TRAINING CONVOLUTIONAL FILTERS FOR ROBUST FACE DETECTION

Manolis Delakis† and Christophe Garcia‡
†University of Crete, PO BOX 1470, 71110 Heraklion, Greece
‡IRISA/INRIA de Rennes
Campus de Beaulieu 35042 Rennes Cedex, France
Phone:+33 299842593
E-mails:delakis@csd.uoc.gr, cgarcia@irisa.fr

Abstract. We present a novel face detection approach based on a convolutional neural architecture, designed to detect and precisely localize highly variable face patterns, in complex real world images. Our system automatically synthesizes simple problem-specific feature extractors from a training set of face and non face patterns, without making any assumptions or using any hand-made design concerning the features to extract or the areas of the face pattern to analyze. Experiments on different difficult test sets have shown that our approach reach superior overall detection results, while being significantly faster than state-of-the-art approaches that require dense scanning and local preprocessing.

INTRODUCTION

Human face detection is becoming a very important research topic, because of its wide range of possible applications, like security access control, model-based video coding, content-based video indexing or advanced human and computer interaction. It is also a required preliminary step to face recognition and expression analysis. Numerous approaches for face detection have been proposed in the last years, which are presented in two interesting recent surveys by Yang *et al.* [11] and Hjelmas *et al.* [4].

Most face detection methods are based on local facial feature detection and classification using statistical and geometric models of the human face. Low level analysis first deals with the segmentation of visual features using image properties such as edges, intensity, color, motion or generalized measures [10, 3]. Then, visual features are organized into a more global concept of face through facial feature and constellation analysis using face geometry constraints [2].

The main drawback of feature-based approaches is that either little global constraints are applied on the face template or extracted features are strongly influenced by noise, occlusion and variations in face expression or viewpoint. In order to handle difficult scenarios where multiple faces of different sizes and poses have to be detected in heavily cluttered background, neural network based approaches have proved to be very efficient. They have also the clear advantage of learning underlying rules contained in the highly variable face patterns from large training sets of images. The first advanced neural approach that reported results on a large and difficult dataset was by Rowley *et al.* [7]. Their system incorporates face knowledge in a retinally connected neural network, looking at windows of 20×20 pixels, where different units look at different areas of the face window. A large number of adjustable weights (2,905) are learnt through standard backpropagation. Osuna *et al.* [6] developed a support vector machine (SVM) approach to face detection. A SVM with a 2nd-degree polynomial is used as a kernel function and approximately 2,500 support vectors are obtained and use for face detection. Féraud *et al.* [1] proposed an approach based on constrained generative models (CGM), which are large auto-associative fully connected MLPs, trained to perform a nonlinear PCA. Classification is obtained by combining and analyzing the reconstruction errors of CGMs.

All these methods are based on a costly exhaustive multi-resolution window scanning technique. The input image is successively subsampled by a factor of 1.2, giving a pyramid of images. In each subsampled image, a window (of size 20×20 approximately) is scanned at every position, and its content is preprocessed via lighting correction and histogram equalization like in the Sung and Poggio's system [8], before being processed by the neural architecture.

In this paper, we propose a novel image-based approach based on a convolutional neural architecture [5]. Our approach differs from most of the previous ones on several main points. Our system automatically derives simple problem-specific feature extractors from a training set of face and non face patterns, without using any assumptions or hand-made design concerning the features to extract and the areas of the face pattern to analyze. Moreover, once trained, our system acts like a fast pipeline of simple convolutions and subsampling modules that treat the raw input image as a whole for each analyzed scale and does not require any costly local preprocessing of extracted windows.

SYSTEM ARCHITECTURE

The convolutional neural network, shown in Fig. 1, consists of a set of three different kinds of layers. Layers Ci are called convolutional layers, which contain a certain number of planes. Layer C1 is connected to the retina, receiving the image area to classify as face or non face. Each unit in a plane receives input from a small neighborhood (biological local receptive field) in the planes of the previous layer. The trainable weights (convolutional mask) forming the receptive field for a plane are forced to be equal at all points in

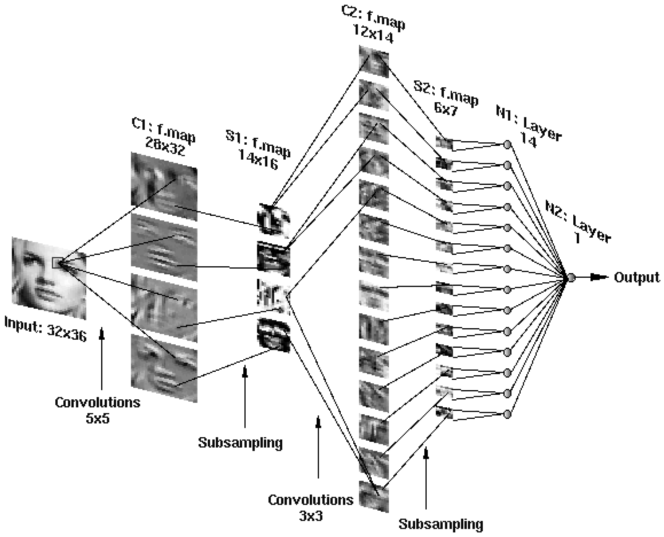


Figure 1: The convolutional architecture

the plane (weight sharing). Each plane can be considered as a feature map that has a fixed feature detector that corresponds to a pure convolution with a trainable mask, applied over the planes in the previous layer. A trainable bias is added to the results of each convolutional mask. Multiple planes are used in each layer so that multiple features can be detected.

Once a feature has been detected, its exact location is less important. Hence, each convolutional layer Ci is typically followed by another layer Si that performs local averaging and subsampling operations. More precisely, a local averaging over a neighborhood of four inputs is performed followed by a multiplication by a trainable coefficient and the addition of a trainable bias. This subsampling operation reduces by 2 the dimensionality of the input and increases the degrees of invariance to translation, rotation, scale, and deformation of the face patterns.

In our implementation, layers C1 and C2 perform convolutions with trainable masks of dimension 5x5 and 3x3 respectively. Layer C1 contains 4 feature maps and therefore performs 4 convolutions on the input image. Layers S1 and C2 are partially connected. Mixing the outputs of feature maps helps in combining different features, thus in extracting more complex information. In our system, layer C2 has 14 feature maps. Each of the 4 subsampled feature maps of S1 is convolved by 2 different trainable masks 3x3, providing 8 feature maps in C2. The other 6 feature maps of C2 are obtained by fusing the results of 2 convolutions on each possible pair of feature maps of S1.

Layers N1 and N2 contain simple sigmoid neurons. The role of these layers is to perform classification, after feature extraction and input dimensionality reduction are performed. In layer N1, each neuron is fully connected to every points of one feature map only of layer S2. The unique neuron of layer N2 is fully connected to all the neurons of the layer N1. The output of this neuron

Figure 2: Some samples of the training set. The last row shows face-like false alarms produced during bootstrapping.

is used to classify the input image as face or non face. For training the network, we used the classical backpropagation algorithm with momentum modified for being used in convolutional networks as described in [5]. Desired responses are set to -1 for non-faces and to +1 for faces.

In our system, the dimension of the retina is 32x36. Because of weight sharing, the network has only 951 trainable parameters, despite the 124,741 connections it uses. Local receptive fields, weight sharing and subsampling provide many advantages to solve two important problems at the same time: the problem of robustness and the problem of good generalization, which is critical given the impossibility of gathering in one finite-sized training set all the possible variations of the face pattern. This topology has another decisive advantage. In order to search for faces, the network must be replicated (or scanned) at all locations in the input image, as done in the above mentioned approaches [1, 6, 7, 8]. In our approach, since each layer essentially performs a convolution with a small-size kernel, a very large part of the computation is in common between two neighboring face window locations in the input images. This redundancy is naturally eliminated by performing the convolutions corresponding to each layer on the *entire input image at once*. The overall computation amounts to a succession of convolutions and non-linear transformations over the entire images.

TRAINING METHODOLOGY

The face examples used to train the network were collected from various sources over the Internet or scanned images from newspapers. This collection effectively capture the variability and the richness of natural data in order to train our system for operating in uncontrolled natural environments. Some of the 3,702 original collected face patterns are shown in the first row of Fig. 2. Using manually labeled eye and mouth positions, face images were cropped and normalized to the size of 32×36 pixels, after de-rotation and re-scaling in order to position roughly the two eyes around the same locations inside the retina. More precisely, the central part of the face was set to roughly 20×20 pixels, leaving space for the borders of the face and some background portions. The system is therefore fed with some additional information about the face shape and some border effects that may arise in the convolutions are canceled.

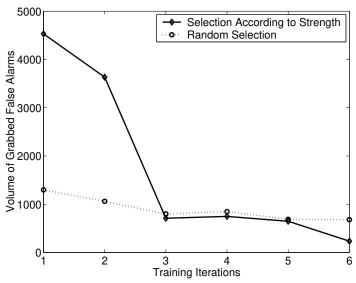
1. Create a validation set of 400 faces and 400 non-faces randomly extracted and excluded from the initial training set. It will be used to choose the best performing weight configuration during steps 3 and 8.
2. Set $BIter = 0$, $ThrFa = 0.8$.
3. Train the network for 60 learning epochs. Use an equal number of positive and negative examples in each epoch. Set $BIter = BIter + 1$.
4. Gather false alarms from a set of 692 scenery images with network answers above $ThrFa$. Collect at maximum 5000 new examples.
5. Concatenate the newly created examples to the non-face training set.
6. If $ThrFa \geq 0.2$ set $ThrFa = ThrFa - 0.2$.
7. If $BIter < 6$ go to step 3.
8. Train the network for 60 more learning epochs and exit.

TABLE 1: THE PROPOSED BOOTSTRAPPING SCHEME.

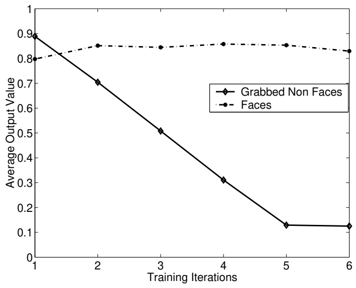
No preprocessing was applied on the cropped faces, such as overall brightness correction and histogram equalization that are performed in [1, 6, 7, 8]. Moreover, in order to create more examples and to enhance the tolerance to small rotations and variations in intensity, a series of transformations were applied to the above initial set of face examples, including mirroring, rotation, smoothing and contrast reduction. The last two transformations help to teach the system how to cope with situations where the retina is fed with a quite weak signal (e.g. over-smoothed or with weak edges). Some examples of the transformed patterns are shown in the second row of Fig. 2. Finally, the training set reached the number of $25,212$ face patterns, partially occluded (glasses, hair, etc...), unequally lighted, turned up to ± 60 degrees, rotated up to ± 20 degrees and with average intensity values varying from dark to light.

Collecting a representative set of non-faces is more difficult as virtually any non-face image could belong to it. A practical solution to this problem consists in a bootstrapping strategy [8], in which the system is iteratively re-trained with false alarms produced when applied to a set of scenery images, that do not contain faces. In the proposed approach, we improved this strategy in some points. Before proceeding with the bootstrapping, an initial training set of 6,422 non-face patterns was built by selectively cropping images. Most of these images contain part of faces as it was noticed in some early experiments that this kind of images is a serious false alarm source. The proposed bootstrapping procedure is presented in table 1.

In step 1, a validation set is built and used for testing the generalization ability of the network during learning and, finally, selecting the weight configuration that performs best on it. This validation set is kept constant through all the bootstrapping iterations, in contrast with the training set which is updated. In step 3, the backpropagation algorithm is used with the addition of a momentum term for neurons belonging to the N1 and N2 layers. Stochastic learning was preferred versus batch learning. For each learning epoch, an equal number of examples from both classes are presented



(a)



(b)

Figure 3: The evolution of the proposed training-bootstrapping procedure.

to the network giving no bias toward one of the two classes.

The generation of the new patterns that will be added to the non-face training set is carried out by step 4. The false alarms produced in this step force the network, in the next iteration, to refine its current decision boundary for the face class. At each iteration, the false alarms, giving network answers greater than $ThrFa$, and therefore strongly misclassified, are selected. As the network generalizes from these examples, $ThrFa$ is gradually reduced until reaching 0. In this way, some redundancy is avoided in the training set. The learning process is stopped after six iterations, where convergence is noticed, i.e. when the number of false alarms remains roughly constant. This procedure helps in correcting problems arising in the original algorithm proposed in [8] where false alarms were grabbed regardless of the strength of the network answers. Finally, the controlled bootstrapping process added 25,087 non-face examples to the training set. Some of them are shown in the last row of Fig. 2.

The performance of the bootstrapping procedure is described in Fig. 3. In particular, one can observe how the network was boosted in terms of false alarms rejection. The first graph (Fig. 3(a)) presents the *volume* (sum of all network outputs corresponding to the false alarms) of the grabbed false alarms with respect to the training-bootstrapping iterations. This metric characterizes the strength of the false alarms. We analyze the evolution of the false alarm volume when the false alarms are grabbed according to their strength and when they are grabbed with random selection, i.e. when $ThrFa$ is constantly 0. In the first case, the first iteration produces very strong false alarms, as expected for the first run of the network over the scenery images. During the following iterations, the network learns quickly how not to produce strong false alarms, as illustrated by the sharp decrease of the volume in iteration 3. Thereafter, the behavior remains roughly constant, which indicates that the bootstrapping procedure can safely terminate. In the case of random selection, we notice in all iterations low false alarm volumes as the majority of false alarms correspond to low network outputs. The evolution of the training remains roughly stagnant, which indicates that the rejections abilities of the network do not improve during bootstrapping. Indeed, in all iterations, the system failed to run over the complete set of the scenery

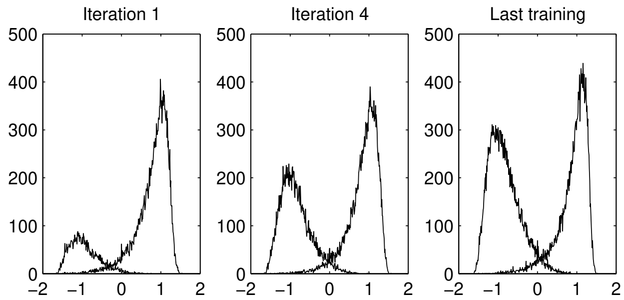


Figure 4: The evolution of the separation of the classes.

images before reaching the limit of 5,000 false alarms.

Fig. 3(b) presents the evolution of the mean network output over the false alarms and the positive examples of the training set. The roughly constant high mean response to faces (around 0.85) contrasts with the gradually decreasing mean response to false alarms, summarizing the results of the training. We can remark that the mean false alarm activation strictly follows the linear decreasing of the $ThrFa$ value.

A direct way to realize how well the network separates the two classes is to compute the histogram of the network responses over the complete training set. The evolution of this histogram over the bootstrapping iterations is displayed in Fig. 4. We see clearly in all the histograms two distributions corresponding to the face and the non-face classes, with peaks to the respective target values ($+1$ for faces and -1 for non-faces). Even though the separation between these two distributions is not absolute, which would imply a zero training error, it is satisfactory as only a very small percentage of the distributions overlaps. The distribution of the face responses remains roughly constant, which agrees with Fig. 3(b). The distribution of the non-face responses naturally expands as more non-face examples are added to the training set in each iteration.

CHOICE OF THE NETWORK TOPOLOGY

In table 2, we present the error rates of the proposed topology (referred as N4) on the training and on the validation sets, along with the error rates of other convolutional topologies (N1 and N5) and the linear classifier. For each topology, the number of errors, the Mean Square Error (MSE) and the number of false alarms generated during the bootstrapping are given. The training of the proposed topology resulted in 4% misclassifications in both training and validation sets. The N1 topology is a convolutional network with one convolutional/subsampling layer only, identical to the first convolutional layer of topology N4. It corresponds to the topology of the convolutional network used in [9]. The N1 topology has 1,009 weights, more than in N4, as subsampling is performed only once. The N5 topology has two convolutional/subsampling layers, as in N4, but with the addition of one more

	Training Set		Validation Set		
Topology	Errors	MSE	Errors	MSE	False Alarms
N1	2607	0.17	60	0.26	43309
N4	**1921**	**0.17**	**32**	**0.18**	**25087**
N5	1540	0.15	24	0.17	32155
Linear	11558	0.65	185	0.66	-

TABLE 2: ERROR RATES FOR DIFFERENT CONVOLUTIONAL TOPOLOGIES AND FOR THE LINEAR CLASSIFIER.

feature map in the C1/S1 layers. It also has 6 more feature maps in the C2/S2 layers as it uses the same connection strategy as in N4 between the S1 and C2 layers. This topology uses 1,346 weights. N1 and N5 are two topologies that represent variations of the proposed topology (N4), as we add or remove "structure".Finally, results for the linear classifier which is a single-layer perceptron with $32 \times 36 + 1 = 1,153$ weights, applied on the complete training set produced by the N4 topology, are also presented.

First, we can notice that the N4 and N5 topologies perform best and are roughly equivalent. N5 has a slightly better performance but generates more false alarms: it required two more bootstrapping iterations before convergence. As the N4 topology has fewer weights, we choose it as a simpler and lighter solution. Furthermore, the N4 topology performs slightly better in the testing phase. The topology N1 generated much more false alarms than the other two. This topology finally failed to converge during the bootstrapping in the sense that it was not able to process all the scenery images with less false alarms than the upper bound (5,000) in all the iterations. It also performs poorly on the validation set, having an error rate roughly double than the ones of N4 and N5. Finally, we can easily notice the very poor performance of the linear classifier. It gives roughly 10% error rate in the training set and 23% in the validation set, compared the respective 4% and 4% that the N4 topology gives. This demonstrates how complex and non-linear is the problem to solve.

EXPERIMENTAL RESULTS

Our method has been evaluated using the *CMU* test set [7] which is, so far, the most widely used data set in the literature. It consists of 130 images with a total of 507 near frontal faces. This data set includes 23 images of the second data set used by Sung and Poggio [8], referred as *MIT*. Most of the images in these data sets have complex backgrounds with faces covering a variable part of the total image area. Faces in these data sets present a large variability in size, illumination, facial expression, pose, and may be partially occluded. In order to detect faces of different sizes, the input image is repeatedly subsampled by a factor of 1.2, resulting in a pyramid of images, which are entirely convolved at once by the network, providing a result image for each scale. Candidate faces (pixels with positive values in the result

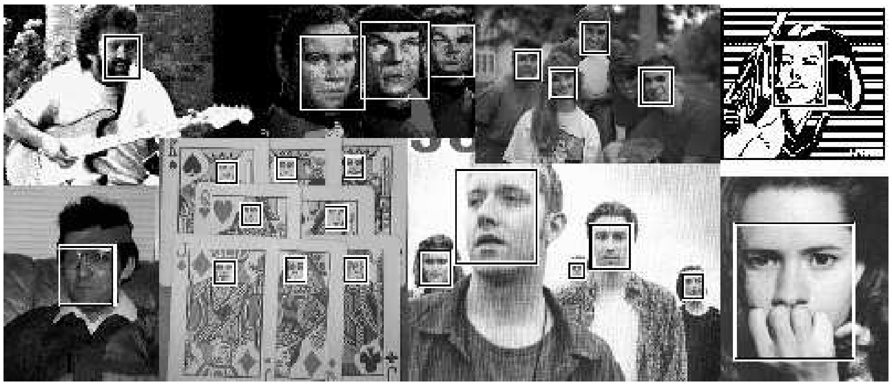


Figure 5: Some results obtained on the *CMU* test set.

Face Detector	**CMU**	**MIT**
`Rowley` *et al.* `[7]`	86.2%/23	84.5/8
`Féraud` *et al.* `[1]`	86.0%/8	
`Sung and Poggio [8]`		79.9%/5
`Osuna` *et al.* `[6]`		74.2%/20
Our approach	**89.95%/8**	**87.8%/4**

Table 3: Results Reported in Terms of Percentage of Good Detection / Number of False Alarms, on the *CMU* and *MIT* test sets.

image) in each scale are mapped back to the input image scale and then grouped into representative faces according to their proximity in image and scale spaces. A local search procedure is finally performed in a small pyramid around each face candidate center in image-scale space, in order to perform fine localization and eventually false alarm dismissal. Finally, the *volume* of positive answers in the local pyramid is used to take the classification decision. Based on numerous experiments and ROC curve analysis, a face candidate is classified as face if its corresponding volume is greater than $ThrVol = 22.0$.

Table 3 shows the detection rates of our method and the reported results of several detection methods on the *CMU* and *MIT* test sets [4]. Our results are given for $ThrVol = 22.0$, which provides balanced detection and false alarm rates on different test sets. Fig.5 presents some results of the proposed face detection scheme on some images of the *CMU* test set.

Our system can also be tested using our interactive demonstration available on the Web at *www.csd.uoc.gr/˜cgarcia/FaceDetectDemo.html*, allowing anyone to submit images and obtain the detection results on-line.

CONCLUSION

Our experiments have shown that using an architecture based on convolutional neural networks for face detection improves significantly the detection results on difficult benchmarked test sets. Because of its convolutional nature and the use of a single network, our system is moreover significantly faster than state-of-the-art image-based approaches which require costly local preprocessing. It can process images at a rate of approximately $2.5e - 05$s. per pixel on a PC-Pentium IV at 1.6GHz, which gives a processing time of 2.5s for a video image of 288x352 pixels. A straightforward implementation on classical image processing boards will allow low-cost near real time applications. Moreover, our approach is not restricted to vertical semi-frontal faces. It is able to detect highly variable faces turned up to ± 60 degrees, rotated up to ± 20 degrees in image plane, and partially occluded.

REFERENCES

[1] R. Féraud, O. Bernier, J. Viallet and M. Collobert, "A Fast and Accurate Face Detector Based on Neural Networks," **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 23, no. 1, pp. 42–53, 2002.

[2] C. Garcia, G. Simandiris and G. Tziritas, "A Feature-Based Face Detector Using Wavelet Frames," in **Proceedings of the International Workshop on Very Low Bit Coding**, 2001, pp. 71–76.

[3] C. Garcia and G. Tziritas, "Face Detection Using Quantized Skin Color Regions Merging and Wavelet Packet Analysis," **IEEE Transactions on Multimedia**, vol. 1, no. 3, pp. 264–277, 1999.

[4] E. Hjelmås and B. K. Low, "Face Detection: A Survey," **Computer Vision and Image Understanding**, vol. 83, pp. 236–274, 2001.

[5] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," **Proceedings of the IEEE**, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] E. Osuna, R. Freund and F. Girosi, "Training Support Vector Machines: An Application to Face Detection," in **Proceedings of IEEE Conference on Computer Vision and Pattern Recognition**, 1997, pp. 130–136.

[7] H. Rowley, S. Baluja and T. Kanade, "Neural Network-Based Face Detection," **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 20, no. 1, pp. 23–38, 1998.

[8] K.-K. Sung and T. Poggio, "Example-Based Learning for View-Based Human Face Detection," **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 20, no. 1, pp. 39–51, 1998.

[9] R. Vaillant, C. Monrocq and Y. LeCun, "Original Approach for the Localisation of Objects in Images," **IEE Proceedings on Vision, Image, and Signal Processing**, vol. 141, no. 4, pp. 245–250, 1994.

[10] G. Yang and T. S. Huang, "Human Face Detection in Complex Background," **Pattern Recognition**, vol. 27, no. 1, pp. 53–63, 1994.

[11] M. Yang, D. Kriegman and N. Ahuja, "Detecting Faces in Images: A Survey," **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 24, no. 1, pp. 34–58, 2002.