searching across recent publications which benchmarks were used the most frequently. The following sections describe the list of the benchmarks that I identified in chronological order.

3.2.1 LFW

The Labeled Faces in the Wild (LFW) database is considered by most as the first significant 'in-the-wild' dataset and is still considered as one of the reference benchmarks in face verification. The dataset was released back in 2007 in [42] and updated in 2014 as described in [53]. It is composed of 13,233 250x250 images of faces from 5,749 different individuals, 4,069 of which appear in only one image. This little remark is correlated to the goal of this database which is to study face verification rather than face identification. In the original paper, the author actually spoke about studying the "unseen pair match" problem defined as "given two pictures, each of which contains a face, decide whether the two people pictured represent the

same individuals" with the condition that "no images of test subjects are available at training time and the decision for all test pairs are made independently". However, in the update paper, "based on feedback from the vision community", they "adopted the more widely used term of *face verification*".

The authors describe their dataset as "spanning the range of conditions typically encountered by people in their everyday lives" by opposition to previous datasets created under controlled conditions to "facilitate the study of specific parameters". Their dataset exhibits therefore a "natural variability" in a series of characteristics such as pose, illumination, expression, etc. The paper "Names and Faces in the News" [10] describing the dataset on which LFW was built does not provide any more information about the distribution of those characteristics adding only that pictures were obtained from "Yahoo News website over a period of roughly two years" and that their "face recognition dataset is more varied than any other to date". LFW uses the raw images from this dataset and then performs a series of operations. First, they run the Viola&Jones [94] face detector on each image which gives same a series of detections. They then delete from this detections the ones that are false positives, that correspond to an unidentified person or that are duplicates. The next step consists in labeling the faces with the correct name. Finally, for each of those labeled faces, the corresponding detection region is expanded by a factor of 2.2 before being rescaled to 250x250 pixels.

Concerning training and testing sets, the data is divided differently based on two views. The authors mention that "View 1 is for algorithm development and general experimentation, prior to formal evaluation. View 2, for performance reporting, should be used only for the final evaluation of a method." In View 1, the data is divided into two subsets, one for training and one for testing while in View 2, the data is divided in 10 subsets for cross-validation. Each of the subsets in both View and View 2 are composed by a series of pairs of matched (i.e. of the same identity) and mismatched images. Moreover, they state that "for any given training-testing split, the people in each subset are mutually exclusive." As we are in the case of face verification, the ROC curve (True Positive Rate - False Positive Rate) is used to report performances.

A final aspect of the LFW evaluation is that it is divided into 6 protocols based on the characteristics of the face recognition algorithms. Originally, there were only two of those protocols: the image-restricted and unrestricted training protocols. The main difference between those two is that under the second one, algorithms could be trained on additional matched (i.e. of the same identity) or unmatched pairs built from the training data while in the first one, only the original pairs provided in the training data could be used. However, as the authors mention in their update paper, "certain unforeseen issues have arisen" due mainly to algorithms using unsupervised techniques and algorithms using outside data (not part of the LFW training sets). To solve this problem, the 6 following protocols were established:

1. Unsupervised.
2. Image-restricted with no outside data.
3. Unrestricted with no outside data.
4. Image-restricted with label-free outside data.
5. Unrestricted with label-free outside data.
6. Unrestricted with labeled outside data.

The first protocol is used to evaluate unsupervised learning algorithms and the two following ones correspond to two original protocols. The fourth and fifth protocols are similar to the second and third ones except that the algorithm must only be trained on label-free outside data. This means that "the outside data cannot contain any information about whether two images are 'same' or 'different'" and "the outside data cannot contain the identity of any individual, since this can be used to create 'same' and 'different' pairs". Finally, the last protocol is the most permissive allowing to train on any outside data as long as they do not overlap in terms of identities with the testing set.

The dataset, the train-test splits, and the evaluation code are provided at `http://vis-www.cs.umass.edu/lfw/`.

3.2.2 PubFig

The PubFig datasets [50] was an effort to provide an alternative to LFW containing fewer individuals (200) but more pictures per individuals with a total of 60,000 pictures. This larger number of images per person allowed the authors to "construct subsets of the data across different poses, lighting conditions, and expressions, while still maintaining a sufficiently large number of images within each set."

Concerning the dataset construction, images were gathered using search queries with the person's name on search engines like Google Images and Flickr. Face and "fiducial point locations" detection was then applied to the images to obtain cropped face images before finally rectifying the images "using an affine transformation". It is not mentioned if and how the detections were manually checked.

The dataset can be arranged following two protocols. The first evaluation protocol corresponds to View 2 approach of LFW based on 20,000 pairs. The second protocol is based on subsets of all these pairs divided by "pose, lighting, and expression". Moreover, in this case, "training is performed using the same data as in the first evaluation, but the testing pairs are split into 'easy' and 'difficult' subsets for each type of variation." However, the division between 'easy' and 'difficult' set is not further explained.

Finally, the paper refers to the following website to obtain their dataset: `http://www.cs.columbia.edu/CAVE/databases/pubfig/`. However, on this website, the data is only provided as URLs and everything has to be downloaded using this URLs.

3.2.3 YTF

I will now describe a dataset that is not focusing on face recognition in unconstrained images but in unconstrained videos called YouTube Faces and released in 2011 in "Face Recognition in Unconstrained Videos with Matched Background Similarity" [102]. This dataset is relevant in the context of the RAGI project which is working with sequences of frames, in particular, if the team wants to develop some tracking techniques. In total, the dataset is composed of 3,425 videos of 1,595 subjects. The authors detail these numbers by adding that "an average of 2.15 videos are available for each subject." and that "the shortest clip duration is 48 frames, the longest clip is 6,070 frames, and the average length of a video clip is 181.3 frames."

The collection process is detailed precisely in a concise way in the article: "We begin by using the 5,749 names of subjects included in the LFW dataset to search YouTube for videos of these same individuals. The top six results for each query were downloaded. We minimize the number of duplicate videos by considering two videos' names with edit distance[11] less than 3 to be duplicates. Downloaded videos are then split to frames at 24fps. We detect faces in these videos using the VJ [Viola&Jones [94]] face detector. Automatic screening was performed to eliminate detections of less than 48 consecutive frames, where detections were considered consecutive if the Euclidean distance between their detected centers was less than 10 pixels. This process ensures that the videos contain stable detections and are long enough to provide useful information for the various face recognition algorithms. Finally, the remaining videos were manually verified to ensure that (a) the videos are correctly labeled by subject, (b) are not semi-static, still-image slide-shows, and (c) no identical videos are included in the database." In addition to the entire frames, the dataset contains also preprocessed data obtained from those frames. Firstly, detections are stored similarly to LFW by expanding them, resizing them and cropping them to obtain 100x100 pixels images centered on the face. In addition to that, these images are converted to grayscale and aligned. Secondly, descriptors of these detections, obtained via different face description algorithms, are also provided.

The benchmark is, as in LFW, based on standard ten-fold cross-validation and image-constrained and unconstrained protocols are used. The main difference between YTF and LFW is that in this case, the goal is not to identify if two images correspond to the same identity but if a two videos correspond to the same identity.

The dataset can be found on the following website:
`https://www.cs.tau.ac.il/\simwolf/ytfaces/index.html`.

3.2.4 CFP

The following dataset is interesting in what it tries to emphasize about face recognition. The main goal of the Celebrities in Frontal Profile (CFP) dataset [76] is to highlight the differences between frontal to frontal face verification and frontal to profile face verification, in an in-the-wild context. In total, the dataset contains 10 frontal and 4 profile images of 500 individuals.

The dataset was obtained by selecting a list of individuals (with an equal number of males and females and a variety in terms of race) and downloading "hundreds of images for frontal and profile respectively", using keywords as 'profile face' or 'side view' in the second case. A cleanup process, to suppress wrong identities and poses, was realized via the AMT workforce. The workers were asked to separate faces between frontal and profile according to the following criteria: frontal pose images are defined as "those images where both sides of the face are almost the same area of the image" while profile pose images are considered as "those images where one eye is completely visible and less than half of the second eye is visible." Finally, images are cropped and annotated with facial key-points using automatic tools for frontal faces while requiring the workforce of AMT for profile faces on which automatic tools did not work properly.

The dataset is accompanied by an experimental protocol based once again on the View 2 protocol of LFW. The dataset is therefore divided into 10 folds. 50 individuals are assigned to each fold and for each of these individuals, "7 same and 7 not-same pairs" are generated leading

[11] Edit distance: `https://en.wikipedia.org/wiki/Edit_distance`

to 700 pairs per split. However, this protocol is actually divided into two sub-experiments: Frontal-Frontal (where pairs are composed only of frontal pictures) and Frontal-Profile (where pairs are composed of one frontal picture and one profile picture). This leads to a total of 7,000 pairs across all folds for the two experiments. The results of both experiments are reported using ROC curves and can be visualized on the download website of the dataset: `http://www.cfpw.io/`

3.2.5 CACD

One variation factor that the previous datasets do not take into account is age. This is the gap that the authors of "Cross-Age Reference Coding for Age-Invariant Face Recognition and Retrieval" [14] decided to fill by releasing their Cross-Age Celebrity Dataset (CACD). Other face recognition across age datasets such as FGNet [2] and MORPH [72] had been proposed previously to CACD but the latter "contains a larger number of images of different people in different ages." The dataset is indeed composed of "160,000 images of 2,000 celebrities with age ranging from 16 to 62".

The first step of the dataset construction was the collection of celebrities name from the IMDb[12] website. Compared to other datasets, this collection was made following the criteria that the selected people should "have different ages". This was done by "collecting names with different birth years" ranging from 1951 to 1990. For each of the years in this period, the top 50 celebrities were selected leading to a number of 2,000 individuals. The image collection was then undertaken using Google Image Search by using a combination of the individuals' names and a year between 2004 and 2013 as keyword. This collection leads to a total of 200,000 images from which 40,000 duplicates were removed. Finally, for a subset of 200 celebrities, the images were manually checked to remove the noisy ones (e.g. images of other celebrities).

The testing protocol to adopt with this dataset is not clear. Even if not explicitly stated, the protocol is designed for face identification evaluation. The dataset is divided so that 200 celebrities can be used for evaluation while testing results should be reported using another subset of 120 celebrities. This testing is then divided between probe and gallery sets along three different versions. In all three versions, the probe set is made of images taken in 2013 while the gallery sets contain images taken between 2004-2006, 2007-2009 and 2010-2012 respectively. The performances of all methods are reported in terms of Mean Average Precision (MAP). To compute this metric, "for the retrieval results of each query [(i.e. probe)] image, precision at every recall is computed level and averaged to get average precision (AP). MAP is then computed by averaging AP for all query images".

The dataset is available at `http://bcsiriuschen.github.io/CARC/`.

3.2.6 PIPA

The People In Photo Albums (PIPA) dataset presented in "Beyond Frontal Faces: Improving Person Recognition Using Multiple Cues" [116] "consists of 37,107 photos containing 63,188 instances of 2,356 identities". The peculiarity of this dataset is that it is deployed with the intent of using methods for person recognition rather than just face recognition. The main difference is that in the former, heavy use of the context (body, clothes, ...) of the face is used

[12]IMDb: `https://www.imdb.com/`

to identify a person.

The images were collected from "public photo albums uploaded to Flickr" by 111 different users. The collection can be decomposed into 5 main steps. At the start, thousands of albums from Flickr were downloaded and the first step was to filter out those "where person co-occurrence is very low". Then, for each selected albums, annotators were asked to draw bounding boxes around faces of people that appeared at least twice in this album, using a different color for each identity. For faces that are occluded, the bounding box was drawn around the "region of where the head should be". Moreover, no more than 10 people were annotated per photo. The third step consisted in merging similar identities across albums. After that, individuals for which less than 10 faces had been identified were discarded and for individuals that had more than 99 faces identified, only 99 of them were kept. Finally, the data was split between training (50%), validation (25%) and testing (25%) sets.

The authors do not mention any particular evaluation protocol based on this division. The dataset with its annotations is available at
`https://people.eecs.berkeley.edu/~nzhang/piper.html`

3.2.7 MegaFace

Released in 2016, the goal of the MegaFace Benchmark [47] was to take face recognition at scale. This dataset contains more than 1,000,000 images of faces taken in-the-wild, originating "from Yahoo's recently- released database of Flickr photos" [89] containing 100M creative commons photographs and named YFCC100M. The authors of this dataset claim that the singularity of their dataset is to be broad rather than deep (i.e. "contain many different people rather than many photos of a small number of people") with more than 690,000 unique identities. The benchmark also presents protocols for evaluating both face verification and identification.

To extract useful data from the YFCC100M dataset, the authors of MegaFace made use of the metadata accompanying each picture (i.e. "Flickr identifier, owner name, camera, title, tags, geo, media source") and in particular of the Flickr IDs. Drawing from the hypothesis that pictures from different users should contain faces of different identities and "assuming that [if] two or more faces appear in the same photo, they are likely to be different identities, they built a dataset with a maximum of unique identities". They proceeded in the following way. They went through the list of 500K different user IDS selecting "the first photo with a face larger than 50x50 [pixels] and adding it to the dataset". Moreover, if the photo contained other faces above this resolution, they were also added to the dataset. This process was then repeated for the second, third, etc. photo of each user "until a sufficient number of faces were assembled". A total of 1,296,079 faces were gathered this way using the HeadHunter algorithm [63]. According to the authors, as face detection can have "up to 20% of false positives", taking this number of detections would ensure that at least 1,000,000 of them were faces. After that, blurry faces were removed to obtain a total of 690,572 faces "having a high probability of being unique individuals". Moreover, even if not guaranteed, the remaining 310K images "likely also contain additional unique identities". Finally, it is important to mention that each detected face was stored so that "50% of the faces spans the photo height" and that 49 fiducial points, yaw and pitch angles were also estimated.

In the evaluation protocol proposed by MegaFace, this dataset actually constitutes what the

authors call 'distractors' i.e. "faces of unknown people". These distractors are used exclusively in the gallery of faces against which probe faces are compared during testing. The tests are composed of two face recognition scenarios: identification and verification. In identification, a probe is compared against a "gallery containing at least one photo of the same person". The probe is compared against all faces in the gallery and those are ranked "based on similarity to the probe". More specifically, the authors state that "the probe set includes N people; for each person, we have M photos. We then test each of the M photos (denote by i) per person by adding it the gallery and use each of the other M-1 photos as a probe." The results are then displayed using a CMC curve. To evaluate verification, all the pairs between the gallery and probe set are computed and the results are reported using a ROC curve.

The MegaFace Challenge uses two probe sets to evaluate algorithm performances: FaceScrub [67] and FGNet [2]. In the FaceScrub paper, authors state this dataset contains 141,130 faces of 695 different public figures. The procedure for obtaining those images was to select celebrities names on IMDb.com and search for each name in a search engine like Google. Then, the Viola&Jones [94] (that I will denote as VJ) detector was run to detect faces which were then aligned and resized to 96x96 pixels frames. Finally, they use a special methodology to remove outliers (i.e. frames that do not correspond to the queried person) in an efficient way. For FGNet, I could not find a paper or website describing it in detail. However, MegaFace mentions that is composed of 975 photos of 82 people with age ranges of more than 40 years.

Finally, it is important to specify that the MegaFace challenge is divided into two sub-challenges called Challenge 1 and Challenge 2. The first challenge allows the competitors to train their algorithm on any dataset while in the second case the training can only be done using the MegaFace dataset. The MegaFace Challenge is hosted on `http://megaface.cs.washington.edu/` and the MegaFace, FaceScrub and FGNet datasets are downloadable from there. Evaluation code is also provided for both of the challenges.

3.2.8 IJB-C

As described in section 3.1.7, the IARPA Janus Benchmarks are designed for evaluating both face detection and recognition. As I already covered how the datasets were built, I will only focus on the face recognition protocol. Moreover, the benchmarks contain both evaluation protocols for face verification and identification but I will focus on the latter. In IJB-A [48], the same 10 splits as for the face detection protocol are used. For each split, every of the 167 testing subjects has their imagery randomly sampled into either the probe or the gallery set. Moreover, in each split, 55 randomly selected subjects have their imagery removed from the gallery so that algorithms cannot be built on the assumption that each person has a mate (i.e. a face associated to his/her identity) in the gallery. The evaluation metrics are the same as the one proposed in FRVT [31]: CMC and DET curves.

For IBJ-B [101], 3 of the 10 protocols are designed for face recognition. These protocols are not based on the same data division as in IJB-A. Two disjoints gallery sets are created containing respectively 931 and 914 subjects using 3,081 and 3,376 still images. The three protocols are then defined based on the way the probe set is constructed. This was done in order to "have a thorough understanding of an algorithm's strength and weaknesses". I will not enter into the details of each protocol as I do not think it is useful for this thesis.

Looking at the information on the NIST website, it seems like IJB-C uses these same protocols. However, as specified for face detection, the overwhelming size of IJB-C prevents me from using it.

3.2.9 Summary and choice of testing benchmark

Information about the benchmarks is summarized in Table 3.3 and 3.4. Similarly to the face detection part, this information describes the sizes, origins, annotations and testing protocols of the previously-described benchmarks. Concerning the annotation, the main information covers how the faces were annotated.

In the face detection section, I chose the dataset mainly on its precision with regard to annotation. In this part, none of the dataset I found provide precise additional annotations information except for age. However, studying the variation of performance according to age or robustness of face recognition across a large range of years is not of main interest in the case of RAGI. Indeed, firstly, RAGI is aimed to be developed in public workspaces where the main public is composed of adults, so getting performances for children is not of paramount interest. Secondly, the system is aimed mainly at helping regular workers and not at recognizing people that have not been seen for years.

Another interesting aspect regarding annotations is the way that the bounding box where detected in the first place. The oldest dataset (until CACD) used a commodity face detector such as VJ to detect faces. The problem with this type of detector is that it limits the pose variation as it is very good at detecting frontal faces but gets much worse when the faces are tilted away from the camera. The authors of CACD tried to overcome this by manual annotating profile faces while PIPA and IBJ-C manually labeled all the instances. The case of MegaFace is a bit more special as annotations are done automatically but with a much more recent face detection algorithm which performs, for instance, two times better than VJ on the MALF dataset. However, for its FaceScrub probe set, the faces were detected using VJ. Concerning size, MegaFace clearly leads the march even though the other datasets still provided a large range of people and pictures per people. In the end, based on those criteria and the fact that the MegaFace benchmark provides full evaluation tools, I decided to focus on this dataset to test algorithms in section 6.2. IJB-C possess similar qualities but as said previously, its size is prohibitive (>325GB). As a comparison, MegaFace is only 64GB.

Name	Nb faces	Nb people	Origin
LFW	13,233	5,749	Yahoo News Website
PubFig	60,000	200	Google Images and Flickr
YTF	3,245 (videos)	1,595	YouTube
CFP	7,000	500	Search engine
CACD	160,000	2,000	Google Images
PIPA	63,188	2,356	Flickr albums
MegaFace	1M+141k+975	690k+695+82	Flickr+Search Engine
IJB-C	>138,836	$\geq$ 1845	CC, Google Yahoo, YouTube

Table 3.3: Summary of the characteristics of the face recognition benchmarks including a total number of faces in the dataset, number of different identities, faces and origin of the images. For MegaFace, the number of faces and of people is divided between the MegaFace dataset, FaceScrub, and FGNet.

Name	Recognition Type	Annotation Technique	Testing Code
LFW	Ver	Automatic (VJ)	Yes
PubFig	Ver	Automatic	No
YTF	Ver	Automatic (VJ)	Yes
CFP	Ver	Automatic + Manual (AMT)	No
CACD	Ver	Automatic	No
PIPA	Ver	Manual	No
MegaFace	Ver + Id	Automatic (HeadHunter)	Yes
IJB-C	Ver + Id	Manual	Unknown

Table 3.4: Summary of the characteristics of the face recognition benchmarks including which type of face recognition can be evaluated on the benchmark (Verification or Identification), techniques used to annotate the faces and availability of testing code.

Chapter 4

Algorithm Selection

As mentioned in the previous section, I decided to search for algorithms by looking at the results reported on some of the benchmarks that were just described. For each benchmark providing sufficiently new results, I selected a small set of best-performing algorithms. Then once these algorithms had been selected, the second step was to remove those that do not actually provide open source code and model.

4.1 Face detection

In the list of benchmarks presented for face detection, the only ones that provide evaluation for a sufficient number of recent algorithms on their website are FDDB, MALF, and WIDER FACE. For each of these benchmarks, I will briefly describe how algorithms are referenced and explain how I selected the best performers.

4.1.1 Benchmark-based selection

FDDB

FDDB presents two lists of algorithms: one of the published methods and another with unpublished methods. The second list is therefore of no use for us as we can not even know how the algorithms work. The first list is composed of 47 algorithms from the oldest one the newest one. The algorithms are then tested based on the discrete and continuous metrics as mentioned earlier. It is quite difficult to determine exactly how the algorithms perform relatively the each other by looking at the displayed ROC curves. Indeed the curves are stacked over one another with colors being reused for several algorithms. One can click on the figure to zoom in but the zoom is fixed on the top-left corner, as shown in Figure 4.1. Nevertheless, it is still possible to constitute the list of top performers on this dataset.

Table 4.1 contains the list of top-performers as of 14-04-2018[1] in approximate order of performance (starting with the best performer) with their abbreviated name and the research paper in which the technique is described.

[1]Since then, a new algorithm denoted as FD-CNN as been added.

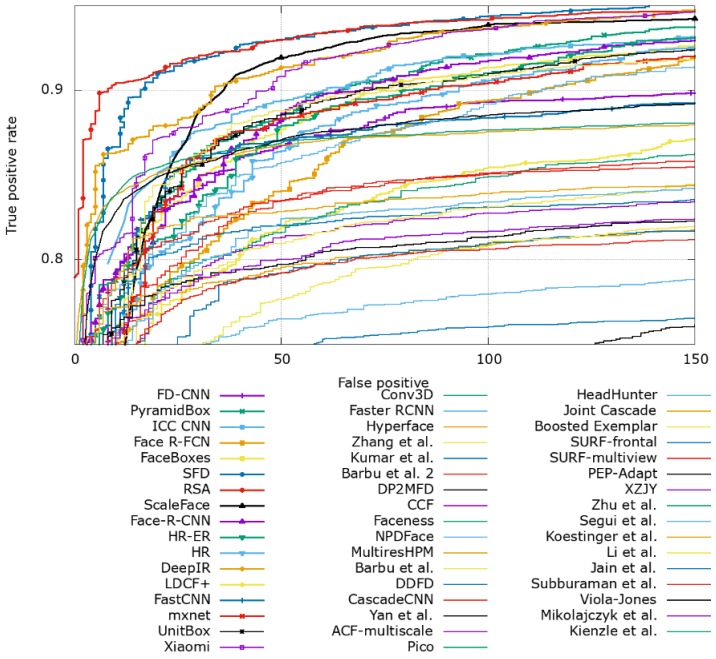


Figure 4.1: Zoom on the top-left corner of the discrete-score ROC curves obtained by the published face detection algorithms on the FDDB benchmark on 07-06-2018.

Algorithm	Reference
RSA	Recurrent Scale Approximation for Object Detection in CNN [59]
SFD	S^3FD: Single Shot Scale-invariant Face Detector [117]
ScaleFace	Face Detection through Scale-Friendly Deep Convolutional Networks [112]
DeepIR	Face Detection using Deep Learning: An Improved Faster RCNN Approach [82]
Xiaomi	Bootstrapping Face Detection with Hard Negative Examples [95]
ICC-CNN	Detecting Faces Using Inside Cascaded Contextual CNN [115]
PyramidBox	PyramidBox: A Context-assisted Single Shot Face Detector [88]

Table 4.1: List of best published performers on the FDDB benchmark on 14-04-2018.

As DeepIr corresponds to a commercial product, I am not going to consider it further.

MALF

In MALF, results for commercial, published and unpublished methods are presented under the same ROC curves. There are several of these curves according to the different division proposed by the benchmark (whole, easy-hard, small-large faces, small-medium-large yaw). For each of this curves, as shown in Figure 4.2, the area under the curve is also reported in % which allows an easier comparison of the algorithm performances.

If we analyze the different graphs, we can easily see that among the top 12 best algorithms the same 6 published methods are always present. As of 14-02018, Table 4.2 shows this list in performance order (starting with the best algorithm) based on the whole dataset test with their abbreviated name and the corresponding paper.

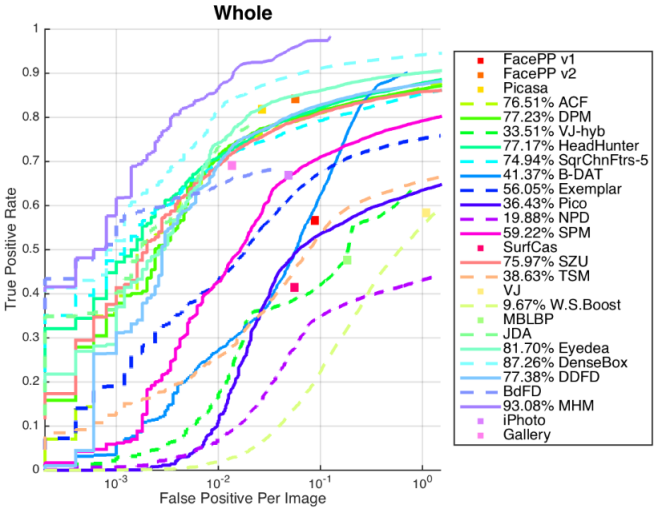


Figure 4.2: ROC curves obtained by the face detection algorithms on the whole MALF benchmark on 07-06-2018.

Algorithm	Reference
JDA	Joint Cascade Face Detection and Alignment [16]
DDFD	Multi-view Face Detection Using Deep Convolutional Neural Networks [25]
DPM	Face detection without bells and whistles [63]
HeadHunter	Face detection without bells and whistles [63]
ACF	Aggregate channel features for multi-view face detection [107]
SqrChnFtrs-5	Face detection without bells and whistles [63]

Table 4.2: List of best published performers on the MALF benchmark on 03-06-2018)

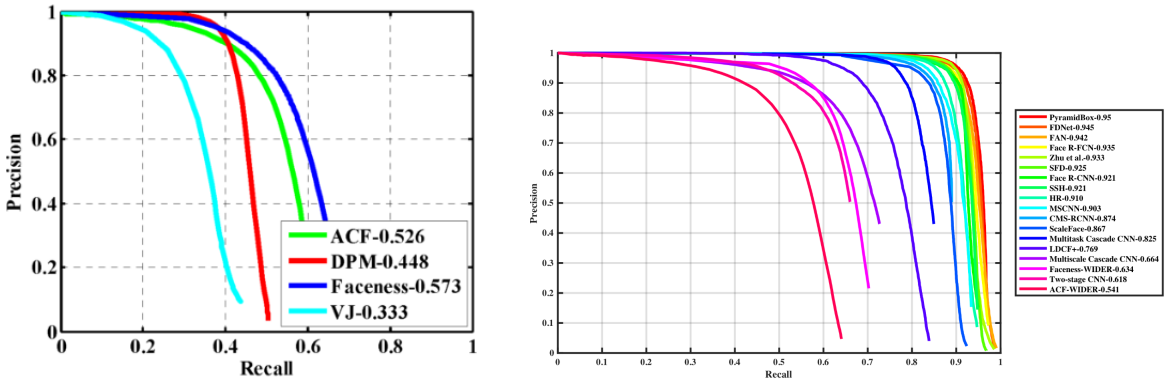
WIDER FACE

Concerning WIDER FACE, there are also two lists of algorithms. The first list is tested under Scenario-Ext (i.e. A face detector is trained using any external data and tested on the WIDER FACE test partition) for the easy, medium and hard subsets. Results are displayed using precision-recall curves with in addition average precision for each algorithm. This list contains only 4 algorithms (i.e. the ones that were tested in the WIDER FACE paper) which are in order of performance on the medium dataset, as shown in Figure 4.3 (a):

Algorithm	Reference
Faceness	From Facial Parts Responses to Face Detection: A Deep Learning Approach [110]
ACF	Aggregate channel features for multi-view face detection [107]
DPM	Face detection without bells and whistles [63]
VJ	OpenCV implementation of Viola-Jones face detector [94]

Table 4.3: List of face detection algorithms tested on the WIDER FACE benchmark with the Scenario-Ext protocol.

The second list of algorithms is tested under Scenario-Int (i.e. A face detector is trained using WIDER FACE training/validation partitions, and tested on WIDER FACE test partition). 18 algorithms are tested but only 17 are referenced (i.e. FDNET is not referenced). Across



(a) Scenario-Ext (b) Scenario-Int

Figure 4.3: Precision-recall curves obtained by the face detection algorithms on medium subset of the WIDER FACE benchmark using (a) Scenario-Ext and (b) Scenario-Int on 07-06-2018.

the results for the three subsets, we can easily identify 10 top-performers that achieve similar performances ($AP > 90\%$ on the easy and medium sets and $AP > 80\%$ on the hard set) both on the validation set and on the test set. The list of those algorithms classified by performance on the medium set, as shown in Figure 4.3, is shown in Table 4.4.

Algorithm	Reference
PyramidBox	PyramidBox: A Context-assisted Single Shot Face Detector [88]
FDNet	*Not Referenced*
FAN	Face Attention Network: An Effective Face Detector for the Occluded Faces [98]
Zhu et al.	Seeing Small Faces from Robust Anchor's Perspective [119]
Face R-FCN	Detecting Faces Using Region-based Fully Convolutional Networks [99]
SFD	S^3FD: Single Shot Scale-invariant Face Detector [117]
Face R-CNN	Face R-CNN [98]
SSH	SSH: Single Stage Headless Face Detector [65]
HR	Finding Tiny Faces [40]
MSCNN	A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection[11]

Table 4.4: List of best performers on the WIDER FACE benchmark with the Scenario-Int on the 03-06-2018.

We can note that, in this second scenario, ACF and Faceness, both retrained on WIDER FACE, were also tested and their performances are among the worst ones. This is why in the following section we are not considering the algorithm tested in the first scenario (except for VJ as explained later).

4.1.2 List of potential algorithms to test

The previous lists are sometimes redundant. This is why I list in Table 4.5 the list of all algorithms considered worth to test. In this list, I added information about my second criteria for algorithm selection: an open-source code. Finding an open-source code for all these algorithms is not guaranteed. It is not an easy task as the benchmarks generally do not give a link to the repository where the code comes from and that the paper associated with the algorithm do not always provide an implementation of the algorithm. When this is not the case, we can

sometimes still rely on an unofficial implementation. To be considered relevant, this unofficial implementation must at least be based on the same technical characteristics. In the following list, I will, therefore, provide information about whether open-source code exists for each algorithm, if it is an official (i.e. directly referenced in the research paper or owned by one of the authors) or unofficial implementation and specify the link to the corresponding repository. To save some space, the URLs to the repositories are not explicitly written but accessible through hyperlinks under the 'Official/Unofficial' keywords.

I kept in the table VJ, the OpenCV Haar-Cascade detector derived from the Viola&Jones algorithm [94], because it is considered as the first practical face detector and is still widely used. In addition to these algorithms, I decided also to test Dlib[2] detector based on Histograms of Oriented Gradients (HOG) [19] which is widely popular.

Algorithm	Code
ACF	Unofficial
DDFD	Unofficial
DPM	Official
Face R-CNN	Unofficial
Face R-FCN	Unofficial
FAN	None
HeadHunter	Official
HR	Official
ICC-CNN	None
JDA	None

Algorithm	Code
MSCNN	Official
PyramidBox	None
RSA	Official
ScaleFace	None
SFD	Official
SqrChnFtrs-5	None
SSH	Official
VJ	Official
Xiaomi	None
Zhu et al.	None

Table 4.5: List of potential face detection algorithms to test.

4.2 Face recognition

As for face detection, I want to test state-of-the-art face recognition algorithm with open source code. To find state-of-the-art algorithms, I decided to explore the results of the previously-described face recognition benchmarks. The two benchmarks proposing the largest and most recent list of algorithms are LFW and MegaFace. I will, therefore, focus on the algorithms performing well on those datasets. In this case, I did not display any graphs because they were mixing published and commercial algorithms and therefore more difficult to exploit.

4.2.1 Benchmark-based selection

LFW

As stated in section 3.2.1, the LFW benchmark is divided into 6 protocols. For each of these protocols, a list of algorithms with their references and their ROC curve is provided. The most interesting one is the 'Unrestricted, Labeled Outside Data' protocol on which the most diverse set of algorithms was tested and that corresponds best to the needs of the RAGI project. The algorithms' names are either written in gray, red or green and as specified on the result

[2]Dlib: `http://dlib.net/ml.html`

page: “Results in red indicate methods accepted but not yet published (e.g. accepted to an upcoming conference). Results in green indicate commercial face recognition systems whose algorithms have not been published and peer-reviewed. We emphasize that researchers should not be compelled to compare against either of these types of results.” The only algorithms of interest are therefore those listed in gray which makes a total of 16 algorithms. However, in this list, some algorithms are clearly less performing than others so I decided to keep only the 10 bests listed in chronological order in Table 4.6. To save some space, I sometimes simplified the abbreviations used to refer to the algorithms. I added a ‘*’ next to the algorithms whose name were modified. In addition, I had to change the font in which certain references were written.

Algorithm	Reference
Tom-vs-Pete*	Tom-vs-Pete Classifiers and Identity-Preserving Alignment for Face Verification [9]
High-dim LBP	Blessing of Dimensionality: High-dimensional Feature and Its Efficient Compression for Face Verification [15]
TL JB*	A Practical Transfer Learning Algorithm for Face Verification [13]
DeepFace*	DeepFace: Closing the Gap to Human-Level Performance in Face Verification [87]
POOF*	Part-Based One-vs-One Features for Fine-Grained Categorization, Face Verification, and Attribute Estimation [8]
DeepID	Deep Learning Face Representation from Predicting 10,000 Classes [83]
FaceNet	FaceNet: A Unified Embedding for Face Recognition and Clustering [74]
MMDFR	Robust Face Recognition via Multimodal Deep Face Representation [23]
P+S+E Aug.*	Do We Really Need to Collect Millions of Faces for Effective Face Recognition? [62]
Dlib-R*[3]	Dlib implementation of Deep Residual Learning for Image Recognition [20]

Table 4.6: List of best, published performers on the LFW benchmark with the ‘Unrestricted Labeled Outside Data’ protocol on the 14-04-2018.

MegaFace

For MegaFace, I will focus on the results obtained on the Challenge 1 with the FaceScrub dataset as probe dataset. The Challenge 1 is more interesting because competitors can train their algorithm on any dataset. The results when the probe set is FGNet are not as interesting for us as this dataset focus mainly on age variation and is also smaller than FaceScrub. In addition to that, for each challenge, the results are displayed both for identification and verification. I will focus on the results presented in the identification protocol. A problem with the MegaFace website is that no difference is made between commercial and scientific developments and more over these methods are not referenced. I had, therefore, to search manually on the Internet by typing the name of the methods in Google search to find the corresponding algorithms. Doing this research, the only open source algorithms that I could identify are the following. Note that I specify the name of the algorithms not of the methods presented on the MegaFace results page.

Algorithm	Reference
ArcFace	ArcFace: Additive Angular Margin Loss for Deep Face Recognition [21]
SphereFace	SphereFace: Deep Hypersphere Embedding for Face Recognition [57]
FaceNet	FaceNet: A Unified Embedding for Face Recognition and Clustering [74]

Table 4.7: List of best, identifiable performers on MegaFace Challenge 1 Identification protocol with FaceScrub as probe set on the 14-04-2018.

4.2.2 List of potential algorithms to test

This time none of the algorithms in the two lists are the same. In Table 4.7, I exactly copied the algorithms of the two previous lists and highlighted which of them do not provide an open-source implementation and if they do, if this implementation is official or unofficial. Note that for the algorithms taken from the MegaFace dataset, I could not make a direct link between the methods and the algorithms so I will consider an algorithm to be official with regards to the scientific paper that explains the algorithm.

In addition to those algorithms, I will also consider OpenFace which is another open source implementation of FaceNet available at `http://cmusatyalab.github.io/openface/`

Algorithms	Code
ArcFace	Official
DeepFace	None
DeepID	Unofficial
Dlib-R	Unofficial
FaceNet	Unofficial
High-dim LBP	Unofficial

Algorithms	Code
MMDFR	None
P+S+E Aug.	Unofficial
POOF	None
SphereFace	Official
TL JB	None
Tom-vs-Pete	None

Table 4.8: List of potential face recognition algorithms to test.

Chapter 5

Algorithm Description

In this chapter, I will describe the algorithms that will be tested in the two following chapters. Moreover, as some of the developments of algorithms are sometimes related to each other, the description are sorted in chronological order of the methods publication.

My initial goal was to test all the algorithms for which I had found an official or unofficial implementation. However, due to lack of time, I did not achieve this goal. I, therefore, focused at first on algorithms with official implementations. Unfortunately, I did not manage to run all of them. The problems I encountered came principally from compatibility reasons with CUDA and cuDNN or in the case of algorithms written in Matlab, systematic fatal errors, when running the algorithms, that I could not explain. In total, I managed to run 5 face detection algorithms and 3 face recognition algorithms.

5.1 Face detection

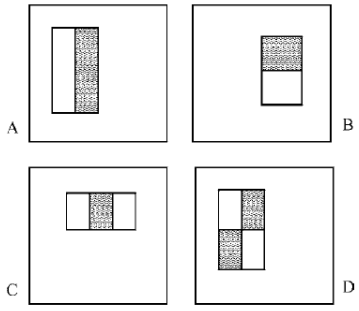
The 5 algorithms that will be described for face detection are VJ, HOG, FRCNN, SFD, and SSH.

5.1.1 VJ

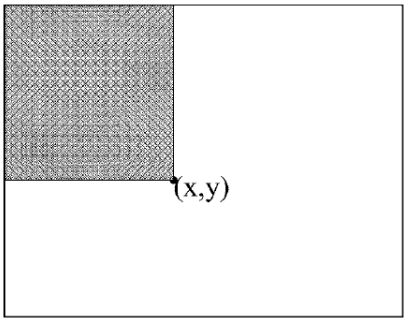
The algorithm proposed by Viola & Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" [93] and refined for face detection in "Robust Real-Time Face Detection" [94] was one of the first real-time object (and face) detection method. The algorithm is based on three main concepts: Haar features and the integral image, feature selection via Adaboost [27] and the attentional cascade. To summarize, a sliding window is passed over the image on which face detection is performed and for each of the sliding window's position, features are generated from the pixels in the sliding window. Then these features are passed through a cascade of classifiers to identify if these pixels correspond to a face. I will now enter into some more details.

The features used in the Viola & Jones algorithm are "reminiscent of Haar basis function" and are, in summary, computed from the difference of the sum of the pixels in different numbers (2, 3 and 4) of adjacent rectangles. In a sub-window of 24x24 pixels, 160,000 such features can be computed. To compute this features more efficiently on a given gray-scale image, it is first converted to an 'integral image' where the value of a pixel at position (x,y) is simply the sum

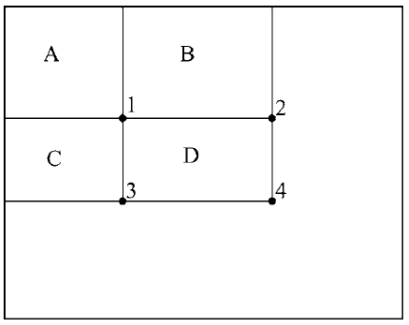
of the values of the pixels above and to the left of (x,y) in the original image.



(a) "Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature."



(b) "The value of the integral image at point (x, y) is the sum of all the pixels above and to the left."



(c) "The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is A + B, at location 3 is A + C, and at location 4 is A + B + C + D. The sum within D can be computed as 4 + 1 − (2 + 3)."

Figure 5.1: Illustrative figures from [94] showing (a) examples of features, (b) how the integral image is computed and (c) how the integral image can be used to compute the sum of pixels contained in the rectangles used to compute the features.

The features must then be passed through a classifier to distinguish between faces and non-faces. The classifier chosen in the paper is a modified version of Adaboost. The modification is intended to take care of the overwhelming number of features generated per sub-window. To select the best discriminating features, the idea is to draw "an analogy between weak classifiers and features". The basic idea is that each weak classifier is using only one feature and that the better the classifier the better the feature. With this technique, the authors were capable of extracting 200 features from the initial 160,000.

However, to boost performances while keeping a good frame rate, the algorithm was modified by replacing the simple classifier by a cascade of classifiers. This cascade classifier works similarly to "a degenerated decision tree". The subwindows features are scattered between a series of classifiers that are used in series and at each step, some of the corresponding subwindows are rejected as non-face. The first classifiers are simple and aimed at rejecting a majority of sub-windows to scale the time up and then more complex classifiers are used to achieve low false positive rates. Using this technique, the final model uses 6060 features divided across 38 layer cascade of classifiers while maintaining high frame rate. Figure 5.2 shows an example of the two first selected features.

In the original paper, two parameters are put forward. They are relative to the sliding window process in which a detector is scanned over the image. Indeed, faces can be situated anywhere on an image and at a variety of different scales. The scaling process is managed

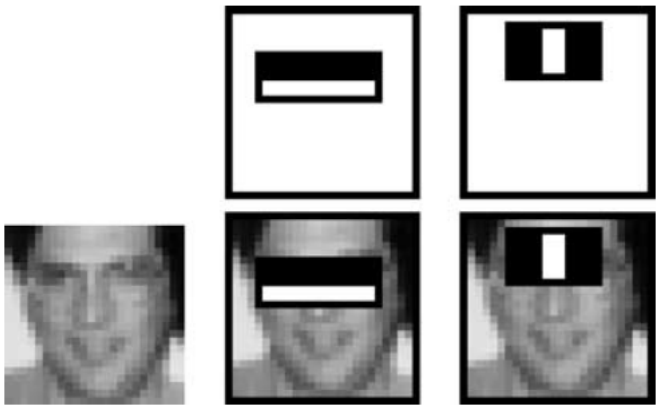


Figure 5.2: Illustrative figure from [94] of the features used in VJ. Description: "The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose."

by scaling the size of the detector (whose minimum size is 24x24 in the original paper) by a constant factor (set at 1.25). When the value of this parameter decreases, faces of more diverse sizes can be detected but the algorithm gets slower. To move the detector across the image, one can simply use a step of one pixel or increase this step. Once again, the smaller this step, the more face can be detected but the slower the algorithm.

In addition, the paper deals with the problem of multiple detections of the same face. As specified in the paper, "Since the final detector is insensitive to small changes in translation and scale, multiple detections will usually occur around each face in a scanned image". The solution to this problem is to group all the detections into disjoint subsets, two detections being in the same subset if their bounding regions overlap. Then for each of these subsets, a single bounding region, whose corners are "the average of the corners of all detections in the set", is produced.

5.1.2 HOG

Histogram of Oriented Gradients (HOG for short) technique was presented as an object, more specifically in this case, human detection method in the paper "Histograms of Oriented Gradients for Human Detection" [19].

The idea behind this algorithm is that "local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions." The goal of the algorithm is then to produce a vector of features for a given image subwindow based on those gradients and to use those features to classify a subwindow as human or not. The complete feature extraction procedures can simply be transposed to face detection, the only difference being that a new classifier must be trained with the features.

I will briefly explain the details of the algorithm. I refer here to the characteristics of the default algorithm. Other characteristics are studied in the paper. Given a detection window, the first step is to divide it into a number of small spatial regions called cells, which are 8x8 pixels squares. For each of those cells, a "1-D histogram of gradient directions or edge orientations" is computed. The idea is to apply a simple $[-1, 0, 1]$ mask over the pixels of the cell to get the intensity of the gradient in each direction. Then, "each pixel calculates a weighted vote for an edge orientation histogram channel based on the orientation of the gradient element centred on it, and the votes are accumulated into orientations bins over [...] cells". The algorithm could stop there by using the concatenation of the histograms of all cells as a feature vector but the authors apply first some normalization to the data. Their normalization technique is "based on grouping cells into larger spatial blocks and contrast normalizing each block separately". In the default configuration, each of this block is composed of 16x16 pixels containing 4 cells. Moreover, the blocks typically overlap meaning that "each scalar cell response contributes several components to the final descriptor vector, each normalized with respect to a different block." In the default configuration, the block spacing stride is of 8 pixels meaning that each cell is covered 4 times. The set of features is finally classified using a soft linear SVM.

During the testing phase, the detection window slides over the image and applies this algorithm at each position. In the paper, it seems that only one size (64x128) of detection window is used and it is not clear how the window is slid across the image (the only indication being that on an image of 320x240 pixels, 4000 different positions are tested).

5.1.3 FRCNN

Face R-CNN is a deep learning based approach to face detection proposed in the paper "Face R-CNN" [97] and based on the successful object detection method "Faster R-CNN" [71]. The open-source code that I found is actually not an implementation of this paper (or at least there is no mention of that) but rather a version of Faster-RCNN applied to face detection. However, I decided to test this algorithm anyway for the following reason. Faster R-CNN is the second main update of a method called R-CNN. This algorithm presented in [29] was the first to reapply CNN's to object detection and to achieve state-of-the-art. The introduction of this method spurred the development of a series of CNN based approach trying to improve the drawbacks of this method. For this reason, I thought it would be convenient to describe briefly this method and the updates that lead to Faster R-CNN as well as making tests using an algorithm based on this approach. For simplicity, I will denote Faster R-CNN as FRCNN even though it might cause confusion with Face R-CNN.

R-CNN stands for *Regions with CNN features*. When this method was proposed, the state-of-the-art techniques in object detection were based on HOG or SIFT [61] features. As presented in the previous section, this techniques use a sliding window approach in which, for each window position, a vector of features is generated and then classified using some sort of classifier (SVM for example). CNN's had also been applied in this case as a feature extractor. This article proposes an alternative to the sliding-window approach for the localization problem called the "recognition using regions" paradigm as argued in [32]. The basic idea is to replace the window sliding step by a region proposal step that will generate a set of "category-independent region proposals for the input image". Features are then extracted and classified only for those preselected regions.

To summarize, the R-CNN framework is composed of three modules as exemplified in Figure 5.3: "The first generates category-independent region proposals. These proposals define the set of candidate detections available to our detector. The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region. The third module is a set of class-specific linear SVMs." An additional component called bounding box regression is actually deployed to reduce localization errors from step 1. Inspired by a technique used in [26], its goal is to "predict a new detection window" by using the feature generated by the last pooling layer of the network through linear regression.

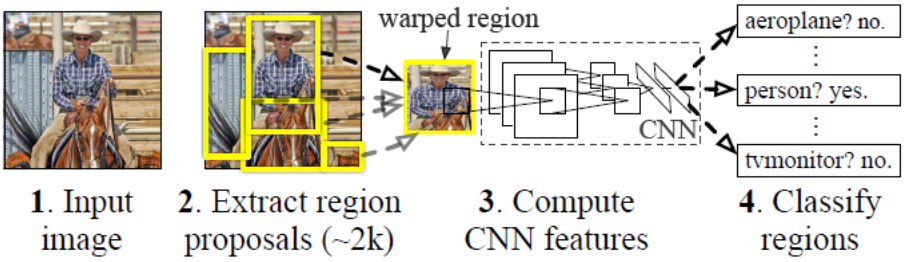


Figure 5.3: Illustrative figure from [29] for the R-CNN algorithm. Description: "Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs."

An inconvenient requirement of R-CNN is that the input image of the CNN must be of fixed size because it ends with fixed-length fully-connected layers as mentioned in "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition" [35]. The trick used in R-CNN to deal with that problem is to simply "warp all pixels in a tight bounding box around it to the required size", while applying some dilution prior to that. The solution proposed by [35] is the introduction of a 'spatial pyramid pooling' (SPP) layer in the CNN, as shown in Figure 5.4, to get rid of this requirement. More precisely, the technique is to "add an SPP layer on top of the last convolutional layer. The SPP layer pools the features in local spatial bins having sizes proportional to the image size and generates fixed-length outputs, which are then fed into the fully-connected layers (or other classifiers)". In other words, the authors "perform some information 'aggregation' at a deeper stage of the network hierarchy (between convolutional layers and fully-connected layers) to avoid the need for cropping or warping at the beginning." The authors call their network using this technique, SPP-Net. Another problem of R-CNN is that "the feature computation in RCNN is time-consuming, because it repeatedly applies the deep convolutional networks to the raw pixels of thousands of warped regions [(i.e. the proposals)] per image." SPP-net improves drastically this computing-time (10 to 100 times faster at test time) by "computing a convolutional feature map for the entire input image and then classifies each object proposal using a feature vector extracted from the shared feature map".

This last point is evoked in the paper "Fast R-CNN" [28] presenting a method of the same name and written by the authors of R-CNN. Their observation is that SPP-Net improved R-CNN but that it still has a major drawback which is that training is a multi-stage pipeline (i.e. CNN fine-tuning, SVM fitting and learning of bounding-box regressors). Fast R-CNN uses "a single-stage training algorithm that jointly learns to classify object proposals and refine their spatial locations" based on a multi-task loss learning. As illustrated in Figure 5.5, the

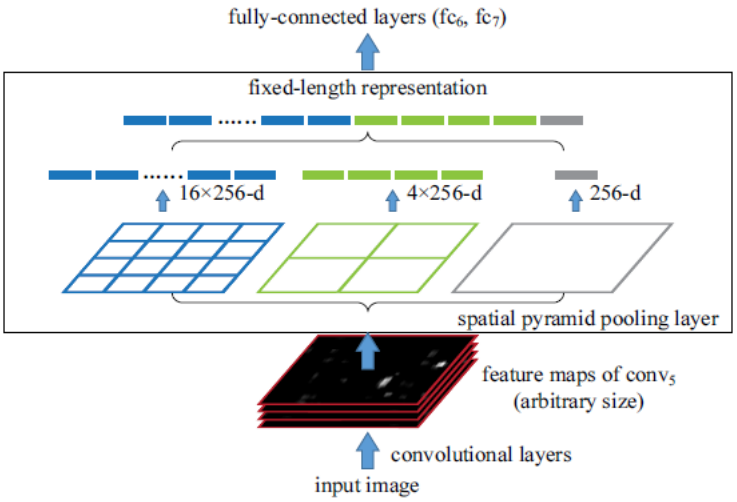


Figure 5.4: Illustrative figure from [35] showing the introduction of an SPP layer in a CNN.

main idea is to end the CNN with two sibling output layers: "one that produces soft-max probability estimates over K object classes plus a catch-all 'background' class and another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes." The first layer removes the need for a SVM classifier while the second layer performs bounding box regression. In addition, [28] replaced the SPP layer by the 'ROI layer' which is a "special-case of SPP layer in which there is only one pyramid level". Using this technique, Fast R-CNN can train deep networks "9x faster than R-CNN and 3x faster than SPP-Net".

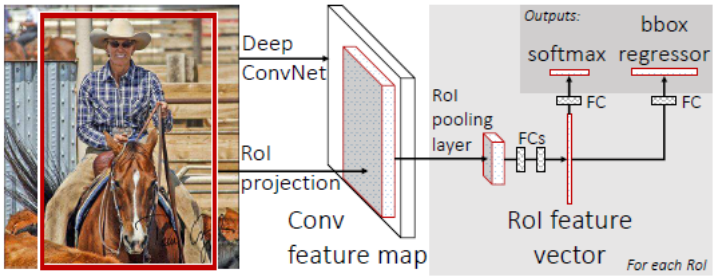


Figure 5.5: Illustrative figure from [28]. Description: "Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss."

SPP-Net having dealt with the detection computing time and Fast R-CNN with the training computing time, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"[71] exposed the fact that for these detection networks, the bottleneck was then situated at the region proposal phase but also observe that "the convolutional (conv) feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals." They, therefore, introduce novel *Regions Proposal Networks* (RPNs) that share convolutional layers with the object detection networks. Basically, a RPN "takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness

score." In more details, as illustrated in Figure 5.6, the process is the following. "To generate region proposals, we slide a small network over the conv feature map output by the last shared conv layer. This network is fully connected to a *nxn* spatial window of the input conv feature map. Each sliding window is mapped to a lower-dimensional vector." "This vector is fed into two sibling fully-connected layers—a box-regression layer (reg) and a box-classification layer (cls)." "For each sliding-window location, the reg layer predicts k region proposals via 4k outputs encoding the coordinates of these boxes. The k proposals are actually parametrized relative to k reference boxes, called 'anchors'. Each anchor is centered at the sliding window in question and is associated with a scale and aspect ratio. In the meantime, the cls layer outputs 2k scores that estimate the probability of object/not-object for each proposal." Finally, to optimize the training procedure, "RPN and Fast R-CNN can be trained to share the convolutional features".

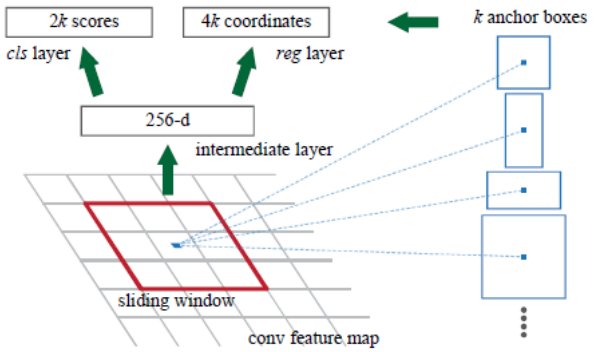


Figure 5.6: Illustrative figure from [71] of the RPN.

The authors of Faster R-CNN mention, to finish their introduction of RPN, that some of the generated proposals will highly overlap with each other, similarly to what happened with VJ. To deal with this, they adopt a technique called non-maximum suppression (NMS). Referred as greedy NMS in [29], it is defined as "rejecting a region if it has an IoU overlap with a higher scoring selected region larger than a learned threshold". In the case of Faster R-CNN, this threshold is fixed to 0.7.

The last aspect of Faster R-CNN that will have an influence on computing time is the rescaling that is applied on images before being fed to the network at training and testing time. Generally, training the network at multiple images scales lead to better performances because the algorithm learns to detect objects of various size. However, according to the authors, the 'multi-scale feature extraction may improve accuracy but does not exhibit a good speed-accuracy trade-off". In the original algorithms, the images are thus "rescaled such that their shorter side is 600 pixels long".

Relying on this framework, the authors of the Face R-CNN paper propose "a robust deep face detection approach" by "exploiting new techniques including new multi-task loss function design, online hard example mining, and multi-scale training strategy to improve Faster R-CNN in multiple aspects." However, as stated before, I could not found an official implementation of this paper. However, I did find the official python implementation of the Faster R-CNN method at `https://github.com/rbgirshick/py-faster-rcnn` (derived from the official Matlab implementation) and also an implementation of face detection based on this repository at `https://github.com/playerkk/face-py-faster-rcnn`.

5.1.4 SFD

The *Single Shot Scale-invariant Face Detector* (S^3FD) is presented in [117] as a real-time anchor-based face detector, "which performs superiorly on various scales of faces with a single deep neural network, especially for small faces". On the one hand, this network follows the new approach proposed in "Faster R-CNN" of using so-called anchors from which the region proposals are defined. In addition, the authors mention that they were also inspired by another branch of anchor-based models exemplified in the *Single Shot MultiBox Detector* (SSD) method introduced in [56]. On the other hand, the authors of S^3FD criticize those methods especially on the subject of scales of face that are detected. I will therefore first present rapidly the second branch of anchor-based methods by which the method is inspired and then go through the improvements that S^3FD provides.

SSD is based mainly on the Faster R-CNN framework in that it uses "the convolutional priors (or anchor boxes)". In the case of Faster R-CNN, those anchors are used to make 'box-regression' (i.e. defining region proposals parametrized relatively to the anchors) and 'box-classification' (i.e. class the corresponding proposal as object or non-object) using the RPN network. The object detection part of the network then uses those proposals to predict if they contain the object of interest. Based on the YOLO [70] and OverFeat [77] networks, the idea of SSD is to merge those two steps in one by directly learning "to predict both the offsets over these priors and the confidences for multiple categories, instead of treating all categories as class-agnostic object". This approach thus "avoids the complication of merging RPN with Fast R-CNN and is much easier to train". The basic idea is therefore not just to predict if a region proposal is object or non-object but directly give it a score for each considered class.

As illustrated in Figure 5.7, S^3FD identifies three reasons common to both Faster R-CNN and SSD that make them bad at detecting small objects and proposes solutions to these problems. Firstly, the authors consider that there is a "biased framework". The two main ideas are that "the stride size of the lowest anchor-associated layer (i.e. feature map associated on which anchor are defined and from which region proposals will be generated) is too large" and that "anchor scale mismatches receptive fields and both are too large to fit small faces". For the authors, this has had a negative impact on the detection of small and medium faces. Their solution is to propose a "scale-equitable face detection framework". This solution is basically composed of tiling anchors using more stride sizes and to design the anchor scale according to the receptive field.

The second drawback that is pointed out is the discrepancy between discrete anchor scales and continuous face scale illustrated in Figure [117]. The authors specify that as a consequence "those faces whose scale distribute away from anchor scales cannot match enough anchors [...] leading to their low recall rate." To improve that, they propose a "scale compensation anchor matching strategy with two stages. The first stage follows current anchor matching method but adjusts a more reasonable threshold. The second stage ensures that every scale of faces matches enough anchors through scale compensation". Finally, trying to detect small faces poses a new problem which is that "plenty of small anchors have to be densely tiled on the image" leading to a "sharp increase in the number of negative anchors on the background, bringing about many false positive faces". The solution to this is to use a "max-out background level for the lowest detection layer".

Using these corrections, S^3FD achieves state-of-the-art results on AFW, PASCAL face, FDDB and WIDER FACE and "can run at 36 FPS on a Nvidia Titan X (Pascal) for VGA-resolution images" (640x480). To achieve this, it also filters out all detections with confidence

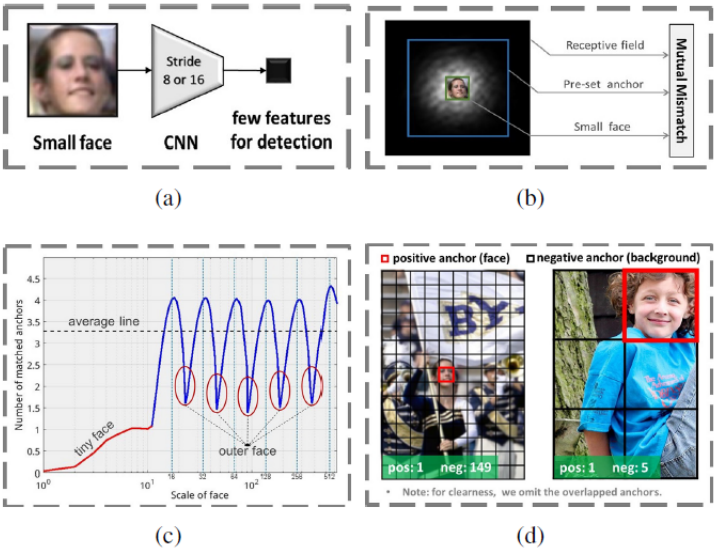


Figure 5.7: Illustrative figure from [117]. Description: "Reasons behind the problem of anchor-based methods. **(a) Few features:** Small faces have few features at detection layer. **(b) Mismatch:** Anchor scale mismatches receptive field and both are too large to fit small face. **(c) Anchor matching strategy:** The figure demonstrates the number of matched anchors at different face scales under current anchor matching method. It reflects that tiny and outer faces match too little anchors. **(d) Background from small anchors:** The two figures have the same resolution. The left one tiles small anchors to detect the small face and the right one tiles big anchors to detect the big face. Small anchors bring about plenty of negative anchors on the background."

threshold lower than 0.05 and applies NMS with a threshold of 0.3.

5.1.5 SSH

"SSH: Single Stage Headless Face Detector" [65] was published around the same time (i.e August 2017) than S^3FD and proposes a similar approach to SSD in the sense that it is also a single-stage detector that directly outputs a score for each class for each of the region proposals.

The two additional features of SSH is that it "is scale-invariant by design and can incorporate context efficiently". To deal with faces of various scales, the authors take inspiration from [55] by applying detection modules on three different convolutional layers as shown in Figure 5.8. The output of these three modules is similar to the one of SSD with boxes coordinates and scores for multiple categories. We can see that this approach is very different from SFD where the technique consists mainly in multiplying anchors by varying their sizes and strides.

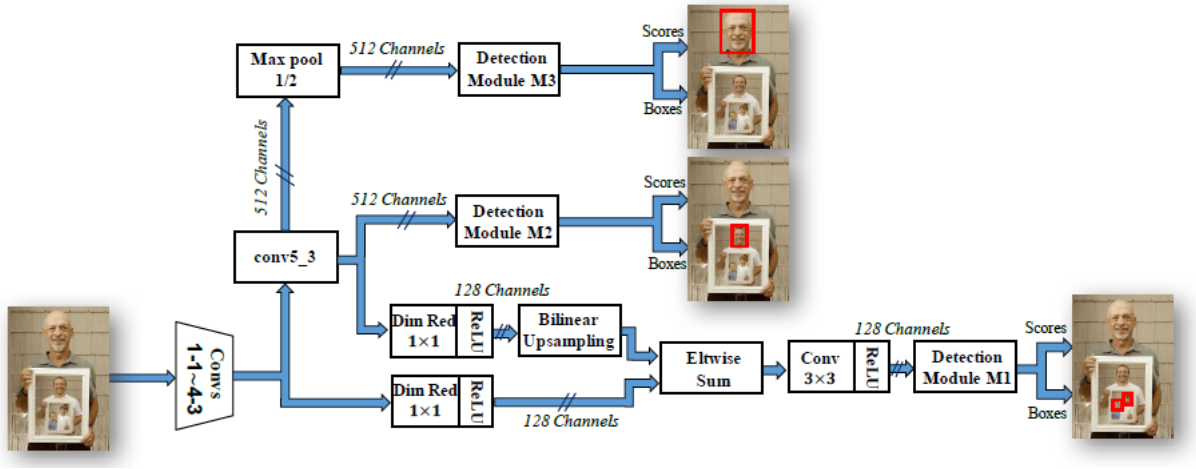
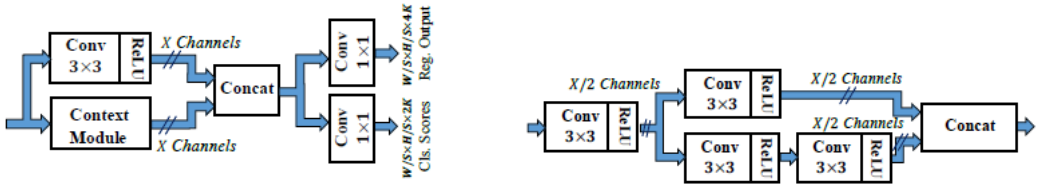


Figure 5.8: Illustrative figure from [65] of the network architecture of SSH.

Context seems to be beneficial for face detection or at least that is what we can infer from the claim that "in two stage-detectors [such as Faster R-CNN], it is common to incorporate context by enlarging the window around the candidate proposals". SSH follows this trend but implements this contextualization directly in the network by integrating a convolutional "context layer" in the detection modules as shown in Figure 5.9 (a). This context module consists mainly in applying a larger filter of size $5x5$ and $7x7$ rather than $3x3$. However, as shown in Figure 5.9 (b), these large convolutional filters are replaced by sequential 3x3 filters in order to reduce the number of parameters, which is a method similar to the one proposed for the Inception network in [86].



(a) (b)

Figure 5.9: Illustrative figures from [65] of the detection (a) and context (b) modules.

Then, similarly to Faster R-CNN, the basic implementation of SSH uses as input rescaled images. They preprocess the images by "rescaling the shortest side to 1,200 pixels while keeping the largest side below 1600 pixels without changing the aspect ratio". Another possibility is to use SSH with an image pyramid. The idea is simply to apply detection on the input image rescaled at a variable number of different sizes, four in this case. Of course, this has a major effect on the computation time but improves the performances.

Finally, once again to deal with multiple detections of the same face, SSH makes use of NMS with a threshold of 0.3. SSH shows state-of-the-art performances on FDDB, Pascal-Faces and WIDER FACE. Moreover, inference time ranges from 48 ms for 400x800 images to 182 ms for 1200x1600 images when performed on a NVIDIA Quadro P6000 GPU.

5.2 Face recognition

The 3 algorithms described in this section are referred to as OpenFace, Dlib-R, and ArcFace. As a short introduction and reminder, those three algorithms are based on deep neural networks architectures and their goal is to project pictures of images in a discriminative feature space where similar identities are clustered and separated from different identities.

5.2.1 OpenFace

As stated in [5], the OpenFace face recognition library is an effort to bridge the accuracy gap that exists between state-of-the-art private and publicly available face recognition systems. In this paper, they briefly present the application of deep learning to face recognition and explain the main ideas of two deep learning approaches called DeepFace [87] and FaceNet [74]. They finally describe their own method based on FaceNet. I will therefore first describe the FaceNet approach developed by researchers at Google and considered as a breakthrough paper by many public media and then go over the modifications brought in the context of OpenFace.

FaceNet is a system that "directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity". The feature extraction is done using a deep neural network as in previous work such as DeepFace [87] or DeepId2+ [103]. The authors specify that their method achieves greater "representational efficiency" because the output of their method is directly a "128-D embedding" while previous methods used an intermediate bottleneck layer as representation leading to 1000s of features. To achieve this, the authors do not consider the problem as a mere classification problem and use a specific loss function that they call triplet loss.

To goal of triplet loss is to ensure that "an image x_i^a (*anchor*) of a specific person is closer to all other images x_i^p (*positive*) of the same person than it is to any image x_i^n (*negative*) of any other person." Mathematically, it means that for each triplet i we want the Euclidean distance between the embeddings of x_i^a and x_i^p to be strictly smaller within a constant than the Euclidean distance between x_i^a and x_i^n. Mathematically, we want

$$||x_i^a - x_i^p||_2^2 + \alpha < ||x_i^a - x_i^n||_2^2 \; \forall \; (x_i^a, x_i^p, x_i^n) \in \mathcal{T} \tag{5.1}$$

where $\mathcal{T}$ is the set of all possible triplets in the training set. In addition to that, authors also explain the technique to select "hard triplets" that will provide fast training convergence. Indeed, they specify that "all possible triplets would result in many triplets that are easily satisfied" and thus "these triplets would not contribute to the training and result in slower convergence". Finally, they "constrain the embedding to live on the d-dimensional hypersphere so that its norm is equal to 1.

This learning process is then applied to a pre-defined neural network architecture. In particular, the two following architectures are tested. The first one is based on the Zeiler&Fergus model [114]. This model consists of "multiple interleaved layers of convolutions, non-linear activations, local response normalizations and max pooling layers" and is complemented by adding several "1x1xd convolution layers" "between the standard convolutional layers" resulting "in a model 22 layers deep". The second is based on "GoogleNet style Inception [86] models" which use "mixed layers that run several different convolutional and pooling layers in parallel and concatenate their responses". This second model is much lighter with "20x fewer parameters"

and "5x fewer FLOPS". From these two base models, they built in total 6 different models by varying several parameters.

Finally, nearly no preprocessing is applied to the images before they are fed to the network during training or testing. Indeed, "the thumbnails are tight crops of the face area, no 2D or 3D alignment, other than scale and translation is performed".

This last point marks the main difference between OpenFace and FaceNet. Indeed, in the case of OpenFace, 68 face landmarks are detected using Dlib's face landmark detector [46] and they are then used to apply a simple 2D affine transformation in order to make the "eyes and nose appear in similar locations for the neural network input" The justification for this extra alignment is that OpenFace does not have the same means to deal with pose and illumination variations than FaceNet. Indeed, FaceNet can deal with this problem by training their algorithm on a huge private dataset of "100-200M images" while OpenFace is trained with only "500k images" coming from the CASIA-WebFace [113] and FaceScrub datasets.

5.2.2 Dlib-R

Dlib is an open-source library proposing a series of machine learning algorithms. As mentioned on its website[1], "Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real-world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high-performance computing environments". During an update trying to introduce "deep metric learning tooling", a face recognition program was added to the library as presented in `http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html`. However, Dlib's documentation is really scarce. Applied examples are the only source of information with some undocumented prototype.

As with OpenFace, the goal of the face recognition system is to produce a low-dimension feature representation of faces. The underlying neural network architecture is a "ResNet network with 29 conv[ulational] layers". It consists basically in a simplified version of ResNet-34 [36] with "a few layers removed and the numbers of filters per layer reduced by half". On top of that network, "a type of pair-wise hinge loss that runs over all pairs in a mini-batch" is used. The goal of this loss is to "project all the identities into non-overlapping balls of radius 0.6" which is similar to FaceNet. Finally, the network is trained on FaceScrub, the VGG dataset [68] and "a large number of images" that the author "scraped from the Internet".

5.2.3 ArcFace

ArcFace is the name of a loss function and of the algorithm that use it as described in "ArcFace: Additive Angular Margin Loss for Deep Face Recognition" [21] which is one of the most recent research paper on face recognition (Jan 2018). As with each new paper on face recognition, the goal of the authors was to improve the state-of-the-art results. To do so, they identify three main attributes that differentiate face recognition models: the training data employed to train the model, the network architecture and settings and the design of the loss functions. They then try to work on these three attributes.

[1] Dlib: `http://dlib.net/`

Regarding the training data, the main work was to refine one of the chosen training dataset, MS-Celeb-1M. This cleaning process was made semi-automatically. The main idea was to use a face recognition algorithm to "rank all face images of each identity by their distances to the identity center" and to automatically remove the face images "whose feature vector are too far from the identity's feature center [22]". Moreover, the face images that were around the threshold are manually checked. This refined dataset is used for training the final model that is tested on the MegaFace dataset. However, the VGG2 dataset is also used to train other networks to make a series of experiments. The tests on MegaFace are actually made on two versions of this benchmark: the original one and a refined one. The authors show with examples that there is noise both in FaceScrub (i.e. faces associated with the wrong identity) and that there is overlap between FaceScrub and the MegaFace dataset over identities. This process can improve accuracy by about 1%. For the first problem, 605 (on 100k) noisy faces were replaced by faces of the right identity. For the overlap problem, the authors identified 707 noisy face images and by taking care of these faces correctly, the accuracy was improved by up to 15%. The authors mention that this list of noisy images would be put online, I could not find this list.

The authors then compared different network architectures by varying a series of parameters. All these networks follow some common structure. They all take as input a 112x112 RGB images that are obtained by aligning, cropping, resizing and normalizing the original face images. The first difference comes from the output size which can be set to 7x7 (denoted with a L at the beginning of the network) or 3x3. Then 5 different options are tested for the embedding setting. Then two different versions of the ResNet network are tested. The second version is obtained by replacing the residual units by an improved version called 'IR' (for improved residual). Finally, a series of other network architectures are compared including MobileNet [38], Inception-ResNet-V2 [85], DenseNet [41], Squeeze and excitation networks [39] and Dual Path Network [17]. The result of their study gives that the ResNet with the L output, using a BN-dropout-FC-BN layer after the convolution layer as embedding setting and IR residual units gives the best result. The final model that they use to do the comparison of performance between the different losses is the LResNet100E-IR.

The final test consists in comparing the performance of algorithm using different loss functions. Before that, the authors make a review of the different loss functions that have been developed for face recognition using deep learning. These losses can be divided into two main classes. The earlier class that the authors call "Euclidean margin-based loss" contains the softmax loss used for example in DeepFace but also center [100], range [118] and marginal [22] loss that are all losses that need to be combined with softmax. These losses are used when considering the problem of face recognition as a classification problem. The issue with these losses is that they suffer from "massive GPU memory consumption when the identity number increases to million level" and that "balanced and sufficient training data for each identity" is preferred. An alternative to those losses in the same class is exemplified by the triplet loss [74] or the contrastive loss [84] which uses "pair training strategy", the issue being now to select "effective training samples". The second class of losses are based on modifications on the softmax loss and called "angular and cosine margin-based loss". L-softmax [58] or additive cosine margin [96] are exemplars of this class and ArcFace is also part of it. The authors specify that the losses of this second class are better because they add "discriminative constraints on a hypersphere manifold, which intrinsically matches the prior that human face lies on a manifold." Compared to its class predecessors, the authors claim that ArcFace, or additive angular margin, has "better geometrical interpretation" which allows it to obtain "more discriminative deep features".

Finally, the test shows that indeed ArcFace performs better on several benchmarks than other losses including the one of the same class. It is interesting to mention also that the authors show that fine-tuning the network with triplet-loss after having training it with softmax loss improves the performances further.

Chapter 6

Algorithm Evaluation

This chapter contains a series of results obtained through the evaluation of the face detection and face recognition algorithms that were described in the previous section on two benchmarks that were selected in chapter 3. For both face detection and recognition, before entering the analysis of the results per-say, I explain in more details the evaluation protocol that will be applied and I detail how the time performances will be evaluated. In addition, for face detection, as the WIDER FACE has been annotated with a variety of facial attributes, I will analyze the influence of each of these factors on the performances of the algorithm.

6.1 Face detection

6.1.1 Evaluation protocol

Based on the elements analyzed in section 3.1.8, I decided to use WIDER FACE as a testing benchmark. Before presenting the results, I am going to explain in more details the testing approach used by the evaluation toolbox of WIDER FACE.

The toolbox comes with a series of functions coded in Matlab for evaluating and plotting. In addition to that, three '.mat' files contain data structures defining which faces in each image compose the 'easy', 'medium' and 'hard' subsets and providing the meta-data associated with each face. By looking at the list of faces that compose each of these subsets, we can actually notice that the 'hard' set is a superset of the 'medium' set which is, in turn, a superset of the 'easy' dataset. I decided to keep these sets as they were even though I could have restructured them so that they become disjoint. The total number of faces to detect in each subset is 7,211, 13,319 and 31,958 for the easy, medium and hard subset respectively.

Every face is associated to one value for each of the following attributes: blur (clear:0, normal blur:1, heavy blur:2), expression (typical:0, exaggerate:1), illumination (normal:0, extreme:1), occlusion (none:0, partial:1 or heavy:2) and pose (typical:0 or atypical:1). In addition to that, I used the ground truth bounding boxes to create similar labels for size with the following notation: $height > 300 : 0$, $height \in]50; 300] : 1$, $height <= 50 : 2$. The following figure shows the distribution of faces according to those labels and across the three subsets (easy in green, medium in blue and hard in red) and will be discussed in more details in section 6.1.5.

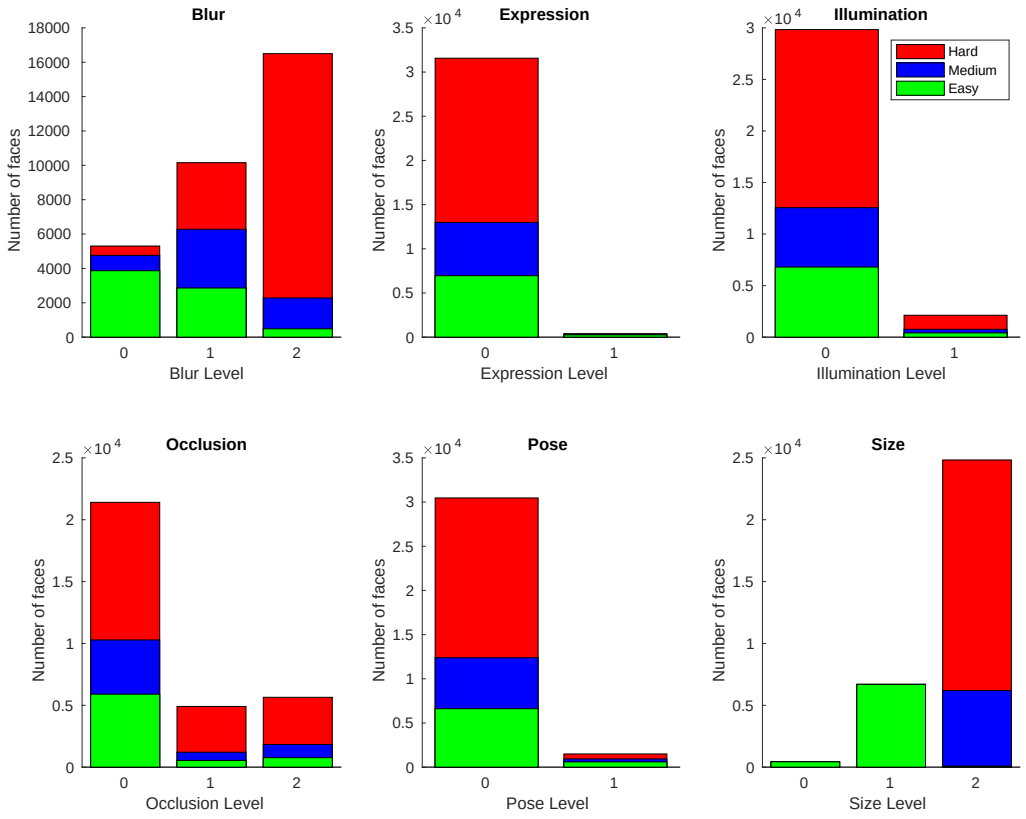


Figure 6.1: WIDER FACE dataset statistics. The figures show how many faces are annotated with each value (ranging from 0 to 1 or 0 to 2) of 6 different features. For each of these values, the color bars indicate how many faces associated with this value are contained in each of the three subsets of WIDER FACE.

Finally, an invalid flag (valid:0, invalid:1) is associated to each face. However, none of the faces in the subsets are set to 1 so we do not need to pay attention to this flag.

The '.mat' files also contain the ground truth for every face, even the non-tested or invalid ones. The ground-truth consists of rectangular bounding boxes reported as 'x1, y1, w, h'. x1 and y1 define the position of the upper-left corner of the rectangular bounding box, x1 being the position in pixels from the left side of the image and y1 the position in pixels from the top side of the image. w is the width and h is the height, both in pixels, of the image.

To evaluate the face detection performance of an algorithm, the evaluation function expects to receive for each image containing a face in the evaluation set, a list of bounding boxes in the same format as the ground truth, each of which must be associated with a confidence score. The list needs to be sorted in decreasing order according to this score. The score has ideally to be between 0 and 1 but a function is provided to normalize it if need be. During the evaluation, thresholds are defined ranging from 0 to 0.999 with a step of 0.001 and compared against the confidence scores in order to define a series of precision-recall points. Indeed, at each threshold,

all the bounding boxes associated with a score equal or below this threshold are not considered for evaluation. This will, therefore, make the relative number of true positive, false negative and false positive vary and therefore precision and recall too. It is important to notice that as the maximum value of the thresholds is 0.999, the precision-recall curve might not reach the point (1,0).

For a given threshold and a given image, a list of predictions will be compared against the ground-truth bounding boxes and from this comparison, we can compute precision and recall values for this threshold and image. However, a problem appears when a prediction corresponds to a face that should not be considered for the currently evaluated subset (which can happen because we are not considering the hard subset or because it was marked with an ignore flag). To deal with this problem, predictions that match with such a face are simply ignored for the precision-recall computations.

6.1.2 Time evaluation

As stated in section 2.6, an important part of this thesis is to find if state-of-the-art algorithms can work in a practical environment with a given amount of computing power. I will therefore in this evaluation section compare the computing times of each face detection algorithms. I will briefly explain what was exactly measured and how.

The time was measured during the tests on the validation set of WIDER FACE. This dataset is composed of images of constant 1024 pixels height but of varying width with a median of 754 pixels. Because of this characteristic, I decided to measure the detection time for each frame individually so that I could observe the influence of size on the computation time while also being able to take the median time over all frames in order to get rid of outliers.

For each image, the time was measured over the whole face detection phase and only that, meaning that the time takes into account:

- All preprocessing (resizing, ...) and postprocessing (NMS, ...) operations
- Not the loading of libraries or networks
- Not the saving of the data.

A question that I asked myself was which time to measure. Indeed, for algorithms written in Python 2.7, one can easily have access to both CPU and real time (through the functions `clock` and `time` of the `time`[1] library respectively) while in C++[2], I could only have access to CPU time through the `clock`[3] function of the `ctime` library. Another possibility to measure time is to use the `time`[4] command in the Linux terminal. However, this command allows only to compute the whole execution, real and CPU, time which does not allow a precise time evaluation.

In my opinion, both real and CPU times are interesting. Indeed, the first one gives an idea of the practical capabilities of an algorithm while the second allows a fairer comparison of

[1]Python 2.7 `time` library: `https://docs.python.org/2/library/time.html`
[2]Those are the two languages in which the algorithms were written as explained further.
[3]C++ `clock` function: `http://www.cplusplus.com/reference/ctime/clock/`
[4]Linux `time` command: `https://linux.die.net/man/1/time`

those capabilities across different computing architectures. I, therefore, decided to keep both times (except in the case of C++ implementations). These times will either be expressed by the number of seconds to analyze one image or as the number of frames that can be processed per seconds (FPS) depending on the situation. The second one being computed by dividing 1 by the first one, and vice-versa.

Furthermore, to be able to analyze those results in more details, I decided to keep the information output by the `htop`[5] command indicating memory and CPU usage and the `nvidia-smi`[6] command providing data about the GPU usage. To not overwhelm the results with data, I will only refer to those when I consider it appropriate.

6.1.3 Per-algorithm results

VJ

The best-known and most used implementation of VJ is the one of the computer vision library, OpenCV. The first important part of the implementation is the cascade classifier. OpenCV proposes a series of pre-trained classifiers stored in publicly available XML files. The second part is the algorithm in itself. In OpenCV, the algorithm is coded, both in C++ and Python, in a function called `detectMultiScale` which takes a series of parameters listed on this page: `https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html`.

Among these parameters, four of them are of particular interest. First, `scaleFactor` is the parameter evoked earlier that defines by how much the sliding window detector is scaled at each step. Related to this parameter are the `minSize` and `maxSize` parameters that define respectively the initial and final size of the detector across the scaling process. The fourth parameter is called `minNeighbours` and is defined as "specifying how many neighbors each candidate rectangle should have to retain it". However, when looking at the two papers, there is no mention of neighbors. Strangely enough, there does not seem to be more precise documentation about this parameter. However, by scanning through forums, this parameter would be linked to the multiple detection problem. Indeed, this parameter would define the minimum number of overlapping neighbors a bounding region would need to have not to be removed from the retained detections. By increasing this parameter, we lower the number of false positives by removing redundant detections but this could be at the price of missing some faces.

Following this implementation, there is a number of choices that can be made to analyze the performances of VJ. First, we have to decide which classifier to use. For frontal face detection, the library proposes 6 different classifiers[7]. Then we have to fix the value of the scaling and multiple detection parameters. As the RAGI team is already using this algorithm with fixed values for these parameters, I will make choices based on theirs. Here is the list of parameters they used:

- `Cascadeclassifier`: haarcascade_frontalface_alt2.xml
- `scaleFactor` : 1.3

[5]Linux `htop` command: `http://man7.org/linux/man-pages/man1/htop.1.html`

[6]Linux `nvidia-smi` command: `https://developer.nvidia.com/nvidia-system-management-interface`

[7]There is not a lot of information about the differences between those models except for a few lines at the beginning of the .xml files defining each of these models.

- `minNeighbors` : 7
- `minSize` : 64x64
- `maxSize` : Default = Size of the image

For the analysis, we could use only those parameters but they would give us a single point on the precision-recall curve. For me, two parameters could be interesting to modify. The first one is the `minNeighbors` parameter. Indeed, this is the only parameter that allows playing directly on the ratio of false and true positives enabling therefore to draw a precision-recall curve. To this regard, I must point out that I had to change the evaluation method slightly. Indeed as explained in section 6.1.1, the precision-recall curve is computed using a series of threshold on the confidence level. As there is no confidence level for VJ, I artificially set it to 1 for all predictions. This has the effect to produce only one single precision-recall value for a given value of `minNeighbors`. To obtain sufficient points on the precision-recall curve, I will test values ranging from 0 to 10 for this parameter.

The second parameter is `minSize`. Indeed, the WIDER FACE benchmark evaluates algorithm on faces as small as 10 pixels in height. It would, therefore, be interesting to test this algorithm with `minSize` set to 10x10 so that its performances are comparable to other algorithms. The last parameter that could be modified but whose impact is quite straightforward is `scaleFactor`. Indeed, without too much surprise, decreasing this parameter would increase the number of detections but at the price of heavier computational time. I will not study the impact of this parameter at first.

Finally, the implementation of this function exists and is similar in both C++ and Python. I will therefore test the implementation in both languages. With these considerations in mind, I list here-under the four main cases that I will test. Note that for each of those configurations, I will obtain one precision-recall curve by modifying the `minNeighbors` parameter which is why this parameter does not appear in the names.

Name	Coding Language	`scaleFactor`	`minSize`
VJ-C++-scale-1.3-min-10	C++	1.3	10x10
VJ-C++-scale-1.3-min-64	C++	1.3	64x64
VJ-python-scale-1.3-min-10	Python	1.3	10x10
VJ-python-scale-1.3-min-64	Python	1.3	64x64

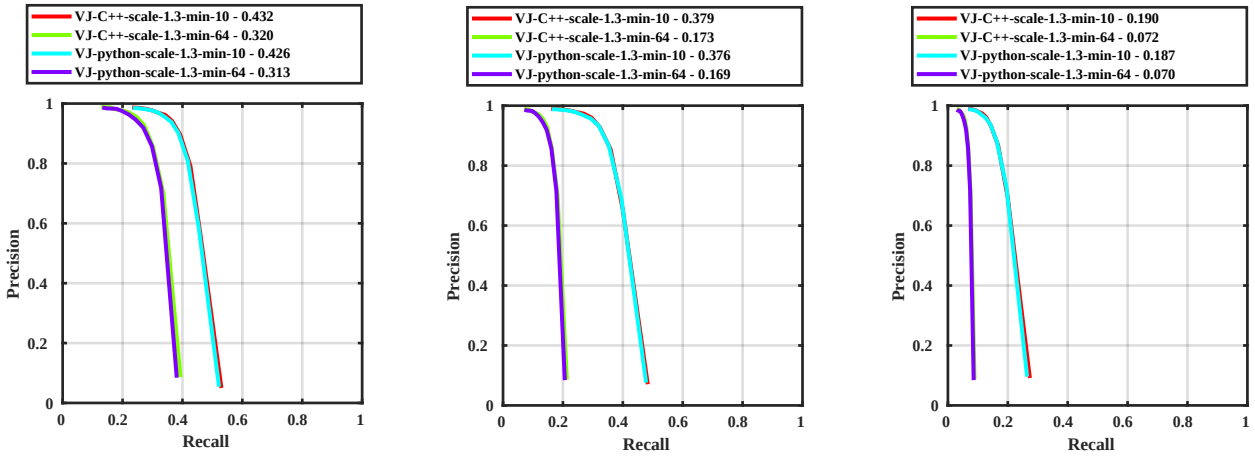
Table 6.1: List of tested versions of the VJ algorithm with varying coding language and `minSize`.

Figure 6.2 shows the ROC curves for these four configurations on the easy, medium and hard subsets of WIDER FACE while Table 6.2 contains the number of frames that can be processed per second, for both CPU and real time.

The first observation we can make is that changing the implementation language does not affect substantially the performances of the algorithm, even though the C++ implementation seems to generally perform better. However, in terms of computation efficiency, it seems like the python implementation performs better than the C++ one by improving CPU time by 12,3% and 1,3%, for `minScale` = 10x10 and 64x64 respectively. We can see that the smaller `minScale` is the greater the gap in computation time between the two implementations. However, `minScale` affects the two implementations in the same way: when it decreases, meaning

that the algorithm can try to detect smaller faces, their performances improve while their computation times increase. We see that in terms of real-time FPS, we still achieve a reasonably good number when we use `minScale` = 10x10 while we can see a drastic improvement in the performances, especially for the harder subsets. Overall, we can see that using VJ we can obtain good precision but at the default of low recall which even for the easy subset painstakingly reaches 0.5 if we set `minNeighbors` to 0.

Another interesting characteristic in Table 6.2 is to see the large difference between CPU and real time demonstrating that the implementation makes good use of multi-threading. Finally, an aspect that is not present in this table is the variation of computation time according to `minNeighbors`. This is due to the fact that, according to my tests, this factor does not influence significantly the computation time.



(a) Easy subset (b) Medium subset (c) Hard subset

Figure 6.2: Precision-recall curves obtained with the C++ and Python OpenCV implementation of the VJ detection algorithm with varying values of `minSize` on the three subsets of WIDER FACE.

Algorithm	CPU FPS	Real FPS
VJ-C++-scale-1.3-min-10	2.28	/
VJ-C++-scale-1.3-min-64	7.51	/
VJ-python-scale-1.3-min-10	2.56	35.98
VJ-python-scale-1.3-min-64	7.60	61.38

Table 6.2: Medians of the number of frames, coming from WIDER FACE, that can be processed per second, in CPU and real time, by the C++ and Python OpenCV implementation of the VJ face detection algorithm with varying values of `minSize`.

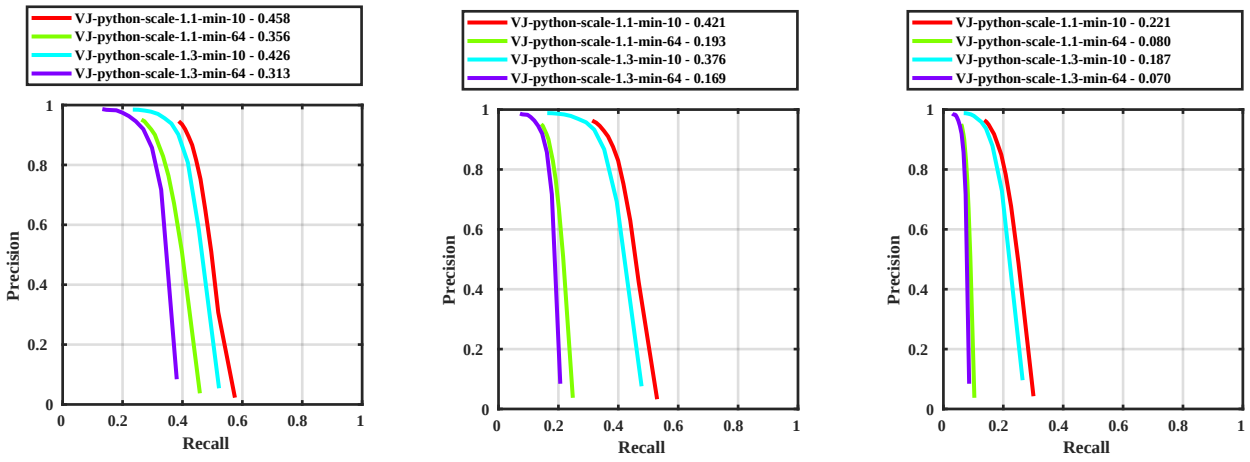
Having seen these first results and the high level of FPS that could be reached even with a small value of `minSize`, I decided to make some additional test by setting `scaleFactor` to its default value, 1.1. The goal of this experiment is mainly to see if there is an interest in decreasing the value of this parameter. As the Python implementation was the fastest and that its performances are similar to the C++ implementation, I decided to only test two additional

settings listed in Table 6.3 and to compare them to the previous python settings. Figure 6.3 and Table 6.4 illustrate this comparison. The main observation is that there is a performance improvement compared to the previous setting but that this improvement is significantly smaller compared to the one obtained when reducing the value of `minSize`. Moreover, the computation time degrades badly. Indeed, we can see that the number of FPS is divided by more than two both in terms of CPU and real time.

Name	Coding Language	`scaleFactor`	`minSize`
VJ-python-scale-1.1-min-10	Python	1.1	10x10
VJ-python-scale-1.1-min-64	Python	1.1	64x64

Table 6.3: List of additional tested versions of the VJ algorithm written in Python with varying `scaleFactor` and `minSize`.

According to these results, I will focus only on the VJ-python-scale-1.3-min-10 version for comparison with other algorithms as it furnishes one of the highest performance levels with more than reasonable speed.



(a) Easy subset (b) Medium subset (c) Hard subset

Figure 6.3: Precision-recall curves obtained with the Python OpenCV implementation of the VJ face detection algorithm with varying values of `minSize` and `scaleFactor` on the three subsets of WIDER FACE.

Algorithm	CPU FPS	Real FPS
VJ-python-scale-1.1-min-10	1.08	14.39
VJ-python-scale-1.1-min-64	4.04	27.99
VJ-python-scale-1.3-min-10	2.56	35.98
VJ-python-scale-1.3-min-64	7.60	61.38

Table 6.4: Medians of the number of frames, coming from WIDER FACE, that can be processed per second, in CPU and real time, by the Python OpenCV implementation of the VJ detection algorithm with varying values of `minSize` and `scaleFactor`.

HOG

This algorithm was implemented in the Dlib library introduced in section 5.2.2. Dlib provides implementations of the algorithm in Python and C++ but for both of them the documentation is very scarce except for few examples.

By looking at those sources, we can see that the face detection algorithm takes only two arguments: the image and a parameter called `adjust_threshold`. I could not find the meaning of this threshold and it does not seem to affect performances according to my tests. In the C++ implementation, a function name `pyramid_up` is used in order to increase the size of the image. This method is apparently used to cope with the fact that the initial detector only looks for faces of size larger than 80x80 pixels. However, using this method consequently slows down the computation time. In this regards, I am only going to test the basic method as proposed in the examples. The problem with this decision is that we are going to obtain only one single precision-recall point which will make it more difficult to compare the performances of this method to others.

Table 6.5 shows the precision-recall results for the two implementations on WIDER FACE and Table 6.6 shows the FPS obtained in real and CPU time. We can see that quite similarly to VJ, HOG can achieve high precision but with a low maximum recall rate of 51.6% on the easiest subset. This value then decreases for the harder subsets. Precision does not change across subsets for the C++ implementation. This would mean that the algorithm does not produce detections for faces that are only in the medium and hard datasets. This is totally plausible as all faces above 50 pixels in height are in the easy subset (see Figure 6.1) and that the C++ implementation does not detect faces under 80x80 pixels. Quite surprisingly, for the Python implementation, the precision even increases. By comparing the two implementations, we can see that the Python implementation is superior to the C++ one, especially in terms of recall.

However, when we look at computation time, the C++ implementation is nearly 5 times faster than the Python implementation. The latter reaches only 3 FPS which is very slow for a practical implementation. For this reason, I will continue the further analysis with the C++ implementation. We can finally note that the number of FPS in real time is very close to and even smaller than this number in CPU time meaning that the CPU time is inferior to real time. This seems to indicate that the algorithm is not running using parallel processing and by looking at the output of `htop`, this seems to be the case for both implementations. It is important to keep that in mind because it probably implies that the algorithm could actually run faster.

Subset	Easy		Medium		Hard	
Algorithm	Precision	Recall	Precision	Recall	Precision	Recall
HOG-python	0.9444	0.5160	0.9619	0.4153	0.9621	0.1744
HOG-C++	0.9611	0.3599	0.9611	0.1948	0.9611	0.0812

Table 6.5: Precision-recall values obtained with the Dlib Python and C++ implementations of the HOG face detection algorithm.

Algorithm	CPU FPS	Real FPS
HOG-python	3.22	3.07
HOG-C++	15.94	/

Table 6.6: Medians of the number of frames, coming from WIDER FACE, that can be processed per second, in CPU and real time, by the Dlib Python and C++ implementations of the HOG face detection algorithm.

FRCNN

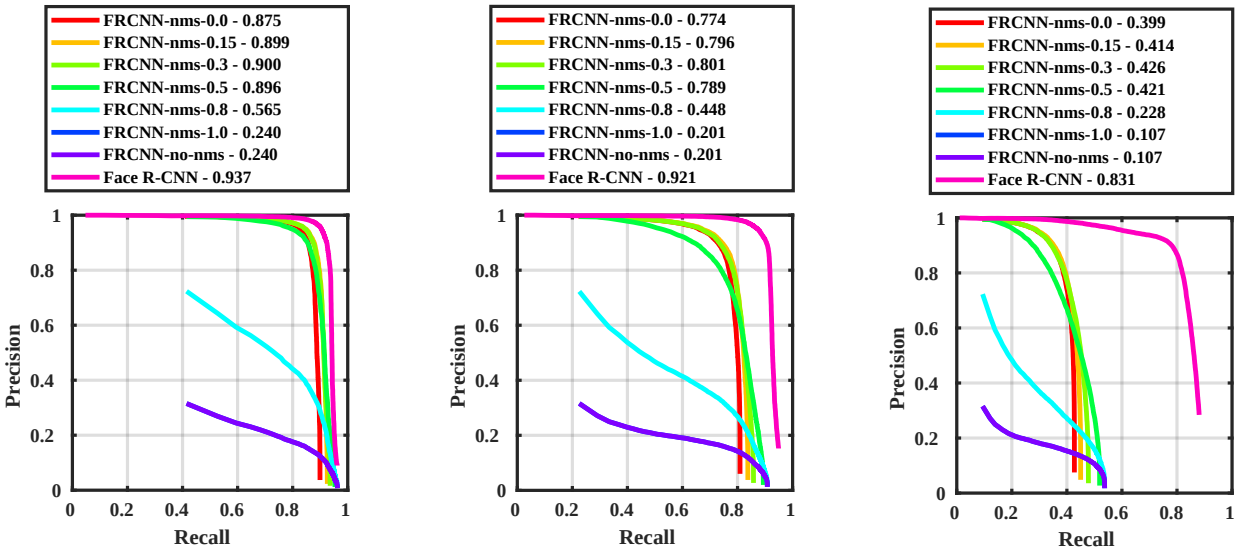
The implementation found at `https://github.com/playerkk/face-py-faster-rcnn` was written in Python and uses Caffe deep learning framework. The repository contains training and testing code as well as a pretrained face detection model based on the VGG16 architecture [79] and trained on the WIDER FACE training set. We need to take this into account when comparing algorithm as they are tested on the WIDER FACE dataset (even though not on the same partition).

As described in the original Faster R-CNN paper, the algorithm outputs for each image a list of bounding boxes with confidence score ranging from 0 to 1. The testing algorithm then performs two additional steps. First, it removes all detections associated with a score below a certain threshold. However, as I am trying to obtain the most complete precision-recall curve and due to the way the evaluation procedure works, I set this parameter to 0 so that all the detections are kept. Then, NMS is applied. As this filtering also relies on a threshold, I will modify its value to see its influence on performance and time. I will use as values 0 (suppress everything but one detection), 0.15 (default), 0.3, 0.5 , 0.8 and 1 (suppress nothing). I will also try to not use NMS to see mainly the influence on computation time. Finally, the WIDER FACE evaluation toolbox comes with a series of pre-computed results for a series of algorithms. These algorithms contain Face R-CNN and I will, therefore, display its precision-recall to compare it to the ones I obtain with FRCNN.

The main information to gather from Table 6.7 is that the variation of the NMS threshold value does not affect the computation time, probably because applying NMS is computationally negligible compared to the detection phase as we can see that not applying NMS does not even improve the computation time compared to certain cases where it is used. For FRCNN, this detection phase is still quite efficient because we can obtain a rate of more the 15 FPS. Finally, we can see the effect of parallel processing by looking at the difference between CPU and real time FPS. This difference is smaller than with VJ but that might be explained by the fact that, in the case of FRCNN, the computation occurs also on the GPU. This time is therefore only considered as real time which makes the number of real time FPS decrease compared to CPU time FPS.

Figure 6.4 shows us that FRCNN can reach much higher recall rate than the two previously tested methods with more than 90% on the easy subset for some values of the NMS threshold. When looking at the variation of the NMS threshold, we can see that for the three subsets the behavior is identical. When the NMS threshold increases, at low-recall levels, precision decreases while at low precision-level, recall increases. Moreover, we can see that, up to a threshold of 0.5, the increase of recall still compensates the decrease in precision, with an average precision (AP) staying at a reasonable value, but that increasing the threshold worsen the performances.

Then we can notice that the results for a NMS threshold value of 1.0 and when not using NMS are the same as expected considering the workings of NMS. Another remarkable feature in this figure is that while the performances of the tested algorithms decrease significantly when increasing the difficulty of the subset, the curves of Face R-CNN are quite consistent and lead to very good performances even on the hardest subset. Finally, if we focus mainly on the AP as performance measure, it seems like a value of 0.3 for the NMS threshold is the best value to chose.



(a) Easy subset (b) Medium subset (c) Hard subset

Figure 6.4: Precision-recall curves obtained with the FRCNN algorithm using different values of the NMS threshold, or no NMS, on the three subsets of WIDER FACE.

Algorithm	CPU FPS	Real FPS
FRCNN-nms-0.0	8.83	15.69
FRCNN-nms-0.15	8.82	15.59
FRCNN-nms-0.3	8.84	15.69
FRCNN-nms-0.5	8.84	15.69
FRCNN-nms-0.8	8.83	15.62
FRCNN-nms-1.0	8.82	15.51
FRCNN-no-nms	8.83	15.65

Table 6.7: Medians of the number of frames, coming from WIDER FACE, that can be processed per second, in CPU and real time, by the FRCNN algorithm using different values of the NMS threshold, or no NMS.

SFD

The implementation that I use for this algorithm is the one referenced in the paper presenting the method, which corresponds normally to the one tested on the WIDER FACE dataset. This implementation is written in Python and uses Caffe as deep learning framework. Moreover, the implementation of this repository is in part based on the SSD reposi-

tory `https://github.com/weiliu89/caffe/tree/ssd`. The repository contains code both for training and testing the method on the four previously mentioned dataset. For the testing part, a pre-trained VGG network [79] on the WIDER FACE training set is furnished.

Concerning the test implementation in itself, the repository furnishes prediction scripts dedicated for each of the four pre-cited databases. While the tests are quite simple for the other datasets, in the case of WIDER FACE, it seems like the authors tried to innovate. First, a shrinking factor is computed to take care of images that would be too large to use with the network, according to the script. This factor is computed by this formula: $max\_im\_shrink = \sqrt{\frac{0x7fffffff/577}{w*h}}$ where w and h are respectively the width and height of the image in pixels. If this value is superior to 1, then the image is not too large and the final factor $shrink$ is set to 1 so that no shrinking occurs, in the other case $shrink = max\_im\_shrink$.

Then the detection is performed. The script actually proposes three variations of the face detection algorithm. The first one is the basic algorithm. The image is resized by shrinking its height and width by $shrink$ using linear interpolation and face detection is applied to this image. The second variation is similar to the first one except it uses the mirror image. Finally, the third variation is coded in a function called `multi_scale_test` and aims at detecting faces at smaller sizes by resizing multiple times the original image.

In the original script, the three variations (denoted `det0`, `det1` and `det23` respectively) are used and the detections obtained are stacked together. A last set of operations is applied to these stacked detections through a function called `bbox_vote`. The goal of this function, as I inferred it from the code, is to combine detections that overlap too much, similarly to NMS. The difference with NMS is that the bounding boxes that overlap too much with a maximum confidence bounding box are not just deleted but all the bounding boxes are replaced by a bounding box whose coordinates are a weighted average of the coordinates of the overlapping bounding boxes, similarly to what was described in the VJ paper. This technique relies also on a threshold to compare to the IoU. Initially, this value is fixed to 0.3 but it is quite easy to change the function to modify this value.

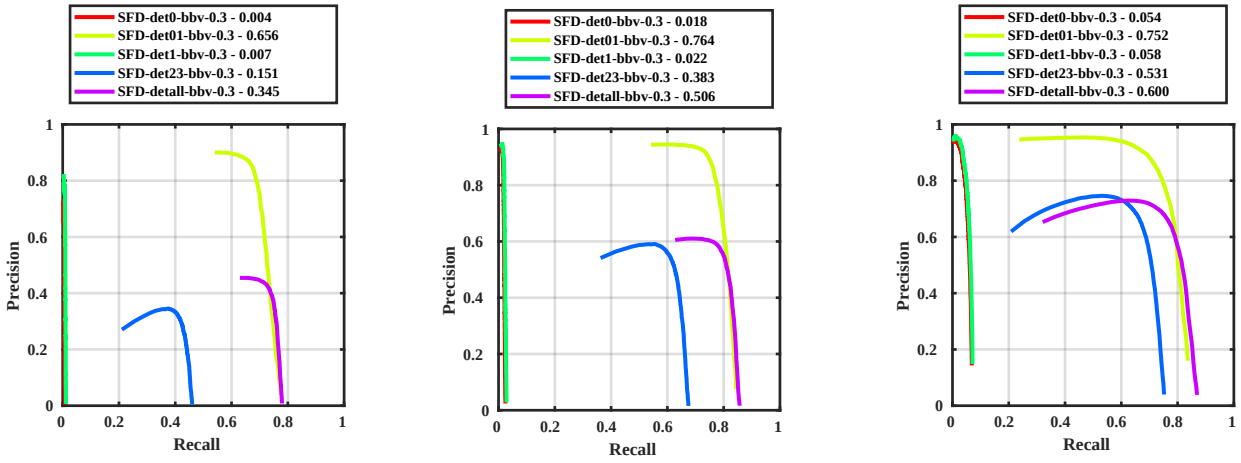
The different variations of the algorithm open the opportunity to a series of test by combining them together. My original idea was to test first the combination of the three variations (denoted as `detall`), by concatenating the predictions and then applying `bbox_vote` to all of them. Then I wanted to test each one of them independently. Finally, I had the plan to test other combinations to find the best compromise between performance and speed. However, my experiments did not turn out as I expected, as the results that I obtained were very strange. To understand these results I made diverse tests that I will try to summarize here-under.

First, I tried what I just presented. Figure 6.5 shows the result for the four cases evoked before plus the combination of first and second variations (that I call `det01`). The `bbox_vote` threshold was at 0.3 for all these experiments. There is a lot to say about this graphs. First, one strange thing but that may be explained by the way the subsets are defined is that the performance of the algorithm seems to be better on harder subsets. This seems strange as all the other algorithms tended to work worse on harder datasets but as the difficulty of the subset was defined by using a face detection algorithm, this could be possible.

Then we see that if we combine all the variations together (i.e. `detall`), we obtain a curve

that has a quite good recall for low precision values but that when the recall rate decreases, the precision rises and then starts to come back down again. This is especially visible on the hard subset. This behavior means that when we increase the decision threshold (i.e. the threshold above which the confidence level of the bounding boxes must be to be considered in the computation of precision and recall), the number of TP decrease more rapidly than the number of FP. Another way to put it is that at higher confidence levels the ratio of TP to FP is smaller than at lower confidence levels. This behavior was already strange but still explicable.

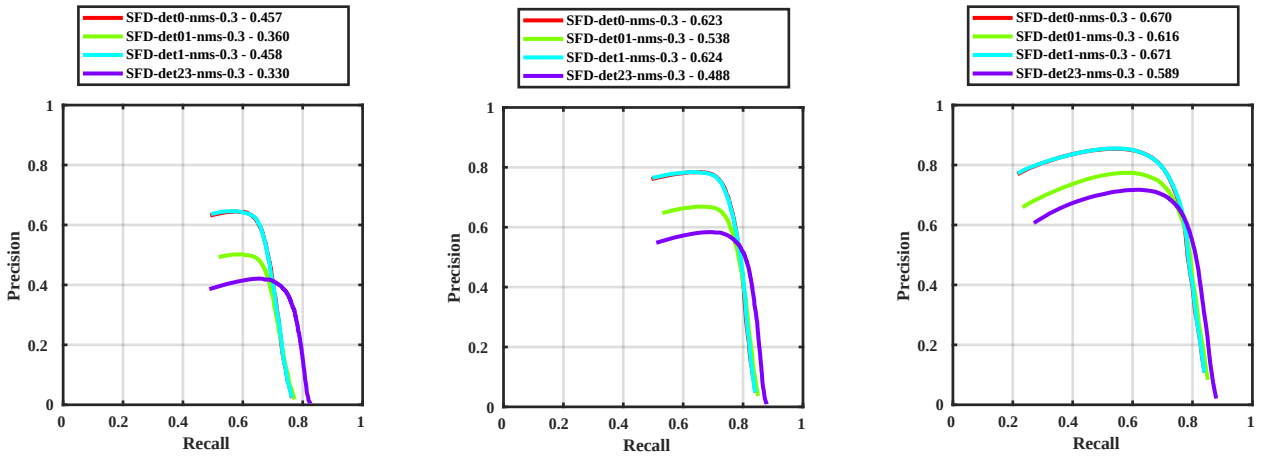
However, looking at the other curves started to make me very doubtful about the implementation of the algorithm, or the way I was using it. Indeed, when looking at the `det23` curve, we still have the same behavior but with a bit less accuracy which seems plausible. We could infer that adding the result of `det0` and `det1` would improve the `det23` alone. However, the increase in performance seems quite huge when we look at the curves for `det0` and `det1`. Indeed, their recall level is nearly 0 for the easy dataset. However, when adding their results to `det23`, we obtain an increase in recall of more than 20%. But the strangest part comes from the `det01` curve which displays the results from the combination of the results of `det0` and `det1` algorithm.



(a) Easy subset (b) Medium subset (c) Hard subset

Figure 6.5: Precision-recall curves obtained with the variations using bbox_vote of the SFD algorithm on the three subsets of WIDER FACE.

The implementation of `bbox_vote` being a bit suspect to me I decided to replace it by the NMS technique, which was originally used in the research paper. The results are shown in Figure 6.6 (`detall` is not present anymore for computational time reasons). However, they obtained are not much better. Indeed, most problems that appeared with `bbox_vote` are still present. The only improvement is that the results for `det0` and `det1` lead to a recall that seems more reasonable. However, we see that the combination of those two variations now leads to a worse performance than when using the two algorithms independently. In addition, we can see that as the performance curves of `det0` and `det1` nearly overlap. This can probably be explained by the fact that `det0` and `det1` produce similar predictions which seems legitimate as the only difference between the two variations is that the images are flipped.



(a) Easy subset (b) Medium subset (c) Hard subset

Figure 6.6: Precision-recall curves obtained with the variations using NMS of the SFD algorithm on the three subsets of WIDER FACE.

To try to understand more deeply the reasons behind these poor results, I started looking at the predicted bounding boxes. I discovered that the algorithm outputs some bounding boxes that make no sense with out-of-bound coordinates or dimensions (with regards to the images on which they were detected). I assume this problem comes directly from the pretrained network and is therefore not solvable.

To finish with this algorithm, Table 6.8 shows the computation time per frame and FPS for the different version of the algorithm. The table only displays results when using `bbox_vote` but the results obtained with NMS are nearly the same, with a maximum difference of 0.2 FPS in favor of NMS. Moreover, we can see that once again multi-threading does not seem to be activated. Anyway, due to the inexplicable results that I obtained, I decided not to further analyze this algorithm nor to compare it to other methods.

Algorithm	CPU FPS	Real FPS
SFD-det0-bbv-0.3	10.17	10.17
SFD-det1-bbv-0.3	9.95	9.95
SFD-det01-bbv-0.3	5.04	5.04
SFD-det23-bbv-0.3	0.88	1.08
SFD-detall-bbv-0.3	0.76	0.91

Table 6.8: Medians of the number of frames, coming from WIDER FACE, that can be processed per second, in CPU and real time, by the variations using `bbox_vote` of the SFD algorithm.

SSH

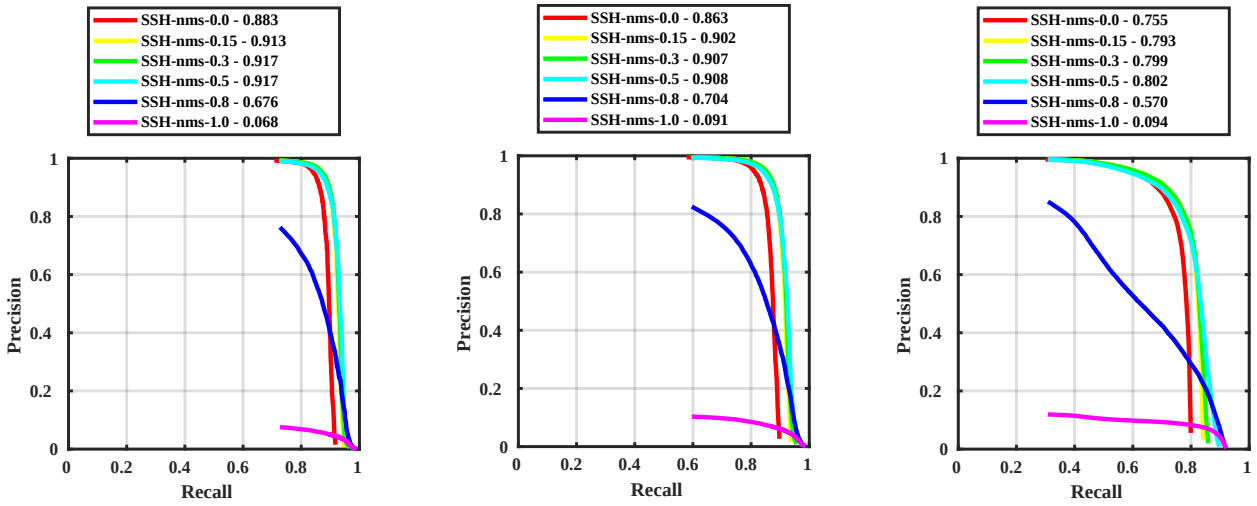
The authors of [65] do not provide a link to an official implementation of their method in their paper. However, I found an implementation of SSH on the GitHub repository[8] of the first author, Mahyar Najibi. The repository contains code for training a model and testing on the

[8]SSH repository: `https://github.com/mahyarnajibi/SSH`

WIDER FACE dataset. The pretrained network that is furnished was trained on the WIDER FACE training set. The name of the architecture of the pretrained network is not mentioned in the repository but I assume that it must be a VGG16 [79] as in the research paper.

The implementation of the testing protocol is very similar to the one of FRCNN. Indeed, in this case, it is also possible to use NMS with different threshold values and also to vary the confidence level under which we do not want to keep the detections. As for FRCNN, we will set the confidence level to 0 and test the values 0, 0.15, 0.3, 0.5, 0.8 and 1 for the NMS threshold. In addition to that, there is the possibility to rescale the input images at a fixed scale or using a scale pyramid. I decided to only explore the first option as it was considered the best trade-off between speed and accuracy in the SSH paper.

Looking at Figure 6.7, we can see that the behavior of the curves varying with the NMS threshold is very similar to the FRCNN case (i.e. when the threshold value increases, at low recall, precision decreases and at low precision, recall increases) with the best threshold value being 0.5 in this case, as inferred from the value of the AP. However, the main difference comes from the fact the performances for SSH are better than with FRCNN and decrease much less rapidly when testing onto harder subsets. Unfortunately, this is at the default of slower computation time going from 8.8 FPS to 4 FPS in CPU time and from 15.6 to 6.7 FPS in real time as shown in Table 6.9. Moreover, we can see once again that the NMS threshold value does not seem to affect the computation time and that multi-threading performs normally.



(a) Easy subset (b) Medium subset (c) Hard subset

Figure 6.7: Precision-recall curves obtained with the SSH algorithm using different values of the NMS threshold, or no NMS, on the three subsets of WIDER FACE.

Algorithm	CPU FPS	Real FPS
SSH-nms-0.0	4.07	6.68
SSH-nms-0.15	4.07	6.67
SSH-nms-0.3	4.06	6.68
SSH-nms-0.5	4.09	6.68
SSH-nms-0.8	4.06	6.67
SSH-nms-1.0	4.06	6.66

Table 6.9: Medians of the number of frames, coming from WIDER FACE, that can be processed per second, in CPU and real time, by the SSH algorithm using different values of the NMS threshold, or no NMS.

6.1.4 Overall results

Before analyzing the influence of face attributes, Figure 6.8 and Table 6.10 summarize the result of the different algorithms by taking the best version in each case.

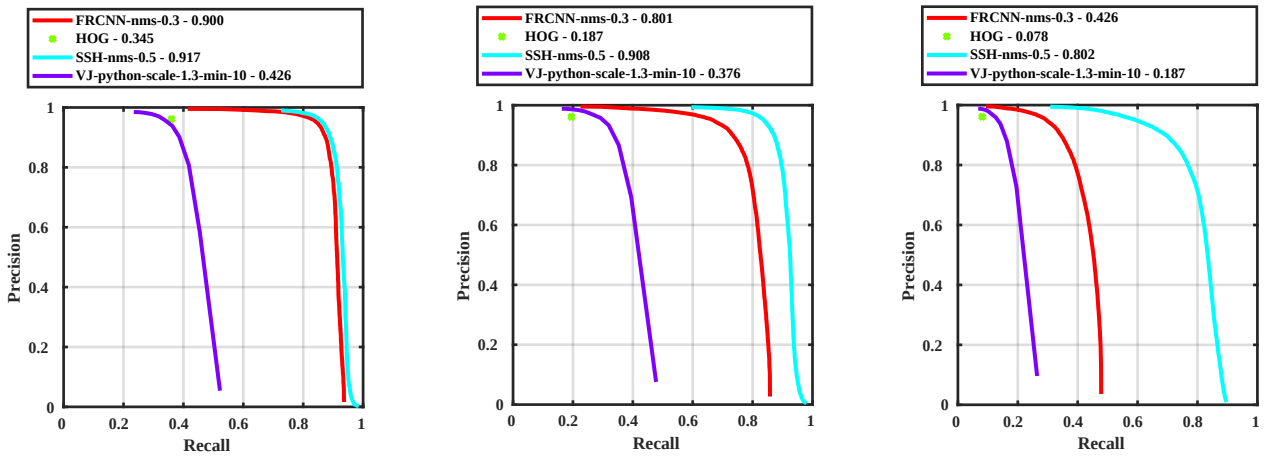
In terms of precision-recall performance, the clear winner is SSH which is the only one that manages to reach recall values superior to 50% on each of the three subsets. FRCNN produces similar performances on the easy and medium set but stalls on the hardest datasets. Concerning VJ and HOG, their performances are nearly the same but one level below FRCNN and SSH.

Concerning inference time, if we focus on real-time, we can see that the better the algorithm the smaller the number of frame per second. It is more difficult to compare HOG because we don't have access to real-time data and also because as explained before, it did not benefit from parallel processing. If we focus only on FRCNN and SSH, the former is more than two times faster than the latter. This difference is supported by fact that SSH makes a much heavier use of GPU when running. Indeed, while FRCNN only uses around 2000 MB of GPU memory, SSH uses nearly 4500 MB. This shows that SSH is a much more complex model which therefore needs greater computation time.

Finally, Figure 6.9 shows, for the 4 algorithms, the evolution of CPU time with the number of pixels per image. Even if the data is quite noisy, we can see that both for VJ and HOG, the time seems to increase linearly with the number of pixels. However, for FRCNN and SSH, the relation is far from linear. This phenomenon is actually linked to the fact that the images are reshaped before being fed to the networks. However, overall, time seems to increase with the number of pixels even if in these two cases it is more difficult to extrapolate for larger values.

Algorithm	CPU FPS	Real FPS
VJ-python-scale-1.3-min-10	2.56	35.98
HOG-C++	15.94	/
FRCNN-nms-0.3	8.84	15.69
SSH-nms-0.5	4.09	6.68

Table 6.10: Number of frames, coming from WIDER FACE, that can be processed per second, in CPU and real time, with the best versions of the 4 tested algorithms.



(a) Easy subset (b) Medium subset (c) Hard subset

Figure 6.8: Precision-recall curves obtained by the best versions of the 4 tested algorithms on the three subsets of WIDER FACE.

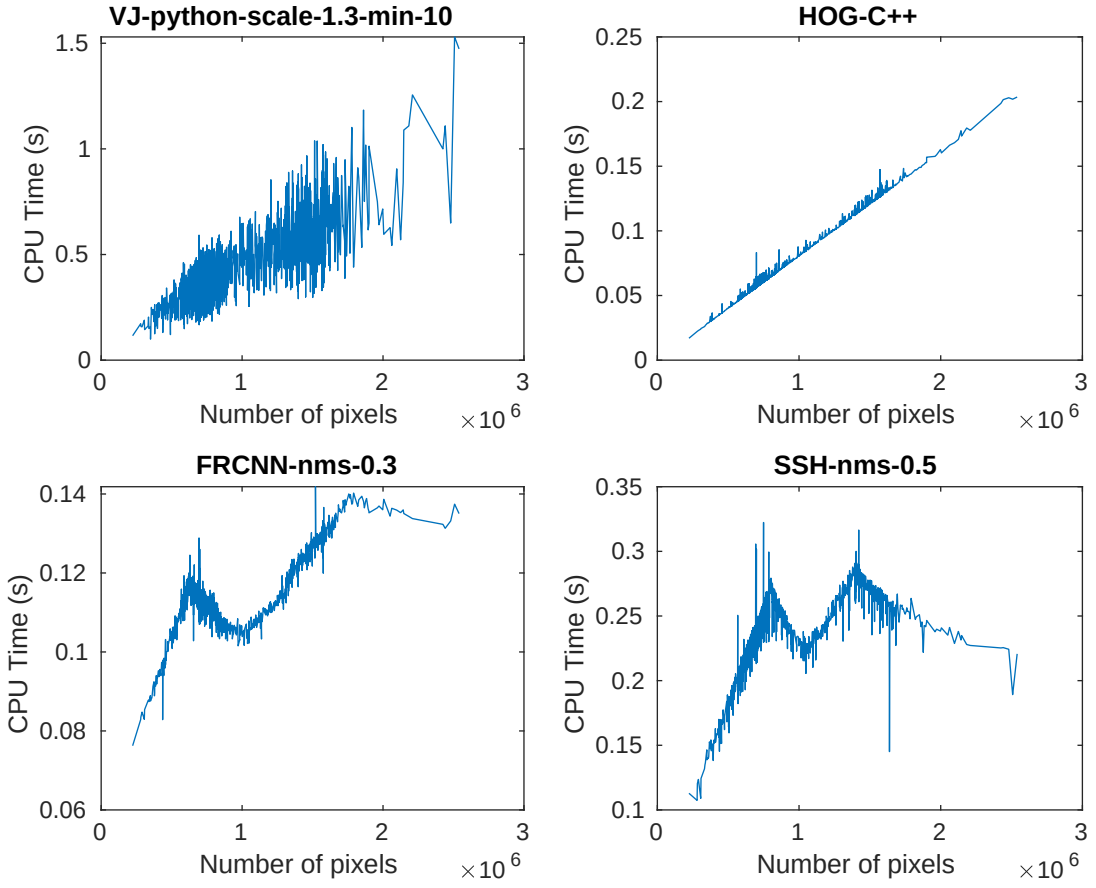


Figure 6.9: Evolution of CPU computation time per image with the number of pixels per image for the 4 tested algorithms.

6.1.5 Influence of parameters on overall results

Those first results give us a general view of the respective performances that each algorithm can reach on the WIDER FACE dataset. However, the faces in this dataset vary according to a large number of parameters which affect the performance. As stated earlier, thanks to the great labeling done by annotators, we can actually study these influences separately. As a reminder, Figure 6.10 shows the distribution of across the three subsets for 6 different face attributes. The meaning of the values of these attributes is explained earlier in sections 3.1.6 and 6.1.1. The main information to remember is that smaller values of the parameters refer to easier cases (e.g. size 0 refers to large faces while size 2 refers to small faces). In this section, I will focus on analyzing these distributions and subsequently the influences of the attributes.

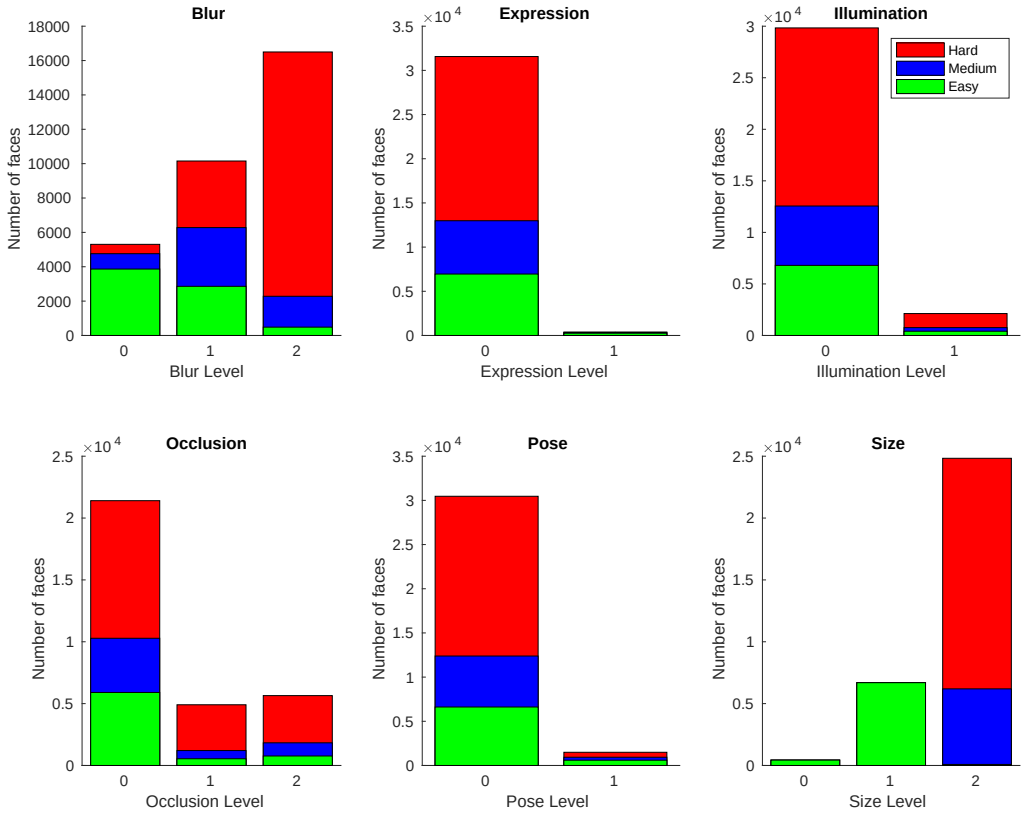


Figure 6.10: WIDER FACE dataset statistics. The figures show how many faces are annotated with each value (ranging from 0 to 1 or 0 to 2) of 6 different features. For each of these values, the color bars indicate how many faces associated with this value are contained in each of the three subsets of WIDER FACE.

First of all, for the expression, illumination and pose feature, the proportions of faces labeled with 0 is very high (99%, 94%, and 96% respectively). This means that for those features the results obtained when considering faces characterized by a value of 0 will be nearly equivalent to the global results. For each of these attributes, only the results for faces associated with the value of 1 will bring new information. For blur and occlusion, this cannot be considered as true

and it might be interesting to study each of the values of those features. We can note that for blur, the proportion of faces in the hard subset increases when increasing the blur level while the number of faces in the easy subsets shrinks. This shows that the level of blur is correlated with the difficulty level of the subsets, which was not the case with the other previously mentioned features. When dividing faces according to their size, we see that the medium and hard subsets actually do not contain any additional faces of size level 0 or 1 (i.e. any additional faces with height superior to 50 pixels). There seems to be an even greater correlation for this attribute than with blur.

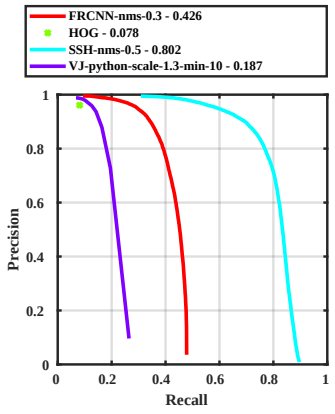
Studying the effect of some of those attributes seems more useful for the ones we have the most control over. For instance, studying the influence of expression or occlusion is not very useful in the case of the RAGI project because the only people that can control these parameters are the users of the application. Pose is also one of those parameters that are a priori controlled by the user. However, one could consider that we only want to recognize people facing directly the camera and in that sense, its influence is interesting to study. On the opposite, blur or size are features that can be dealt with by choosing appropriate camera models or parameters. Even though illumination can vary quite a lot even for a camera placed at a fixed position, it can be useful to know its influence to be able to choose appropriate places to position those cameras. All this considered, I still decided to study the effects of all these features, even briefly, for the sake of completeness. Finally, as the hard subset is a superset of the easy and medium subsets and to be more concise, I will only display results on this subset.

Blur

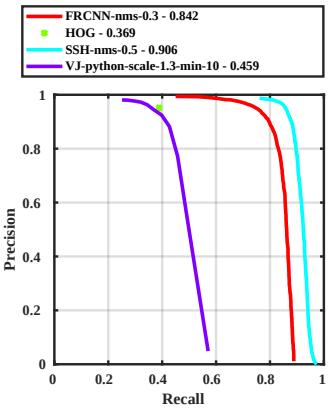
Figure 6.11 shows the precision-recall curve in the general case (a) and for the 3 levels of blur (b, c, d). As expected, the performances degrade with the blur level. We can see that for level 0, FRCNN reaches nearly the same performance as SSH while HOG outperforms VJ. This behavior reverses immediately when we consider faces with light blur with a large drop in performances for all the algorithms except for SSH which maintains a high level of performance. This behavior worsens when we increase the level of blur. In that case, we see that HOG, VJ and even FRCNN lead to performances that make them unpractical. The performance of SSH drops significantly too but it is still better than the ones of VJ and HOG when no blur is considered.

Expression

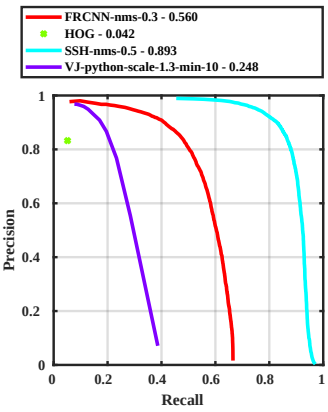
We can see in Figure 6.12 the effects of exaggerated expression which defines the level 1 of the expression feature. The rather unsmoothed look of the curves come from the fact that the number of faces on which the test is made is small. We can see that the effects are not too damaging to the performances even improving the performances for FRCNN. Even if the test set is small and therefore the results not especially reliable, we can infer that exaggerated expression should not affect the performances significantly which is a good news as this is a parameter that we cannot control.



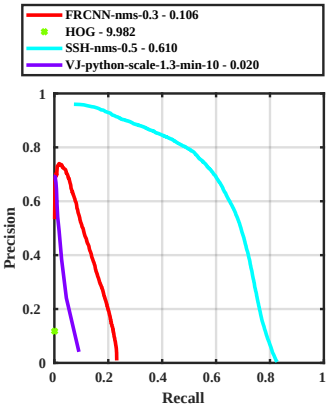
(a) General



(b) Level 0

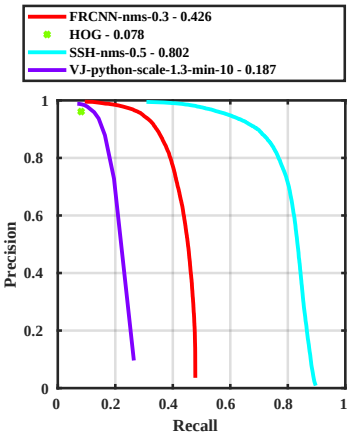


(c) Level 1

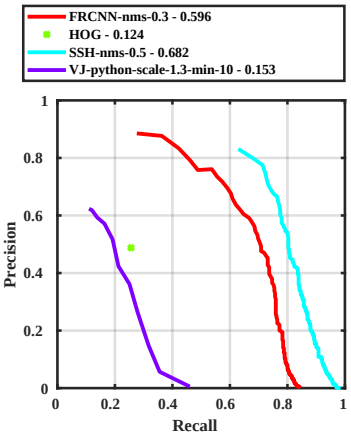


(d) Level 2

Figure 6.11: Blur analysis. Precision-recall curves obtained by the best versions of the 4 tested detection algorithms on whole hard subset of WIDER FACE (a) and when considering only faces annotated with blur level 0 (b), level 1 (c) and level 2 (d) among that subset.



(a) General

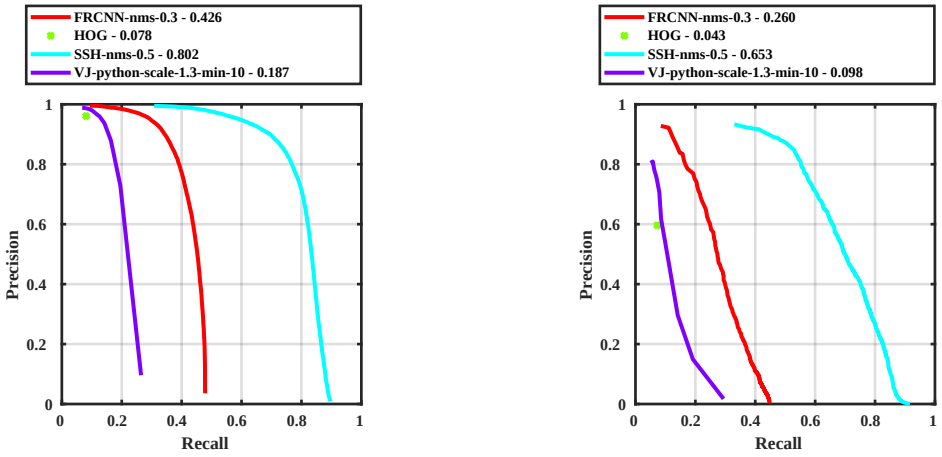


(b) Level 1

Figure 6.12: Expression analysis. Precision-recall curves obtained by the best versions of the 4 tested face detection algorithms on the whole hard subset of WIDER FACE (a) and when considering only faces annotated with expression level 1 among that subset (b).

Illumination

In the case of illumination, extreme illumination (even though not defined more explicitly) seems to have a negative effect on performance. As shown in Figure 6.13, the performances in terms of average precision of all the algorithms are nearly divided by two except for SSH which resists better. It is a shame not to have more information about the definition of extreme illumination which could have helped to avoid such situations.



(a) General (b) Level 1

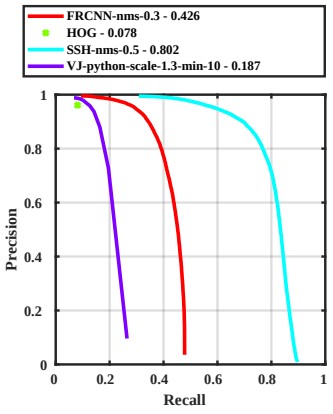
Figure 6.13: Illumination analysis. Precision-recall curves obtained by the best versions of the 4 tested face detection algorithms on the whole hard subset of WIDER FACE (a) and when considering only faces annotated with illumination level 1 among that subset (b).

Occlusion

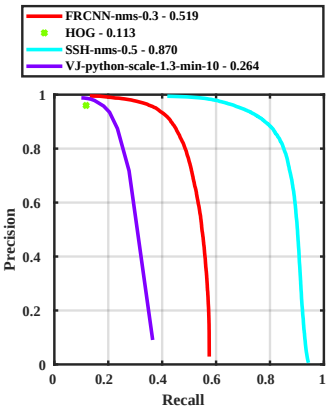
For occlusion, most faces are gathered in level 0 which explains why in Figure 6.14, results in (a) and (b) look so similar but with a notable improvement in (b). However, when considering even a small level of occlusion, the performances become much worse. Even for SSH, we can see that for a precision of 90% we painstakingly reach a recall of 20%. When considering more occlusion, there is again a small decrease compared with level 1. As a reminder, level 1 means that 1-30% of the face area is occluded. With this in mind, we can infer that the algorithm that will be practically used will have a much rougher time detecting people when they wear sunglasses, hats or scarfs and unfortunately that is something we cannot control. We can, however, minimize this problem as the system will be used inside where people tend to wear those type of accessories less often.

Pose

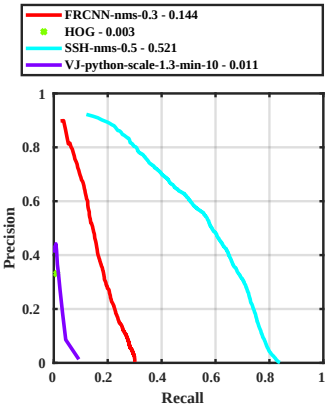
We can see in figure 6.15 that considering faces of pose level 1, i.e. whose roll or pitch degree is larger than 30°; or whose yaw is larger than 90°, affects the performances in a worse manner than heavy occlusion. VJ and HOG detect nearly no faces. In particular, we can see that VJ does not make any detection anymore. This is a well-known default of VJ which is aimed at detecting frontal faces. FRCNN and SSH still manage to find some faces but the overall precision has dropped.



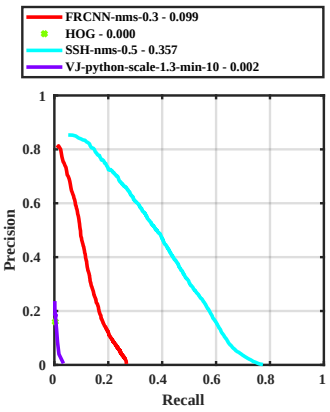
(a) General



(b) Level 0

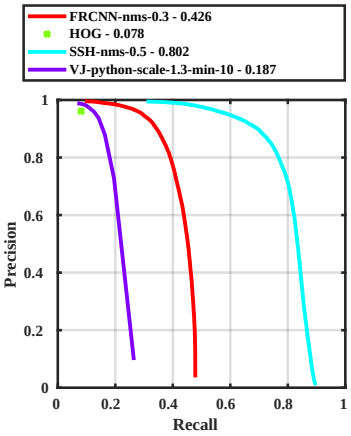


(c) Level 1

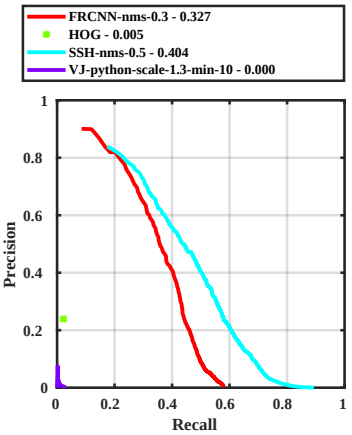


(d) Level 2

Figure 6.14: Occlusion analysis. Precision-recall curves obtained by the best versions of the 4 tested face detection algorithms on the whole hard subset of WIDER FACE (a) and when considering only faces annotated with occlusion level 0 (b), level 1 (c) and level 2 (d) among the subset.



(a) General



(b) Level 1

Figure 6.15: Pose analysis. Precision-recall curves obtained by the best versions of the 4 tested face detection algorithms on the whole hard subset of WIDER FACE (a) and when considering only faces annotated with pose level 1 among that subset (b).

Size

Size is one of the most interesting features to study because it is one of the most important to make a choice in terms of the camera that should be used in the system. Indeed, the greater the resolution, the larger the faces in terms of pixels. As shown in Figure 6.16, when considering faces larger than 300 pixels in height and more surprisingly faces of more than 50 pixels, FRCNN performs nearly as well as SSH. A larger difference appears for the smallest faces which seems logical as SSH is based on Faster R-CNN while trying to be better at detecting small faces. HOG, for faces of size level 1, reaches a recall level of nearly 80% at the price of a slight decrease in precision compared to the general case. However, this good performance does not resist the shrinking of faces. For level 2, it reaches logically an AP of 0 as it cannot detect faces under 80 pixels in height. VJ benefits also from the fact that we only consider large faces but does not reach the same level of performances at the other algorithms. However, compared to HOG it still performs reasonably well on very small faces.

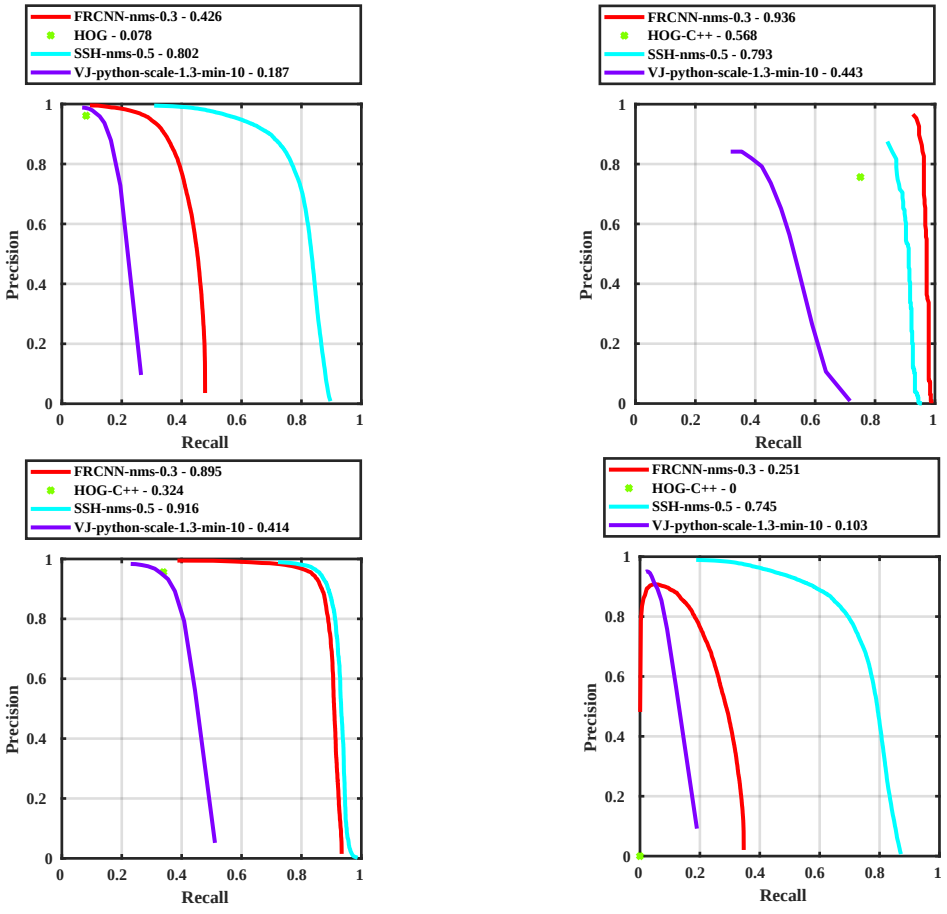


Figure 6.16: Size analysis. Precision-recall curves obtained by the best versions of the 4 tested face detection algorithms on the whole hard subset of WIDER FACE (a) and when considering only faces annotated with size level 0 (b), level 1 (c) and level 2 (d) among that subset.

6.2 Face recognition

6.2.1 Evaluation protocol

As explained in section 3.2.9, I will do the test of face recognition algorithms on the MegaFace benchmark. I am going to describe in more details how these tests were made using the tools provided with this benchmark.

The MegaFace evaluation code allows testing identification of faces from images of the two probe sets, FaceScrub and FGNet, against a gallery containing a varying number of distractors (from 10 to 1000000). For a given probe set and a given number of distractors images, it outputs a CMC curve. As specified earlier, as we are not interested in face recognition across age variation, I will only focus on the FaceScrub dataset. It is important to specify that, for "efficiency" [47], the evaluation protocol uses only a subset of FaceScrub, including 80 identities (40 females and 40 males) with 50 images each. In addition, I decided to limit the test to 10K distractors. Indeed, above that the computing time started to become prohibitive and in the context of RAGI, 100k or 1M distractors did not seem relevant yet.

Finally, I need to add that for both MegaFace (MF) and FaceScrub (FS), bounding boxes information is provided and allows to easily crop each image to keep only the face. This cropped face is then passed through some pre-processing (such as alignment) and then feature extraction.

6.2.2 Time evaluation

There are two main stages in face recognition that need to be analyzed separately in terms of time: feature generation (including all preprocessing steps) and feature classification (obtained via nearest-neighbors classification). It is important to compute the two computations time separately because these two tasks can easily be parallelized. Moreover, the efficiency of those two phases is equivalently important to estimate. Indeed, while we might assume that feature generation is the most variable in terms of computation time, the length of the generated features will affect linearly the computation time of the second phase, which could become substantial when the size of the gallery explodes.

For the first phase, the time was measured for each image of the MegaFace dataset. I did not compute it over the images of FaceScrub because it would have been redundant. It is important to notice here that the number of images is equal to the number of faces. However, if we consider the full face recognition pipeline, each image will possibly contain several faces. To avoid confusion with the FPS used a time metric for face detection algorithms, I will use for face recognition the term FaPS, referring to the number of faces processed per seconds. This per-face time will take into account the alignment and feature extraction step but not the cropping because it does not depend on the chosen algorithm.

For the second phase, it was not possible to obtain computation time per image because the testing code was only available in the form of executable working on the whole probe and gallery set used for the test and that could not be modified. The only possible differentiation was to compute time for different gallery sizes. However, as the testing code was written in Python, I had access to both real and CPU time. Moreover, as this time depends only on the size of the extracted feature vector, they will be the same for each of the tested variations of

the 3 algorithms. I will therefore only analyze it in the general results section.

As for face detection, I measured CPU and real time (when it was possible) plus CPU and GPU usage using the same functions and command.

6.2.3 Per-algorithm results

OpenFace

The OpenFace library provides Python code for training and testing, with pretrained networks, explained at `http://cmusatyalab.github.io/openface/models-and-accuracies/` and provided at `https://github.com/cmusatyalab/openface/`. Among the pretrained networks, the best performing on LFW is 'nn4.small2.v1' which is based on a simplified version of Inception.

OpenFace implementation does not provide any variation of use except for switching off or on alignment. In this case, alignment is performed using OpenFace's `ALignDlib` class presented on this page `http://openface-api.readthedocs.io/en/latest/openface.html`. As its name indicates, this class is based on Dlib's landmark estimation algorithm which is inspired by [46] as referenced on this page: `http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html`. The constructor of this class takes in a pretrained landmark predictor model, which is, in this case, the `shape_predictor_68_face_landmarks.dat` model provided on the following repository: `https://github.com/davisking/dlib-models`

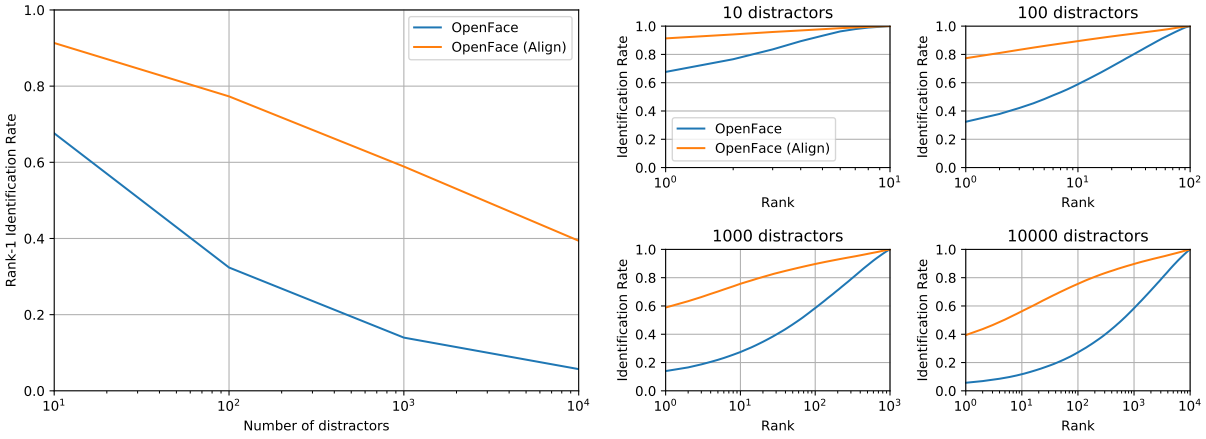
Therefore, I will test 2 variations of the algorithm:

- OpenFace : OpenFace without alignment
- OpenFace (Align) : OpenFace with alignment

Figure 6.17 shows in (a) the rank-1 performance for the different distractors numbers while in (b), it shows how the identification performance evolves with the rank for these 4 different distractors numbers. As expected, the performances decrease with the number of distractors whilst it increases when the rank increases. Alignment seems to improve the performances drastically especially when we consider larger distractors numbers. Indeed, we can see that with 10 distractors the performance of OpenFace without alignment is acceptable with an identification rate at rank-1 around 70%. However, this value drops very rapidly to reach less than 10% for 10,000 distractors. Even for 1,000 distractors, which would seem like a reasonable gallery size in a RAGI project, we only get around 20%. For the three largest distractors numbers, the alignment doubles the identification rate reaching around 60% for 1,000 distractors and still 40% for 10,000 distractors. If we have a look at the second graph, we actually see that for both variations, the identification rate rises quite slowly with the rank.

In terms of computation time, Table 6.11 shows that alignment makes the number of FaPS decrease quite drastically to a mere 25 faces-per-second in real time. The worse identification rate obtained without alignment is compensated by a 4 to 5-fold increase in computation time. This increase in computation time is even bigger when considering CPU time. This certainly means that alignment is realized using CPU rather than a GPU implementation. There is no clear explanation about this in the OpenFace documentation. However, when looking at the GPU activity with alignment or not, there is no difference which seems to confirm this theory.

As the performance of OpenFace without alignment are too bad for a practical application, I will keep the OpenFace (Align) version for further analysis, even though it is a lot slower.



(a) Rank-1 identification rate vs number of distractors.

(b) CMC curves for varying number of distractors.

Figure 6.17: Performances obtained by the OpenFace algorithm, with and without alignment, on MegaFace. (a) shows the evolution of rank-1 identification rate with the number of distractors in the gallery while (b) shows the CMC curves for these different number of distractors.

Algorithm	CPU FaPS	Real FaPS
OpenFace	230.47	110.81
OpenFace (Align)	29.89	24.66

Table 6.11: Medians of the number of faces, coming from MegaFace, that can be processed per second, in CPU and real time, with the OpenFace algorithm, with and without alignment.

Dlib-R

The Dlib library contains a pretrained model based on a simplified version of ResNet-34 and C++ testing code for Dlib-R in the form of an example that allows performin features extraction on a single image containing multiple faces. Alignment can also be performed using Dlib face alignment algorithm, evoked with OpenFace, but with the `shape_predictor_5_face_landmarks.dat` landmark prediction model this time. Then the face recognition accuracy can apparently be improved if "jittering is used when creating face descriptors". The function allowing to add jittering to the process actually just makes 100 copies of the aligned image, "all slightly jittered by being zoomed, rotated and translated a little bit differently. They are also randomly mirrored left to right". Then face recognition is applied to each of these images. This process, unfortunately, slows down the extraction process.

As we have the option to apply alignment and alignment, this leads to 4 different test possibilities:

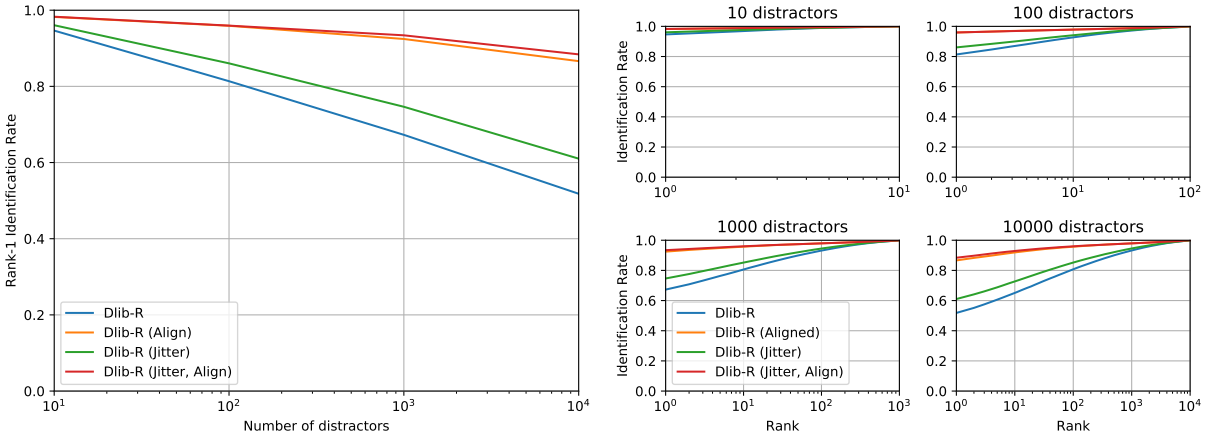
- Dlib-R : Dlib-R without jitter or alignment
- Dlib-R (Align) : Dlib-R with alignment and without jitter

- Dlib-R (Jitter) : Dlib-R without alignment and with jitter
- Dlib-R (Jitter, Align) : Dlib-R with alignment and jitter

Figure 6.18 shows that for a small number of distractors the differences in performance of the 4 variations are very small and that they increase with this number. If we compare the benefits of alignment and jitter, we can see that alignment seems to provide greater improvement. Indeed, when looking at the largest distractors number, we can see that compared to the basic implementation alignment provides 35% improvement while jitter only provides a bit less than 10% improvement. Moreover combining the two, we see that we only gain a mere 1% in identification rate compared to when we only use alignment.

The prevalence of alignment is further motivated by the computation times. Indeed, we can see in Table 6.12 that both the alignment and jitter decrease the number of faces that can be processed per second but that jitter has a much rougher impact. On the opposite, compared to OpenFace, alignment has much less effect on the baseline computation time decreasing it only by a factor 2. This might be caused by the fact that a 5 landmark model is used rather than a 68 landmark or this might come from the fact that Dlib-R is coded in C++ while OpenFace is written in Python. I did not have the time to investigate that more in depth.

According to these results, I will continue further analysis with the Dlib-R (Align) version.



(a) Rank-1 identification rate vs number of distractors

(b) CMC curves for varying number of distractors

Figure 6.18: Performances obtained by the Dlib-R algorithm, with and without alignment and with and without jitter, on MegaFace. (a) shows the evolution of rank-1 identification rate with the number of distractors in the gallery while (b) shows the CMC curves for these different number of distractors.

Algorithm	CPU FaPS
Dlib-R	457.46
Dlib-R (Align)	246.67
Dlib-R (Jitter)	14.14
Dlib-R (Align, Jitter)	13.73

Table 6.12: Medians of the number of faces, coming from MegaFace, that can be processed per second, in CPU and real time, with the Dlib-R algorithm, with and without alignment and with and without jitter.

ArcFace

The implementation of ArcFace was released with a project called 'InsightFace' which is referred as a "2D and 3D Face Analysis Project" and which is part of the GitHub repository 'Deep Insight' (`https://github.com/deepinsight`) conducting a lot of projects around deep learning. More precisely, in this project, they "provide training data, network settings and loss designs for deep face recognition". The training data is composed of the 'normalised' MS-Celeb-1M dataset [33] and VGG2 dataset [12]. The underlying network include several architectures including ResNet implemented using the MXNet library and a series of loss functions are tested including the triplet loss and 'ArcFace'.

InsightFace furnishes training and testing code as well as pretrained network. More precisely, LResNet50E-IR and LResNet34E-IR are provided. However, the second one is only available via Baidu Drive and one needs an account (which I do not have) to obtain it. The testing script allows generating features for FaceScrub and MegaFace by applying alignment before-hand. Therefore, I decided to test ArcFace with and without alignment. The main difference being that the exact alignment process used by ArcFace is a bit more difficult to determine than with OpenFace or Dlib-R. Indeed, the ArcFace library provides a series of functions for aligning the different datasets on which it can be tested. One of the implementations was using the same alignment technique as OpenFace and seem to be the simplest one to use, so I went on with that choice. I named the two tested variations as follows:

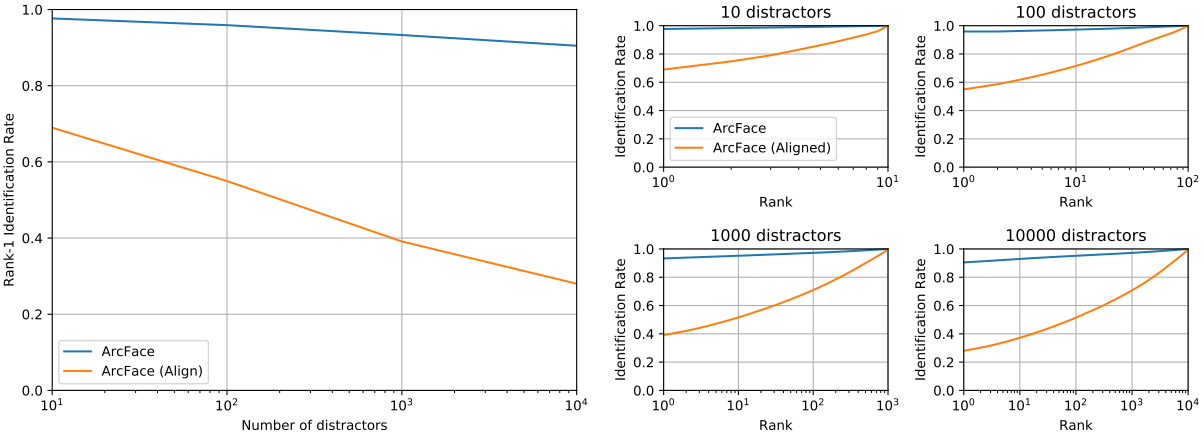
- ArcFace : ArcFace without alignment
- ArcFace (Align) : ArcFace with alignment

By looking at Figure 6.19, we can see that unexpectedly the use of alignment has a negative effect on the identification rate. This must be linked to the fact the alignment technique used in this case is not appropriate for feature extraction step that follows, making the testing input distribution too different from the training input distribution. On the opposite, without using any alignment technique, we obtain surprisingly good results even for the largest distractors set size.

Algorithm	CPU FaPS	Real FaPS
ArcFace	50.98	60.35
ArcFace (Align)	23.14	24.53

Table 6.13: Medians of the number of faces, coming from MegaFace, that can be processed per second, in CPU and real time, with the ArcFace algorithm, with and without alignment.

In terms of computation time, with no surprise, alignment has a negative effect. This combined with the previously mentioned fact makes the case to keep the version without alignment. Of course, it might be interesting, in future work, to explore other alignment techniques to see if they improve the performances.



(a) Rank-1 identification rate vs number of distractors

(b) CMC curves for varying number of distractors

Figure 6.19: Performances obtained by the ArcFace algorithm, with and without alignment, on MegaFace. (a) shows the evolution of rank-1 identification rate with the number of distractors in the gallery while (b) shows the CMC curves for these different number of distractors.

6.2.4 Overall results

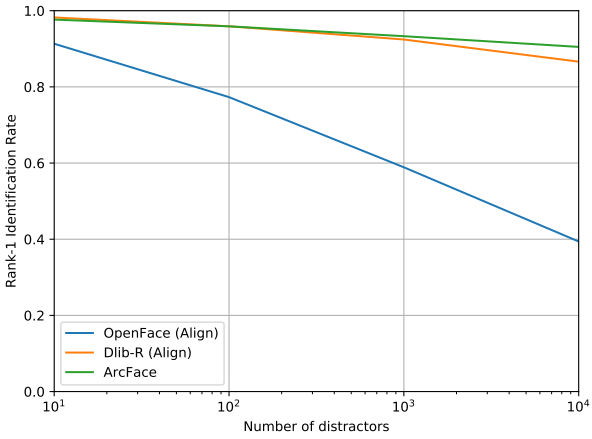
Figure 6.20 shows similar graphs as in the three previous sections but comparing now the best versions of each face recognition algorithm. The main remark that can be made is that Dlib-R and ArcFace are one level above OpenFace. The two former algorithm have a similar level of performance with Dlib-R being slightly better for the smallest number of distractors and ArcFace taking the lead when this number increases.

However, if we look in terms of time, Table 6.14 shows that for CPU time, Dlib-R performs much better. This might be explained by the fact that it is written in C++ rather than in Python and that it is the algorithm that uses the least GPU memory.

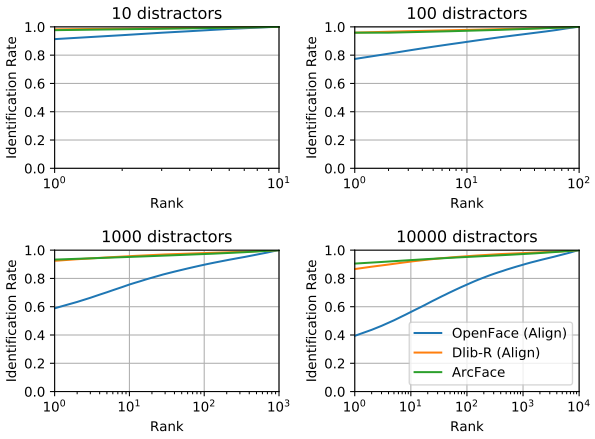
Algorithm	CPU FaPS	Real FaPS
OpenFace (Align)	29.88	24.66
Dlib-R (Align)	246.67	/
ArcFace	50.98	60.35

Table 6.14: Medians of the number of faces, coming from MegaFace, that can be processed per second, in CPU and real time, with the best versions of the 3 tested face recognition algorithms.

I will now briefly analyze the computation time for the classification step. As this time only depends on the size of the feature representation, I will refer to the algorithm by their name without characteristics. Those sizes are displayed in Table 6.15. We can see that Dlib-R and OpenFace generate the same number of features per face while InsightFace generates four times more.



(a) Rank-1 identification rate vs number of distractors



(b) CMC curves for varying number of distractors

Figure 6.20: Performances obtained by the best versions of the 3 tested face recognition algorithms on MegaFace. (a) shows the evolution of rank-1 identification rate with the number of distractors in the gallery while (b) shows the CMC curves for these different number of distractors.

Figure 6.21 shows the evolution of classification time with the number of distractors used. We can see that as expected the time is greater for ArcFace. We can also see that there seems to be a small difference between Dlib-R and OpenFace which is unexpected but seems to be mainly noise. To have a better measure, I should maybe have repeated the experiment and taken the meantime over these different runs. If we analyze the evolution of time with the number of distractors, it seems like we have a linear relation which fits with the complexity of a generic nearest neighbor algorithm. Finally, from this graph, we can also approximately infer the time needed to make a prediction for one image. We know that, in total, the identities of 4,000 probes are predicted. Therefore, for the largest gallery size, one ArcFace prediction would take $3.2/4000 = 0.0008s = 0.8ms$ meaning that $1/0.0008 = 1,250$ identities can be predicted by second with the given testing implementation. If we manage to use a testing implementation that is as efficient as this one, the classification time can be regarded as negligible.

Algorithm	OpenFace	Dlib-R	ArcFace
Feature Vector Size	128	128	512

Table 6.15: Size of the feature vectors generated by the 3 tested face recognition algorithms.

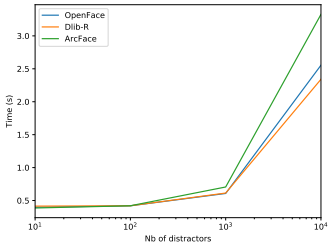


Figure 6.21: Evolution of classification time (expressed in seconds) with the number of distractors for the 3 tested face recognition algorithms.

Chapter 7

Application to RAGI

As shown in the previous section, the face detection and, a priori, face recognition performances vary depending on a number of factors such as pose, exposition or size of the face. Each dataset being defined by different distributions across those characteristics, it is interesting to exactly know what performances we could obtain in the context of the RAGI project. It was therefore decided to build a dataset for the project and to test the best versions of each algorithm on it. To achieve this goal, a series of images were collected and annotated as described in the following sections. Face detection and recognition were then tested using similar protocols than in the previous sections. To finish the analysis, a test on the full face recognition pipeline is realized by combining face detection and face recognition algorithms.

7.1 Dataset construction

The dataset construction consisted of three main steps: data acquisition, annotation for face detection and annotation for face recognition.

7.1.1 Data acquisition

The data acquisition was undertaken at the University of Liège, in the B28 building, also know as Montefiore, on Friday 20-04-2018. The acquisition procedure was the following. RAGI main station, equipped with a camera on its top was brought in three different locations of the building. The camera was a MAKO G-419C of Allied Vision[1] equipped with a wide-angle lens of the model KOWA LM6HC. The white balance and gain were automatically set by the camera while the exposure time was set to 16,666 nanoseconds. With these parameters, the camera takes 2048x2048 images at approximately 25 FPS. At each of this location, a continuous series of frames was taken for about 15 minutes.

The first location where images were taken was the main entrance hall. In this hall, two configurations were tested, the first one with the screen facing the outside and the second one with the screen positioned at approximately a 90° angle to the door. Then, the screen was positioned near the R21 class located next to a secondary entrance and in front of a stare case. Finally, a sequence was registered at the R100 classroom which is located near the catwalk connecting the B28 to the B37. In this configuration, the screen was facing the entrance of an elevator. For each of these scenarios, a dozen of volunteers was asked to move freely in

[1]Allied Vision website: `https://www.alliedvision.com/en/digital-industrial-camera-solutions.html`

front of the screen of the station as they would normally when walking around the university. In addition, volunteers were asked to come at least once close to the screen as if they were interacting with it. Finally, it is important to notice that, as the recordings were done on a school day, a series of other people were also captured during the acquisition.

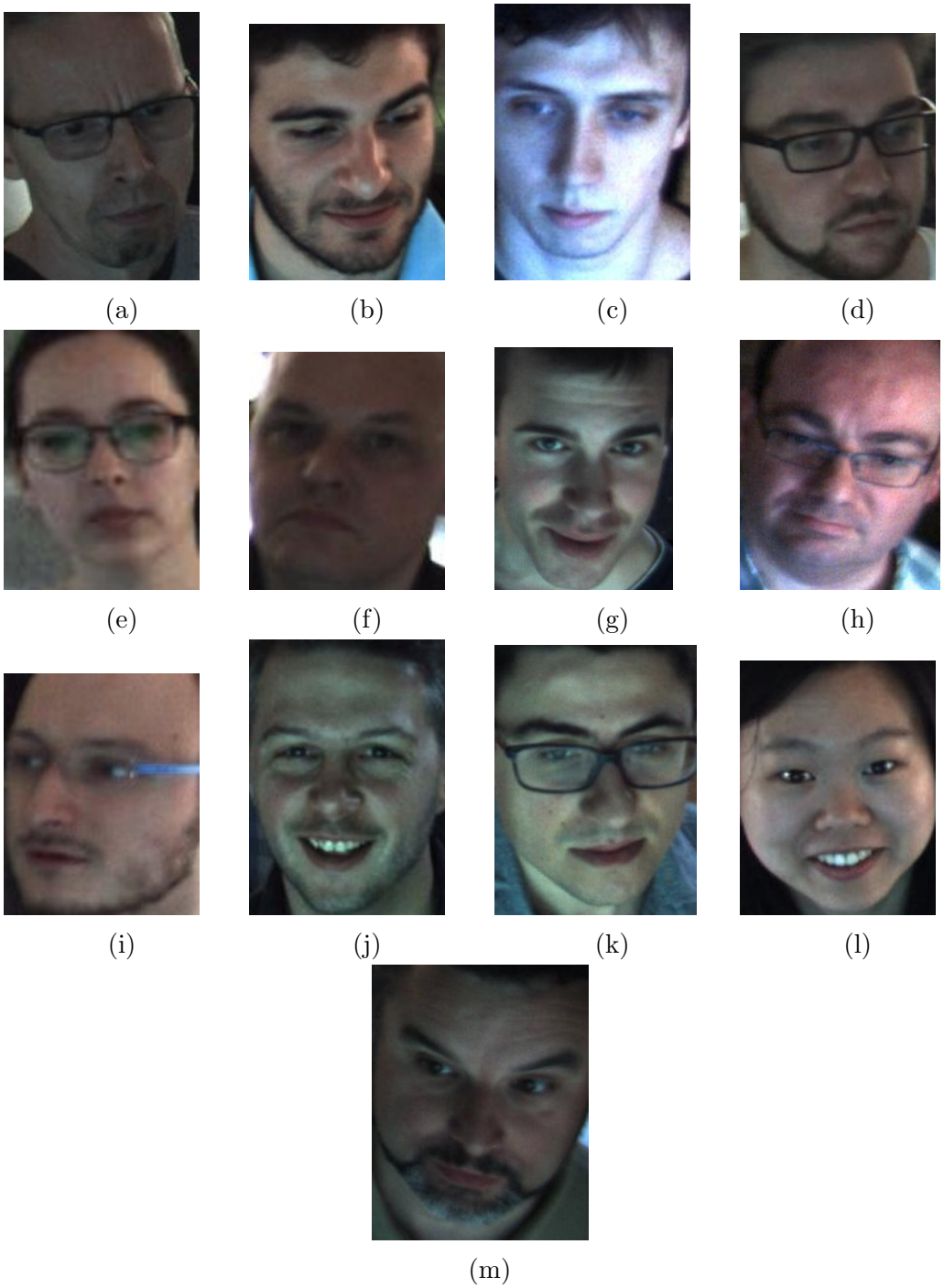


Figure 7.1: The team of volunteers with images taken from the RAGI dataset: a) Bourguignon D., b) Di Bartelomeo M., c) Dubois A., d) Ewbank T., e) Ferire L., f) Frederic M., g) Gerard D., h) Hiard S., i) Mattheus B., j) Van Lishout F., k) Vecoven N., l) Wang L., m) Suplis O. The volunteers are classified by alphabetical order of their family name except for mister Suplis whose name I did not know when the dataset was annotated.

Figure 7.1 presents the team of volunteers with frames coming from the RAGI dataset. It

is important to notice that those photos were taken among the best quality pictures from each person and therefore do not reflect the true distribution of the dataset.

As shown in Table 7.1, the acquisition procedure resulted in a total of 35004 frames divided between the different scenarios.

	Hall 1	Hall 2	R21	R100	Total
Number of frames	12,087	5,896	8,782	8,239	35,004

Table 7.1: Number of frames taken in each scenario composing the RAGI dataset.

As all these frames needed to be annotated manually, I decided to restrict the dataset to a sample of around 500 frames. This was done by selecting 1 frame every 35004/500 rounded up (i.e 71)[2] frames which means that the time between each frame, in a same scenario, is of approximately 71/25 = 2.84 s. By scanning through the 494 selected frames, we can see that the number of faces per images varies from 1 to more than 10 and that there is a large variation in terms of size, pose, illumination, level of blur between the different faces.

7.1.2 Face detection annotation

The next step was to annotate the dataset for face detection. Following the convention of most datasets (except for FDDB and AFLW), I decided to use rectangular bounding boxes as annotations. Then, I had to decide which faces should be annotated. Indeed, it is important to keep in mind that those faces will then be used in a face recognition phase in order to identify the person it belongs to. Therefore, faces that do not contain enough facial characteristics (like eyes, nose, etc.) should not be taken into account. The decision was to focus only on faces for which at least one eye was visible. No restriction was put on the level of blur or any other characteristics. A problem with this approach is that some algorithm could detect faces that we do not consider as recognizable and our evaluation protocol would then consider it as a false positive. To counter this, all heads (with a recognizable face or not) were marked with a bounding box. An 'ignore' flag was then set to 0 for all recognizable faces and 1 for all other bounding boxes.

Finally, one has to decide exactly how to draw the bounding boxes around the faces. Indeed, the larger the bounding box is, relatively to the face, the more characteristics of the face are captured but also the more noise is gathered from the background for subsequent analysis. The decision was to follow the WIDER FACE protocol by drawing the tightest rectangle containing the forehead, the cheeks, and the chin. For unrecognizable faces, a similar approach was taken except when the head was completely turned around. In that case, I tried to approximate this criterion as well as possible. However, it seems unlikely for those heads to be detected by face detectors and therefore the precision of the bounding box is less of a problem.

Considering that there are 494 frames in total and that they contain 5-10 faces/heads on average according to visual estimation, this leads to an approximate total of 2500-5000 bounding boxes to annotate. To speed up this work, the annotation process was divided between 5

[2]I realized later that it would have been more sensible to round this number down so that the dataset could be exactly 500 images instead of only 494. However, the dataset was already annotated by the time I noticed this mistake.

annotators. However, to facilitate the work of the annotators a semi-automatic process was used. The SSH face detector was first run over all the frames to generate bounding boxes with confidence scores. The bounding boxes were then shown to annotators that had the choice between confirming that the bounding box was correct, modifying it to match the above-cited criteria or to reject it. Moreover, for each frame, the annotators had the opportunity to annotate all non-annotated faces with new bounding boxes. This process can be criticized by putting forward that it will necessarily give a positive bias to the SSH detector. I analyze and discuss this problem in section 7.2.

This process led to the annotation of 4638 bounding box from which 3561 are considered as recognizable. More statistics are given in section 7.1.4.

7.1.3 Face recognition annotation

The final step consisted in associating each recognizable face with a name. This was also done semi-automatically by using the OpenFace recognizer and in this case, this cannot cause a bias for this algorithm. The idea of the procedure was to first generate features for all the faces using OpenFace. Then, for each face, its feature representation was compared against all the feature representations of faces that were already annotated to find the closest one and proposing to annotate the new face with the corresponding identity. The annotator had then the choice to confirm this choice or to enter the true identity if the prediction was wrong. All faces that belonged to people that were not volunteers were gathered under the same identity called 'Unknown'.

Even though the chosen detector was not very good with its prediction, even for the last annotations and even when considering that each identity can only be predicted once per image (see following sections), the annotation process was quite rapid. On my own, it took me approximately 5 hours to annotate all the faces. The number of annotated faces per person and other similar statistics are explained in the following section.

7.1.4 Statistics

This section contains a series of statistics about the RAGI dataset.

General

Table 7.2 shows the number of frames and faces for each scenario plus the mean and standard deviation of the number of faces. We can see that these last two values do not vary much across the different scenario with a mean ranging from 5.39 to 8.63 people per picture. Then Figure 7.2 contains the number of faces taken for each person. We see that this number varies quite a lot with a standard deviation of around 95 faces for a mean of approximately 255 faces. We can see that this number is dragged down by people like Bourguignon D. and Suplis 0. which were not present during the whole collection process. This large variety is not a problem as in a real application of RAGI, one could probably imagine that some people are in contact with the system less frequently.

	Hall 1	Hall 2	R21	R100	Total
Number of frames	171	83	123	117	494
Number of faces	1475	522	937	627	3561
Mean nb of faces per image	8.63	6.31	7.60	5.39	7.21
Std of nb of faces per image	1.81	2.29	2.42	1.78	2.42

Table 7.2: This table contains, for each scenario, the number of selected frames, the number of recognizable faces that were annotated and the means and standard deviations of the number of recognizable faces annotated per image.

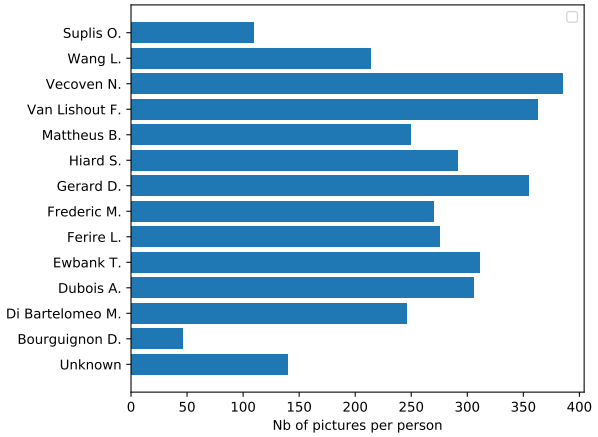


Figure 7.2: Number of face picture for each volunteer and unknown person in the RAGI dataset.

Width and height

Table 7.3 and 7.4 and Figure 7.3 show some statistics about width and height for the whole dataset of faces. We can see that the range of sizes of faces is quite large even though most of the faces are rather small compared to the largest face as suggested in Figure 7.3. We can also notice that mean height is greater than mean width which seems logical.

	Mean	Range
Width	74.31	[14;382]
Height	93.96	[22;477]

Table 7.3: Mean and range of the width and height of recognizable faces in the RAGI dataset.

Deciles	1	2	3	4	5	6	7	8	9
Width	32	39	44	48	56	65	81	104	150
Height	39	49	54	62	72	83	104	132	192

Table 7.4: Deciles of the width and height of recognizable faces contained in the RAGI dataset.

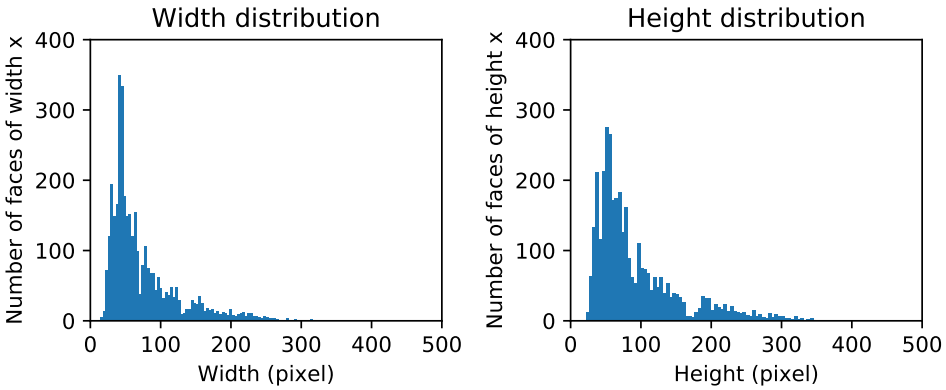


Figure 7.3: Unnormalized distribution of the width and height of recognizable faces in the RAGI dataset.

Figure 7.4 is displayed to emphasize this fact and to show that bounding box annotations were consistent throughout the annotation process. Indeed, we can see that there is a clear correlation between width and height, with the height being generally greater than the width. It is interesting to notice that there are outliers where the width is larger than the height. There are 60 such cases and are mainly linked to faces being tilted towards the ground as shown in Figure 7.5 or very small faces. Another important characteristic that we can derive from this graph is that for further analysis and for testing, it is reasonable to focus only on height or width as those are closely correlated (i.e. correlation coefficient of 0.98). Following the principle used in WIDER FACE, I decided to focus on height.

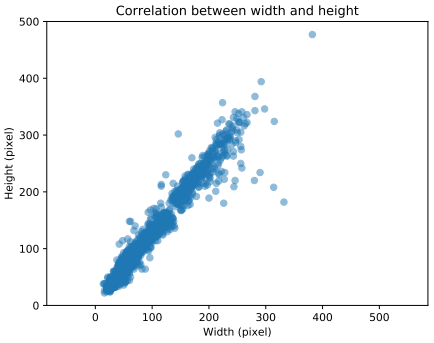
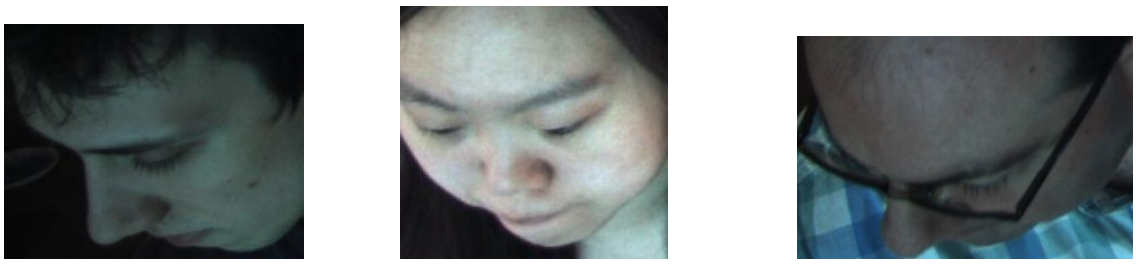


Figure 7.4: Scatter plot of the width and height of recognizable faces in the RAGI dataset.



(a) (b) (c)

Figure 7.5: Examples of faces annotated with an horizontal bounding box in the RAGI dataset.

Figure 7.6 shows the distribution of faces height per person in the dataset by displaying the proportion of faces with a larger height than each of the height deciles from Table 7.4 in different colors. We can first notice that for each person there is a large variety in terms of face heights with everyone having at least one face in nearly each of the ranges. Then there is also a large variation of height distribution between the different volunteers. It is clear that some people spent more time closer to the screen than other. However, every volunteer has at least 5% of their faces that have a height larger than the two last deciles.

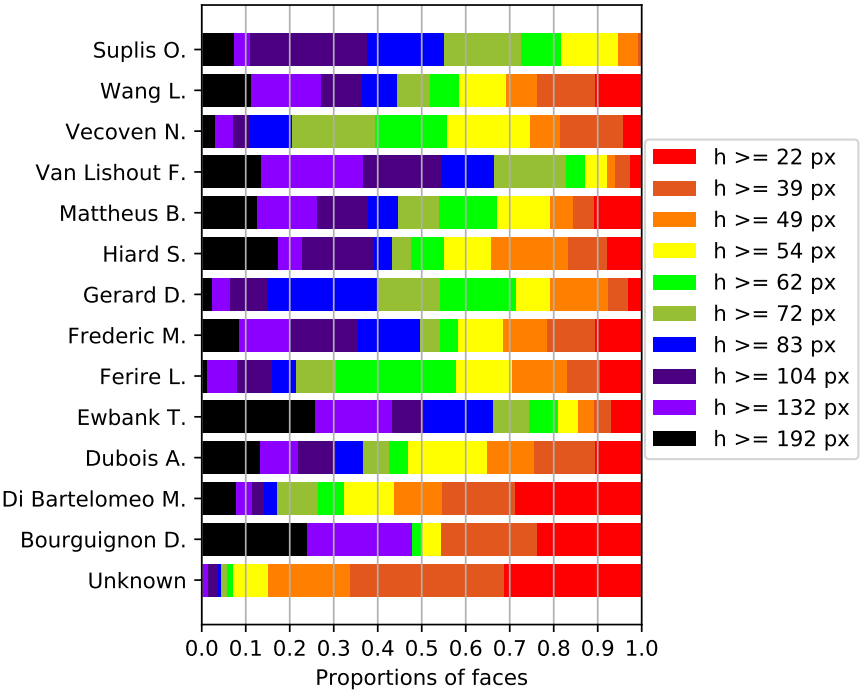


Figure 7.6: Distribution of faces height per person. The distribution for a given person is shown by giving the percentage of faces that are superior to the different deciles.

Scenarios

To finish the statistical part, I decided to look at the level of scenarios. First, Figure 7.7 is there to show that the size distributions observed on the whole datasets are similar if we divide the faces between the 4 scenarios. Indeed, for all the scenarios, we have similar long-tail distributions with small variations. One of the only remarkable difference is that the average size of faces seems a little bit larger in the case of R100.

This similarity does not hold when we consider the number of faces per person as shown in Figure 7.8. This most informative thing on this graph is actually the person for which there are 0 faces in some scenario which is important when considering per-scenario tests as explained later.

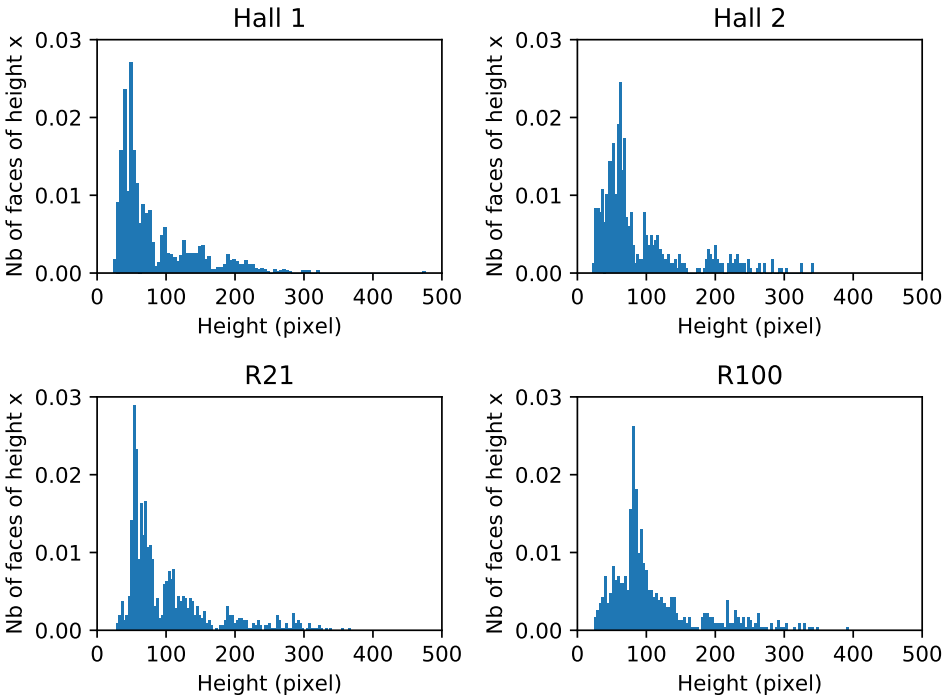


Figure 7.7: Normalized distribution per scenario of the width and height of recognizable faces in the RAGI dataset.

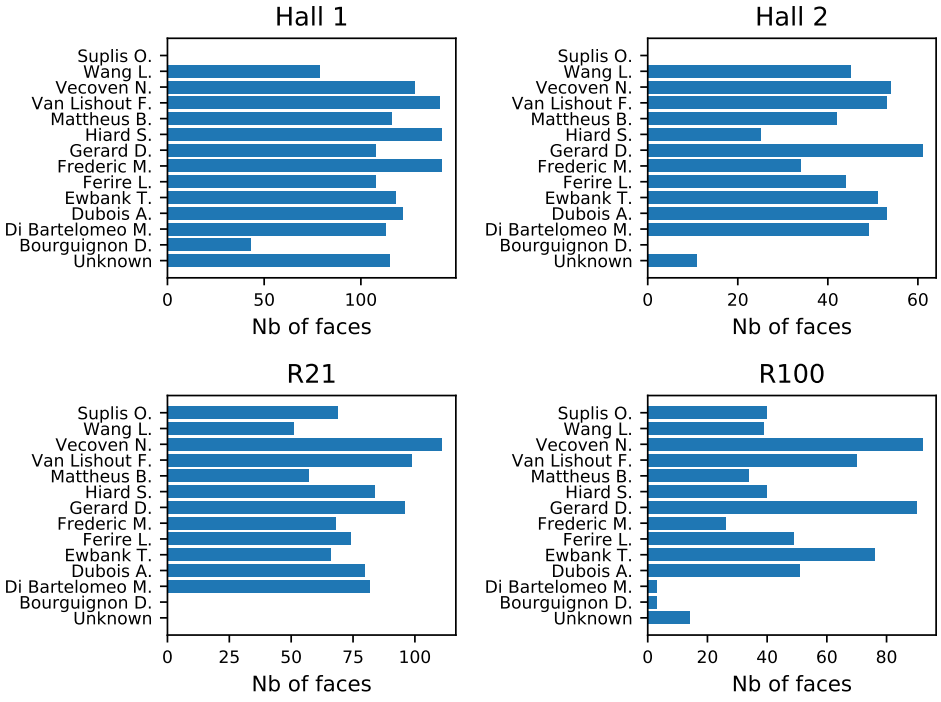


Figure 7.8: Number of recognizable faces per person in each scenario of the RAGI dataset.

7.2 Testing

This section contains a series of tests for evaluating the face detection and face recognition performances that can be reached on the RAGI dataset. The goal of these tests is to determine what performances can be obtained in a practical application while also trying to find ways to optimize those performances.

7.2.1 Face detection

Results on the whole dataset

Figure 7.9 shows the precision-recall curves of the four selected face detection algorithms applied to the RAGI dataset and Figure 7.10 is there to be able to make a comparison with the results obtained on WIDER FACE. We see that there is not a tremendous change in terms of results but we can point out some particularities. First, we can see that performances for VJ and FRCNN on RAGI are situated between their performances on the medium and hard set. The performance of HOG is better than on the medium set but worse than on the easy set. This seems to show that the level of difficulty of the RAGI dataset does not correlate that well with the difficulty-based subset division of WIDER FACE. This has also the impact that HOG performs here even better than FRCNN which was never the case on WIDER FACE. This improvement in the performance of HOG might come from the fact that RAGI is built from larger images than WIDER FACE and contains, therefore, larger faces on average. I will explore this possibility in a coming section.

Anyways, the main difference comes from SSH which performs exceptionally well on RAGI, even better than on the easy subset of WIDER FACE. When seeing this result, the first thing that comes to mind is the way the RAGI dataset was built. Indeed, even if the bounding boxes were readjusted manually, they were first detected by SSH which creates an inevitable bias towards this algorithm. I try to quantify this bias in the following section.

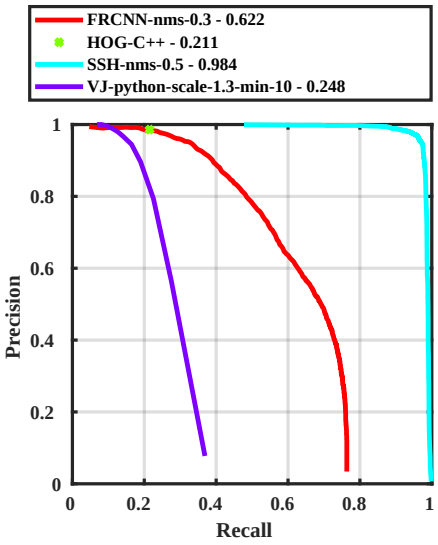
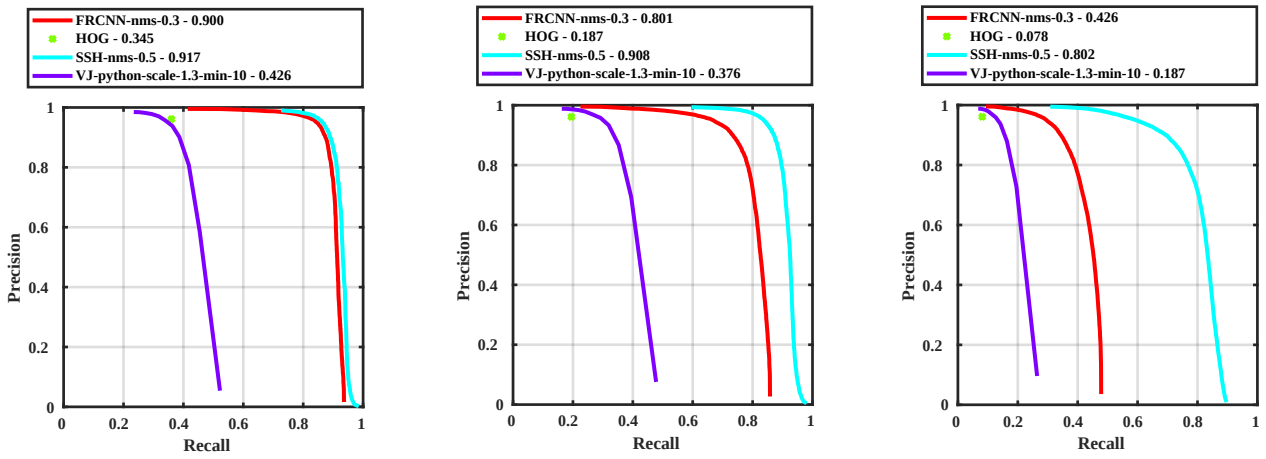


Figure 7.9: Precision-recall curves obtained by the best versions of the 4 tested face detection algorithms on the RAGI dataset.



(a) Easy (b) Medium (c) Hard

Figure 7.10: Precision-recall curves obtained by the best versions of the 4 tested face detection algorithms on the three subsets of WIDER FACE.

Table 7.5 shows the number of frames from RAGI or WIDER FACE that can be processed per second by each algorithm. As a reminder, WIDER FACE frames are 1024 pixels in height with a variable width with a median of 754 pixels while RAGI frames are 2048x2048 pixels. This means that the images in RAGI contain approximately 5 times more pixels than the ones in WIDER FACE. The results are reminiscent of what was obtained in section 6.1.4. Indeed, we can see that the time performances of VJ and HOG are drastically reduced on RAGI compared to WIDER FACE, the FPS being divided nearly by 4 for the first one and by nearly 9 for the other. On the opposite, the number of FPS is not even divided by 2 for the two neural network. This has the effect of closing the gap in time performance between those algorithms and VJ.

Dataset	RAGI		WIDER FACE	
Algorithm	CPU FPS	Real FPS	CPU FPS	Real FPS
VJ-python-scale-1.3-min-10	0.85	11.06	2.56	35.98
HOG-C++	1.85	/	15.94	/
FRCNN-nms-0.3	6.22	9.78	8.84	15.69
SSH-nms-0.5	3.19	5.27	4.09	6.68

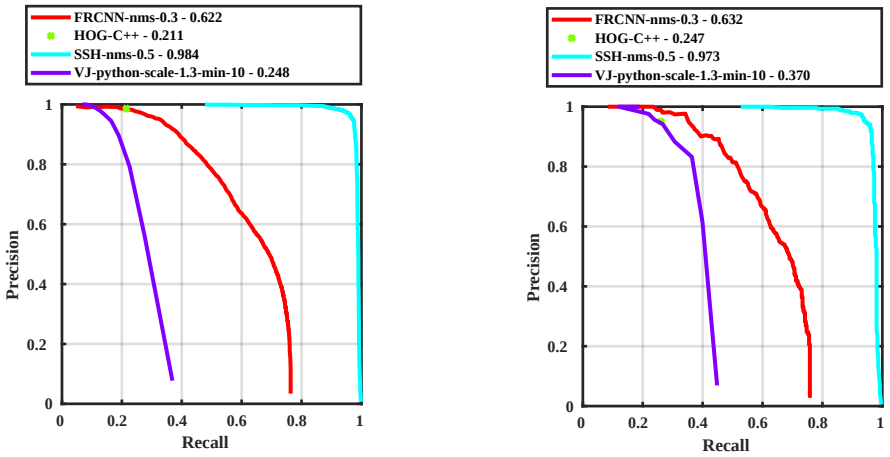
Table 7.5: Medians of the number of frames, coming from the RAGI dataset and from WIDER FACE, that can be processed per second, in CPU and real time, for the best versions of the 4 tested algorithms.

Bias analysis

Regarding positive bias for SSH, the best choice to get rid of it would have been to annotate the whole dataset manually. However, the downside is that the annotation process would take much more time. Moreover, it is important to remind that the dataset was not annotated totally automatically but that each automatically generated bounding box was verified to respect the previously mentioned annotation criteria. Considering that, I thought that the best alternative was to manually annotate a random subset of images to see how much the automatic part of the annotation would influence the result. Per scenario, I took 15 images, composing a

subset of 60 images, that I annotated manually and I tested the algorithms on this new dataset.

Figure 7.11 shows the results on the semi-automatically annotated and manually annotated dataset. Even though the manually annotated dataset is much smaller than the original one and that therefore this is results are not perfectly significant, it seems like the semi-automatic annotation process provides a small but negligible, with respect to the level of performance of SSH, positive bias for SSH. This analysis can be mitigated by the fact that all the other algorithms perform better on the manually annotated subset, meaning probably that the faces of this subset are easier to detect on average than on the whole dataset.



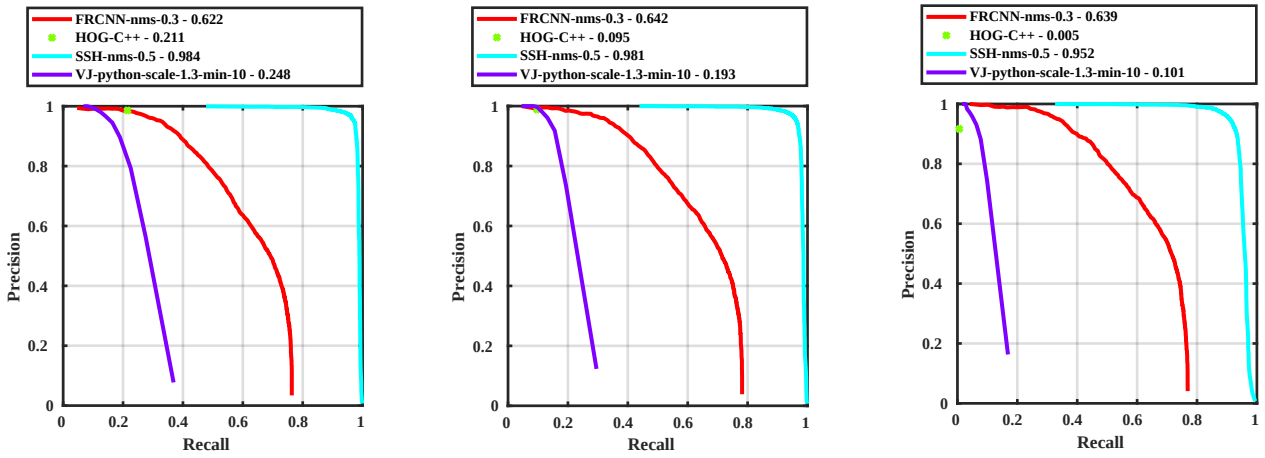
(a) Semi-Automatically Annotated (b) Manually annotated

Figure 7.11: Precision-recall curves obtained by the best version of the 4 tested face detection algorithms on the semi-automatically annotated RAGI dataset (a) and on a manually annotated subset of this dataset (b).

Influence of resolution

An aspect that I did not explore in the case of WIDER FACE is the influence of the resolution of the image. The resolution will definitely influence time but that can be at the price of worse detection performance. Moreover, knowing which resolution is acceptable can be useful to know how good the cameras must be to have a reliable system. In Figure 7.12, the results on the original dataset are juxtaposed with results on images that have been converted to lower resolutions (i.e. 1024x024 and 512x512).

These results confirm the idea, evoked two sections earlier, that HOG was performing better on RAGI than on WIDER FACE because the images were larger. Indeed, when decreasing the resolution, the performance of HOG drops dramatically. This is also the case for VJ and SSH to a smaller extent. The most unexpected behavior comes from FRCNN whose performances are the best for a medium resolution, followed by the smallest one whilst the performance on the original dataset is the worse. These differences might be linked to the rescaling process but it seems that FRCNN copes well with reducing the resolution of the image.



(a) 2048x2048 (b) 1024x1024 (c) 512x512

Figure 7.12: Precision-recall curves obtained by the best versions of the the 4 tested face detection algorithms on three versions of the RAGI dataset varying on the resolution of the images on which the detection is performed.

Table 7.6 shows the number of FPS obtained when using all three of the resolutions. As expected, for each algorithm, this value increases when the resolution decreases. For VJ, this increase is not very interesting because its time performance on the original dataset was already more than acceptable. The number of FPS is multiplied by 7 and then again by 4 when reducing the resolution two times. However, we cannot consider this second increase as a viable option as it would mean imply nearly no detections. The third line of this table is clearly the most interesting as we see that we can reach nearly real-time performance with FRCNN when reducing the resolution to 512x512. For SSH, the magnitude of the increase in FPS is smaller but using a medium resolution provides a gain of 2 FPS without affecting the performances.

Resolution	2048x2048		1024x1024		512x512	
Algorithm	CPU FPS	Real FPS	CPU FPS	Real FPS	CPU FPS	Real FPS
VJ*	0.85	11.06	2.56	30.72	8.07	89.69
HOG-C++	1.85	/	7.3	/	28.25	/
FRCNN-nms-0.3	6.22	9.78	10.16	17.63	12.20	21.92
SSH-nms-0.5	3.19	5.27	4.61	7.13	5.17	8.15

Table 7.6: Medians of the number of frames, coming from the RAGI dataset, that can be processed per second, in CPU and real time, with the best versions of the 4 tested algorithms when keeping the initial resolution (2048x2048) or reducing it to 1024x1024 and 512x512. To save space, VJ-python-scale-1.3-min-10 was renamed VJ*.

Influence of face size

Even though I already studied the influence of face size on performances with WIDER FACE, I wanted to reiterate on RAGI to see if there were some behavioral differences and also to provide a firmer decision ground for the project. To do so, I computed the precision-recall curve for 10 different size levels based on the height deciles. Each of the ten levels corresponds to the face of height between two deciles (and the minimum and maximum height for the two extreme

levels). For more clarity, Table 7.7 shows to which interval of heights, each level corresponds. I decided to display those results through the average precision to not overwhelm the section with too many graphs.

Level	0	1	2	3	4
Height Interval	[192;477]	[132;191]	[104;131]	[83;103]	[72;82]
Level	5	6	7	8	9
Height Interval	[62;71]	[54;61]	[49;53]	[39;48]	[22;38]

Table 7.7: Correspondence between size levels and height intervals of the RAGI dataset.

Figure 7.13 thus shows the evolution of the AP when going from the level corresponding to the largest faces to the one corresponding to the smallest faces. In general, we can see the same behavior as the obtained on WIDER FACE in section 6.1.5. Whilst the APs for VJ, HOG and FRCNN drop quite rapidly when the size shrinks, SSH performance does not seem to be affected before level 6. Even with the smallest faces, it still reaches an AP of 0.8. Interestingly, we can see that for very large faces, VJ and HOG seem to struggle more than with slightly smaller faces.

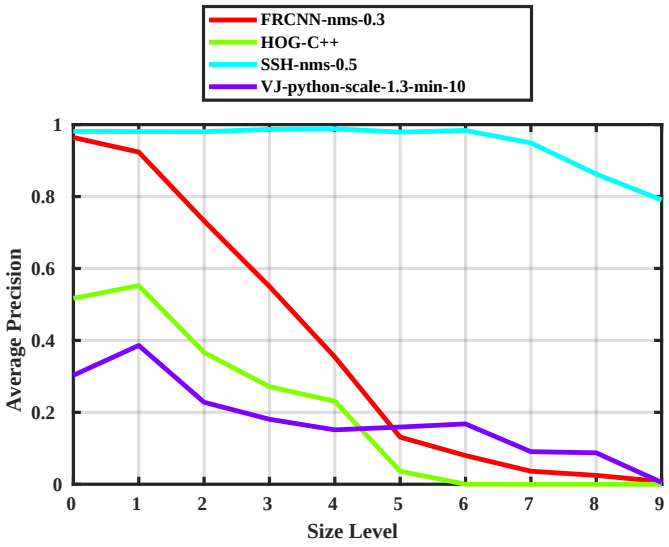


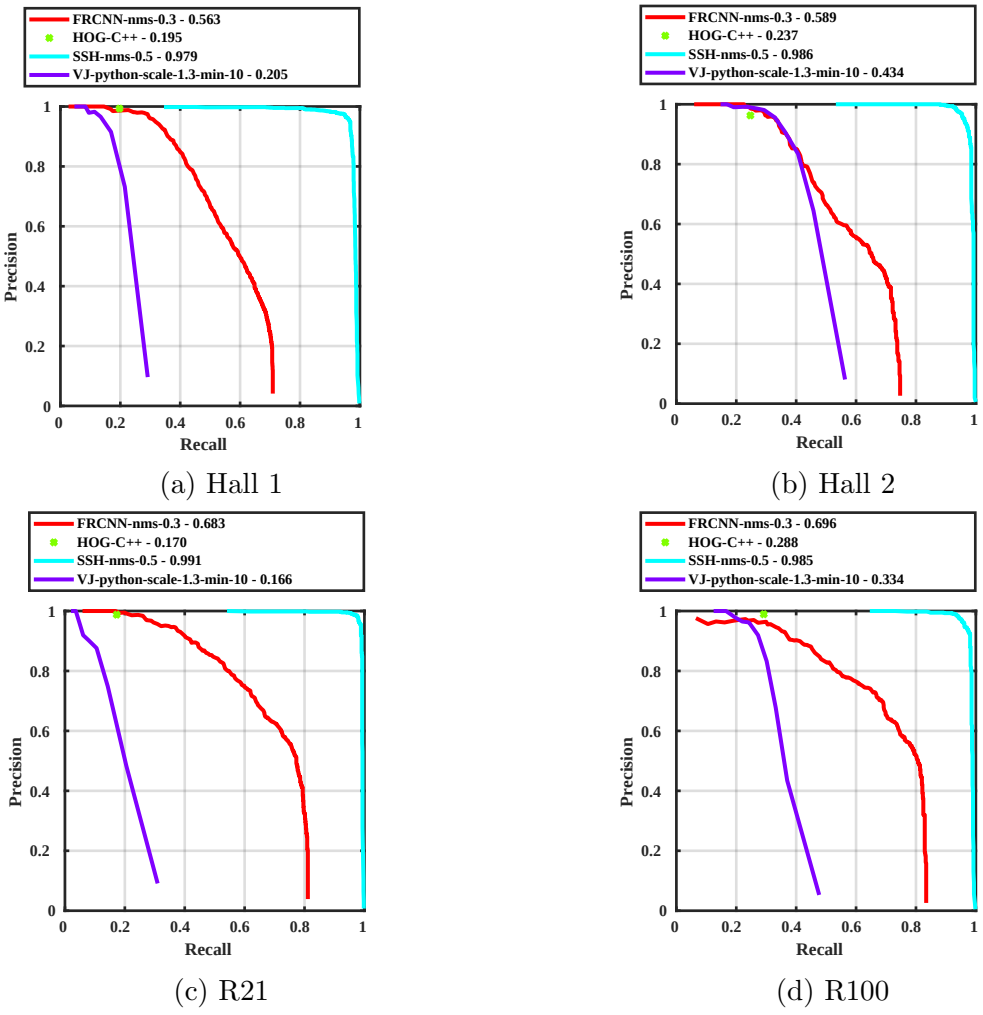
Figure 7.13: Evolution of the average precision of the best versions of the 4 tested face detection algorithms with the size level of faces from the RAGI dataset.

Influence of scenario

To finish with the face detection part, I wanted to analyze how the performances would change from one scenario to the other. The characteristics of each of the scenarios were not quantified and I can therefore not use this analysis to make inference about other scenarios. However, I think it still presents some interest because we have a rough idea of those characteristics (e.g. Hall 1 was much more illuminated than Hall 2) and it might help us in identifying places where it would be less suitable to position a welcome station.

In Figure 7.14, we can see the results for the four scenarios. Overall, it seems like the scenarios do not seem to affect the performances significantly. The best example of that is SSH whose AP is close to constant. However, we can find some patterns in the results. For

instance, the results in scenario R100 are always among the two best results in terms of AP whilst the results on Hall 1 are always among the two worst. The good results for R100 might be explained by the fact that it was situated in a closed space where people that are captured have a tendency to be closer to the screen. Another noticeable trait is that the results for scenario Hall 2 are also always better than in the case of Hall 1. This is probably linked to the illumination conditions where in the case of Hall 1 the camera was facing the outside and has thus to deal with more extreme conditions.



(a) Hall 1 (b) Hall 2

(c) R21 (d) R100

Figure 7.14: Precision-recall curves obtained by the best versions of the 4 tested face detection algorithms on the 4 scenarios composing the RAGI dataset.

7.2.2 Face recognition

To obtain face recognition performance, the RAGI dataset needs to be divided into a probe set and a gallery set. To choose how to do this division, two parameters have to be considered. First, the division should reflect at best how face recognition system of RAGI would work. This is however not clear yet and maybe some clarification will be possible with the following tests. However, I made the assumption that among all the images taken from one person, a random subset would be kept in the gallery for further recognition. Therefore, I decided to keep the same height distribution of faces in both the probe set and gallery. The second parameter corresponds to what proportion of the whole dataset each of those subsets should contain. I

took this decision based principally on the tests I wanted to realize. In particular, I wanted to test the influence of number of faces per person in the gallery with a number up to 40 which meant that the gallery should contain more than 520 faces (which makes approximately 15 %). Then considering the rest of my tests, I decided to settle on a division of 70% for the probe set and 30% for the gallery.

Figure 7.15 shows the resulting number of faces per person in both subsets with their height distribution. We can see that the graphs are quasi-identical except for the values on the x-axis showing that there are more faces in the probe set. I will make an initial test considering the whole probe set and gallery set and then I will try to study the influence of some parameters by varying the images that are contained in the gallery.

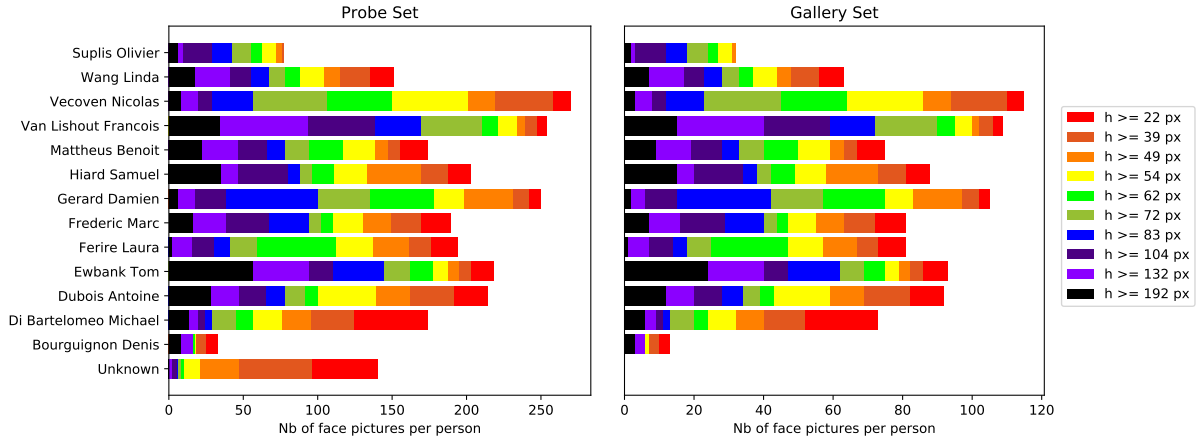


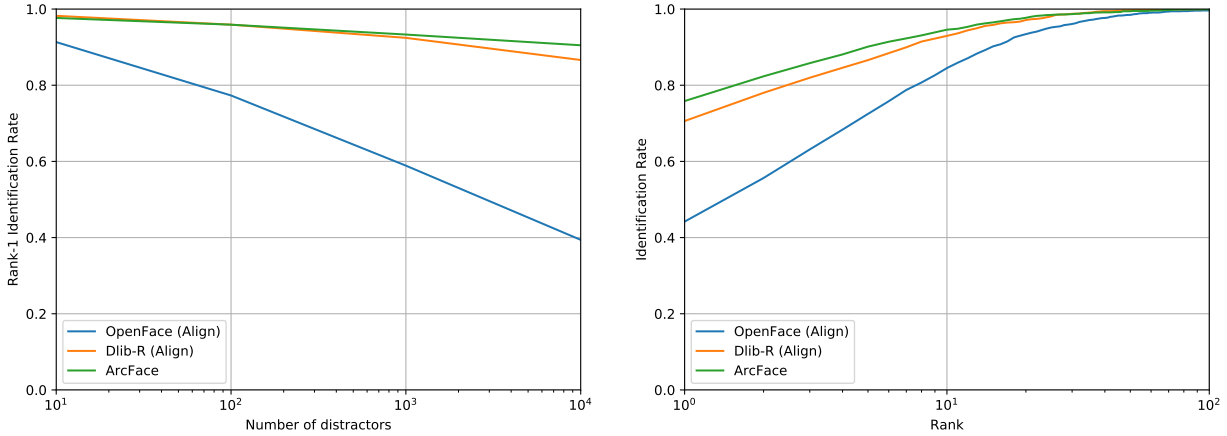
Figure 7.15: Number of faces per person superior to each height deciles in the probe set and gallery set of the RAGI dataset.

The last remark can be made about pictures from unknown people. Basically, all these images went into the probe set as shown in Figure 7.15. Indeed, this follows what would happen in a real scenario where unknown people do not have any pictures of them in the RAGI gallery, which is why they are considered as unknown. As a reminder, to try to prevent predicting a known identity for this unknown people, a threshold is used to filter cases where the closest face in the dataset is too different from the image. However, to simplify tests, I decided not to consider unknown faces initially. The impact of adding them in the face recognition protocol will be studied separately.

Results on the whole dataset

Figure 7.16 contains two very different graphs but they are interesting to compare in order to characterize the general performance of the face recognition algorithms on the RAGI dataset. The first graph shows the evolution of rank-1 identification rate on MegaFace when the number of images in the gallery increases. The second graph is showing the evolution of identification rate across ranks on the RAGI dataset. By looking at the value of this curves at rank-1, we can compare the overall performance on RAGI to the overall performance on MegaFace. The results are quite stunning. Indeed, we can see that the performances on RAGI are worse or just a bit better in the case of OpenFace, than the performance with 10,000 distractors on MegaFace. This is surprising mainly for two reasons. On the one hand, the MegaFace evaluation protocol is such that for each probe that is tested, the gallery only contains one image of the same person

whilst in RAGI this number is on average of 78. On the other hand, the number of distractors is much larger in MegaFace. Indeed, as we have 1,020 face picture in the gallery of RAGI, this means that on average we have around 942 distractors.



(a) Rank-1 vs distractors number on MegaFace (b) CMC curves on the RAGI dataset

Figure 7.16: Comparison between the performance of the best versions of the 3 tested face recognition algorithms on MegaFace and on the RAGI dataset. The performances on MegaFace are reported through the evolution of rank-1 identification rates with the number of distractors in the gallery (a) while the performances on the RAGI dataset are reported through CMC curves (b).

There are several elements that can explain such a difference. The first cause is the way the images composing the MegaFace benchmark were annotated. The probe set contains images from FaceScrub whose faces were detected using Viola&Jones algorithm which is known to detect mostly frontal faces and, as we showed in our experiments, fails to detect faces characterized by more complex features. Then, the gallery set is based on the MegaFace dataset. Even though VJ was not used in this case, faces were still detected automatically using the HeadHunter algorithm. Even if I did not manage to test this algorithm, we can see on the result pages of some of the earlier-mentioned benchmarks that it outperforms VJ but that it does not reach the performances of SSH. If we add this to the fact, that in our case, annotations were additionally manually corrected, we can infer that the RAGI dataset proposes much more challenging faces to recognize.

The second argument that could explain the performance differences is that in MegaFace, the images from the probe and gallery set comes from two different datasets which could, therefore, have a very different underlying distribution. This is supported by the fact that the pictures in those two datasets do not come from the same source. For FaceScrub, pictures of celebrities are downloaded by typing their name in a search engine while MegaFace is built on images coming from Flickr. I assume that this difference in distribution makes it easier to discriminate between the true identity coming from FaceScrub that was inserted in the gallery and the distractors coming from MegaFace. On the opposite, all images in the RAGI dataset where taken in the same context.

Finally, this difference in performance could just come from the fact that the faces in RAGI are just more difficult to recognize than in MegaFace. An argument in favor of this theory is the

median number of pixels per image of the two datasets. While for the MegaFace benchmark, this median is of 22650 for the MegaFace dataset and 30276 for FaceScrub, the median for RAGI is only of 4032 pixels per faces. As shown in the Figure 7.17 based as one the same height levels as previously, the smaller the face the more difficult it is to recognize it.

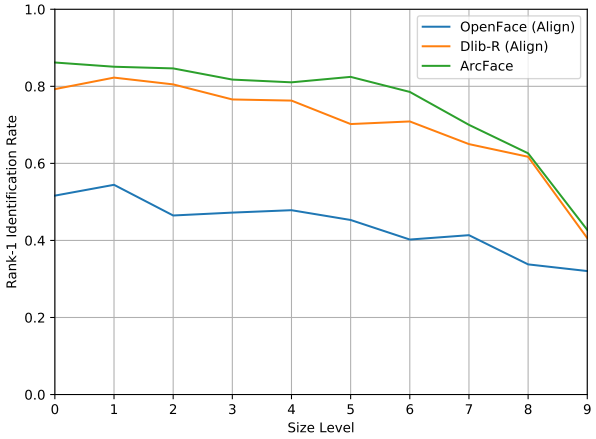


Figure 7.17: Evolution of the rank-1 identification rates of the best versions of the 3 selected face recognition algorithms with the face size level.

This size difference seems also to have some effect on the feature generation time. As shown in Table 7.8, the number of FPS increases when algorithms are used on RAGI. The biggest improvement appears for OpenFace. Its magnitude is so large it made me wonder if something didn't go wrong during the testing on MegaFace. This was the occasion to see how the computing time evolved with the face size. The evolution of the number of FaPS with the size of the face picture is shown in Figure 7.18 and confirms that this large change in computation time for OpenFace was not specific to MegaFace. However, I did not have the time to find out why this was happening and if this was intrinsic to the algorithm or a misuse of the algorithm.

Dataset	RAGI		MegaFace	
Algorithm	CPU FaPS	Real FaPS	CPU FaPS	Real FaPS
OpenFace (Align)	127.08	64.38	29.88	24.66
Dlib-R (Align)	246.67	/	283.13	/
ArcFace	50.71	60.09	50.98	60.35

Table 7.8: Medians of the number of faces, coming from the RAGI dataset and from MegaFace, that can be processed per second, in CPU and real time, by the best versions of the 3 tested face recognition algorithm.

To close this comparison, we can see that the order of performances seems to stay the same with ArcFace and Dlib-R still providing quite close levels of performance.

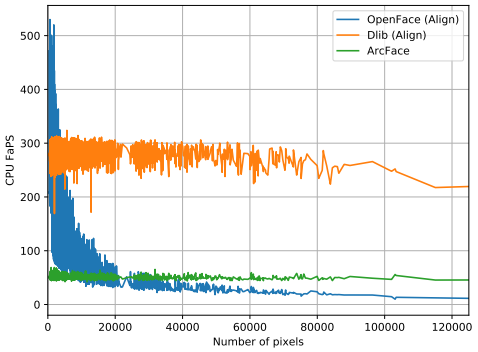


Figure 7.18: Evolution of the number of faces, coming from the RAGI dataset, that can be processed per second, in CPU time, with the number of pixels constituting the face images of the RAGI dataset.

Influence of gallery set size

I mentioned earlier in the tests on MegaFace that the size of the feature vectors and the number of images in the gallery affected the time needed to predict the identity of a given probe. I decided to do a similar test with RAGI particularly to see how the number of face pictures per person in the gallery would affect the performance. To do so, I used the original gallery set and restricted the number of faces per person to a maximum number. If the person had more than this number of pictures in the original gallery, I took a random subset of this size while if he/she had fewer pictures than this number I took all of its pictures. I decided to fix this maximum number to a series of values starting at 10 pictures per person up to 120 with an increment of 10. As the largest number of pictures for a given person in the original gallery set is of 115, the gallery set obtained when fixing the maximum number to 120 is equal to the original gallery set.

This method leads to an average number of faces per person in the gallery that is smaller than the maximum number used to create the gallery. As can be seen in Figure 7.19, I decided to use this average number to show the evolution of the identification rate as it is most representative of the quantity of information contained in the gallery. We can see that the identification rate augments with the size of the gallery but that, around 60 faces per person, it seems to start saturating. Moreover, for the three algorithms, the improvement in performance between the smallest gallery and the largest one is smaller than 15 % which shows the limits of just adding more faces to the gallery to obtain better performances.

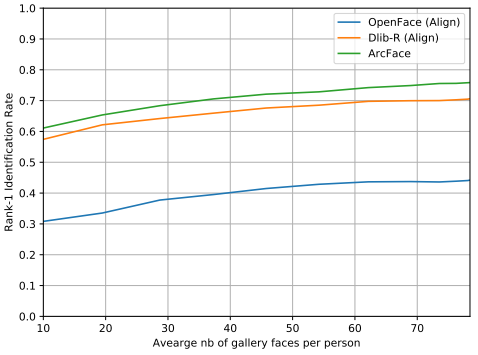


Figure 7.19: Evolution of the rank-1 identification rates obtained by the best versions of the 3 tested face recognition algorithms with the number of faces per person in the gallery.

Influence of gallery face sizes

Whilst I have made the assumption that the probe set and gallery would follow the same size distribution, this is far from guaranteed. Indeed, this distribution depends completely on the way the gallery is furnished with images and updated. Originally, the idea was to take pictures of a user while he/she was registering on the application which would have meant that the gallery images would have mainly been 'large' faces. Another possibility was to collect a series of images in a buffer prior to the registration of a user and to ask this user to confirm that the picture where his/hers when he/she registered. The collection of gallery pictures could be started as soon as the person is visible from the camera and smaller face picture could thus be gathered. In addition, people change across time and it is thus necessary to update their images in the gallery. The simplest way to do so is to replace old detections by adding new ones for which we have a high confidence in terms of identity.

These two last points have the same common problem: both the face detection and face recognition algorithms perform generally better on larger faces, as shown previously. If we follow this simple protocols, larger faces will end up composing the majority of the gallery.

I decided to see how this kind of behavior would influence the face recognition performances. To do so, I realized three tests by keeping successively only the 10, 20 and 30 largest faces of each person in the gallery. To allow a fair comparison, I decided to show this result next to the results obtain with galleries with 10, 20 and 30 random pictures of each people. The rank-1 identification rate is shown for these two different scenarios in Figure 7.20 with the former being denoted as 'Large'. As in the previous figure showing the influence of gallery size, the x-axis is labeled with the average number of pictures per person in the gallery.

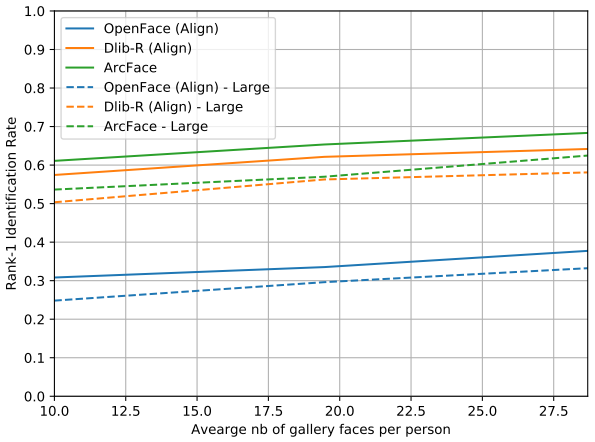
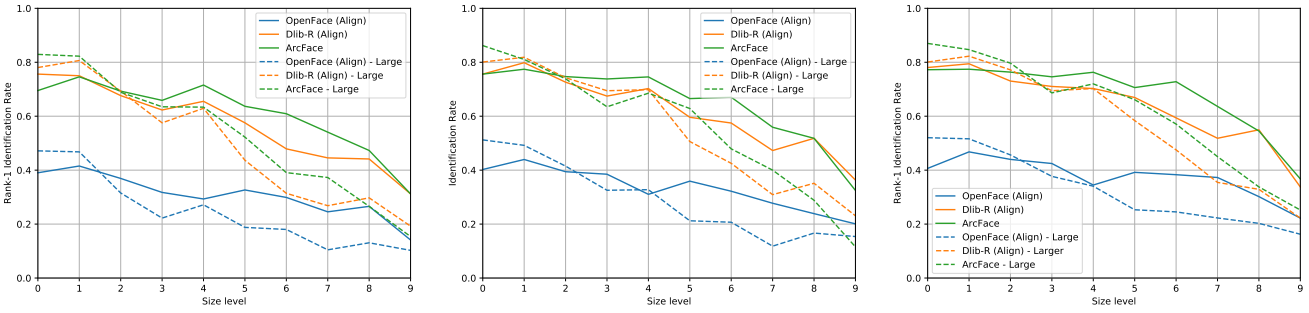


Figure 7.20: Comparison between the rank-1 identification rates obtained by the best versions of the 3 tested face recognition algorithms on the RAGI dataset for different number of faces per person in the gallery selected either randomly or as the largest face picture for each person (denoted as 'Large').

This Figure shows that keeping only large faces results in worse performances for all three algorithms and for all three gallery size. This difference seems to decrease with the gallery size but this might be explained just by the fact that the larger the gallery is the more the two scenarios contain similar images as they are both taken from the same original gallery which is of finite size. We can easily guess that this decrease in performance is mainly due to worse performance in recognizing smaller faces which do not necessarily have the same distribution as large faces. To make sure this is the case, I decided to display the performance per size of face

in Figure 7.21. The behavior is similar for the three gallery sizes and for the three algorithms. We can see that the largest faces are better recognized but that this is overcompensated by a decrease in performance for all faces of size level greater than 4-5.



(a) 10 pictures (b) 20 pictures (c) 30 pictures

Figure 7.21: Comparison between the rank-1 identification rates obtained by the best versions of the 3 tested face recognition algorithms on the RAGI dataset for different face size levels when face pictures in the gallery selected either randomly or as the largest face pictures for each person (denoted as 'Large'). This comparison is done for 3 different number of pictures per person in the gallery set.

All in all, these results emphasize that to recognize faces of a given size, it is important to have faces of a similar size in the gallery. This leaves us with two choices mainly: either we find a way to keep in the gallery a varied distribution of faces in terms of size, or we focus mainly on larger faces.

Influence of scenario

The goal of a per-scenario test is two-fold. Firstly, as was done for face detection, we can compare the performances between each scenario and thus maybe infer some characteristics that are positive or negative for face recognition. The main difference between this test for face recognition and face detection is that for face recognition, to be able to make a fair comparison, we have to pay attention to the number of faces that each scenario contains. Indeed, if for each scenario, we would just take all the pictures and divide them into a probe and gallery set with fixed proportions, scenarios with fewer pictures would be disadvantaged because they would have a smaller gallery as a result. To try to overcome this, I decided to fix the probe set size and gallery size to respectively 23 and 10 pictures per scenario (to respect the ratio of 70%-30%). These numbers were selected based on the number of faces per person in the scenario with the smallest number of faces (i.e. R100). In addition, people that had less than 10 pictures in one scenario where taken out of the corresponding test.

Figure 7.22 shows the per-scenario results. As for face detection, we can see that Hall 2 provides better performance than Hall 1 probably given the difference of luminosity. The R100 scenario which is the one to possess the highest mean of face height gives slightly better performances for Dlib-R and ArcFace than the other scenarios but the performances of OpenFace are less good than in Hall 2.

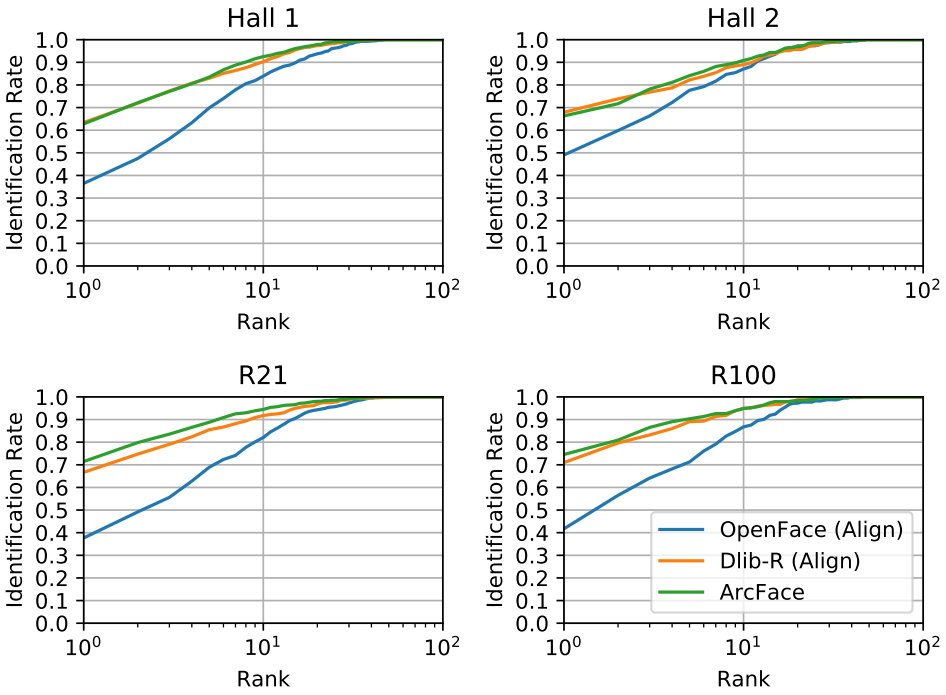


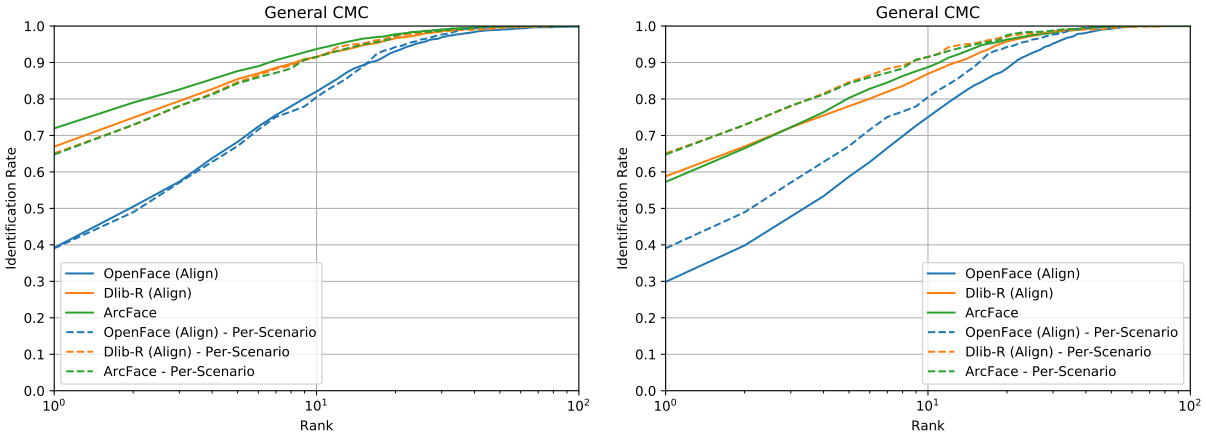
Figure 7.22: CMC curves obtained by the best versions of the 3 tested face recognition algorithms on the 4 scenarios composing the RAGI dataset. Probe faces coming from one scenario are compared against a gallery of 10 faces per person coming only from this same scenario.

Figure 7.23 shows the second goal of per-scenario test which is to see if comparing probes against images taken only in the same context would improve or decrease the performances. Indeed, on the one hand, limiting in this way the face recognition procedure might improve the performances because probes are compared to pictures captured in similar conditions and this might ease the classification process. On the other hand, the performances might be negatively affected by the fact that we lose the diversity of facial features captured across scenarios.

To obtain an answer to this question, I first built one probe set and one gallery per scenario. Each of the galleries was built by selecting 10 faces per person coming from this scenario while the probe sets were built from all the remaining faces of this scenario. I proceeded evaluation on each of these probe-gallery pairs individually to obtain CMC curves that I then combined using a weighted average, whose weights where the number of images in each probe set. To compare this per-scenario result to the original scenario, I combined the per-scenario probe sets together and per-scenario galleries together so to obtain one pair of 'across-scenarios' probe-gallery set where the gallery contains 40 images per person. This comparison is shown in Figure 7.23 (a). We can see that the per-scenario performances are worse than the global ones. However, this is very subtle for both Dlib-R and OpenFace. Only for InsightFace can we see a drop of nearly 10% in rank-1 performance. Moreover, this comparison is actually not very fair. Indeed, when comparing faces per-scenario, the galleries are only composed of 10 faces while in the other case the gallery was made up of 40 faces. However, as we could see in Figure 7.19, the difference of performance when using a gallery of 10 or 40 faces per person was near 10% for each algorithm.

To make a fairer comparison, Figure 7.23 (b) shows the same comparison but where in the 'across-scenarios' case, only 10 randomly chosen faces per person were kept in the gallery set. We can see that the behavior is completely inverted with per-scenario performances becoming

better than the across-scenarios performances. Once again, this results must be mitigated. This inversion seems in some sense logical has the combined galleries of the per-scenario test now contains much more information than in the across-scenarios test. Looking at graph (a), we could then argue that if we have access to these 40 images per person, we should always compare each probe to all these images rather than the per-scenario subset. However, as mentioned in section 6.21, the larger the gallery size against which the probes are compared the larger the classification time. Being constrained by computation time, we might thus want to limit the number of faces per person in the gallery and in that case, it might make sense only to use pictures taken in the same context if there is a sufficient number of such.



(a) (b)

Figure 7.23: Comparison of the performances obtained by the best versions of the 3 tested face recognition algorithms through CMC curves when comparing probes against gallery images from all scenarios or only against gallery images coming from the same scenario (denoted as 'Per-Scenario'). The 'per-scenario' gallery sets are composed of 10 pictures per person in both (a) and (b). In (a), the gallery set of the across-scenario case is made of 40 images per person obtained from the combination of the 'per-scenario' galleries while in (b), 10 images per person are randomly selected out of these 40.

You Appear Once Per Image

One reality that was not considered in the evaluation procedure , for now, is that, in the majority of cases, one person appears only once in a given image which I refer as the 'You Appear Once Per Image' concept or YAOPI for short. For now, the evaluation procedure consisted in taking each face picture independently and to predict as its identity the one attributed to the gallery face with the closest feature representation. However, in a practical application, one can consider that in a given picture, once a given identity has been predicted with sufficient confidence, it should not be predicted again. This concept can be mitigated by two main exceptions. Firstly, a person could appear multiple times because of the presence of reflective surfaces. This is, for example, the case in the R100 scenario where the interior of the elevator contains a mirror. The second exception is not based on a scenario configuration but comes from the problem of multiple detections from which face detection algorithms can suffer. However, this problem can be dealt by using techniques such as NMS with sufficient threshold.

To evaluate algorithms with this concept, the idea is to compute the distances to gallery pictures for all the face pictures in a given image. Then, the prediction process is done iter-

atively by assigning the corresponding identity to the face picture that is the closest to one of the gallery pictures, removing this probe and removing all the distances to gallery pictures associated to the same identity. This is done until a prediction has been made for all identities. In addition, we have to keep in mind that the more faces there are on a given picture, the more efficient this process is. For the result to be meaningful in any context, I changed slightly the computation of the identification rate. Rather than giving a score of 1 or 0 per face, I give a score per image which is computing by adding the score for each face in the picture and dividing it by the number of faces in the picture.

In Figure 7.24, I compare the performances obtained when using this concept (denoted as YAOPI on the graph) or not using it. We can see that using YAOPI always improves the performances but not in a significant way. This is especially the case as we have an average of 7.21 faces per image, which is already quite high, for 13 different identities, which is quite small. In a real application, the number of identities would be much higher, making the impact of YAOPI shrink even more.

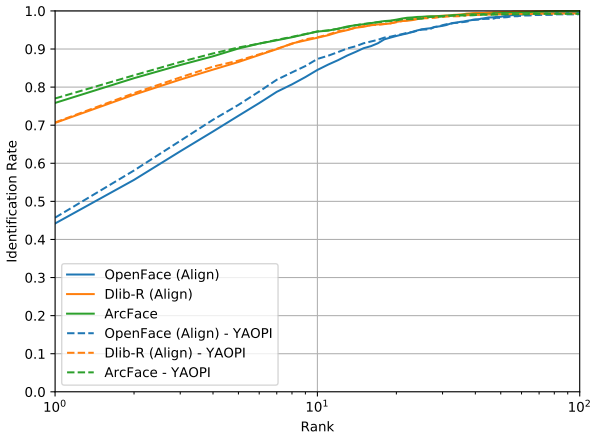


Figure 7.24: Comparison of the performances obtained by the best versions of the 3 tested face recognition algorithms through CMC curves when considering that each person can appear only once in an image (denoted as 'YAOPI') or not considering it.

Results with unknown

A big aspect of face recognition that I have not dealt with yet, neither with RAGI nor with MegaFace, is the face pictures corresponding to unknown identities. Indeed, in the tests presented up to now, I did not use those pictures during evaluation. The main add-on to the face recognition process to take them into account is a threshold on the distance between two feature representations. Basically, if the distance between a probe and the closest gallery picture is greater than this threshold, the face is considered as unknown and otherwise, the identity of this gallery picture is assigned to the probe. This is very similar to a binary classification problem with the little twist that one of the two classes is actually divided into sub-classes corresponding to all the identities contained in the gallery. We can therefore evaluate this classification by using a modified version of the ROC curve where the True Positive Rate and False Positive Rate are respectively replaced by the True Positive Identification Rate computed at rank-1 and the False Positive Identification Rate defined in section 2.2.2. I could have used the DET curve but I thought it was less interesting because it was not showing directly the TPIR.

To obtain a series of points on this curve, one has to do the evaluation using different values of thresholds. Moreover, these values have to be chosen according to the algorithm being tested. Indeed, each algorithm extract feature in a different features space and the magnitude of distances between representations of images can vary from one to another. For each algorithm, I had to choose a series of thresholds. To be the closest possible to a practical application where we only have access to gallery pictures, I decided to use the distances between features representation of gallery pictures. For each identity, I computed all the distances between the gallery pictures associated with this identity and took the percentiles of these distances. This gives me an idea of how dispersed the feature representations of a given identity are.

I then use these percentiles in two different ways. Firstly, I compute the mean of each percentile across all identities and use these mean percentiles as thresholds. I also add two thresholds that I consider as minimum (i.e. 0) and maximum (i.e. a very large value) values for distances. This process results in the solid lines in the following graphs. The second method comes from the observation that different identities might have a different level of dispersion of their feature representations. Therefore the use of a per-identity threshold might be more appropriate. In this second method, for a given probe, I use the percentile associated with the identity that is predicted at rank-1 as threshold. This gives the dotted line denoted as 'Per Id' in the next graphs.

From Figure 7.25, we can see that adding a threshold has very different effects on the performances of the three algorithms. As expected the lower the threshold, the lower TPIR and FPIR. However, compared to Dlib-R and OpenFace, the TPIR drops much less rapidly leading to a much larger difference between Dlib-R and ArcFace for low values of FPIR. For example, if we consider the solid lines, at an FPIR of 0.2, ArcFace still manages to reach a TPIR, which as a reminder is equivalent to rank-1 identification rate, of 0.5 while Dlib-R is only at 0.2.

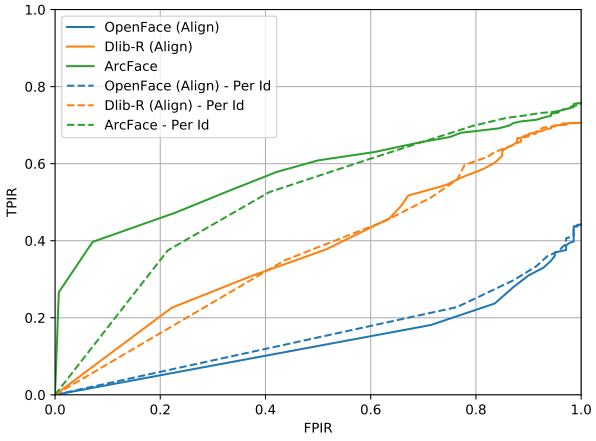
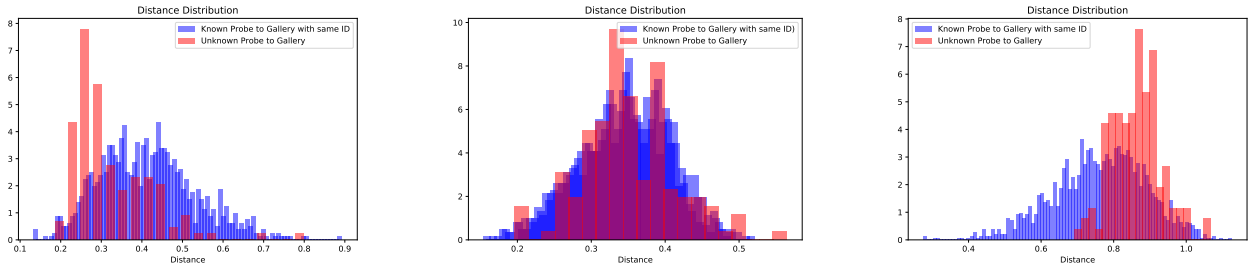


Figure 7.25: TPIR-FPIR curves obtained by the best versions of the 3 tested face recognition algorithms when using as thresholds the mean of the per-person distance percentiles for every probe or by using as thresholds, for a given probe, the distance percentiles corresponding to the predicted identity, denoted as 'Per Id'.

These differences in behavior can be explained by the plots in Figure 7.26. To built those graphs, I took all the pictures in the probe set and compute their distance to all the pictures in the gallery. For each probe, I sorted the distances and kept only the smallest one with the

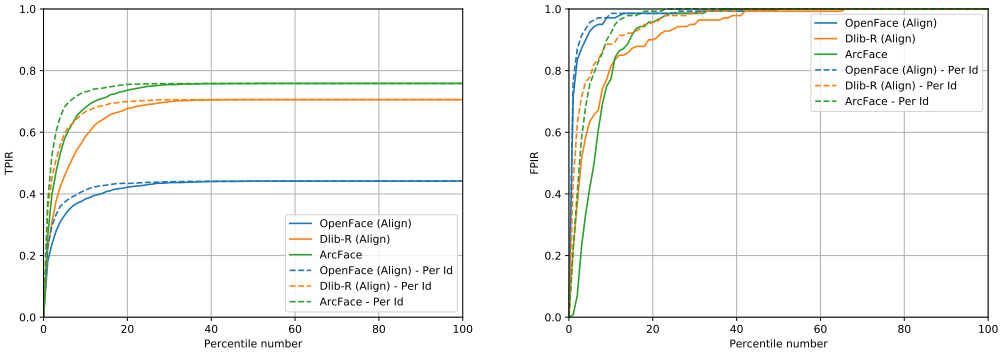
corresponding identity. The red distribution is computed by keeping the distances associated uniquely to an unknown probe while the blue distribution takes into account the distances associated to a probe whose closest gallery was of the same identity. This graph is directly related to the previous figure because if you choose a fixed threshold, the densities of the blue and red distributions on the left this threshold will give you a point on the TPIR-FPIR curve. This shows why for example in the case of ArcFace, when increasing the threshold value, the FPIR takes more time to start increasing than the TPIR.



(a) OpenFace (Align) (b) Dlib-R (Align) (c) ArcFace

Figure 7.26: Distribution of the shortest distances between probe and gallery images generated by the best versions of the 3 tested face recognition algorithms. For the red distribution, probe faces are faces from unknown people. For the blue distribution, the distances are computed between known probes and gallery faces of the same identity.

Figure 7.25 also shows the performance of the classification step if we use a per-identity threshold. We can see that depending on the algorithm, it has either a rather positive, rather negative or no effect. To understand more deeply what is happening I decided to draw the evolution of TPIR and FPIR with the threshold value as shown in Figure 7.27. What is interesting is to see that for a given percentile, the per-identity technique will provide higher TPIR but also a higher FPIR, the effect on the global performance resulting then just of the relative amplitude of the two increase. The main thing to remember is that per-identity seem more permissive than when using the mean of thresholds over all identities.



(a) TPIR vs distance percentile number (b) FPIR vs distance percentile number

Figure 7.27: Evolution of the TPIR and FPIR values obtained by the best versions of the 3 tested face recognition algorithms with the distance percentile number when using as thresholds the mean of the per-person distance percentiles for every probe or by using as thresholds, for a given probe, the distance percentiles corresponding to the predicted identity, denoted as 'Per Id'.

7.2.3 Face detection and recognition

Compared to WIDER FACE and MegaFace, the RAGI dataset presents the unique feat to provide testing for both face detection and recognition. This provides the opportunity for a final test of the full face recognition pipeline combining face detection and face recognition per-say. I will test this pipeline by combining each of the four face detection algorithms with each of the three face recognition algorithms.

Testing procedure

To generate results for the full pipeline, there are a series of steps to follow, mostly taking together all the different procedures that were explained throughout this work.

The first step consists in detecting faces of the dataset with the face detection algorithms. For HOG, there are no parameters and it can be directly used. However, for VJ, FRCNN, and SSH, we need to decide where we want to be situated on the precision-recall curve. To be able to compare the results to the ones of HOG, one can either use the same precision or recall level. I decided to do both by comparing at the same recall level, the CMC curves while comparing at the same precision level, the TPIR-FPIR curves.

I think that CMC curves are a more appropriate comparison choice at same recall level because in that case, algorithms produce an identical number of true positives which lays a fair ground for identification rate comparison. This fixed recall rate comes with a varying precision which implies a varying number of false positives. However, as these false positives contain no faces, I consider them as unknown and in the CMC test I do not consider unknown identities, therefore the number of false positives does not influence the test. Similarly, TPIR-FPIR curves are more suitable to compare performances at same precision level because in that case relative number of false positive to true positive is fixed.

To close on the choice of test, I would just point out that I am talking about precision and recall levels rather than values. Indeed, on the precision-recall curves that were generated on RAGI, there is not an infinite number of points as they correspond to a finite number of parameters. Therefore, there is no guarantee to find the exact same value of precision or recall than HOG. Tables 7.9 and 7.10 show respectively for same recall level and same precision level the value of the parameters that were chosen with the corresponding expected precision and recall values. Note that in the case of FRCNN and SSH the parameter is the threshold to filter out prediction with too low confidence level while in the case of VJ the parameter is the number of neighbors.

The first interesting thing to note about this tables is that we actually get the same values for FRCNN in both cases which is logic as the precision-recall curve of FRCNN actually passes on the precision-recall point of HOG as shown in Figure 7.9. Secondly, we can note that for SSH in the case of the same recall, the smallest recall that was computed was of 0.4774. We can also see that the recall obtained by SSH at same precision level is much higher than for the other algorithms.

Algorithm	Parameter Value	Precision	Recall
HOG	/	0.9858	0.2148
FRCNN	0.989	0.9847	0.2171
SSH	0.999	0.9994	0.4774
VJ	2	0.7935	0.2255

Table 7.9: Parameters and precision-recall values at which the face detection algorithms will be used when considering same recall level.

Algorithm	Parameter Value	Precision	Recall
HOG	/	0.9858	0.2148
FRCNN	0.989	0.9847	0.2171
SSH	0.781	0.9857	0.9132
VJ	7	0.9898	0.1095

Table 7.10: Parameters and precision-recall values at which the face detection algorithms will be used when considering same recall level.

Once the bounding boxes have been generated, the next step was assigning them their true identity. To do so, I compared all the predicted bounding boxes with our annotations and associated the identity corresponding to the bounding boxes with the highest IoU. If the maximum IoU was under 0.5, I considered this face to correspond to an unknown identity. Finally, if the bounding box would match with an annotation that was marked as to be ignored, the only option I had was to get rid of it because those ignored annotations were not assigned an identity during the face recognition annotation process. Once this was done I cropped the original images to obtain the pictures of all the detected faces.

The final step was to generate the probe set and gallery for each algorithm. For the following reason, I decided to recreate both of them rather than using the original gallery and using all detected faces as probes. I want the results to reflect the ones in a real application in which both the gallery and probes would have supposedly generated by the same detector. To generate these two new sets, I proceeded exactly in the same way as in the previous section with 70% of the faces in the probe set and the rest in the gallery.

Finally to draw the TPIR-FPIR curve I used the previously computed threshold distances based on the mean of the percentiles as explained earlier. I did this mainly because the distances distribution should be roughly the same and because these thresholds would not in any way influence the results.

Results

Figure 7.28 and 7.29 both display 12 curves (CMC and TPIR-FPIR), each of them resulting from a face recognition pipeline made up of one detector, whose name is displayed as title of each subfigure, and a face recognition algorithm as noted in the legend.

The main global remark about these figures is that using worse performing face detection algorithms, we can obtain better performances at face recognition. This is totally logical as these algorithms will detect larger, less blurry, less occluded, more frontal faces than the better

detectors which are then easier to recognize.

If we compare the CMC curves, we can indeed see that HOG and VJ are overall better than the SSH and FRCNN. There are some exceptions, however. We can see that while the combination of Dlib-R with HOG reaches nearly 100% at rank-1, the rank-1 identification rate for ArcFace applied on HOG is smaller than when applied on SSH. The very good performance of Dlib-R on HOG might be explained by the fact that these two algorithms are part of the same library and we can thus infer that Dlib-R was probably trained on faces detected with HOG. Another exception is that face recognition algorithms perform worse on FRCNN than on SSH which is a better detector.

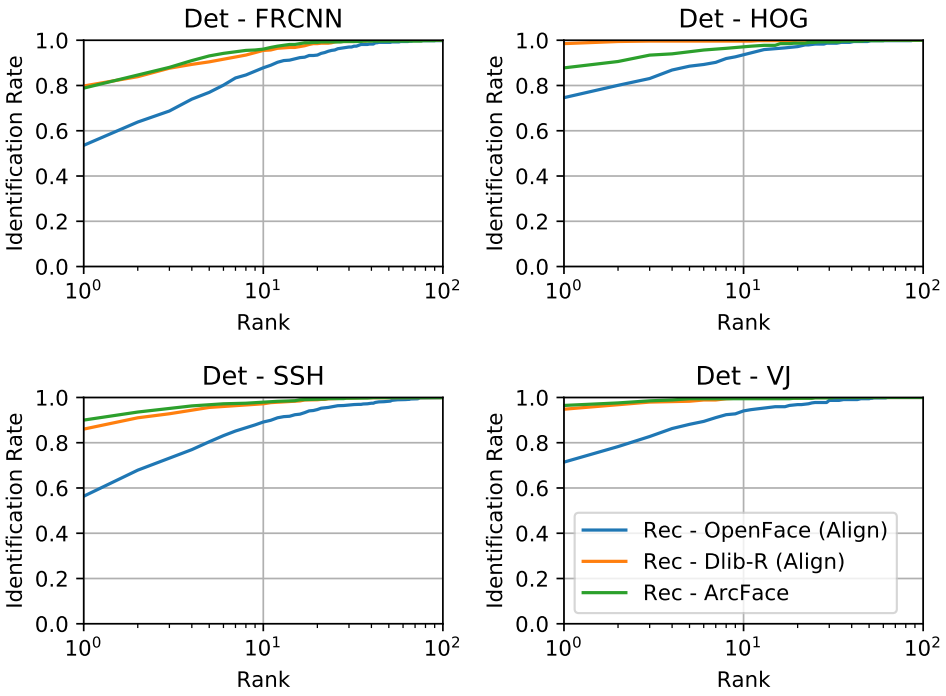


Figure 7.28: CMC curves obtained by the best versions of the 3 tested face recognition algorithms on the detections generated by the best versions of the 4 face detection algorithm working at identical recall level on the RAGI dataset.

When looking at the TPIR-FPIR curves, the relative performances are similar than with CMC curves. The difference in the smoothness of each of the curves comes from the absolute number of unknown probes detected by each algorithm. It differs between algorithms because the precision level is fixed but not the absolute number of false positive. The higher the recall, the more faces are detected, the more false positive, the more unknown faces and the smoother the curve. We can see that for VJ, recall is very low and we, therefore, have a very unsmooth curve that reaches a TPIR close to 1 as soon as the FPIR increases a little bit. In terms of performances over the three face recognition algorithms, it is followed by HOG, then FRCNN and finally SSH. It is interesting to see that, in this case, performances are better on FRCNN than on SSH. This might be explained by the fact that at this precision level, SSH reaches a much higher recall level implying that it has certainly much more complex faces to classify.

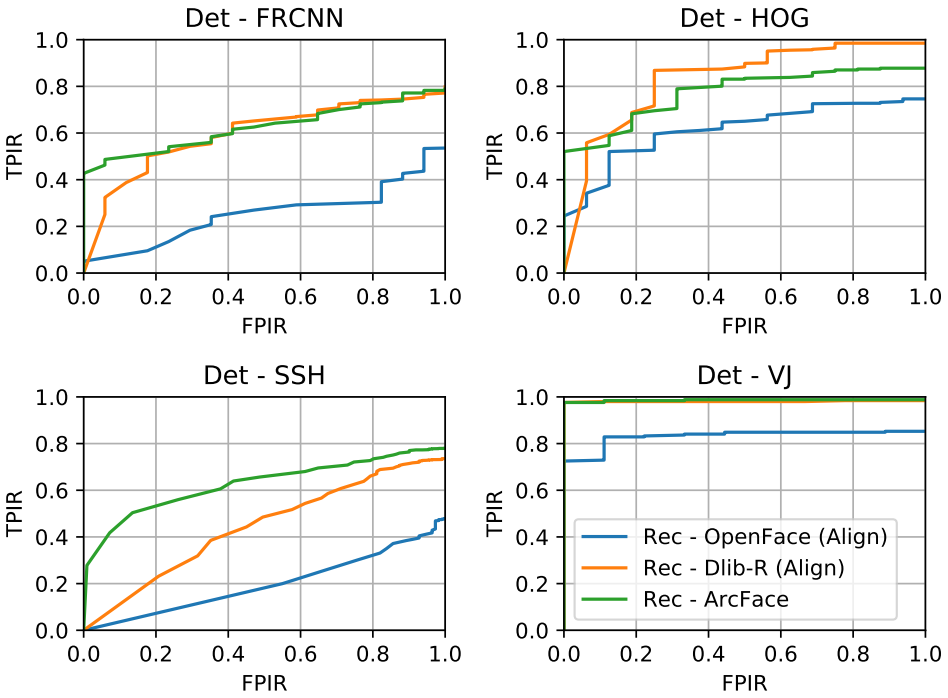


Figure 7.29: TPIR-FPIR curves obtained by the best versions of the 3 tested face recognition algorithms on the detections generated by the best versions of the 4 face detection algorithm working at identical precision level on the RAGI dataset.

Chapter 8

Conclusion

The goal of this master thesis was to provide a firm decision ground for the face recognition system of the RAGI project. Face recognition's main preprocessing step being face detection, I aimed for this goal through the application of intensive research and testing in both of those two tasks.

My work began with the analysis of open source benchmarks for detection and recognition, as reported in chapter 3. The development of those benchmarks is as essential as algorithms development. They support and foster the development of new, better-performing techniques by proposing more and more complex datasets. However, even if my analysis showed that there are some commonalities, there is still a lack of consensus on the protocols that should be used to evaluate algorithms on those datasets.

The goal of my work was two-fold: find algorithms to compare one to another and select face detection and face recognition benchmarks to test them on. Benchmarks define which algorithms are considered state-of-the-art and are thus a great source of research. However, comparing the performances of algorithms tested on different benchmarks is not an obvious task due to their evaluation divergences. The option I chose was therefore to select the best-performing algorithms from several benchmarks and to test them on two individual benchmark datasets, one for detection (i.e. WIDER FACE) and one for recognition (i.e. MegaFace).

In recent years, the complexity of benchmarks has evolved to better represent practical use cases, notably with the switch from controlled to 'in-the-wild' imagery while increasing in size. Nevertheless, this change in paradigm was not always accompanied by a high-level of annotation precision. While for face detection, datasets like WIDER FACE are annotated with information such as face pose, level of blur or even expression level, even though not always clearly determined, face recognition datasets do not provide such rich information yet. Moreover, when such information is available, benchmarks often lack handy evaluation tools to produce detailed results. However, a fine-grained analysis is essential to be able to estimate how well algorithms could perform on datasets with different underlying distributions. This observation fuelled the creation of a dataset that was specific to RAGI as explained in chapter 7.

The selection, in chapter 4, and description, in 5, of face detection and recognition algorithms was the second main step of this thesis. The selection was following two basic criteria: algorithms needed to be well enough documented and open source. Finding algorithms that respect them was not easy. The first obstacle came from the way algorithms are referenced

by benchmarks which is not always clear. Then, many, if not most, of the best algorithms are commercial products and their documentation is not accessible. Moreover, even when I could find a research paper describing precisely the algorithm, it did not guarantee that an open source code was provided. Sometimes, I could find an unofficial version of the algorithm but sometimes not. With this selection process, I found 13 face detection and 8 face recognition algorithms. Finally, before describing the algorithms, I tried installing and running them. This was the last barrier to testing the algorithms and it reduced the number of algorithms to 5 and 3 respectively. The cause of this contraction was the myriad of frameworks in which the algorithms were written and compatibility issues that come with them. This was also one of the reasons why I decided not to train algorithms by myself.

Algorithms were then tested as described in Chapter 6. Each algorithm was tested independently to select the best parameters. These tests led to the abandoning of one of the algorithms which was not reacting correctly. Then, even if I proceeded to various tests to see the influence of various parameters, the results obtained on WIDER FACE and MegaFace were quite consistent and designated the SSH algorithm as the winner for detection while in recognition Dlib-R and ArcFace were performing similarly, one level above OpenFace. The results for detection can be mitigated by the fact that SSH was also the slowest algorithm. Being a deep learning algorithm, this reflects the well-known compromise between speed and accuracy, with neural networks often favoring the latter. Nonetheless, increasing effort has been put into improving the former with some deep learning techniques reaching close-to-real-time performances.

To finish the thesis and obtain results that could be directly used in RAGI, the RAGI dataset was built as described in chapter 7. The construction of this dataset was based on the knowledge I acquired by studying other benchmarks and made me realize all the issues that arise when trying to propose a significant, non-biased benchmark. This dataset has the singularity that it can serve both as a face detection and recognition benchmark. While the rankings obtained on the two previous public datasets were confirmed, the dataset allowed me to analyze the influence of other parameters such as image resolution for detection or the distribution of faces in the gallery for recognition. For recognition, I went a bit further by testing practical concepts that could directly be applied to the RAGI project by including the per-scenario, YAOMI, and unknown tests. I ended this section by studying the combination of the face detection and face recognition algorithms in practical conditions. The new element that this analysis brought was that applying face recognition algorithms on top of face detections obtained with reduced recall may actually lead to better face recognition performances. The reason is that high recall in face detection leads to many low-quality images submitted to the face recognizer. While those parts of images do not contain sufficient information to allow for a reliable face recognition, clever face detectors are still able to spot them.

This last observation puts forward the fact that face recognition is far from being a solved problem. Even if the quality of benchmarks and algorithms has upsurged in the past few years, a lot of compromises have still to be made in a practical context. I hope this work will help the RAGI team in evaluating these compromises. Moreover, new benchmarks and algorithms are constantly appearing. There is still plenty of work to get through in order to optimize face recognition in RAGI. This is supported by the fact that techniques like face tracking, that can sometimes help to improve accuracy, were not studied in this thesis.

Studying face tracking in combination with face recognition would imply to use other bench-

marks like YTF but also to possibly upgrade the RAGI dataset. The time between each annotated frame being of nearly 3 seconds, annotating more frames could open new evaluation possibilities while also generating more reliable results. Indeed, the RAGI dataset being currently composed of 494 images with 3561 faces, it lags clearly behind publicly available datasets. More information could also be provided for each frame specifying some facial attributes for example. Extending the dataset would mean extending the set of identities it contains and could be an opportunity to undertake per-person analysis. Finally, I think this upgrading process should be continuous and harmonized with the day-to-day operations of the RAGI system.

Lastly, the results presented in this thesis needs to be used in the RAGI system. Even though I worked such that those results would be as close as possible to the expected results in this practical concept (e.g. using the same computer architecture, testing on a specific dataset), some additional work is needed. For instance, while studying the efficiency of detection and recognition algorithms separately, I did not make a computing time evaluation of the complete recognition system from frame acquisition to prediction. In addition, the computing architecture of RAGI might change in the future in order to improve efficiency. Studying the effect of different computing architectures on the computing time could be useful to make the best choices. Furthermore, I installed algorithms and wrote my tests with a purely analytic purpose in mind. The RAGI team will need to learn how to properly integrate these algorithms and tests in their system.

Appendix A

Dataset Samples

This chapter contains samples from each of the datasets that were used to test algorithms. For face detection, samples from WIDER FACE. For face recognition, samples are displayed for both the MegaFace dataset, whose images are used as distractors and the FaceScrub dataset, whose images are used as probes. Finally, for the RAGI dataset, examples of full frames, with ground truth bounding boxes, used for the face detection, evaluation and of faces, used for face recognition, are shown.

A.1 WIDER FACE

Figure A.1: Samples from the face detection benchmark WIDER FACE.

A.2 MegaFace

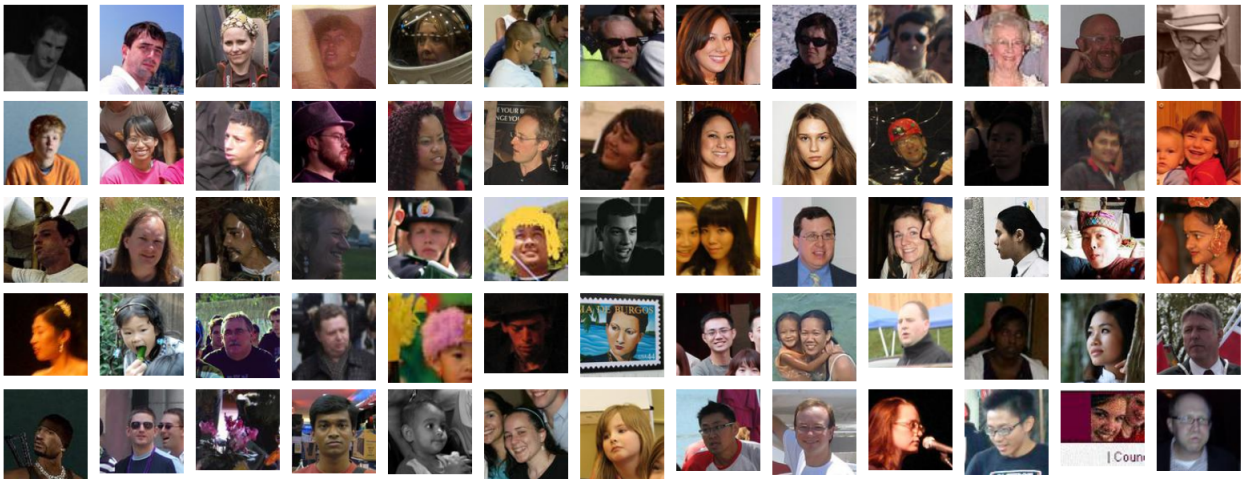


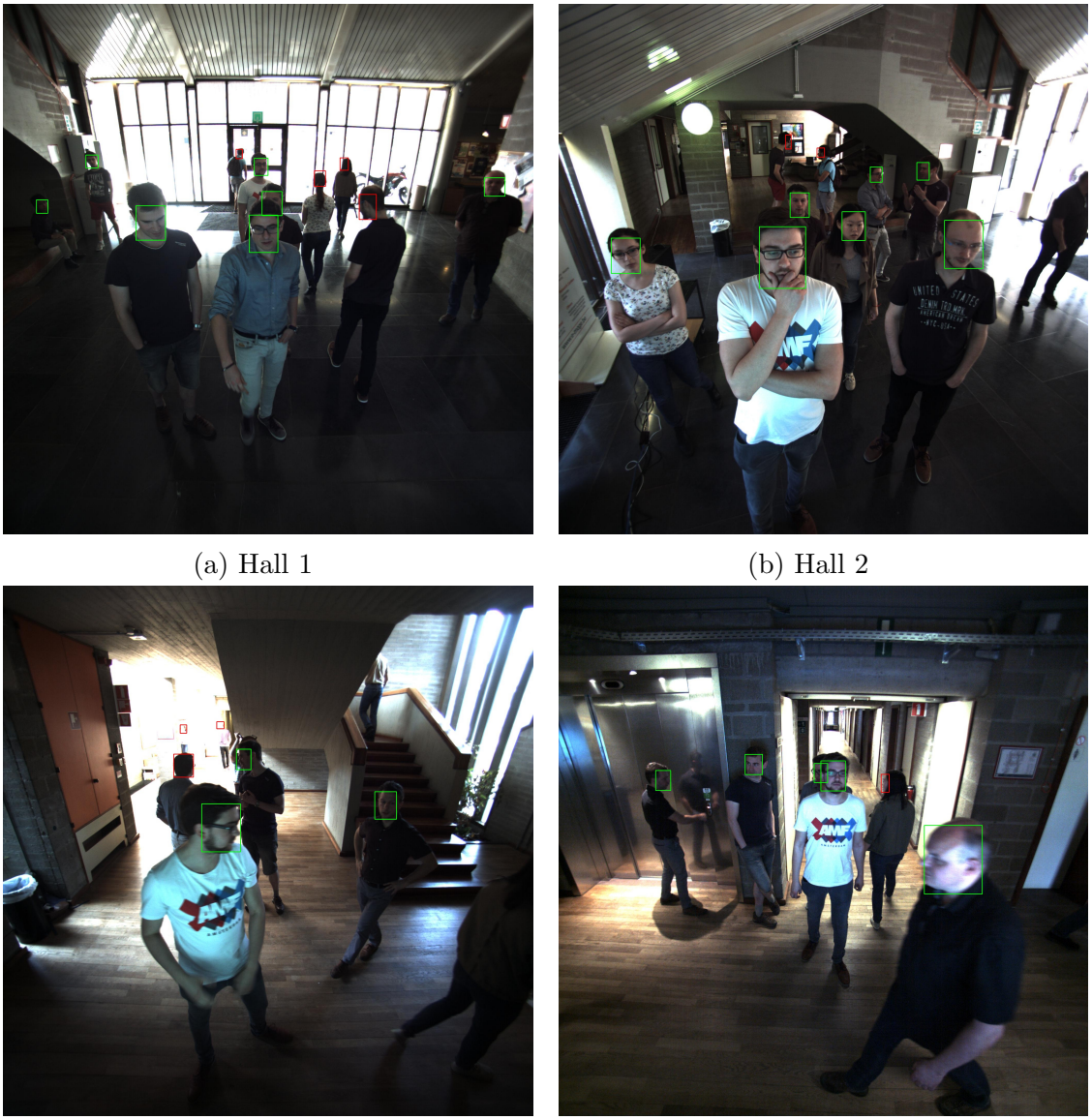
Figure A.2: Samples from the face recognition dataset MegaFace.

A.3 FaceScrub

Figure A.3: Samples from the face recognition dataset FaceScrub.

A.4 RAGI

A.4.1 Face detection



(a) Hall 1 (b) Hall 2

(c) R21 (d) R100

Figure A.4: Samples from the RAGI dataset for the face detection evaluation. One picture from each scenario is displayed. The green bounding boxes correspond to faces considered as recognizable while the red boxes correspond to non-recognizable faces or heads.

A.4.2 Face recognition

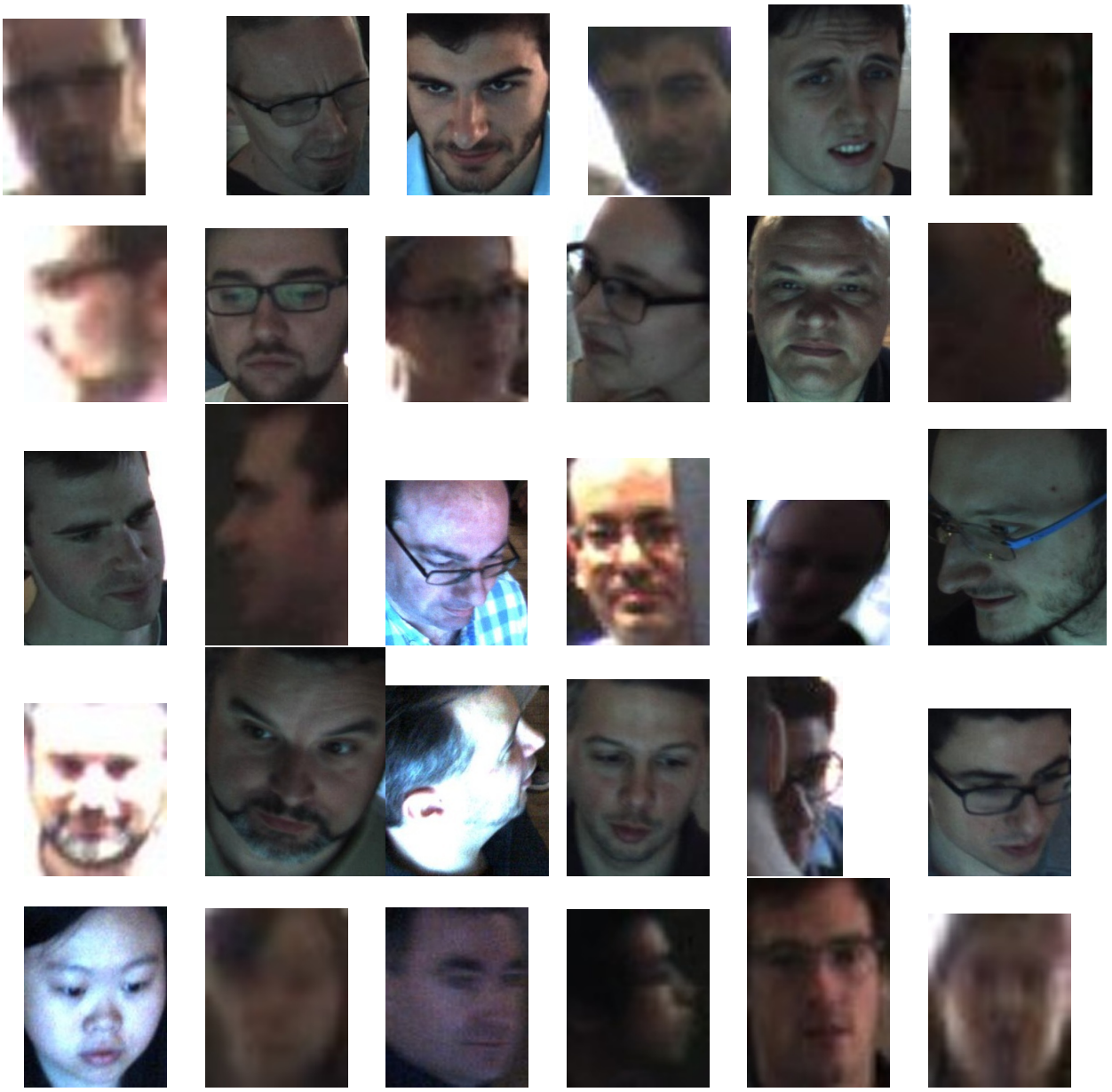


Figure A.5: Samples from the RAGI dataset for the face recognition evaluation. Two pictures are shown for each of the volunteers. One is a good quality picture while the other is affected by an attribute such as pose, blur, occlusion or illumination. The four last pictures correspond to unknown people.

Appendix B

Libraries versions

A series of coding language, libraries and APIs have been used throughout this thesis. I only use in Table B.1 the programs that I used, not the ones that I installed for algorithms that I could not manage to run.

Name	Version
OS	Ubuntu 16.04 - Linux Mint
C++	11
Python	2.7
Matlab	R2018a
OpenCV	3.4.1
Dlib	19.10
NVIDIA driver	390.25
CUDA	9.1
cuDNN	7.1

Table B.1: Version used for each of the coding language, libraries and APIs used in the thesis.

Bibliography

[1] The mplab genki database, genki-4k subset. 1. 12

[2] Face and gesture recognition working group: Fg-net aging database. 2000. 23, 25

[3] A. S. Abdallah, M. A. El-Nasr, and A. L. Abbott. A new color image database for benchmarking of automatic face detection and human skin segmentation techniques. In *Proceedings of World Academy of Science, Engineering and Technology*, volume 20, pages 353–357. Citeseer, 2007. 12

[4] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006. 6

[5] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications. *CMU School of Computer Science*, 2016. 8, 45

[6] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008. 6

[7] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720, 1997. 8

[8] T. Berg and P. Belhumeur. Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 955–962, 2013. 33

[9] T. Berg and P. N. Belhumeur. Tom-vs-pete classifiers and identity-preserving alignment for face verification. In *BMVC*, volume 2, page 7. Citeseer, 2012. 33

[10] T. L. Berg, A. C. Berg, J. Edwards, M. Maire, R. White, Y.-W. Teh, E. Learned-Miller, and D. A. Forsyth. Names and faces in the news. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE. 12, 20

[11] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision*, pages 354–370. Springer, 2016. 31

[12] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman. Vggface2: A dataset for recognising faces across pose and age. *arXiv preprint arXiv:1710.08092*, 2017. 75

[13] X. Cao, D. Wipf, F. Wen, G. Duan, and J. Sun. A practical transfer learning algorithm for face verification. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3208–3215. IEEE, 2013. 33

[14] B.-C. Chen, C.-S. Chen, and W. H. Hsu. Cross-age reference coding for age-invariant face recognition and retrieval. In *European Conference on Computer Vision*, pages 768–783. Springer, 2014. 23

[15] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3025–3032. IEEE, 2013. 33

[16] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun. Joint cascade face detection and alignment. In *European Conference on Computer Vision*, pages 109–122. Springer, 2014. 30

[17] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. In *Advances in Neural Information Processing Systems*, pages 4470–4478, 2017. 47

[18] A. Colombo, C. Cusano, and R. Schettini. 3d face detection using curvature analysis. *Pattern Recognition*, 39(3):444 – 455, 2006. 6

[19] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005. 6, 32, 37

[20] K. Davis. High quality face recognition with deep metric learning. 2017. i, 33

[21] J. Deng, J. Guo, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *arXiv preprint arXiv:1801.07698*, 2018. i, 33, 46

[22] J. Deng, Y. Zhou, and S. Zafeiriou. Marginal loss for deep face recognition. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPRW), Faces "in-the-wild" Workshop/Challenge*, volume 4, 2017. 47

[23] C. Ding and D. Tao. Robust face recognition via multimodal deep face representation. *IEEE Transactions on Multimedia*, 17(11):2049–2058, 2015. 33

[24] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int J Comput Vis*, 88:303–338, 2010. 3, 4, 5, 14, 15, 16

[25] S. S. Farfade, M. J. Saberian, and L.-J. Li. Multi-view face detection using deep convolutional neural networks. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 643–650. ACM, 2015. 30

[26] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 6, 39

[27] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. 35

[28] R. Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. 39, 40

[29] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 6, 38, 39, 41

[30] R. Gross, I. Matthews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. *Image and Vision Computing*, 28(5):807–813, 2010. 9

[31] P. J. Grother and M. L. Ngan. Face recognition vendor test (frvt) performance of face identification algorithms nist ir 8009. Technical report, 2014. 7, 25

[32] C. Gu, J. J. Lim, P. Arbeláez, and J. Malik. Recognition using regions. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1030–1037. IEEE, 2009. 38

[33] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. MS-Celeb-1M: A dataset and benchmark for large scale face recognition. In *European Conference on Computer Vision*. Springer, 2016. 75

[34] Z. Guo, Y.-N. Zhang, Y. Xia, Z.-G. Lin, Y.-Y. Fan, and D. D. Feng. Multi-pose 3d face recognition based on 2d sparse representation. *Journal of Visual Communication and Image Representation*, 24(2):117 – 126, 2013. Sparse Representations for Image and Video Analysis. 6, 8

[35] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *european conference on computer vision*, pages 346–361. Springer, 2014. 39, 40

[36] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 46

[37] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933. 8

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 47

[39] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017. 47

[40] P. Hu and D. Ramanan. Finding tiny faces. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1522–1530. IEEE, 2017. 31

[41] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017. 47

[42] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. 9, 19

[43] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. 3, 4, 5, 12, 13, 14

[44] H. Jiang and E. Learned-Miller. Face detection with the faster r-cnn. In *Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on*, pages 650–657. IEEE, 2017. 6

[45] T. Kanade. Picture processing system by computer complex and recognition of human faces. 1974. 8

[46] V. Kazemi and S. Josephine. One millisecond face alignment with an ensemble of regression trees. In *27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, United States, 23 June 2014 through 28 June 2014*, pages 1867–1874. IEEE Computer Society, 2014. 46, 72

[47] I. Kemelmacher-Shlizerman, S. M. Seitz, D. Miller, and E. Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4873–4882, 2016. i, 7, 24, 71

[48] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain. Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1931–1939, 2015. 17, 25

[49] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 2144–2151. IEEE, 2011. 3, 13

[50] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 365–372. IEEE, 2009. 21

[51] G. P. Kusuma and C.-S. Chua. Pca-based image recombination for multimodal 2d+3d face recognition. *Image and Vision Computing*, 29(5):306 – 316, 2011. 6, 8

[52] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997. 8

[53] G. B. H. E. Learned-Miller. Labeled faces in the wild: Updates and new reporting procedures. Technical Report UM-CS-2014-003, University of Massachusetts, Amherst, May 2014. 19

[54] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5325–5334, 2015. 6

[55] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017. 43

[56] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 42

[57] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. *arXiv preprint arXiv:1704.08063*, 2017. 6, 33

[58] W. Liu, Y. Wen, Z. Yu, and M. Yang. Large-margin softmax loss for convolutional neural networks. In *ICML*, pages 507–516, 2016. 47

[59] Y. Liu, H. Li, J. Yan, F. Wei, X. Wang, and X. Tang. Recurrent scale approximation for object detection in cnn. In *IEEE International Conference on Computer Vision*, 2017. 29

[60] A. C. Loui, C. N. Judice, and S. Liu. An image database for benchmarking of automatic face detection and recognition algorithms. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, volume 1, pages 146–150. IEEE, 1998. 12

[61] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999. 38

[62] I. Masi, A. T. Tran, T. Hassner, J. T. Leksut, and G. Medioni. Do we really need to collect millions of faces for effective face recognition? In *European Conference on Computer Vision*, pages 579–596. Springer, 2016. 33

[63] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *European Conference on Computer Vision*, pages 720–735. Springer, 2014. 24, 30

[64] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 8

[65] M. Najibi, P. Samangouei, R. Chellappa, and L. Davis. Ssh: Single stage headless face detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4875–4884, 2017. i, 31, 43, 44, 61

[66] M. Naphade, J. R. Smith, J. Tesic, S.-F. Chang, W. Hsu, L. Kennedy, A. Hauptmann, and J. Curtis. Large-scale concept ontology for multimedia. *IEEE multimedia*, 13(3):86–91, 2006. 16

[67] H.-W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 343–347. IEEE, 2014. 9, 25

[68] O. M. Parkhi, A. Vedaldi, A. Zisserman, et al. Deep face recognition. In *BMVC*, volume 1, page 6, 2015. 46

[69] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 8

[70] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 42

[71] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 38, 40, 41

[72] K. Ricanek and T. Tesafaye. Morph: A longitudinal image database of normal adult age-progression. In *Automatic Face and Gesture Recognition, 2006. FGR 2006. 7th International Conference on*, pages 341–345. IEEE, 2006. 23

[73] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 746–751. IEEE, 2000. 12

[74] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. 8, 33, 45, 47

[75] M. P. Segundo, L. Silva, O. Bellon, and S. Sarkar. Orthogonal projection images for 3d face detection. *Pattern Recognition Letters*, 50:72 – 81, 2014. Depth Image Analysis. 6

[76] S. Sengupta, J.-C. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs. Frontal to profile face verification in the wild. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016. 22

[77] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 42

[78] P. Sharma and R. B. Reilly. A colour face image database for benchmarking of automatic face detection algorithms. In *Video/Image Processing and Multimedia Communications, 2003. 4th EURASIP Conference focused on*, volume 1, pages 423–428. IEEE, 2003. 12

[79] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 57, 59, 62

[80] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524, 1987. 8

[81] T.-H. Sun and F.-C. Tien. Using backpropagation neural network for face recognition with 2d+3d hybrid information. *Expert Systems with Applications*, 35(1):361 – 372, 2008. 8

[82] X. Sun, P. Wu, and S. C. Hoi. Face detection using deep learning: An improved faster rcnn approach. *arXiv preprint arXiv:1701.08289*, 2017. 29

[83] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1891–1898, 2014. 33

[84] Y. Sun, X. Wang, and X. Tang. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1891–1898, 2014. 47

[85] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, volume 4, page 12, 2017. 47

[86] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015. 44, 45

[87] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014. 10, 33, 45

[88] X. Tang, D. K. Du, Z. He, and J. Liu. Pyramidbox: A context-assisted single shot face detector. *arXiv preprint arXiv:1803.07737*, 2018. 29, 31

[89] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. The new data and new challenges in multimedia research. 24

[90] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991. 8

[91] R. Vaillant, C. Monrocq, and Y. Le Cun. Original approach for the localisation of objects in images. *IEE Proceedings-Vision, Image and Signal Processing*, 141(4):245–250, 1994. 6

[92] F. Van Lishout, A. Dubois, M. L. Wang, T. Ewbank, and L. Wehenkel. Integrating facial detection and recognition algorithms into real-life applications. Montefiore Institute - department of EE & CS, University of Liège, Belgium, 2018. Workshop on the Architecture of Smart Cameras - WASC 2018. 2

[93] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001. 35

[94] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004. 2, 6, 17, 20, 22, 25, 30, 32, 35, 36, 37

[95] S. Wan, Z. Chen, T. Zhang, B. Zhang, and K.-k. Wong. Bootstrapping face detection with hard negative examples. *arXiv preprint arXiv:1608.02236*, 2016. 29

[96] F. Wang, W. Liu, H. Liu, and J. Cheng. Additive margin softmax for face verification. *arXiv preprint arXiv:1801.05599*, 2018. 47

[97] H. Wang, Z. Li, X. Ji, and Y. Wang. Face r-cnn. *arXiv preprint arXiv:1706.01061*, 2017. 38

[98] J. Wang, Y. Yuan, and G. Yu. Face attention network: An effective face detector for the occluded faces. *arXiv preprint arXiv:1711.07246*, 2017. 31

[99] Y. Wang, X. Ji, Z. Zhou, H. Wang, and Z. Li. Detecting faces using region-based fully convolutional networks. *arXiv preprint arXiv:1709.05256*, 2017. 31

[100] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016. 47

[101] C. Whitelam, E. Taborsky, A. Blanton, B. Maze, J. Adams, T. Miller, N. Kalka, A. K. Jain, J. A. Duncan, K. Allen, et al. Iarpa janus benchmark-b face dataset. In *CVPR Workshop on Biometrics*, 2017. 17, 18, 25

[102] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 529–534. IEEE, 2011. 21

[103] W.-S. T. WST. Deeply learned face representations are sparse, selective, and robust. *perception*, 31:411–438, 2008. 45

[104] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 8

[105] Y. Xiong, K. Zhu, D. Lin, and X. Tang. Recognize complex events from static images by fusing deep channels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1600–1609, 2015. 16

[106] J. Yan, X. Zhang, Z. Lei, and S. Z. Li. Face detection by structural models. *Image and Vision Computing*, 32(10):790–799, 2014. 3, 14

[107] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Aggregate channel features for multi-view face detection. In *Biometrics (IJCB), 2014 IEEE International Joint Conference on*, pages 1–8. IEEE, 2014. 30

[108] B. Yang, J. Yan, Z. Lei, and S. Z. Li. Fine-grained evaluation on face detection in the wild. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, volume 1, pages 1–7. IEEE, 2015. 3, 5, 15, 16

[109] M.-H. Yang, D. J. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Transactions on pattern analysis and machine intelligence*, 24(1):34–58, 2002. 3

[110] S. Yang, P. Luo, C.-C. Loy, and X. Tang. From facial parts responses to face detection: A deep learning approach. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3676–3684, 2015. 30

[111] S. Yang, P. Luo, C.-C. Loy, and X. Tang. Wider face: A face detection benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5525–5533, 2016. i, 3, 16

[112] S. Yang, Y. Xiong, C. C. Loy, and X. Tang. Face detection through scale-friendly deep convolutional networks. *arXiv preprint arXiv:1706.02863*, 2017. 29

[113] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014. 46

[114] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 45

[115] K. Zhang, Z. Zhang, H. Wang, Z. Li, Y. Qiao, and W. Liu. Detecting faces using inside cascaded contextual cnn. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3171–3179, 2017. 29

[116] N. Zhang, M. Paluri, Y. Taigman, R. Fergus, and L. Bourdev. Beyond frontal faces: Improving person recognition using multiple cues. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4804–4813, 2015. 23

[117] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. S^3fd: Single shot scale-invariant face detector. *CoRR*, abs/1708.05237, 2017. 29, 31, 42, 43

[118] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao. Range loss for deep face recognition with long-tail. *arXiv preprint arXiv:1611.08976*, 2016. 47

[119] C. Zhu, R. Tao, K. Luu, and M. Savvides. Seeing small faces from robust anchor's perspective. *arXiv preprint arXiv:1802.09058*, 2018. 31

[120] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE, 2012. 3, 14

[121] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014. 16