Fast Face Detection using MLP and FFT

S. Ben-Yacoub, B. Fasel and J. Lüttin
IDIAP
CP 592, 1920 Martigny
Switzerland
email: sby@idiap.ch

Abstract

Neural networks have shown their reliability and robustness for the face detection task (Rowley, Baluja and Kanade, 1995). However, the time consuming process needed by neural networks has prevented them from being a practical tool.

We propose a new technique that significantly speeds up the time needed by a trained network (MLP in our case) to detect a face in a large image. We reformulate neural activities in the hidden layer of the MLP in terms of filter convolution, enabling the use of the Fourier transform for an efficient computation of the neural activities. The method was applied to face detection in still images as well as on live video sequences.

1 Introduction

Face detection is a fundamental step before the recognition or identification procedure. Its reliability and time-response have a major influence on the performance and usability of a face recognition system.

The large variability of human faces causes major difficulties in the design of a model that can encompasses all possible faces [2]. Appearance-based approaches as well as learning-based approaches seem to be best suited for this task. In either case, a set of representative faces is necessary to find an implicit model.

Eigenfaces [10] were used to model the distribution of faces in some large input space (typically the input space is $\mathbb{R}^n$ where n is the size of the image). The main assumption of this approach is that the set of faces are located in a sub-space that can be approximated (using the KL-transform) through a training set. The "faceness" of an input image is determined by its distance to the face sub-space.

A similar approach was proposed by Sung and Poggio [9] where the sub-space was approximated by 6 gaussian distributions. The distances between a given input image and the 6 sub-spaces generated a vector that was used by a perceptron to separate the face space from the non-face space.

The use of neural-networks appears to be one of the most promising methods for face detection [11]. However, the time consuming processing [6, 3] needed by the neural networks has prevented them from being a practical tool.

A neural network based face detector was proposed in [7], and has shown good results. A feed-forward neural network was designed to detect faces using a 20×20 input window. The neural network architecture was optimally designed with receptive fields to specialize a set of neurons to detect eyes, mouth and nose. The negative examples (i.e. non-face images) for the training set were generated using a bootstrap technique. The system has demonstrated excellent detection rates, but it suffers from time consuming computations yielding "slow" responses. Most of the computation time is spent in exploring all the possible sub-images. Some strategies were used to reduce the time complexity, but with the drawback of a lower system performance.

We present an approach that speeds-up the processing time by considering a MLP (Multi Layer Perceptron) as a bank of filters, and by reformulating the processing steps in terms of con-

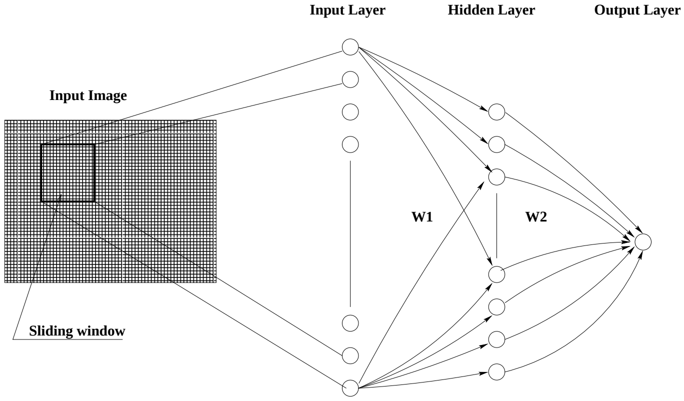


Figure 1: MLP architecture for object detection

volutions. Performing the convolution in the frequency domain is less time consuming. The characteristic of this approach is that it considerably reduces the computation time while maintaining identical performance. This method enables a fast computation of the MLP neuron activities, but does not reduce the recognition performance of a MLP based detector. The method has been validated for face detection in still images as well as in live video sequences.

2 The FFT-Multi Layer Perceptron

We consider a 3-layer feed-forward neural network or MLP (see Figure 1) trained with the classical back-propagation algorithm. The input layer is a vector, and the output layer is a single neuron for the sake of simplicity[1]. Let I be the input layer vector, H the hidden layer vector and O the output. We consider $n \times n$ pixel input images transformed into a column-vector to feed the input layer constituted by n^2 units or neurons. The hidden layer has m neurons, and the output layer is a single neuron which is triggered to 1 if the learned pattern is present and 0 otherwise.

This architecture can be used to detect different objects. Changing the learning set generates a new object detector. The modular property of this architecture makes it very flexible and handy. The activity of a particular neuron i in the hidden layer H can be written as:

$$h_i = g\left(\sum_{j=1}^{n^2} W_{1i}(j) I(j) + b_1(i)\right) \qquad (1)$$

Where W_{1i} is the set of weights of neuron i and $b_1(i)$ is the threshold. Similarly, the output layer activity is:

$$O = g\left(\sum_{j=1}^{m} W_{2j} h_j + b_2\right)$$

Where W_{2j} are the output weights and b_2 the output threshold.

The training algorithm is based on the back-propagation of the error which is now a well-known technique [8]. An example is picked from the training set, the output is computed. The error is computed as the difference between the actual and the desired output. It is minimized by back-propagating it and by adjusting the weights.

During the recognition step of the classical MLP, a sub-image of size $n \times n$ is extracted from the test image of size $N \times N$, and fed to the neural network. This operation must be iterated on all possible different sub-images of the input image. This is the major drawback in the use of neural networks for object recognition. Typically, for an $N \times N$ test image, from which all $n \times n$ sub-images are extracted to compute the activities of m neurons in the hidden layer, $O(N^2 n^2 m)$ computation steps are required.

To reduce the computational burden, we propose to efficiently calculate the activity of hidden units using the Fourier transform. The activity of hidden neuron i, with weight matrix Φ_i, over the whole image $\mathcal{I}$ can be formulated as a cross-correlation operation[2]:

$$\mathcal{H}_i = g(\mathcal{I} \otimes \Phi_i + B_1) \qquad (2)$$

where $B_1(k,l) = b_1(i)\ \forall (k,l) \in [1..n]^2$. The matrix $\mathcal{H}_i$ is the activity matrix of the hidden unit i. From equation (2), we can state that $\mathcal{H}_i(j,k)$ is the activity (or output) of the hidden unit i

[1] Extension to an output with multiple neurons is straightforward.

[2] The cross-correlation operation is represented by $\otimes$

when the observation window is located at position (j,k) in the input image $\mathcal{I}$. The final output activity matrix of the neural network can then be expressed as a linear combination of the hidden units activity:

$$\mathcal{O} = g(\sum_{i=1}^{m} W2_i \mathcal{H}_i + b_2) \qquad (3)$$

Here again, $\mathcal{O}(j,k)$ is the output of the neural network when the observation window is located at (j,k) in the input image $\mathcal{I}$.

In equation (2), the activity is expressed in terms of a cross-correlation between a bank of filters $(\Phi_i)_{i\in 1\ldots m}$ and the input image $\mathcal{I}$. The advantage in this reformulation is that cross-correlation can be performed efficiently in frequency domain using the following relation:

$$\mathcal{I} \otimes \Phi = \mathcal{F}^{-1}(\mathcal{F}(\mathcal{I}) \bullet \mathcal{F}^*(\Phi))$$

The 2D fast Fourier transform (2D FFT) of a $N \times N$ test image $\mathcal{I}$ requires $O(N^2 Log N^2)$ computation steps. The 2D FT of the filters $(\Phi_i)_{i\in 1\ldots m}$ can be computed off-line since they are constant parameters of the network independent from the image. A 2D FT of the test image has to be computed, therefore the total number of FT to compute is $m+1$, yielding a total of $O((m+1)N^2 Log N^2)$ computation steps. The speed-up factor is $\frac{mn^2}{(m+1)LogN^2}$. In our experiments we used 25 hidden units (i,e. m=25) and 25×25 pixels sub-images (i.e. n=25). The curve giving the speed-up factor with respect to image size is shown in Figure (2).

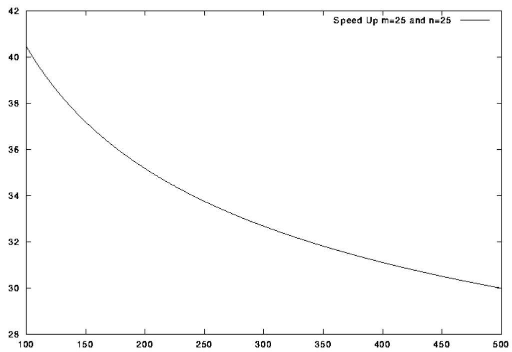
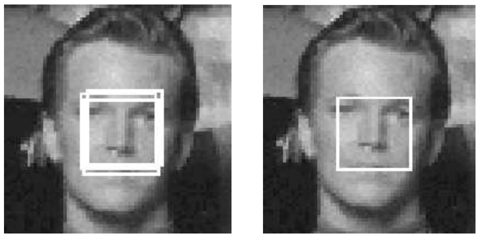


Figure 2: Speed-up curve with respect to N (Image size $N \times N$)

3 Face detection using MLP.

We applied the reformulation of MLP in terms of Fourier transform to the face detection task. The training images are 25×25 pixels large and represent human faces and non-face patterns. The examples were taken from the M2VTS [5] database which contains frontal views of 37 different persons. The negative examples were generated from images without faces (mostly texture images) and also by using a bootstrap procedure [7]. The MLP was trained on 1500 face examples and 4000 non-face examples.

The faces in the training set have slightly different scales. There are also some shifted faces (1 to 4 pixels off center). Face images often lead to multiple detections (see Figure 3 (a)), whereas non-face images mainly lead to single detections in the case of false detection. This information was used to remove a large number of false detections.



(a) (b)

Figure 3: Example of multiple detections

3.1 Grey-level normalization

In order to achieve a detection robust toward illumination changes, we transform the images of the learning set into zero-mean normalized vectors. If $\{X_i, i \in [1..L]\}^3$ is the set of images, the following holds for the transformed images $(\tilde{X}_i)_{i\in[1..L]}$:

$$\| \tilde{X}_i \| = 1 \text{ and } mean(\tilde{X}_i) = 0,\ \forall i \in [1..L]$$

In the recognition step, the data must be also normalized and have zero-mean. A problem arises

[3] L is the size of the training set

here: the normalization and the centering[4] if applied on the whole image will not be guaranteed locally in the sub-windows.

The *normalization and centering must be applied on the extracted sub-image and not on the whole input image* and this is not straightforward since Fourier-based convolution is a global processing. The reformulation of the local normalization and centering of the data in the FT framework in [1] shows that convolving a centered image $\hat{I}$ with a filter ϕ is equivalent to convolving the non-centered image I with the centered filter $\hat{\phi}$. The centered filter can be computed off-line and thus does not generate extra computation.

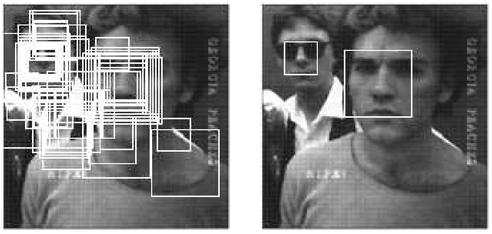
4 Multiscale face detection.

Multi-scale face detection is achieved by building a set of multi-resolution images. The face detector is then applied at each resolution and the final output is a combination of the different levels.

Note that the FT of the lower resolution images is not computed due to the scaling property of the FT. If $f(x, y)$ is the original image and $F(u, v)$ its Fourier transform and $g(x, y)$ is the sub-sampled (by a factor α in each direction) image then we have the following property:

$$FT(g(x,y)) = G(u,v) = \frac{1}{\alpha^2} F\left(\frac{u}{\alpha}, \frac{v}{\alpha}\right) \quad (4)$$

Figure 4 (a) displays the detections at all levels of resolution (the level of resolution is given by the size of the square). The final output is given in Figure 4 (b). It takes into account the number of times a region was detected as face (for more details see [4]).



(a) (b)

Figure 4: Multiscale face detection

[4] *Centered image* stands for zero-mean image

5 Experimental Results

5.1 Still images

Testing was performed on realistic and complex images extracted from the CMU database[5]. Experiments were performed on test set A (40 images) from CMU database. The achieved detection rate using a single MLP (30 neurons in hidden layer) is in the range 70% to 75% of faces detected (see Figure 5 (a)-(i)). The system proposed by Rowley achieves a detection rate ranging from 69% to 85% depending on the used heuristic. Our approach generates more false alarms that could be reduced by combining multiple networks and by increasing the size of the training database. The higher performance obtained by Rowley [7] is likely to be due to a much larger training set and due to a combination of multiple modular networks.

5.2 Video Sequences

This algorithm was also applied to face detection in live video sequences using a camera and a video-board. The algorithm runs on an UltraSparc 30 with an Osprey 1500 video board. The processed frames are 192×144 pixels large. The face detector was tested in an office environment with different illumination conditions and different backgrounds. Each frame is processed in about 0.7 seconds without using any temporal or spatial information: the face detection is performed over the whole frame at each step. We expect to reduce the processing time by introducing temporal knowledge. The system showed good performance even if multiple faces were present in the frame. It was tested with different people (visitors from outside the institute). Some MPEG movies of face detections are available on the WWW[6]

6 Conclusion

We proposed a simple and flexible MLP architecture for fast object detection, (face detection for example). The main contribution of the paper was to describe a method which reduces dramatically the computation time of a MLP based

[5] http://www.cs.cmu.edu/~har

[6] http://www.idiap.ch/vision/facerecognition.html

detector without altering the performances. The reformulation of MLP in terms of filter convolutions enabled us to speed-up significantly the processing time. Classical approaches have to preprocess the data during run-time for normalization purpose, in our case no pre-processing is needed since the normalization is incorporated directly in the weights of the network (i.e. coefficients of the filters). The same algorithm can be used to detect other features (eyes, mouth, nose etc...) separately by changing the learning set.

Acknowledgment

This work was supported by the Swiss Federal Office for Education and Science and the ACTS-M2VTS project. The author used the test database provided by CMU (H. Rowley, S. Baluja and T. Kanade) and AI Lab. MIT (K. Sung and T. Poggio).

References

[1] Souheil Ben-Yacoub. Fast object detection using MLP and FFT. IDIAP-RR 11, IDIAP, 1997.

[2] R. Chellappa, C.L Wilson, and C.S Barnes. Human and machine recognition of faces: A survey. Technical Report CAR-TR-731, University of Maryland, USA, 1994.

[3] M. Collobert, R. Feraud, G. Le Tourneur, and O. Bernier. Listen: A system for locating and tracking individual speakers. In 2^{nd} *International Conference on Automatic Face and Gesture Recognition*, pages 283–288, Oct. 1996.

[4] B. Fasel. Fast multi-scale face detection. IDIAP-COM 98-04, IDIAP, 1998.

[5] S. Pigeon. The m2vts multimodal face database (release 1.00). *CEC ACTS/M2VTS Deliveralble AC102/UCL/WP1/DS/P/161*, 1996. http://www.tele.ucl.ac.be.

[6] M.J.T Reinders, R.W.C Koch, and J.J Gerbrands. Locating facial features in image sequences using neural networks. In 2^{nd} *International Conference on Automatic Face and Gesture Recognition*, pages 230–235, Oct. 1996.

[7] H.A. Rowley, S. Baluja, and T. Kanade. Human face detection in visual scenes. Technical Report CMU-CS-95-158R, Carnegie Mellon University, 1995.

[8] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, 1986.

[9] K. K. Sung and T. Poggio. Example-based learning for view-based human face detecttion. Technical Report AI-Memo 1521, Massachusetts Inst of Technology, Cambridge, MA, USA, 1994.

[10] M. A. Turk and A. P. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3 (1):71–86, 1991.

[11] D. Valentin, H. Abdi, A.J Otoole, and G.W Cottrell. Connectionist models of face processing: A survey. *Pattern Recognition*, 27:1209–1230, 1994.

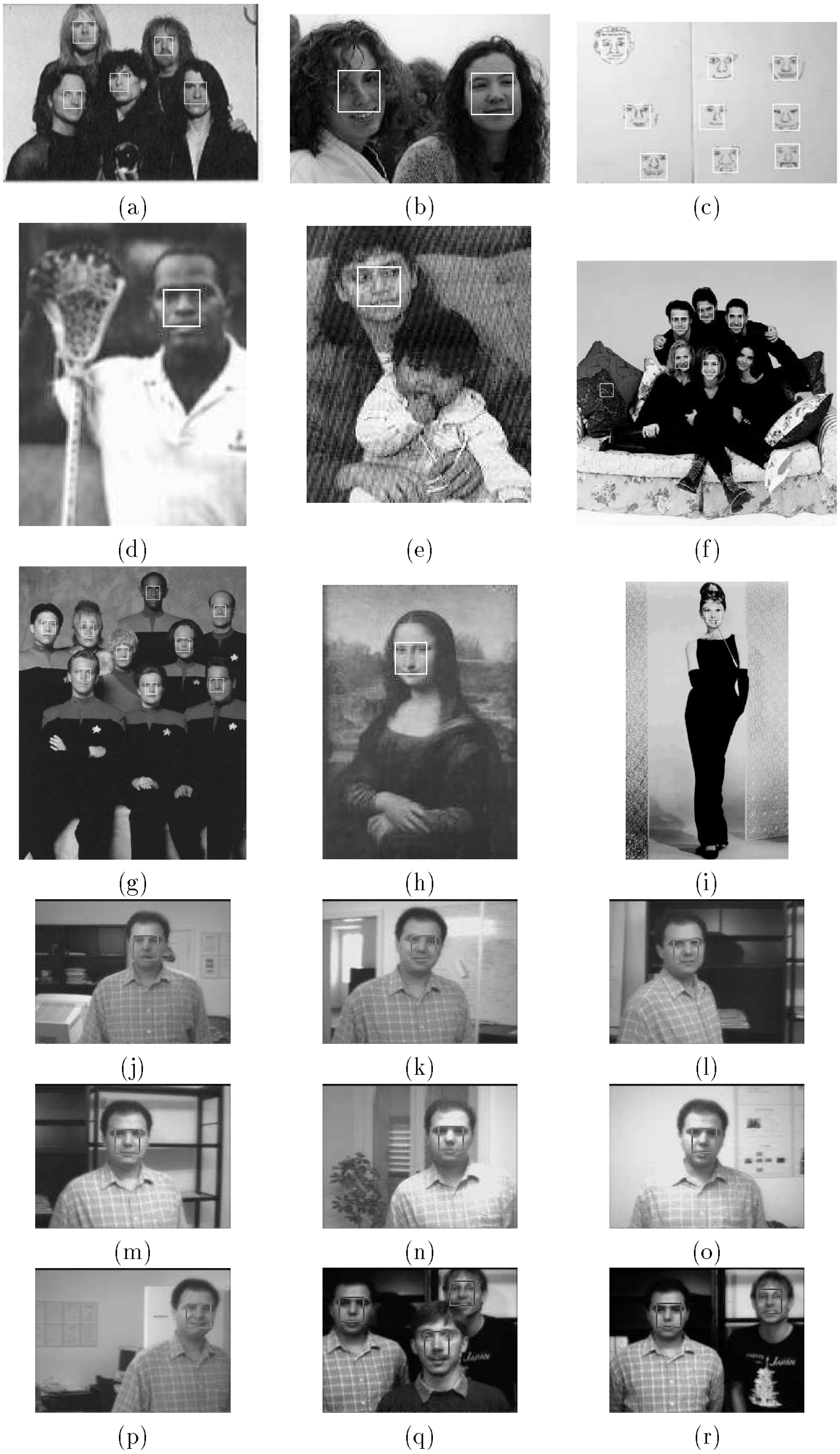


Figure 5: Face detection results on static images and video sequences.