Face Recognition from low-res images with Super-Resolution

Biometric Systems Project 2021/2022

Matteo Orsini
1795119

Fabrizio Rossi
1815023

Abstract - *In this project we explore super-resolution techniques in order to enhance images of faces acquired by a camera with a very low resolution or from a long distance. Our hypothesis is that increasing the images resolution, we can leverage more information and build a model which can perform better in the face recognition task w.r.t. a model which uses the low resolution images. We describe and evaluate several methods to perform the upscaling and compare them with a base model without using super-resolution. Moreover, we propose and test two models, which are Generative Adversarial Networks (GANs), able to perform upscaling from images with a resolution lower than the one used by the most popular state-of-the-art models.*

1 Introduction

In many real case scenarios, like video surveillance, face recognition systems are not provided with images captured in perfect conditions, but instead they are often deployed in uncontrolled environments, posing an additional challenge to the already hard task of face recognition. In fact, in many cases, only low resolution cameras are available to use, or even with high resolution cameras, it could happen that there is the need to recognize a person which is distant from the position where the capturing device is located. In these two settings, we obtain an image which is lacking the valuable information provided by high quality resolution images, making the recognition task harder.

For this reason, in our work we decided to try to perform an open set identification task using a super-resolution image derived from the low resolution image captured, and to check if we could obtain better performances than using a traditional upscaling method like bilinear interpolation.

In order to train and evaluate our proposed system, we used two datasets: Labeled Faces in the Wild (LFW) [1] and CASIA-WebFace [2]. Both datasets contain faces of celebrities or famous people captured in uncontrolled conditions, with varying pose, lighting, and even age. A sample of the two datasets can be seen in Fig. 2.

2 System Architecture

The entire system is composed by three modules: a face localization, a super-resolution and a face recognition module. While these modules are the core of our architecture, we still needed to set up a pipeline in order to process the images correctly and use them in our models.

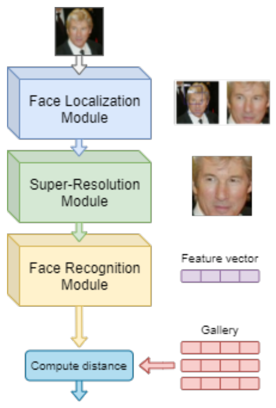

Fig. 1: Pipeline of the whole system

The various steps of the pipeline, illustrated in Fig. 1, are the following:

1. Download and extract the two datasets;
2. Setup a module in order to perform face localization, cropping the regions of the images containing a face, and saving them onto the disk as a new dataset;
3. Setup and use our super-resolution model in order to upscale the cropped images from 32×32 to 128×128;
4. Feed the face recognition module with the super-resolution images in order to produce a feature vector encoding the features of the face image;
5. Evaluate the performance of our system computing the distance between the feature vectors.

The technical details and challenges encountered in all these steps are going to be illustrated in the next sections. The entire system has been implemented in Python with the TensorFlow framework, using the Google Colab platform. An interactive notebook is available here.

(a) Casia-WebFace

(b) Labeled Faces in the Wild (LFW)

Fig. 2: Some samples extracted from the datasets

3 Face Localization

The first real module of our architecture is used in order to extract faces from the images of the datasets. In our implementation we tried to use two different techniques: we first used the Haar Cascade classifier [3] implemented in OpenCV following the Viola-Jones algorithm, and then we also tried the Multi-Task Cascaded Convolutional Neural Network (MTCNN) [4] model.

The first one is a simple model which uses a variant of the AdaBoost approach in combination with Haar features in order to train a classifier able to detect human faces from frontal images. The Haar features are computed using integral images in order to speed up the process. The learning procedure leverages the idea of boosting, so at the end we obtain a strong classifier whose output is a weighted sum of the outputs of the weak classifiers according to their importance. Each weak classifier is trained according to sample weights proportional to the difficulty of a sample w.r.t. the task, improving each time the performance of the previous classifier, focusing on particular features. Once the classifier is trained, a sliding window processes the image in order to detect the regions with the highest scores obtained by the algorithm.

The MTCNN, instead, is trained trying to combine the two tasks of face detection and face alignment, hence the multitasking adjective. In order to improve its efficiency, it is composed by multiple CNNs used in cascade. First of all an image pyramid is created in order to consider the images at different scales. Then those images are passed to the first fully convolutional CNN which is going to output bounding boxes of potential window candidates. Then, a Non-Maximum Suppression (NMS) pass is used to remove highly overlapped regions. The remaining candidates are fed to another CNN that removes further candidates and performs again NMS. In the last stage, candidates are analyzed by a final CNN which this time is trained in a supervised way in order to produce in output facial landmarks.

In our work we compared these two methods to check which one would be the most suited for our needs, since we also have limited computational capabilities. In order to do that, we took in consideration both the qualitative results obtained and the processing speed of the two methods. The results obtained by the Haar Cascade Classifier and MTCNN are comparable, while we measured that the time required in order to process and extract faces from our dataset is much less using the first one. For this reason, at the end we decided to opt for the faster method since we don't lose too much in accuracy and we can save precious processing time.

However, it needs to be mentioned that, as we can see from Fig. 3, the MTCNN seems to be more robust since the Cascade classifier detects much more false positives.

While using the Haar Cascade Classifier, we also noticed that there are multiple problems of detection with low quality images, cluttered background, multiple faces in one image, and profile faces. Sometimes, we have that the algorithm finds more than one face and for this reason we opted to only extract the face with the highest confidence score obtained by the model. Instead, since the classifier is not able to detect profile faces, we decided to simply ignore images where a face couldn't be detected.

We would also like to mention that this choice could have affected the results of the training phase of the models that will be presented in the following sections, since while extracting faces from our datasets, we might have introduced some noise in them in the form of images that are not actually representing human faces but instead some random part of the image. This resulted into making the learning process slightly harder and it has also lowered the metrics, since the models are not going to perform well on these images wrongly cropped present in the validation data.

gallery, and output a final ranking with their score.

In our implementation for the face recognition module we used a deep learning model, which given an image of an enrolled subject will predict the identity of that subject. Then, we can simply discard the last layer of the model and obtain the vector representation.

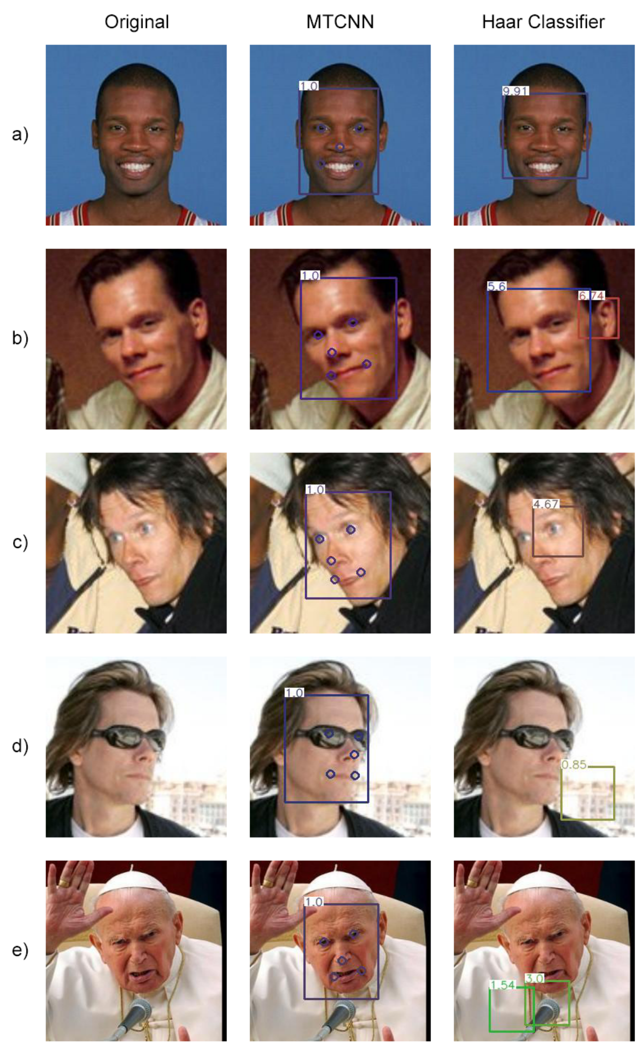

Fig. 3: Comparison of the two face localization methods. The bounding boxes show the results of the model, while the number is the score assigned by the model. In addition, for the MTCNN model, also detected face landmarks are shown. In a) we can see that the two models have comparable performances when dealing with frontal images without face occlusions, which is the most common case. In b) instead we can see that the Haar classifier correctly identified the face, but the ear has a higher confidence score. In c) we can see that since the face is a profile image, the detection is not good and instead an eye is detected. In d) we can see that accessories like glasses can throw the Haar classifier off, which recognizes a part of a building. Finally, in e) we can see that even with some frontal images without occlusions, the Haar classifier can have some difficulties recognizing the face.

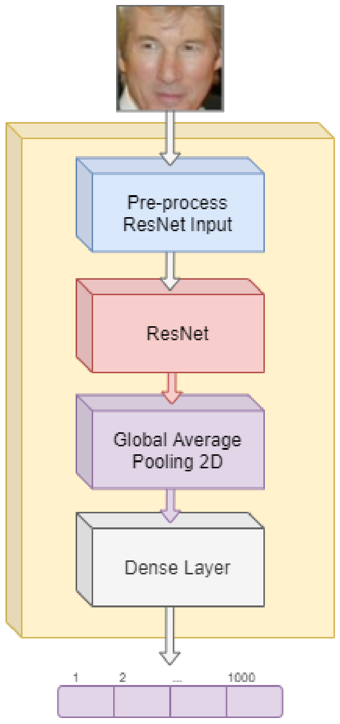

Fig. 4: Architecture of our face recognition module.

4 Face Recognition

The face recognition module is the one in charge of extracting a template from the face image, in the form of a feature vector. This module will be used to generate the templates of both gallery images and the ones of incoming probes at test time. Then, those templates will be used to compute the cosine similarity between the probes and the

In Fig. 4 is shown the architecture of our model. To train our model, since we didn't have dedicated hardware at disposal, we decided to reduce the dataset, so we took the first 1.000 identities of the CASIA-WebFace dataset, resulting in around 73.000 images of size 128×128. We decided to use 80% of the reduced dataset as training set and the remaining 20% as validation set to perform hyper-parameter tuning.

As we can see from the architecture, we used the ResNet50 model, which is very popular and it has shown very good results in similar tasks. The ResNet50 architecture is a deep neural network composed by 50 layers that leverages the concept of skip connections in order to fight the vanishing and exploding gradients problems. The skip connections, shown in Fig. 5, consist in linking a convolutional layer output to the input of another layer deeper in the network using an addition between the two. This improves the training efficiency since the gradients can backpropagate through the network faster using these skips, improving the performance of the network.

Moreover, we used the fine-tuning approach in order to improve the performances and reduce the training time. This technique consist in using the weights of a pre-trained model and re-train it for a new task, generally removing the last layers in order to adapt the model to the new task. In our

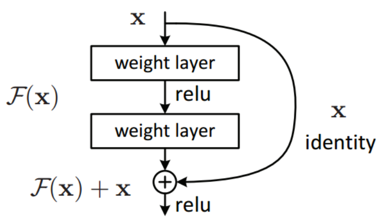
x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 5: A residual block of the ResNet architecture.

case we used the weights of the ResNet50 model which were previously optimized for the ImageNet dataset in order to perform the object recognition task.

Furthermore, we tried several variations of our architecture. For example, we tried to replace the ResNet model with the simpler VGG model, but we obtained worse results. We also tried to use a final fully connected layer and add some dropout layers since our model seems to overfit, but both ideas made the model perform worse than the one defined previously.

In Fig. 6 are shown the results of the model trained for 10 epochs for which we report the loss and accuracy values.

5 Super-Resolution

In this section we will talk about our super-resolution module. Given in input a low-resolution image of a face, this module is going to give in output a 4x higher-resolution image, trying to fill the missing information and creating a more detailed image.

This could be useful for images of faces in low resolution, so for example taken by a low-quality surveillance camera, or captured by a considerable distance, making the resulting face region small. The module is going to upscale those images, in order to make the face recognition module supposedly work better since it is going to have more information available.

In our work we experimented with several methods in order to implement this task, ranging from simple classic interpolations to deep learning models that perform the super-resolution task.

To address this problem we implemented two models based on the Generative Adversarial Network (GAN), but we also tried and compared the following models:

1. **OpenCV Resize**: this model is the simplest one and it just performs a resize operation using the OpenCV library, employing the bilinear interpolation;
2. **Enhanced Deep Super-Resolution (EDSR)** [5]: the architecture of this model starts from a ResNet but the BatchNormalization layers are removed. It uses a high number of filters with some layers to stabilize the training procedure. The model can output images that are 2x, 3x and 4x the resolution of the input image;
3. **Super-Resolution Generative Adversarial Network (SRGAN)** [6]: in this model, since the authors claim that the MSE loss struggles to reconstruct high-frequency details, they tried to recover them using a GAN based on the ResNet architecture. A new perceptual loss is also proposed: instead of using the MSE between the high-resolution image and the one produced by the GAN, they computed the loss on the feature maps extracted by the VGG network;
4. **Enhanced SRGAN (ESRGAN)** [7]: this model is an improved version of the SRGAN. The main differences are that BatchNormalization layers are removed and the blocks are replaced with Residual-in-Residual Dense Blocks (RRDB) that combine multi-level residual networks and dense connections. Moreover, also the discriminator is different, since it doesn't predict if an image is real or fake, but instead tries to predict if an image is more realistic than a fake one (Relativistic Discriminator). In addition, in order to try to remove noise, they interpolated the parameters of two networks, a GAN and a PSNR-oriented network.

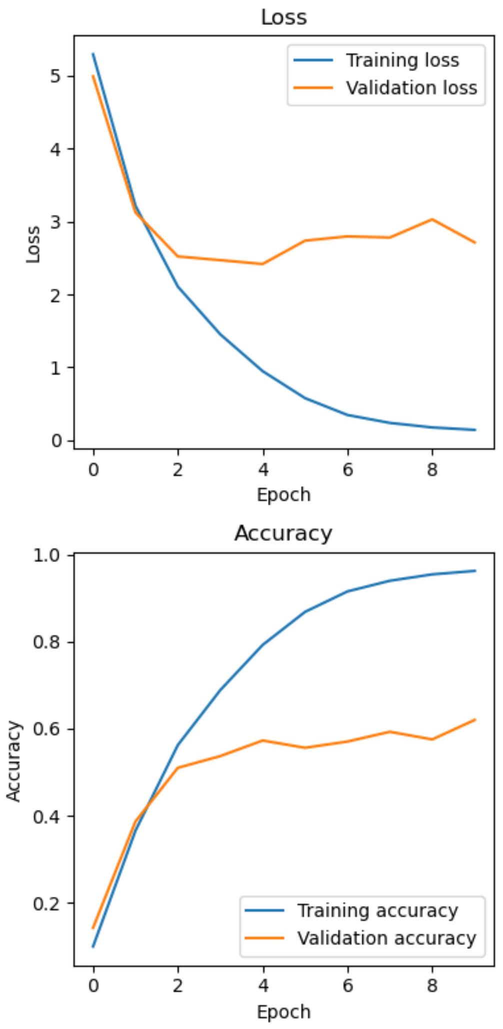

Fig. 6: Results obtained by our face recognition model during the training phase.

However, each of these models has its own limitations (or at least its most popular Python implementation):

- The EDSR model works pretty well but it has a high computational cost, making the prediction pretty slow in comparison to other methods (taking around 30s for an image on our platform);
- The SRGAN and ESRGAN models, as specified in their guidelines, do not perform well with images smaller than 64×64, so the results we obtain in our analysis are not optimal, since we are working with images with a resolution as low as 32×32.

Finally, since we want to work with face images, we can leverage this strong data bias in order to generate better results. This is why we decided to try to implement a custom model that is going to be trained exactly for the task of generating super-resolution images of human faces.

In order to do that, we realized two models. The first one strictly follows the architecture of the SRGAN, but we implemented it again using the TensorFlow library in Python, allowing us to train it from scratch using the CASIA-WebFace dataset. It takes as input a face image normalized in the range $[-1,1]$ and outputs an equally normalized image with super-resolution. However, we didn't used the loss proposed by the original paper, but instead we opted for a weighted sum between the adversarial loss Adv and the MSE between the high-resolution image and the one synthesized by the generator:

$$Adv(\theta) = \min_{\theta_G} \max_{\theta_D} \mathbb{E}[\log D_{\theta_D}(I^{HR})] + \mathbb{E}[\log(1 - (D_{\theta_D}(G_{\theta_G}(I^{LR}))]$$

$$MSE(\theta) = \frac{1}{wh} \sum_{x=1}^{w} \sum_{y=1}^{h} (I^{HR}(x,y) - G_{\theta_G}(I^{LR})(x,y))^2$$

$$\text{Loss}(\theta) = \alpha \cdot Adv(\theta) + \beta \cdot MSE(\theta)$$

Once we analyzed its results we saw that besides having some artifacts that we couldn't manage to avoid, even training it for less or more epochs, it was lacking a general sharpness on the edges of the face. For this reason, we thought about using multitask-learning in order to regularize the learning of the super-resolution images. In fact, we trained our model by adding the task of edge recognition using a supervised approach. To do that, we opted for the Canny operator which is an algorithm to detect various types of edges. It works by first smoothing the image applying a Gaussian filter, then it computes the gradients of the image, applies thinning using non-maximum suppression, filters the found edges using two thresholds and finally applies hysteresis removing the remaining weak edges.

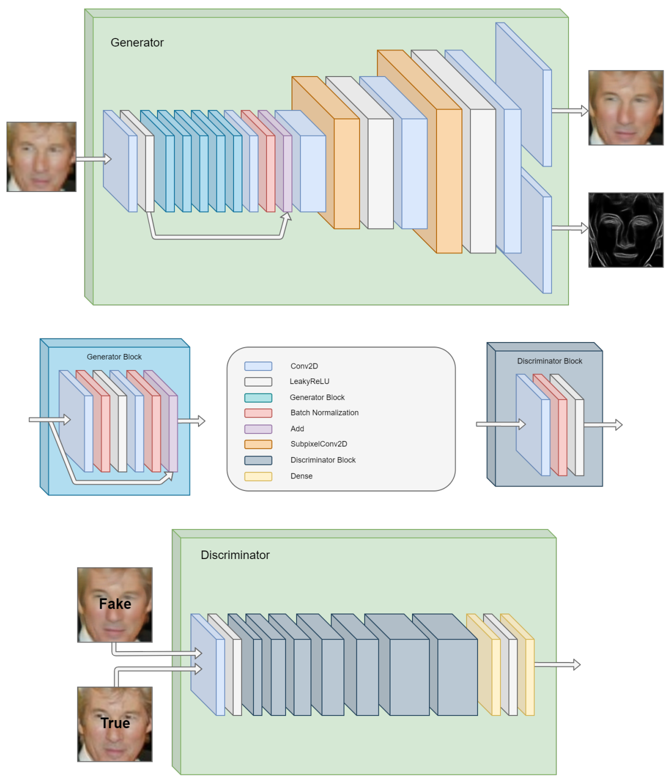

Fig. 7: Our GAN architecture with Edge detection Multi-task learning

In our model we pre-processed the training images using the previously mentioned operator with thresholds 50 and 150 in order to produce labels for this new task. Accordingly, as we can see from the final architecture in Fig. 7, we added to the generator of our model another final layer to perform edge detection, which has a smaller kernel size (3×3) and the sigmoid as activation function in order to produce results in the range $[0,1]$. To add this new task, we also added a new term *Canny* to the loss function, which is the binary cross-entropy between the Canny-preprocessed high-quality image I^{Canny} and the synthesized edge detection output of the generator:

$$Canny(\theta) = -\frac{1}{wh} \sum_{x=1}^{w} \sum_{y=1}^{h} I^{Canny}(x,y) \cdot \log(G_{\theta_G}^{Canny}(I^{LR})(x,y)) + (1 - I^{Canny}(x,y)) \cdot \log(1 - \log(G_{\theta_G}^{Canny}(I^{LR})(x,y)))$$

$$\text{Loss}(\theta) = \alpha \cdot Adv(\theta) + \beta \cdot MSE(\theta) + Canny(\theta)$$

In order to train both GANs we used a shuffled subset of the CASIA-WebFace dataset cropped as described in the previous section. We randomly selected 5.000 images avoiding low quality images using a threshold-based method that computes the Laplacian operator on the images and then the variance, which roughly indicates their sharpness. The models were trained for 100 epochs using a learning rate of 10^{-4}

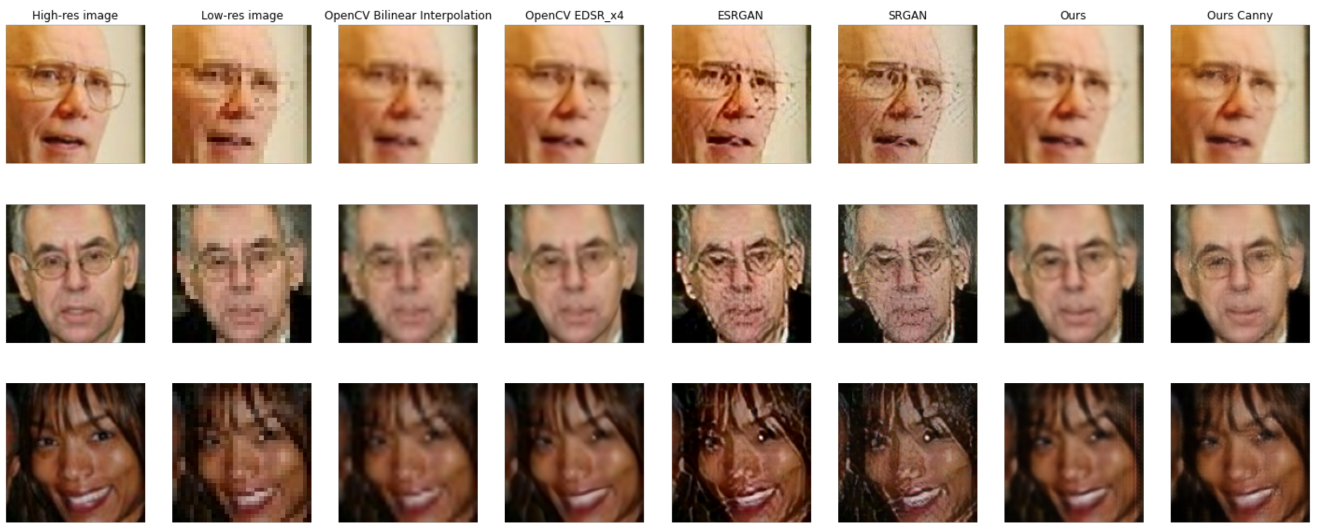

Fig. 8: A comparison of super-resolution images obtained with the different models.

and using as loss weights $[\alpha = 10^{-4}, \beta = 1]$ respectively for the adversarial (*Adv*) and image reconstruction loss (*MSE*).

Finally, in Fig. 8 is shown a comparison among the results obtained by all the different models we tried on some images of the LFW dataset. As we can see, as previously said, both the SRGAN and ESRGAN models are not suited for the low resolution images, so the results obtained are far from usable. Instead, the results obtained with EDSR and our GANs are quite better than the normal bilinear interpolation that is usually employed when using a model that accepts images of a fixed resolution. Moreover, our results are comparable to the EDSR implementation, and while our base GAN obtains pretty much the same quality, our Canny GAN seems to generate slightly sharper images.

6 Evaluation

In this section we are going to explain our evaluation methodology and present the results obtained by our models.

When evaluating a biometric system, it is necessary to define a probe and a gallery set. The gallery set is going to contain the templates of subjects which are already enrolled in the system, while the probe set contains the captured biometric traits that the system needs to process in order to output a decision. In particular, from each probe a template is extracted and compared with one in the gallery according to the selected operating mode of the system, which can be either verification or identification. Each mode can also be closed-set or open-set: in the first case each probe must belong to an already enrolled subject, meaning that there is a matching template in the gallery, while in the second case, an incoming probe could also belong to a subject not enrolled in the system. In our work, we examined our models using the LFW dataset and two evaluation methodologies: an All-Against-All approach in an open-set identification scenario, and the standard LFW evaluation approach, as defined in [1].

In the All-Against-All (AAA) approach, we treat in turn each template extracted from the LFW dataset as a probe, keeping all the others as the gallery set. Moreover, in turn, we also test the ability of the system to recognize impostor attempts by ranking the distances between the current probe and the gallery set composed by all the templates except the ones of the identity associated to the probe. In order to carry out the AAA validation method, we must compute the distance matrix which contains the distances among all the possible pairs of templates. The diagonal of this matrix represents the distance between each template and itself, and it is discarded since we don't want to consider the case where the probe is inside the gallery set as it would be trivial. Moreover, the AAA validation method is computed for several thresholds in order to determine the best one for the model.

In the standard LFW evaluation protocol, instead, we are provided with a fixed test set which consists of pairs of face images. There are two types of pairs: matching pairs which contain two images of the same subject, and mismatching pairs which contain two images of different subjects. The system should be able to recognize those two different cases and output if the two subjects in the pair are the same or not. In the provided test set are present 10 folds, each consisting of 300 matching pairs and 300 mismatched pairs, allowing us to evaluate the models averaging the results under different conditions.

In our first experiment we wanted to compare the performance of our face recognition model feeding as input different versions of the test images. The results obtained are shown in Fig. 9. First, we evaluated the performance of our model on the original LFW dataset with cropped faces of resolution 128×128 (blue line), and compared with the same images downscaled to 32×32 (orange line). We assumed that the first one would be the maximum result achievable, and we can see that in fact using the images 128×128 has a huge performance boost in comparison to using the 32×32 images, which contain much less information, confirming our initial hypothesis. Then, we started from the low-resolution images and tried to measure the performance

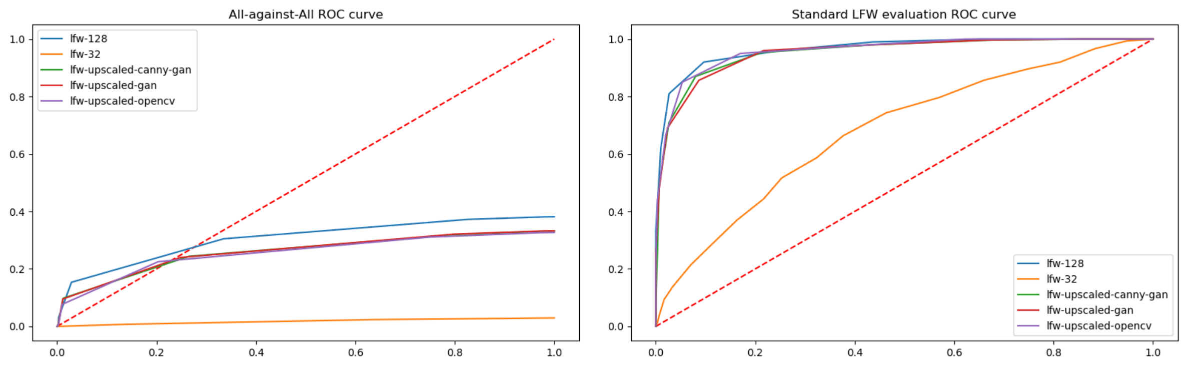

Fig. 9: Comparison of different versions of input images fed to our face recognition model with the All-Against-All and standard LFW validation methods.

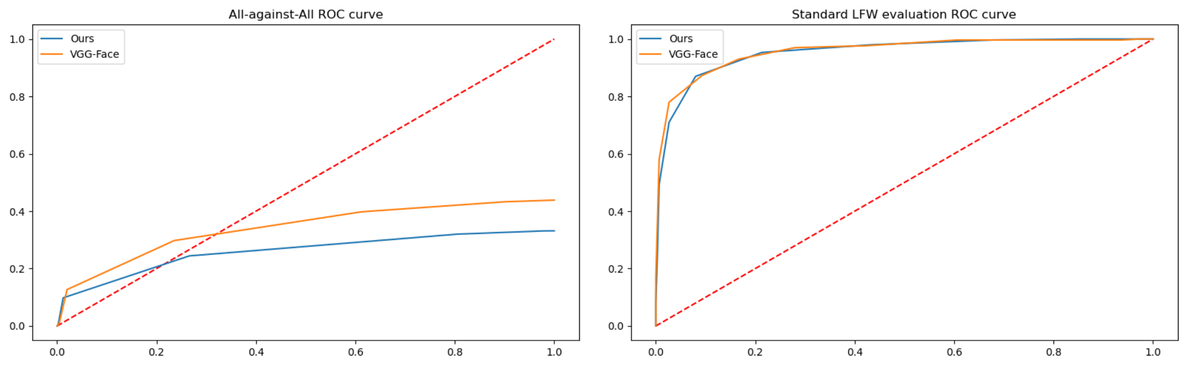

Fig. 10: Comparison of best model performances with the VGG-Face implementation using the All-Against-All and standard LFW validation methods.

using super-resolution through three different methods: the OpenCV bilinear interpolation, our normal GAN, and our Canny GAN. Unfortunately, we can see that the performances of our GANs are very similar to the simple bilinear interpolation implemented in OpenCV, suggesting that the amount of face details added by the GANs did not improve the face recognition performance.

Instead, in the second experiment, we compared our solution against a VGG-Face Python implementation using as input the 32×32 images. VGG-Face is the implementation of a VGG model with the goal of performing the task of face recognition and it was trained with more than 2 million images. The VGG-Face, in contrast to our model, resizes the images to the size 224×224 using bilinear interpolation before computing the template for an input face. In the results shown in Fig. 10 we can see that our solution is distant from the performances of the VGG-Face when evaluated with the All-Against-All method, which could be explained simply by the fact that our model wasn't trained as much as the VGG-Face. Instead, when we evaluate the two models using the standard LFW evaluation method we can see that they perform similarly.

Finally, we also reported some metrics acquired during the evaluation phase that are shown in Table 1 and Table 2. As we can see every model performs much worse in the All-Against-All comparison with respect to the LFW standard approach. This is probably due to the fact that the task tested in the second evaluation method is much easier since it require only to be able to distinguish between two subjects.

Dataset name	**RR**	**DIR @5**	**DIR @15**	**GRR**	**EER**	**Best Th**
Original-128	0.37	0.44	0.47	0.13	0.63	0.25
Original-32	0.03	0.07	0.12	0.04	0.97	0.20
Canny-GAN	0.32	0.40	0.42	0.15	0.68	0.25
GAN	0.32	0.40	0.42	0.16	0.68	0.25
Bilinear Int.	0.31	0.38	0.40	0.20	0.69	0.25
VGG-Face	0.40	0.44	0.46	0.26	0.60	0.20

Table 1: Table with the Recognition Rate (RR), Detection and Identification Rate at rank K (DIR@K), Genuine Rejection Rate (GRR), Equal Error Rate (ERR) and the best distance threshold for each dataset computed with the All-Against-All validation method.

Dataset name	**Accuracy**	**AUC**	**Best Th**
Original-128	0.88	0.93	0.60
Original-32	0.62	0.66	0.40
Canny-GAN	0.86	0.91	0.60
GAN	0.86	0.91	0.60
Bilinear Int.	0.86	0.91	0.60
VGG-Face	0.87	0.93	0.65

Table 2: Table with the Mean Accuracy, Area Under the Curve (AUC), and best similarity threshold for each dataset computed with the standard LFW validation method.

7 Conclusions

In this work we studied how the input image resolution can affect a face recognition system, with a particular focus on how the system performs using super-resolution images. In order to do so, we first had to extract faces from our datasets, and we compared two famous techniques which are the OpenCV implementation of the Haar Cascade Classifier, and the MTCNN deep learning model. The results showed that the latter is more robust, but slower than the former, which has shown acceptable performances for our task, and so our final decision was to use the faster technique. Then, we tried to implement different kind of super-resolution models in order to produce high-quality images for the face recognition module, and we compared their performances against the normal low resolution images. In particular, we designed two GAN models based on the SRGAN architecture. The second model is an improvement of the first, obtained by increasing the sharpness of the output images thanks to the addition of an edge detection term to the loss. In the evaluation phase, we presented two different methodologies, the All-Against-All method and the Standard LFW Evaluation method, comparing the performances obtained both with low, high and super-resolution images. From the results we found out that the GANs proposed do actually increase the resolution and details of the low-quality images, but do not actually help to improve the performance of the entire system. At the end, we also compared our performance with one state-of-the-art face recognition model, the VGG-Face, showing comparable performance even if a little bit worse.

References

[1] Huang, G., Mattar, M., Berg, T., and Learned-Miller, E., 2008. "Labeled faces in the wild: A database for studying face recognition in unconstrained environments". *Tech. rep.*, 10.

[2] Yi, D., Lei, Z., Liao, S., and Li, S. Z., 2014. Learning face representation from scratch.

[3] Viola, P., and Jones, M., 2001. "Rapid object detection using a boosted cascade of simple features". In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Vol. 1, pp. I–I.

[4] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y., 2016. "Joint face detection and alignment using multitask cascaded convolutional networks". *IEEE Signal Processing Letters*, pp. 1499–1503.

[5] Lim, B., Son, S., Kim, H., Nah, S., and Lee, K. M., 2017. "Enhanced deep residual networks for single image super-resolution". *CoRR,* **abs/1707.02921**.

[6] Ledig, C., Theis, L., Huszar, F., Caballero, J., Aitken, A. P., Tejani, A., Totz, J., Wang, Z., and Shi, W., 2016. "Photo-realistic single image super-resolution using a generative adversarial network". *CoRR,* **abs/1609.04802**.

[7] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Loy, C. C., Qiao, Y., and Tang, X., 2018. "ESRGAN: enhanced super-resolution generative adversarial networks". *CoRR,* **abs/1809.00219**.