IDIAP RESEARCH REPORT

IDIAP
Martigny - Valais - Suisse

Fast Object Detection using MLP and FFT

Souheil Ben-Yacoub [a]

IDIAP–RR 97-11

November 1997

submitted for publication

Dalle Molle Institute for Perceptive Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

[a] IDIAP, CP592 1920 Martigny, Switzerland. email: sby@idiap.ch

IDIAP Research Report 97-11

Fast Object Detection using MLP and FFT

Souheil Ben-Yacoub

November 1997

SUBMITTED FOR PUBLICATION

Abstract. We propose a new technique that speeds up significantly 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 Fourier transform for an efficient computation of the neural activities. A formal proof and a complexity analysis are presented. Finally, some examples illustrate the approach.

1 Introduction

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

The large variability of human faces causes major difficulties in the design of a model that could encompass all possible faces [2]. Appearance-based approaches as well as learning-based approaches seem to be better suited for such a task. A set of representative faces is necessary to find the implicit model.

Eigenfaces [9] were used to modelize 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 is that the set of faces are localized 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 [8] were 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 investigations on neural networks as a tool for face detection have shown their reliability and robustness for such a task [10]. However, the time consuming processing [4, 3] needed by the neural networks has prevented them from being a practical tool.

A neural network based face detector was proposed in [5], 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. Although some strategies were used to reduce the time complexity, this also reduced the performance of the system.

We present an approach that can speed-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 convolutions. Performing the convolution in the frequency domain enables to achieve a speed-up ranging from a factor 8 to 14 for image sizes ranging from 100×100 pixels to 300×300 pixels. The characteristic of this approach is that it reduces considerably the computation time while maintaining identical performances. This methods enables a fast computation of the MLP neuron activities, but does not reduce or improve the performances of a MLP based detector. It can be applied to any object detection system based on MLPs.

In Section 2, we shortly describe the MLP architecture and introduce the basic notations. Section 3 presents the reformulation of MLP in terms of filter convolution processing and a complexity analysis. Section 4 addresses face detections with MLP, and outlines the algorithm and complexity analysis. Section 5 presents the multiple scale detection issue and its computational complexity. Some experimental results are illustrated in Section 6. We conclude in the last section and outline further developments and improvements.

2 The 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

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

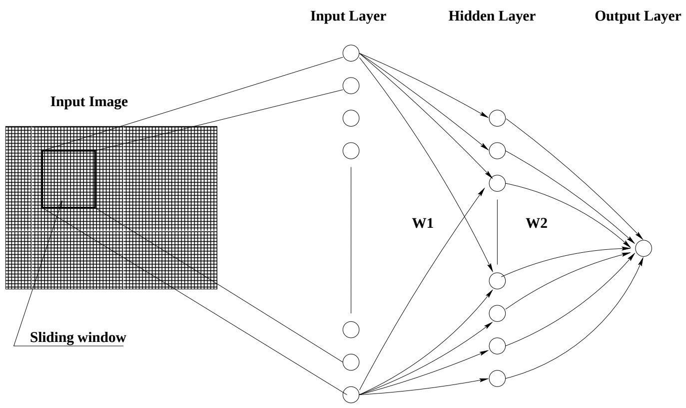

Figure 1: MLP architecture for object detection

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.

W_1 be the weight matrix between the input layer and the hidden layer, and W_2 be the weight matrix between the hidden layer and the output unit. Let b_1 and b_2 be the thresholds associated to the hidden and output units and g be the activation function. The activities (or output) of the hidden units and the output unit are given by:

$$H = g(W_1 I + b_1)$$

$$O = g(W_2 H + b_2)$$

The activity of a particular neuron i in the hidden layer H can be written:

$$h(i) = g(\sum_{j=1}^{n^2} W_{1i}(j) I(j) + b_1(i)) \tag{1}$$

Similarly, the output layer activity is:

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

The training algorithm is based on the back-propagation of the error which is now a well-known technique [6]. 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 minimised by back-propagating it and by adjusting the weights.

During the recognition step, 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.

Our new approach is intended to reduce this computational burden. Rewriting the MLP in terms of filter convolutions enables to alleviate the time complexity issue.

3 MLP as a bank of filters.

Let $w1_i$ be the vector of weights needed to compute the activity of the hidden neuron i. The vector $w1_i$ of size n^2 can be represented as a $n \times n$ matrix Φ_i. The equation (1) in a 2D space, will look like:

$$h_i = g(\sum_{j=1}^{n}\sum_{k=1}^{n}\Phi_i(j,k)I(j,k) + b1(i)) \tag{2}$$

The expression is obtained for a particular sub-image I. We can extend the equation to the global input image $\mathcal{I}$. Let's suppose that we are processing the sub-image I located at position (r,c) in $\mathcal{I}$. The activity of the hidden unit i for a sub-image located at (r,c) is now:

$$h_i(r,c) = g(\sum_{j=-\frac{n}{2}}^{\frac{n}{2}}\sum_{k=-\frac{n}{2}}^{\frac{n}{2}}\Phi_i(j,k)\mathcal{I}(r+j,c+k) + b1(i)) \tag{3}$$

The equation (3) gives the activity of the hidden neuron i with regard to the receptive field located at (r,c). This equation can also be formulated as a convolution:

$$\mathcal{H}_i = g(\Phi_i \otimes \mathcal{I} + B_1), \text{ where } B_1(k,l) = b_1 \ \forall (k,l) \in [1..n]^2 \tag{4}$$

The matrix $\mathcal{H}_i$ is the activity matrix of the hidden unit i. From equation (4), we can say that $\mathcal{H}_i(j,k)$ is the activity (or output) of the hidden unit i 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:

$$\mathcal{O} = g(\sum_{i=1}^{m} w2_i\mathcal{H}_i + b_2) \tag{5}$$

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 Section 2, to compute the final output, a sliding window was parsing the whole image yielding a time-consuming computation of the hidden units activity. In the formulation we propose in equation (4), the activity is expressed in terms of convolution between a bank of filters $(\Phi_i)_{i\in 1...m}$ and the input image $\mathcal{I}$. The advantage in this reformulation is that convolution can be performed efficiently using Fourier transformation.

3.1 Complexity analysis.

The convolution can be expressed in terms of a Fourier transform[2] $\mathcal{F}$ [1]:

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

The 2D Fourier transform (2D FT) of a $N \times N$ test image $\mathcal{I}$ requires $O(N^2Log^2N)$ computation steps. The 2D FT of the filters $(\Phi_i)_{i\in 1...m}$ 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 $O((m+1)N^2Log^2N)$ computation steps. The speed-up factor is $\frac{mn^2}{(m+1)Log^2N}$. 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).

[2] $\mathcal{F}^*$ is the conjugate Fourier transform

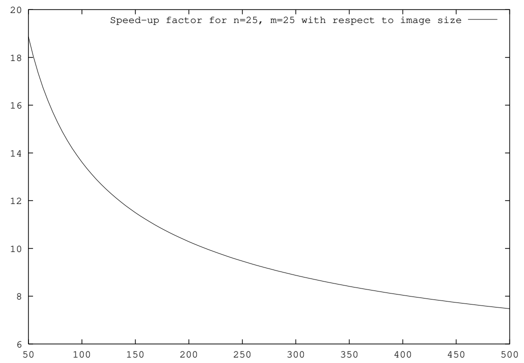

Figure 2: Speed-up curve with respect to n (Image size $n \times n$)

4 Face detection using MLP.

We applied the reformulation of MLP in terms of Fourier transform for the face detection task. We have created a training set of face images. The size of the training images is 25×25 and contain the face of the person as shown on Figure 5. The examples were taken from the M2VTS [7] database which contains frontal views of 37 different persons. The negative examples were generated from images without faces (mostly texture images). We have used a learning set with 200 faces and 450 non-face examples.

In order to achieve a detection robust toward illumination changes, we have to 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 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. We have to reformulate the local normalization and centering of the data in the FT framework.

4.1 Centering and Normalizing locally.

Let $[\tilde{X}]_{rc}$ be the zero-mean normalized sub-image located at (r, c) in the input image $\mathcal{I}$,

$$[\tilde{X}]_{rc} = \frac{[X]_{rc} - \bar{x}_{rc}\mathbf{1}_{nn}}{\| [X]_{rc} - \bar{x}_{rc}\mathbf{1}_{nn} \|}$$

where $\bar{x}_{rc}$ is the mean value of the sub-image located at (r, c), and $\mathbf{1}_{nn}$ is a $n \times n$ matrix where every element is 1. We are interested in computing the convolution between the sub-image $[\tilde{X}]_{rc}$ and the filter $\mathbf{\Phi}_i$:

$$[\tilde{X}]_{rc} \otimes \mathbf{\Phi}_i = \frac{[X]_{rc} \otimes \mathbf{\Phi}_i - \bar{x}_{rc}(\mathbf{1}_{nn} \otimes \mathbf{\Phi}_i)}{\| [X]_{rc} - \bar{x}_{rc}\mathbf{1}_{nn} \|} \tag{6}$$

$$\bar{x}_{rc} = \frac{1}{n^2} \sum_{j=1}^{n} \sum_{k=1}^{n} X_{rc}(j, k) = \frac{[X]_{rc} \otimes \mathbf{1}_{nn}}{n^2} \tag{7}$$

[3] L is the size of the training set

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

The norm of the centered sub-image at location (r, c) can also be computed using Fourier transforms:

$$\begin{aligned} \| \, [X]_{rc} - \bar{x}_{rc}\mathbf{1}_{nn} \, \| &= \sqrt{\sum_{i=1}^{n^2} (x_{rc}(i) - \bar{x}_{rc})^2} \\ &= \sqrt{(\sum_{i=1}^{n^2} x_{rc}(i)^2) - n^2 \bar{x}_{rc}^2} \end{aligned} \tag{8}$$

$$\sum_{i=1}^{n^2} x_{rc}(i)^2 = [X_{rc}^2] \otimes \mathbf{1}_{nn} \tag{9}$$

Combining equations (6), (7), (8) and (9), we get the expression:

$$[\tilde{X}]_{rc} \otimes \Phi_i = \frac{[X]_{rc} \otimes \Phi_i - [X]_{rc} \otimes \mathbf{1}_{nn} \frac{\mathbf{1}_{nn} \otimes \Phi_i}{n^2}}{\sqrt{[X_{rc}^2] \otimes \mathbf{1}_{nn} - \frac{([X_{rc}] \otimes \mathbf{1}_{nn})^2}{n^2}}} \tag{10}$$

In equation (10), the expression $\frac{\mathbf{1}_{nn} \otimes \Phi_i}{n^2}$ is indeed the mean value of the coefficients of the filter Φ_i, that we will from now call σ_i. Thus equation (10) is equivalent to:

$$[\tilde{X}]_{rc} \otimes \Phi_i = \frac{[X]_{rc} \otimes (\Phi_i - \sigma_i \mathbf{1}_{nn})}{\sqrt{[X_{rc}^2] \otimes \mathbf{1}_{nn} - \frac{([X_{rc}] \otimes \mathbf{1}_{nn})^2}{n^2}}} \tag{11}$$

Finally, equation (11) gives the expression (in terms of convolution) of the activity of the hidden unit i when the sub-image is located at position (r, c) in image $\mathcal{I}$. One can observe from this equation that convolving a centered image with a filter is equivalent to the convolution of the non-centered image with the centered filter[5]. We can therefore save the normalization and centering step for the test images by centering and normalizing the filters off-line.

In order to have the activity of the hidden unit i when considering all possible sub-images, we get the matrix activity $\mathcal{N}_i$:

$$\mathcal{N}_i = \frac{\mathcal{I} \otimes (\Phi_i - \sigma_i \mathbf{1}_{nn})}{\sqrt{\mathcal{I}^2 \otimes \mathbf{1}_{nn} - \frac{(\mathcal{I} \otimes \mathbf{1}_{nn})^2}{n^2}}} \tag{12}$$

The element $\mathcal{N}_i(r, c)$ represents the activity of the hidden unit i when the sub-image is located at (r, c). The activity matrix is indeed a convolution between the image and the centered filter (zero mean) divided by the local norm of the sub-image.

The final output of the neural network is a simple linear combination of the different activity matrices.

$$O = g(\sum_{i=1}^{m} w2_i \mathcal{N}_i + B_2)$$

4.2 Algorithm and complexity analysis.

In order to compute the activity matrix defined in equation (12), a certain number of Fourier transforms must be computed. The FT of the centered filter can be computed off-line since it is a constant of the algorithm, the same holds for the Fourier transform of $\mathbf{1}_{nn}$. We can now outline the algorithm:

[5] Mathematically these is: $[\tilde{X}]_{rc} \otimes \Phi_i = [X]_{rc} \otimes \tilde{\Phi}_i$

- *Data*:
  - $n \times n$: filter size.
  - m: number of hidden units (or neurons).
  - FT of the centered filters $\tilde{\Phi}_i = \Phi_i - \sigma_i \mathbf{1}_{nn}$
  - FT of $\mathbf{1}_{nn}$
- *Input*:
  - image $\mathcal{I}$ with size $N \times N$.
- *Computing steps*:
  - $\mathcal{F}(\mathcal{I})$ and $\mathcal{F}(\mathcal{I}^2)$.
  - $\mathcal{I}^2 \otimes \mathbf{1}_{nn} = \mathcal{F}^{-1}(\mathcal{F}(\mathcal{I}^2) \bullet \mathcal{F}^*(\mathbf{1}_{nn}))$
  - $\mathcal{I} \otimes \mathbf{1}_{nn} = \mathcal{F}^{-1}(\mathcal{F}(\mathcal{I}) \bullet \mathcal{F}^*(\mathbf{1}_{nn}))$
  - For $i \in 1..m$.
  - compute $\mathcal{I} \otimes \tilde{\Phi}_i = \mathcal{F}^{-1}(\mathcal{F}(\mathcal{I}) \bullet \mathcal{F}^*(\tilde{\Phi}_i))$
  - compute $\mathcal{N}_i$ according to equation (12).
  - End For

Figure 3: Outline of the algorithm

The complexity analysis shows that the total number of computation steps is in the order of $O((m+4)N^2 Log^2 N)$. The achieved speed-up is in the order of $O(\frac{mn^2}{(m+4)Log^2 N})$. The Figure 4 shows the speed-up curve for m=25 and n=25 (values of our experiments). At run-time no pre-processing of the data is needed since normalization is taken into account in the filter coefficients, this saves computation steps also.

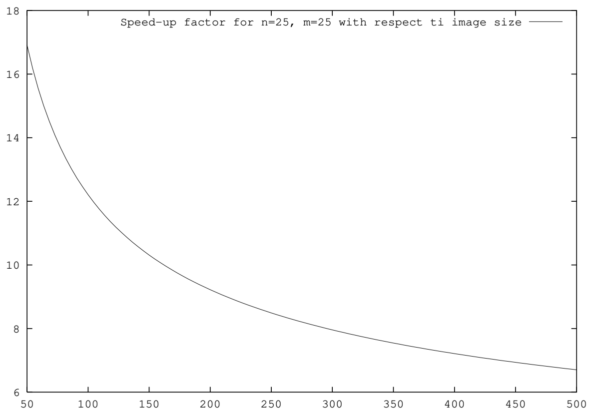

Figure 4: Speed-up curve (considering normalization) with respect to image size

5 Multiple scales face detection.

The neural network was designed and trained to detect faces with small sub-images (typically 20×20 to 30×30), that is for a given scale. Classical solutions build a pyramid of the input image, which generates a set of images at different resolutions. The face detection is then performed at each resolution and the final result is a consensus between the different scales. This approach increases the number of windows to be observed and hence the sliding-window approach will take much more time, and the number of processing steps will increase by an order of 33% (corresponding to the pyramid size).

In the approach we propose, based on a FT processing, the same algorithm as described in the previous section will be applied at different scales. One could expect to have also an increase by an order of 33%. There is however a little difference, since the FT of the new scales do not need to be computed. This is due to a property of the Fourier transform. If $f(x,y)$ is the original image and $g(x,y)$ is the sub-sampled (by a factor of 2 in each direction) image then we have the following property:

$$g(x,y) = f(2x,2y) \tag{13}$$
$$F(u,v) = FT(f(x,y)) \tag{14}$$
$$FT(g(x,y)) = G(u,v) = \frac{1}{4}F(\frac{u}{2},\frac{v}{2}) \tag{15}$$

This implies that we do not need to re-compute the Fourier transform of the images, it can be directly obtained from the original FT. This has as consequence that the processing needs $O((m+2)N^2 Log^2 N)$, thus the speed-up factor will be $O(\frac{mn^2}{(m+2)Log^2 N})$ for the upper levels. It can be shown that the hierachical processing with our method increases the number of processins steps by less than 33%.

6 Experimental Results

We evaluated the method for a small face detection task. The purpose of the test was to verify the computation time and performance of the system rather than perform a state-of-art face detection. The training was performed on a subset (200 face images) of the M2VTS face database [7]. It contains frontal views of Caucasian males and females on an almost uniform background. People with glasses and beard were also present in the database. The training set of 25×25 pixels large images was processed to have centered and normalized images.

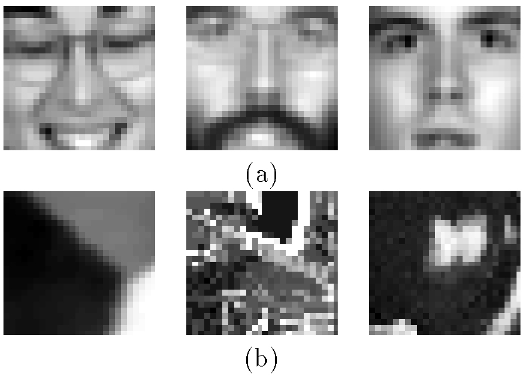

Figure 5: Samples from the training set, (a) faces, (b) non-faces

Negative examples (non face images) were taken from textured images. Training was stopped when the output error was below some threshold (0.1 in our experiments). The activation function used for the MLP is the sigmoid defined as follow:

$$g(x) = \frac{1}{1+e^{-x}}$$

The testing of the MLP face detector was first performed on the ORL[6] face database. This database contains 400 mug-shots of 40 different people (10 images per person). Although the images seem not to be complicated for a face localization, the database contains 60% of non-frontal views with different illuminations. It was a difficult test for our network which was trained on frontal views only. It also contains multi-racial images, whereas our network was trained on white Caucasian people only. The detection rate achieved by the MLP face detector was 95% on the ORL database. It achieved a surprisingly good detection rate for non-frontal views.

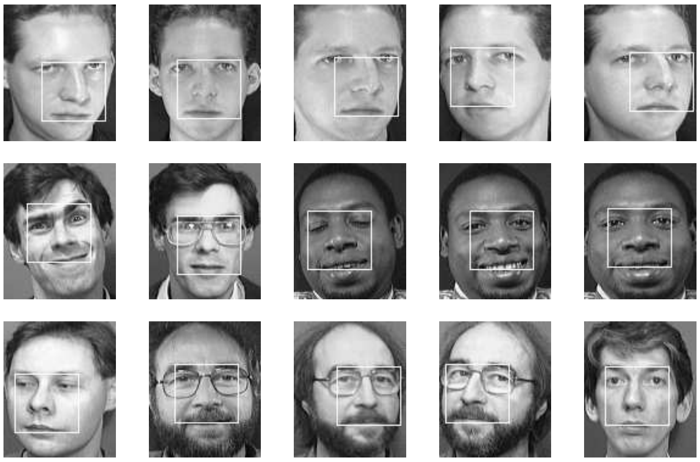

Figure 6: Testing results on the ORL database

Testing was also performed on more realistic and complex images, like the ones of the CMU database[7]. The network detected almost all faces on these examples. The network was also able to detect "cartoon-like" faces whereas it was trained only real human faces. The false detection rate is quite high when compared to standard methods. This can be explained by the size of our training set which is much smaller than the one used in [5]. This can be improved by training on a larger database with a high number of non-face examples and using a bootstrap technique.

[6] http://www.cam-orl.co.uk/facedatabase.html
[7] http://www.ius.cs.cmu.edu/IUS/har1/har/usr0/har/faces/test/

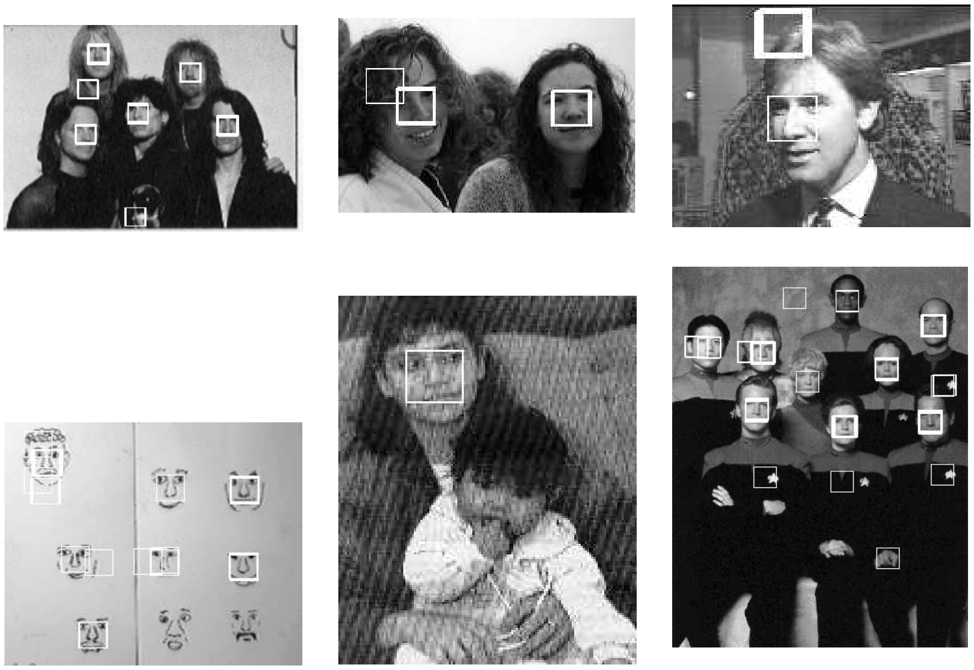

Figure 7: Some face detection results of our network

7 Conclusion

We proposed a simple and flexible MLP architecture for object detection, i.e. face detection. The main contribution of the paper was to describe a method which reduces dramatically the computation time of a MLP based detector without altering the performances. The reformulation of MLP in terms of filter convolutions enabled us to speed-up significantly the processing time, a formal proof of the gain in speed was proposed. Classical approaches have to preprocess the data during run-time in order to normalize them, in our case no pre-processing of the data is needed since the normalization is taken into account 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.

This simple algorithm was applied on test examples and the first results showed good detection rate. The main drawback is the high false detection rate that can be reduced by further improvements:

- Increasing the size of the learning set, and especially the "non-face" examples by using a bootstrap technique.
- Using a consensus between networks trained on different data sets (like in [5]).
- Training different networks on faces, eyes, nose, mouth. The false detections of faces could be removed by checking if at least one eye or nose or mouth was detected in the same area (or observation window).

Acknowledgment

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). The face databases were provided by Olivetti Research Lab. and M2VTS. The author thanks J. Luettin for his comments and useful discussions.

References

[1] K.R Castleman. *Digital Image Processing.* Prentice Hall, 1996.

[2] R. Chellappa, C.L Wilson, and C.S Barnes. Humand 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] 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.

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

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

[7] 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.

[8] 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.

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

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