Adaptive Logic Networks for Facial Feature Detection

D.O. Gorodnichy , W. W. Armstrong, X. Li

Dept. of Computing Science, University of Alberta, Edmonton, AB, Canada T6G 2H1
Email: {dmitri,arms,li}@cs.ualberta.ca

Abstract. The task of automatic facial feature detection in frontal-view, ID-type pictures is considered. Attention is focused on the problem of eye detection. A neural network approach is tested using adaptive logic networks, which are suitable for this problem on account of their high evaluation speed on serial hardware compared to that of more common multilayer perceptrons. We present theoretical reasoning and experimental results. The experiments are carried out with images of different clarity, scale, lighting, orientation and backgrounds.

1 Introduction

Facial feature detection is very important because of its application to 1) videotelephony, 2) multimedia databases and of course, 3) automatic face recognition. In most of these applications this detection must be done in real time. Most conventional approaches such as template matching, contour following, Karhunen-Loeve expansion, algebraic moments and Hidden Markov Models are quite complex and slow in operation (e.g. see [1, 2]), yet they still do not exhibit high accuracy and robustness. Problems are observed with images of different clarity, scale, lighting, orientation and background.

Artificial neural networks (NNs) can be applied where algorithmic methods are too hard or costly to develop or are not known to exist (e.g. see [3]). NNs can discover underlying statistical regularities of training patterns. This suggests that if a NN is given a facial image as a pattern of intensity values, then it may be able to find the regularities of the face: the eyes, mouth etc.

In the next section we describe the neural network approach in detail. In Section 3, we introduce Adaptive Logic Networks (ALNs) and show the advantages of ALNs over other networks. In Section 4, we show how ALNs can be applied to the problem of eye detection, giving data obtained by simulations. Further steps for improving performance are discussed at the end the paper.

2 Neural Networks and Face Recognition

A lot of papers have been written about applying NNs to face recognition. Research in this area can be divided into two main categories:
1: *using NNs for face recognition*, when a whole picture of a face is considered as a pattern and the objective is to identify the face [4], and

2: *using NNs for locating facial features*, where a facial image is considered as a set of patterns, each of which contains information about some part of the face, and the objective is to extract this information, i.e. to recognize those patterns that contain a feature of interest.

In this paper we consider the second task, the presentation of which follows.

We are given a set of head-and-shoulder images quantized to 256 grey levels. Some of the images are put into a training set. During the training stage, a network is provided with both "eye" and "non-eye" patterns. Each pattern is a vector $\mathbf{x} \in \Re^N$, $x_i \in [0, 255]$, obtained by scanning an image using a small window of size N, called a *peephole mask*. This is extended with the desired output value y, which depends on whether the mask is centered on an eye or not. Eyes are located manually, while "non-eyes" are picked automatically (see Fig. 1). After training, an image is scanned using the same peephole mask, and for pixel-pattern $\mathbf{x}$ the network produces output y.

Several questions arise:

Q1: How many faces should be used as prototypes?

Q2: What should the size N of the peephole mask be? — It is desirable that it be as small as possible but large enough to contain all pixels important for recognition.

Q3: What should the shape of this mask be? — It is clear that some pixels around an eye are more informative than others.

Q4: What should the output values be? — Binary, i.e. $y \in \{0, 1\}$ ($\{\text{No}, \text{Yes}\}$) or discrete, i.e. $y \in \{y_0, y_1, ..., y_k\}$, where values between 0 and 1 are assigned to masks close to the centre of an eye?

Q5: How many "non-eye" patterns should be used to for training? — It is obvious that using all non-eye patterns is redundant, as many patterns will be the same, e.g. background, hair.

Q6: Is it worth preprocessing sample patterns before presenting them for training? — e.g. by normalizing or resampling?

In order to resolve these questions let us first study the results obtained by other researchers.

Most of the papers consider the multi-layered perceptron with the Back-Propagation learning rule (BP) as suitable for the task of eye detection [5, 8, 6, 7]. This is because its architecture provides a feedforward association of pixel values in an input layer and an answer ("eye", "non-eye") in the output layer, which consists of a single "neuron". This is also the case with an ALN. Some other types of NNs have also been applied to eyes: Radial Basis Function networks [9] and Self-Organizing Maps (SOM) [5, 4].

Following the order of questions in Section 3.1, let us describe the observations and solutions proposed by other researchers. First, experiments have been carried out with 256x256, 128x128 and 64x64 images, and it has been found [7] that processing of low resolution images is more robust than that of high-resolution images, since low resolution eliminates some small regions of "bad" information, while considerably increasing the calculation rate.

The number of images in most works is 60, 16 of which are used for training. The images are scanned with a *peephole mask*, the size of which is chosen to be either 16x16 or 8x8. The window is homogeneous and either full (all points are used) or sparse (every second or third point is used). This results in a 64-dimensional input vector. However, the interesting result has been obtained in [6], that for 64x64 images, a 15-dimensional input vector suffices. This result was obtained by calculating the fifteen largest eigenvalues of the covariance matrix of the original 64-dimensional vector.

In case of BP NNs, both binary and discrete output schemes are used — neither appeared to be preferred. Another interesting result [5] is that the normalization of faces does not significantly improve the performance of BP NNs. The paper [4] observes that preprocessing input vectors does not improve the performance of SOM. In subsequent sections we will show that this is not the case for Adaptive Logic Networks.

3 Adaptive Logic Networks

The *Adaptive Logic Network* (ALN) [12] can be considered as a tool for approximation of any continuous real-valued function $y = f(\mathbf{x})$ on N-dimensional space given a set of its points $x^m \in \Re^N$, $y^m \in \Re$, $m = 1..M$. In neural network terminology these points are usually referred to as *labeled sample pattern vectors.*

The main difference and advantage of the ALN over other approximation techniques is that it utilizes piecewise linear surfaces only (see Fig. 2a). Because only a few pieces are involved in computing a particular output, as may be determined using a decision tree on the components of the input, a considerable speed-up in processing is obtained. Control over the weights of the pieces can also lead to good generalization [12]. The way the pieces are put together is determined by a tree of maximum and minimum operators (Fig. 2b) acting on linear functions, which computes a y given x.

The training of an ALN consists of many multiple linear regressions working in concert with the goal of fitting the training data with low error, where error is computed by summing the squares of the distances between the approximating surface and the output values.

3.1 Advantages of the ALN

– It possible to integrate into the training procedure qualitative knowledge about the desired function; this may aid generalization and prevent a learned function from having nonsensical properties. It also tends to reduce the amount of training data required;
– It is easier to understand the function that ALN is computing compared to other neural networks since it is made up of linear pieces put together in a simple way. This is especially important for preprocessing input vectors and in designing output values for sample points.
– Generalization can be explicitly controlled using jitter, whereby additional

training data is generated by adding random amounts to the components of the x-vector.
– ALNs are very fast in evaluation. In fact, the decision tree approach enables the system to narrow the computation to about N linear pieces [12], while a BP NN has to examine all its weights. ALNs are rapidly trained too (see Section 4).

It is clear that ALNs can be applied to any problem where the data have some regularities, or in other words, are determined by some function. So far the ALN has been successfully applied in predicting financial events, machine failure and real-time control including rehabilitation of patients with spinal cord injury [10, 11].

Let us describe now how these advantages of the ALN can be used in the problem of eye detection.

4 The Performance of the ALN

In our experiments we used Atree 3.0 ALN Development System (ALN DS) [13]. In training, it takes about 30 sec for ALN DS running on HyperSPARC 90MHz to calculate all weights, i.e. to build a function $F(\mathbf{x})$, for the network with 14-dimensional input and with 500 sample patterns. In evaluation, one 64x64 image is processed in 0.8–1.2 sec (i.e. the output $y = F(\mathbf{x})$ for each pixel is calculated in about 0.4 msec).

Image database The experiments were organized as outlined in Section 2.2. The set of images consisted of twelve 256x256 images, four of which were used in training. We decreased the resolution of the original images using the *Multi-Resolution Pyramid* proposed in [7]. Thus, the actual size of processed images was 64x64. Figure 4 shows these low-resolution images. For the original images, we simply ascend in the pyramid (see [7]).

Images are picked from the database used in previous experiments [2]. In evaluation we also used two images (images 9, 10) scanned from blurred out-of-focus photo and one image (image 11) scanned from a photo not taken in front of a plain background. The results of the experiments were judged by visually determining the accuracy of eye detection.

Peephole mask Figure 3a shows peephole masks we have used in the experiments. The size of fourteen peepholes was chosen as suggested by [6]. We started with a homogeneous mask (mask 1) and this resulted in recognizing many "false" eyes, like corners of the mouth, brows etc. Then, taking into account that some pixels around an eye are more important for making a decision than others, manually designed masks were tested.

The design of a mask is determined by the desire to distinguish real eyes from "false" ones, or in other words, by what we humans think defines an eye. Mask 3 was found to be the best. We consider an eye to be a dark spot (points 1,3,4,5,6), which is usually below another dark spot – the brow (points 7,8), and which is surrounded by light tones: the cheek (point 2), forehead (points 11,12) and points 9,10. Points 13 and 14 are added to distinguish the eye from the corners of the mouth. All the results below were obtained using this mask.

Sample points In each of four training patterns, two "eye" patterns were located manually. Besides this, every fourth pixel was used to obtain a "non-eye" pattern[1]. Then we augmented the training set with manually generated "non-eye" patterns, such as noses, brows, mouths. In both training and evaluation we reduced the area of the search to the inner box $[\frac{1}{6}64,\ (1-\frac{1}{6})64]\mathrm{x}[\frac{1}{5}64,\ (1-\frac{1}{5})64]$. Overall the number of sample patterns used was in the range 130–160 per facial image.

Using ALN features With ALNs we have clear understanding of how functions are represented, in contrast to BP networks, where one cannot say what the function to be learned will look like. This helps us in choosing appropriate inputs and outputs.

Output values scheme Three output value schemes were considered: a binary scheme and two discrete schemes, depicted in Figure 3b. These schemes were designed with the following idea in mind: those input vectors that are close in 14-dimensional input space should also have close output values, as this will make the function to be learned smoother. Indeed, the results obtained with scheme 1 were much better than those obtained with the binary scheme, while those obtained with scheme 2 appeared to be the best. In further discussions we deal with the last output scheme. Sample vectors which are shown in Figure 1 are obtained with mask 3, output scheme 2 and without preprocessing.

Preprocessing We have considered two preprocessing procedures:
— *shift*: instead of the original intensities of the pixels in a mask, the differences between the intensity of a pixel and the central pixel were used (as in [4]).
— *scale*: the input vector was prenormalized by dividing each pixel by the average intensity of the pixels in a vector (as in [1]).

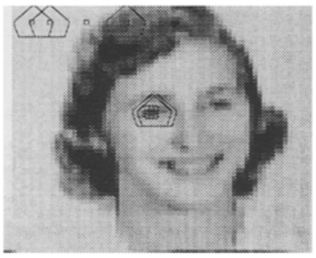
Both these procedures improved the recognition, with *shift* preprocessing achieving the most noticeable result. The results presented below are obtained using *shift* preprocessing.

Evaluation — picking "suggested eyes" During the *evaluation*, an image is scanned pixel by pixel using the peephole mask, and for each pattern obtained $\mathbf{x}^{new}$ the ALN produces an output value $y = F(\mathbf{x}^{new})$. Figure 4 shows outputs produced by the ALN for each pixel of the image. The figure shows outputs obtained for all images by the network using mask 3, output scheme 2, and *shift* preprocessing.

It has been observed that the greatest outputs of the ALN don't always correspond to eyes. Because of this, more than two pixels with the greatest outputs were picked[2]. These pixels are referred to as *"suggested eyes"* and they can be processed further if required.

The result of evaluation of the best network is shown in Figure 4, with "suggested eyes" superimposed. As can be seen, in most cases eyes are detected correctly (although sometimes along with spurious eyes). This result is noticeable, providing that it is achieved in real-time mode and with images of different quality.

[1] Using every third pixel does not appear to yield a significant difference.
[2] In our experiments we pick more than four and fewer than eight pixels.



X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	Y
85	102	111	181	118	174	123	244	212	203	197	73	221	241	10
111	85	123	185	142	102	163	237	218	214	171	74	243	234	7
102	174	85	199	117	214	111	245	199	186	163	75	199	251	7
118	117	142	85	212	172	157	181	149	188	170	144	214	228	4
181	199	185	244	85	224	179	254	118	193	221	149	218	249	4
142	118	157	111	218	117	180	185	159	211	129	117	235	227	2
117	172	118	102	199	211	142	199	129	187	180	173	195	231	2
185	181	179	237	111	199	194	254	142	217	200	159	253	240	2
199	224	181	245	102	217	185	251	117	191	194	129	198	255	2
147	205	55	144	135	234	163	137	111	251	201	74	249	197	0
128	198	182	123	137	226	205	133	144	255	217	135	240	217	0
139	153	202	143	133	196	199	154	123	253	207	137	190	215	0
...														

Figure 1. *Obtaining sample patterns for training.*

5 Discussion

In this paper we showed how to build an ALN, which is one of fastest neural networks, for the problem of eye detection in frontal-view, ID-type pictures. We highlighted those features of ALNs that make them more attractive than other networks. In particular, we showed that ALNs can benefit from using preprocessed input vectors and appropriately designed output value schemes. We also presented techniques to improve the performance of the network, such as designing a peephole mask and creating a training set.

Further possible steps for improvement of performance follow: 1) Use more facial images for training. 2) Augment the training set with more manually shown non-eyes. Here we may make use of the geometrical appearance of outputs produced by the ALN (like those in Figure 4). 3) The output scheme can be elaborated to make the function to be learned smoother. An analytical approach can be used for understanding which pixels lie close to each other in N-dimensional input space. 4) Other peephole masks can be tried. Studying psychological expects of human perception of eyes will be helpful. 5) Other heuristics should be used to pick real eyes from the limited number of "suggested eyes"; e.g. pick only those pixels which lie approximately on the same row in the image; or ignore isolated "eyes". 6) Instead of picking a predefined number of "suggested by ALN eyes" we may use an adjustable threshold for picking them: pixels with output greater then a threshold are picked, the threshold being decreased until a pair of eyes is found.

So, there are quite a few ways to improve the performance of the approach on eyes. Similarly, ALNs can be used for recognition of the mouth, brows etc.

References

1. R. Brunelli and T.Poggio Face recognition: Features versus templates, IEEE Trans. on Pattern Analysis and Machine Intelligence, 15(10), pp. 1042-1052, 1993
2. N. Roeder and X. Li, "Accuracy analysis for facial feature detection", Pattern Recognition, Vol 29, No.1, pp.143-157, 1996.
3. D. O. Gorodnichy, A Way to Improve Error Correction Capability of Hopfield Associative Memory in the Case Of Saturation, HELNET 94-95 International Workshop on Neural Networks Proceedings, Vol. I/II, VU University Press, Amsterdam, pp.198-212, 1996
4. S. Lawrence, C. L. Giles, A. C. Tsoi, A. D. Back. Face Recognition: A Hybrid Neural Network Approach, IEEE Trans. on Neural Networks, special issue on Pattern Recognition, accepted for publication.

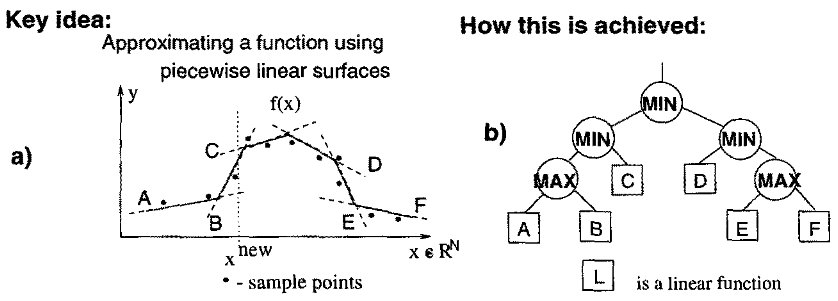


Figure 2. *Main idea behind the ALN.*

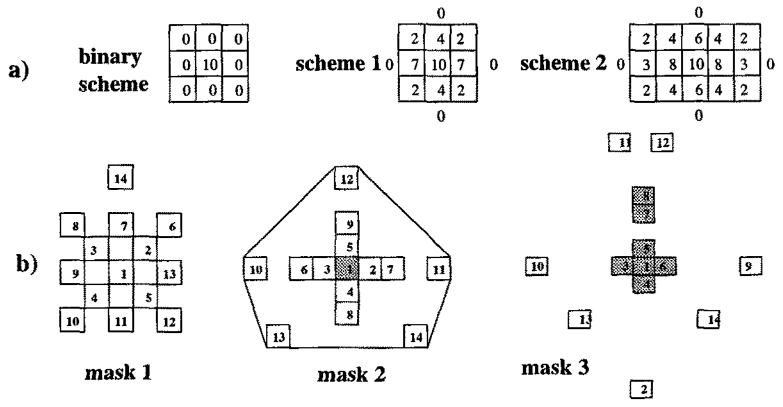


Figure 3. *Peephole masks (a) used to obtain a 14-dimensional input vector and output value schemes (b) with the numbers inside boxes indicating output values associated with the pixels.*

5. R. Hutchinson, W. Welsh. Comparison of Neural Networks and Conventional Technoques for Feature Locationin Facial Images, Proc. First IEE International Conference on ANN, pp.201-205, October 1989.
6. J. B. Waite, Training Multi-Layered Perceptron for facial feature location: a case of study, in Neural networks for vision, speech, and natural language. 1st ed. BT telecommunications series; 1. London: Chapman & Hall, 1992.
7. C C Hand, Artificial Neural Networks feature detector using Multi resolution Pyramid, ibid.
8. J. M. Vincent, Image Feature Location in Multi-Resolution Images, ibid.
9. R. M. Debenham, The detection of eyes in facial Images Using Radial Basis Fuctions, ibid.
10. W. W. Armstrong, C. Chu, M. M. Thomas, "Using Adaptive Logic Networks to Predict Machine Failure" in Proc. of the 1995 Workshop on Environmental and Energy Applications of Neural Networks", World Scientific, Richland, USA, pp. 97-107, 1995.
11. W. W. Armstrong, A. Kostov, R. B. Stein, M. M. Thomas, Adaptive Logic Networks in Rehabilitation of Persons With Incomplete Spinal Cord Injury, pp. 154-171, ibid.
12. W.W Armstrong, M.M.Thomas, Adaptive Logic Networks, sect. in C1.8 in Handbook of Neural Computation, E. Fiesler, R. Beale eds, Institute of Physics Publishing and Oxford University Press - USA, 1996, ISBN 0-7503-0312-3 (looseleaf)
13. W. W. Armstrong, M. Thomas et al, The Atree 3.0 Educational Kit with User Guide available via anonymous ftp from ftp.cs.ualberta.ca [129.128.4.241] in pub/atree/atree3/atree3ek.exe.

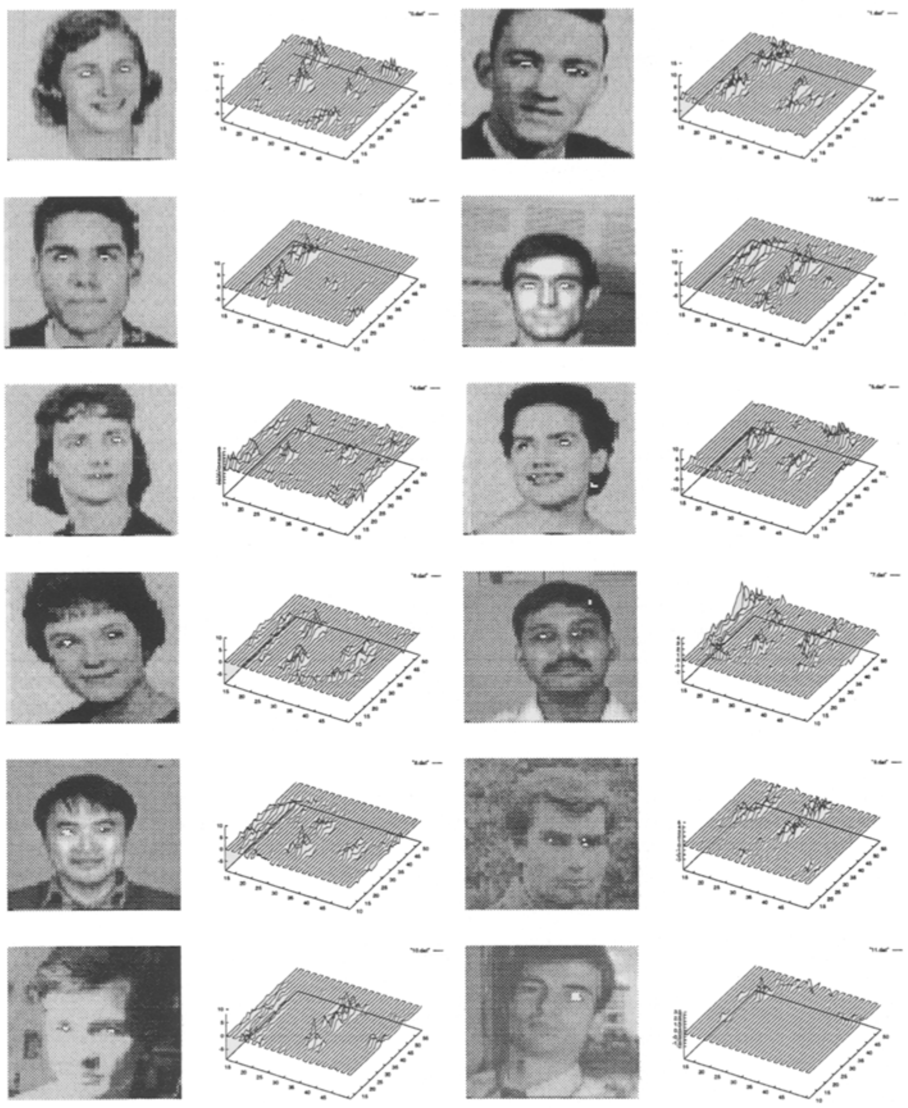


Figure 4. *The result of evaluation: Outputs produced by the ALN are shown along with the images having "suggested eyes" superimposed in white. The first four images are used in training.*