International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor (2018): 7.426

Performance Comparison of Face Detection and Recognition Algorithms

Mohsin Furkh Dar, Dr. Sarvottam Dixit

Faculty of Science and Technology, Mewar University, Chittorgarh, Rajasthan, India

Abstract: ***Recognition of faces is a fundamental cognitive ability that forms an important basis for our social interactions. This paper aims to optimize the existing face recognition system by comparing the results of different algorithms. To achieve this goal, I have analyzed 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 [1] and MegaFace [2]) are also used for evaluating those algorithms. The results on these benchmarks allows to determine which algorithms perform better, that is to say SSH [3] for detection and both Dlib-R [4] and ArcFace [5] for recognition. All the 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.***

Keywords: SSH, Dlib-R

1. Introduction

The area of face recognition has strongly motivated many researchers to find the challenges in image processing systems. During this process of facial image recognition, the researchers pored on the factors which can handle human faces in a better way under different conditions like illumination, pose, lightning effects. Face detection got popular in the year 2000 when Paul Viola and Michael Jones devised a method to detect faces. Their method was fast enough to run on cheap cameras. However, today much more reliable solutions exist now.

Face Detection

As presented in [1], "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 [6]". 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.

To define when we consider a detection to be correct, we must beforehand decide how a detection is represented. As reported in [7] and illustrated in Figure1, 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 [8]". Most of the recent datasets (e.g. [1], [9], [8], [10], [11], [12]) 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 paper.

Figure 1: Illustrative figure from [7]. 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

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 [9], "the overlap ratio between a predicted bounding box B_p and ground truth bounding box B_{gt} is given by

$$IOU = area\ (B_p \cap B_{gt})/area\ (B_p \cup B_{gt})$$

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 paper, a detection is considered 'correct' when this value is greater than 50%. This choice was generally made following the PASCAL VOC [9] protocol.

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). 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).

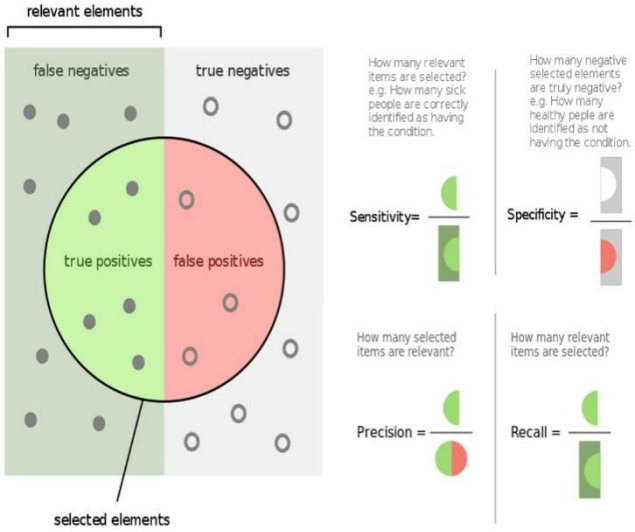


Figure 2: Combination of images coming from the Wikipedia pages 'Sensitivity and Specificity' and 'Precision and Recall' showing how the different binary classification measures are obtained

This is nearly never the case and one has to make a compromise between those two values. To quantify this compromise, the benchmarks propose generally two approaches. The first one is to use adapted Receiver Operating Characteristic (ROC) curves. The second approach is based on precision-recall curves where precision is computed as TP/(TP+FP). The authors of [9] 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.

Face Recognition

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." [13] In our case, we are typically trying to solve a face identification problem under the open-set protocol. Indeed, our goal 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.

There are several ways to evaluate the quality of face recognition algorithms. 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 [2]. A more detailed approach of the quantification of the performances of face recognition algorithms is given in the Face Recognition Vendor Test (FRVT) [14].

The goal of this work was to search for and evaluate state-of-the-art algorithms in both face detection and recognition in order to take informed decisions regarding the implementation choices in 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 paper is organized as follows. Section I contains the introduction of face detection and face recognition problem and the purpose of my work, Section II contain the related work of both face detection and face recognition, Section III explain the methodology and terminology used for both facedetection and face recognition, Section IV describes

results and discussion of this work, and Section V concludes research work with future directions.

2. Related Work

Face detection

Anyone who has some knowledge in the field of face detection would have heard of the Viola & Jones Haar-cascade algorithm [15] which is often considered as the first practical face detector. As mentioned in [16], "the use of hand-designed features methods continued with for instance SURF [17], LBP [18] or HOG [19]. These features were combined with Deformable Parts Model [20] to produce significant advances." However, the real burst in performance came with the renewed use of neural networks based on deep architectures. According to [16], "CNN's had already been applied to face detection as far back as 1994 in [21]" but since 2012, deep architectures started being used such as in [22] or [23].

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 [24], 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 [25] by doing curvature analysis or by combining the two such as in [24]. [26] 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" [27]. 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.

Face Recognition

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 [28], "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 [29]. 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) [30] represent faces as a combination of eigenvectors [31]. Eigenfaces [32] and fisher faces [33] are landmark techniques in PCA-based face recognition. Lawrence et al. [34] 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 [35]. Finally, as for face detection, research has also turned to 3D imagery. Techniques also vary between combining 2D and 3D ([36], [37]), transforming 3D data to 2D data ([27]) or extracting features directly from 3D data ([38], [39], [40]). 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.

3. Methodology

Face detection

The choice for selecting the testing benchmark for face detection was made keeping in mind the goals of thesis. 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 for 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 terms of size and has the advantage of furnishing evaluation tools. So, based on the elements analyzed, we decided to use WIDER FACE as a testing benchmark.

1) Testing Approach used by the evaluation toolbox of WIDER FACE

For evaluation and plotting the tool box of WIDER FACE comes with a series of functions coded in MATLAB. 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. The total number of faces to detect in each subset is 7211, 13319 and 31958 for the easy, medium and hard subset respectively.

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.

2) Time Evaluation

The important part of the project was to find if state-of-the-art algorithms can work in a practical environment with a given amount of computing power. 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. To observe the influence of size on the computation time, detection of each frame was measured individually. It also enables us to take 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.

Both the real and CPU time was measured. The real time gives an idea of the practical capabilities of an algorithm whereas the CPU times allows a fairer comparison of those capabilities across different computing architectures. 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. To analyze these results in more detail, htop command was used to keep the information output, which indicates memory and CPU usage. Also, nvidia-smi command is used to provide data about CPU usage.

The algorithms written in Python 2.7 can easily have access to both CPU and real time using functions clock and time library respectively. In C++, we can access only CPU time through clock function of the ctime library.

Face recognition

The tests of face recognition algorithms are carried out on the MegaFace benchmark. The reason for choosing MegaFace benchmark is that it provides precise annotations. These annotations are done automatically and also provides full evaluation tools. IJB-C possess similar qualities but its size is prohibitive (>325GB) as a comparison, MegaFace is only 64GB.

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.

Finally, 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.

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.

The CPU and real timeplus CPU and GPU usage is measured using the same functions and commands as in face detection.

4. Results and Discussion

This section contains a series of results obtained through the evaluation of the face detection and face recognition algorithms that were selected in the previous section.

Face Detection

The overall result of the different face detection algorithms is summarized in figure 3 and table 1. In each case we took the best version of each algorithm.

Table 1: Number of frames that can be processed per second, in CPU and Real Time with the best versions of four tested algorithms, from WIDER FACE.

Algorithm	CPU FPS	Real FPS
VJ-python-scale-1.3-min-10	1.98	34.56
HOG-C++	14.45	/
FRCNN-nms-0.3	7.25	14.96
SSH-nms-0.5	3.21	5.23

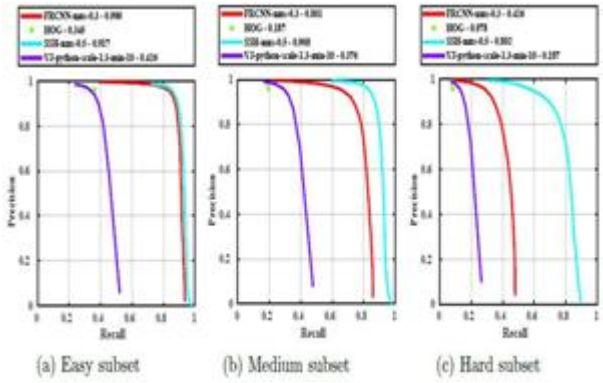


Figure 3: Precision-recall curves obtained by the best versions of the 4 tested algorithms on the three subsets of WIDER FACE.

Clearly SSH performs better in terms of precision-recall performance. On easy and medium set FRCNN produces similar results but fails on the hardest datasets. On the other hand, VJ and HOG perform equally but one level below FRCNN and SSH. Now in terms of computation time, for real-time computation better the algorithm the smaller the number of frames per second. FRCNN is more than two times faster than SSH because SSH makes a much heavier use of Graphics Processing Unit (GPU) when running. More precisely FRCNN uses only around 2000 MB of GPU memory whereasSSH uses nearly 4500 MB. Hence SSH needs greater computation time.

Finally, Figure 4 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.

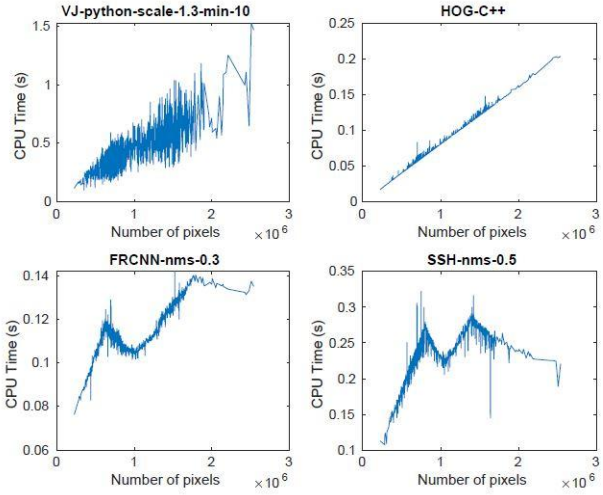
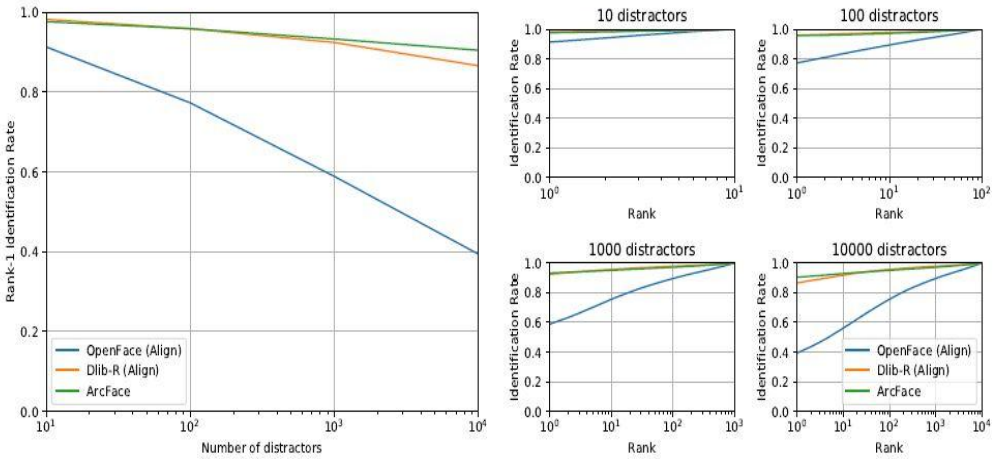


Figure 4: Development of CPU computation time per image with the number of pixels per image for the 4 tested algorithms.

Face Recognition

The face recognition algorithms selected were tested based on MegaFace benchmark. MegaFace allows testing identification based on two probe sets. FaceScrub and FGNet. The later one is used to evaluate face recognition across age variation. So here we are only taking FaceScrub dataset into consideration. Figure 5 shows the graphs comparing the best versions of each face recognition algorithm we tested. The main remark that can be made is that Dlib-R and ArcFace are one level above OpenFace. The two former algorithms 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.



(a) Rank-1 identification rate vs number of distractors (b) CMC curves for varying number of distractors

Figure 5: 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

However, if we look in terms of time, Table 2 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.

Table 2: 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

Algorithm	CPU FaPS	Real FaPS
OpenFace (Align)	27.98	23.56
Dlib-R (Align)	238.84	/
ArcFace	46.54	58.68

As far as computational time is concerned, for classification step time only depends on the size of the feature representation. As shown in table 3, Dlib-R and OpenFace generate the same number of features per face while InsightFace generates four times more.

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

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

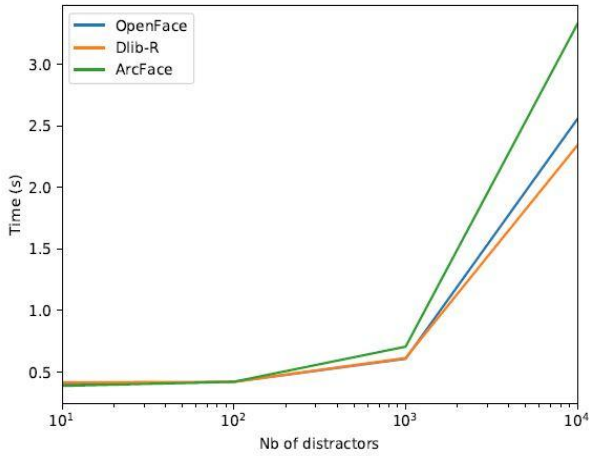


Figure 6: Evolution of classification time (expressed in seconds) with the number of distractors for the 3 tested face recognition algorithms

Figure 6 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. 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.

5. Conclusion and Future Scope

In this paper I presented a way to search for and evaluate state-of-the-art algorithms in both face detection and recognition in order to allow any team to take informed decisions regarding the implementation choices of their computer vision system. 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 choose 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.

The selection and description of face detection and recognition algorithms was the second main step of my work. The selection was following two basic criteria: algorithms needed to be well enough documented and open source.

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. 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 up surged in the past few years, a lot of compromises have still to be made in a practical context. Moreover, new benchmarks and algorithms are constantly appearing. There is still plenty of work to get through in order to optimize face recognition. This is supported by the fact that techniques like face tracking, that can sometimes help to improve accuracy. 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. 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.

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 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.

References

[1] 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
[2] 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
[3] 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
[4] K. Davis. High quality face recognition with deep metric learning. 2017. i, 33
[5] 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
[6] 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
[7] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. 3, 4, 5, 12, 13, 14
[8] 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
[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

[14] 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

[15] 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

[16] 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

[17] 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

[18] T. Ahonen, A. Hadid, and 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

[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] 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

[21] 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

[22] 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, pages 580-587, 2014. 6,38,39,41

[23] 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

[24] 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

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

[26] 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

[27] 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

[28] 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

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

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

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

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

[33] 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

[34] 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

[35] 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

[36] 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

[37] 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

[38] 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

[39] 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

[40] 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.