Face Detection with Neural Networks

Samuel Rota Bulò

Università Ca' Foscari Venezia
DAIS

October 15, 2012

Contents

Problem description

The face detection problem consists in finding the position of faces within an image.

- Important stage because it is auxiliary to other higher level stages, e.g., face recognition.
- The research focused his attention on this topic mainly since the 90s.

State-of-the-art

- 70s: simple techniques with frontal faces and uniform background (portraits) [12]
- 70s-90s: low interest
- 90s: growing interest towards face recognition and consequently face detection [1]
- We distinguish two main classes of techniques
  1. feature-based: low-level visual features extraction (color, edges, etc.) followed by detection of face features (mound, lips, nose, eyes, etc.) and its relative placements.
  2. image-based: reformulation of the face detection problem as a pattern recognition problem; supervised learning.

Feature-based face detection

- Extraction of low-level visual features (edges, brightness, color, movement, etc.) from pixel properties (position, color, etc.)
- Using our knowledge of the face geometry, we abstract low-level features into high-level face features (mound, nose, eyes, etc.)
- Used for real-time systems
- Main techniques
  1. Feature searching: we look for face features and its relative placements in order to detect a face. [3, 5]
  2. Constellation analysis: facial features are grouped into face-like constellations using more robust face models based on statistical analyses. [8, 14]
  3. Active Shape Models: techniques where the face features are located with "active shapes", i.e. models that if released near a feature, interact with the local image and modify themself to take the shape of the feature. [6, 15, 2]

Image-based face detection

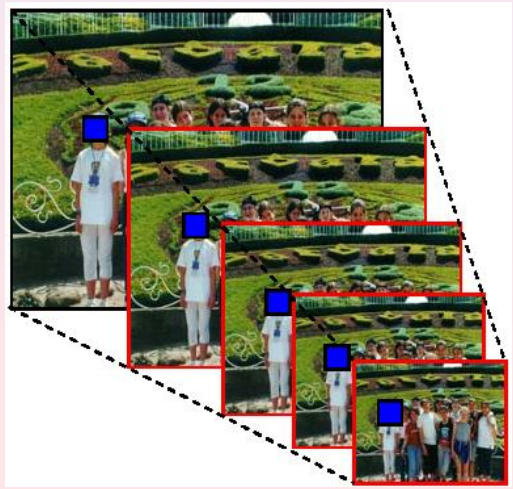
- Face detection=pattern recognition
- No direct knowledge about faces is given (face knowledge), but it is inferred from examples.
  - $\uparrow$ avoid errors deriving from wrong or incomplete face models
  - $\downarrow$ crucial is the training stage
- More robust to environment or physical changes of a face (illumination, background, glasses, moustache, etc.)
- Often associated with image scanning and scaling techniques
- Main techniques
  1. Linear Subspace Methods: *Principal Component Analysis* (PCA) , *Linear Discriminant Analysis* (LDA) e *Factor Analysis* (FA) [9, 13] .
  2. Statistical approaches: information-theory based systems, support vector machines e Bayesian decision rules.
  3. Neural networks: neural networks based techniques are popular for pattern recognition and are inspired by the human brain. [4, 7, 10, 11]

Proposed solution

- From H.A. Rowley, S. Baluja, T. Kanade, "Neural Network-Based Face Detection", TPAMI, 1998. [10]
- We focus on a simplier subproblem.

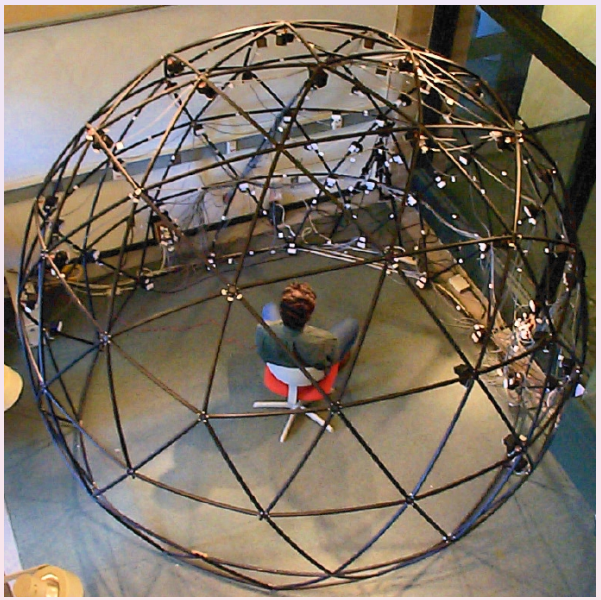
Given a $n \times m$ window on the image, classify its content as face or not-face.

- To solve the original problem we move the window across the image at different scales and apply the binary classifier.



Proposed solution

- The face/not-face classifier consists in a multilayer-perceptron neural network with 2 hidden layers
- The training/validation sets consist in a database of frontal faces of different people at varying orientations, scales and illuminations.



Further problems

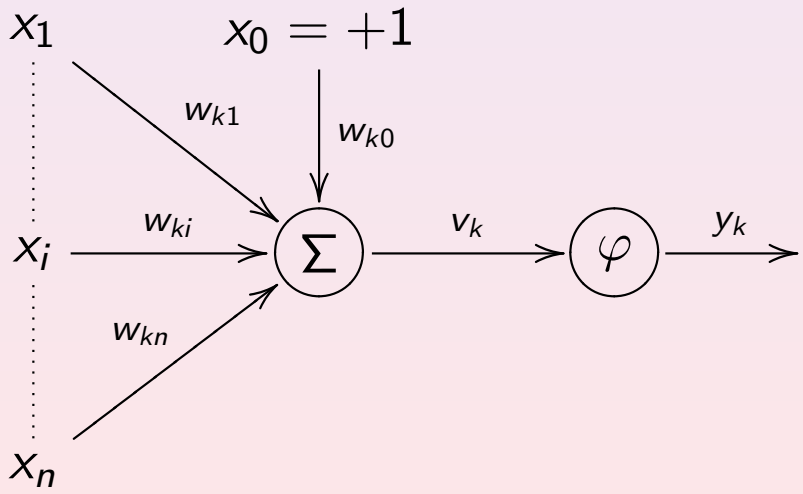
- to render the training and detection stages invariant to varying illumination conditions
- possibly optimal tuning of the network free parameters, including the training policy (training and test sets choice, etc.)
- to deal with false detections (false-positives[1] and false-negatives[2])
- since a face can lead to multiple local detections in the image (also at different scales), we have to cluster somehow together detections relative to the same face

[1] Not-faces classified as faces
[2] Faces classified as not-faces

McCulloch-Pitts' Neuron

- Fundamental processing unit of the neural network
- Characterized by 4 components:

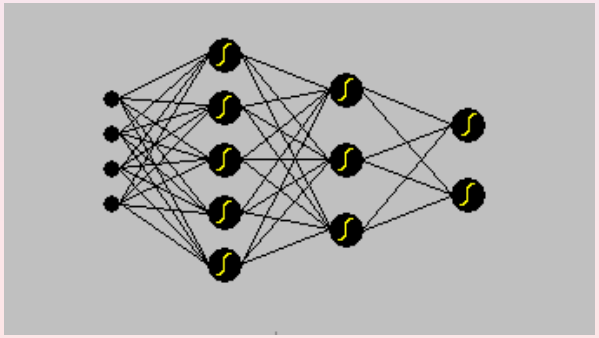
1. synapses or connections with a positive or negative weight
2. a summing unit that computes the inner product between inputs and the synapse's weights (net input)
3. a threshold that increases/reduces the net input
4. an activation function that reduces the output variance of a neuron by mapping the thresholded net input generally within the interval $[0, 1]$ or $[-1, 1]$



$$v_k = \sum_{i=0}^{n} w_{ki} \cdot x_i \quad y_k = \varphi(v_k)$$

Multilayer Perceptron

- It is a layered neural network with 3 types of layers
  1. the set of inputs (input layer)
  2. one or more hidden layers of neurons (hidden layers)
  3. the set of output neurons (output layer)
- the signal is generated in the input layer, propagated through the hidden layers until it reaches the output layer
- the net has generally high connectivity
- the activation function is continuous
- an efficient training algorithm is the back-propagation method



Back-propagation ...

- Training method for a multi-layer perceptron
- we distinguish 2 kinds of signal
  1. function signal: it is an input signal that is propagated through the network from the input to the output layer, where it results in an output
  2. error signal: it is an error signal that is originated in the output layer and is propagated through the network in reversed sense, i.e., from the output layer to the input layer.
- The goal is to minimize
  - the error relative to the last presented example (on-line training)
  - the sum of the errors relative to all examples in the training set (off-line training)
- (we will deal with the on-line training)

. . . Back-propagation. . .

- Error minimization through steepest descent method
- The error function relative to the nth training sample is

$$E(n) = \frac{1}{2} \sum_j e_j(n)^2$$

  where $e_j(n) = d_j(n) - y_j(n)$ is the difference between desired and effective output at the jth output neuron
- The parameters to optimize are the weights w_{ji}.
- The weights are updated in the direction that is opposed to the gradient

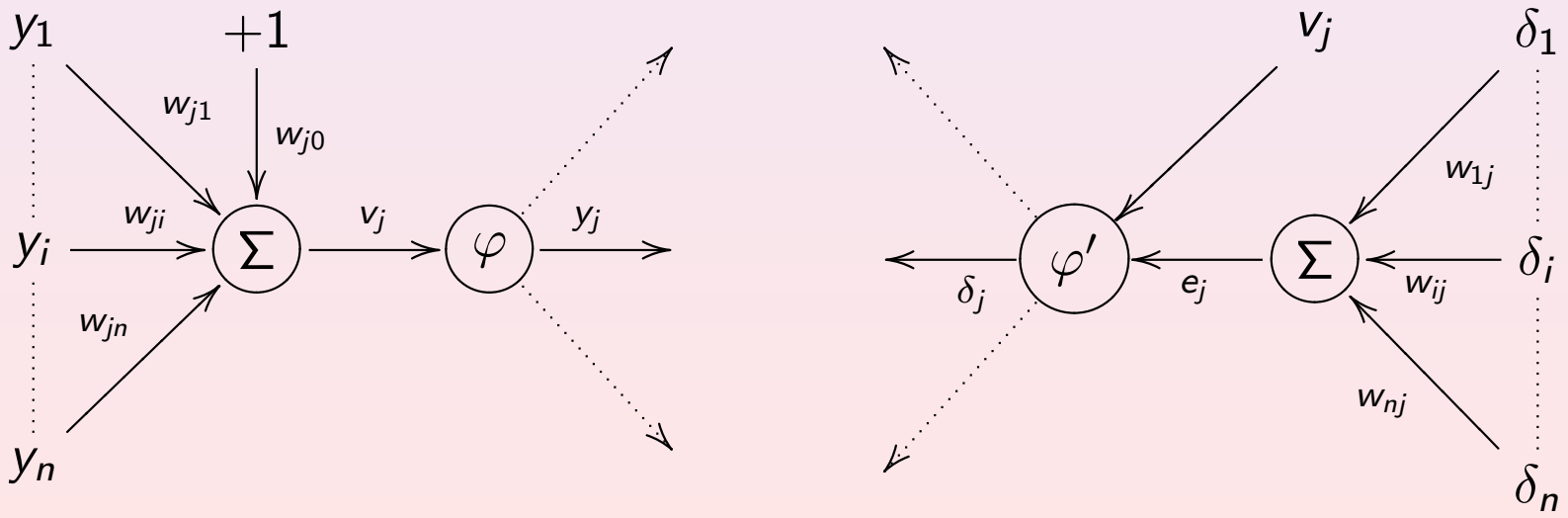
$$w_{ji}(t+1) = w_{ji}(t) - \eta \frac{\partial E(t)}{\partial w_{ji}}$$

. . . Back-propagation

- More details about the weights updating process

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = \eta \delta_j y_i$$

$$\delta_j = \begin{cases} \varphi'(v_j) e_j, & \text{if } j \text{ is an output neuron} \\ \varphi'(v_j) \sum_k \delta_k w_{kj}, & \text{otherwise} \end{cases}$$



Some expedients . . .

1. Weights update with momentum
   - The learning parameter η has great influences on the learning algorithm (speed of convergence, oscillations, etc.)
   - Using the momentum in the learning rule we can get rid of these problems

$$\Delta w_{ji}(t+1) = \underbrace{\alpha w_{ji}(t)}_{\text{momentum}} + \eta \delta_j y_i$$

   - It is preferable to chose the parameter α within $[0, 1)$
2. Choose an antisymmetric activation function, i.e. such that $\varphi(-x) = -\varphi(x)$.
   We used the hyperbolic tangent

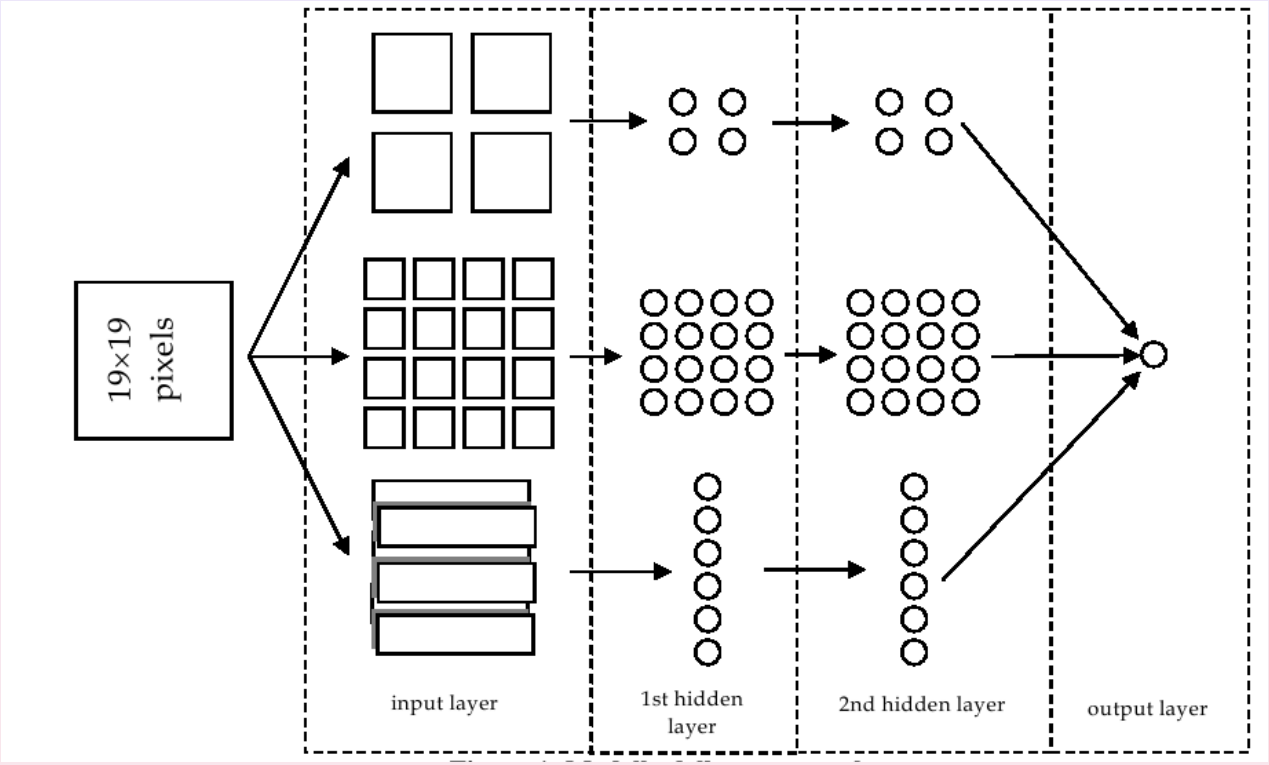
$$\varphi(x) = a \tanh(bx)$$

   with $a = 1.7159$ and $b = \frac{2}{3}$. With this choice, it is efficient to calculate $\varphi'(v_j) = \frac{b}{a}[a - y_j][a + y_j])$

. . . Some expedients

3. desired response within $[-1, +1]$ in order to avoid a possible divergence of the weights to infinity (saturation)
4. input signals within $[-1, +1]$ in order to limit eventual oscillatory effects of the network
5. the weights initialization stage is also crucial. Too large weights lead to saturation; too small weights lead to slow convergence. Good initializations can be achieved considering the weights of a neuron j equally distributed with zero mean and variance $\frac{1}{m_j}$ where m_j is the number of synapses of neuron j.

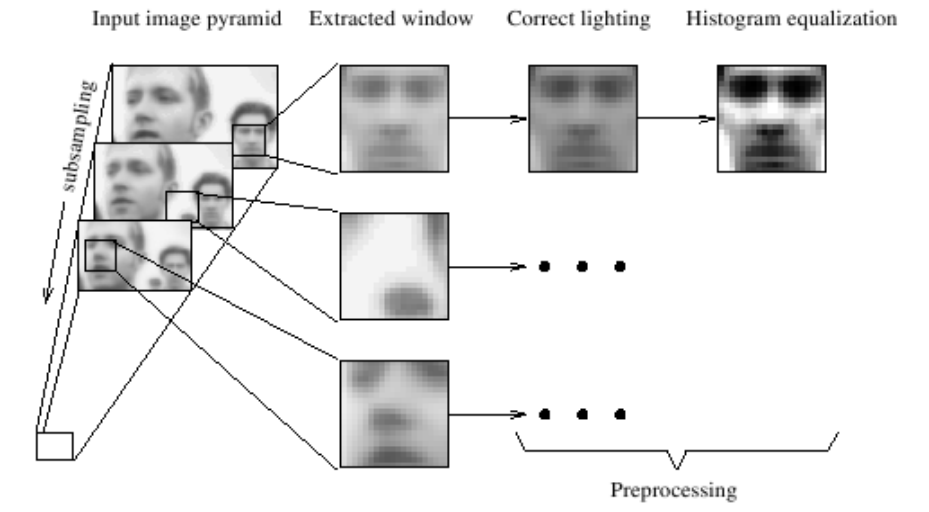
Structure of the neural network



- Activation function: $\varphi(x) = 1.7519 \tanh\left(\frac{2}{3}x\right)$
- Input signal and desired response within $[-1, +1]$
- Training with learning rate $\eta = 0.05$ and momentum $\alpha = 0.2$.

Image normalization

- Gray-level conversion of the whole image
- For each training example
  1. Illumination normalization stage, in order to have an illumination invariant training.
  2. Equalization stage, in order to use the whole brightness spectrum and increase the contrast so that the face details are more evident



Illumination normalization

- We assume that the brightness contribution from the environment is linear.
- We compute the brightness plane through least-square approximation of the brightness intensity of the image. The coefficients a_0, a_1, a_2 have to be optimized in order to minimize the following function

$$\varphi(a_0, a_1, a_2) = \sum_{i=0}^{h-1} \sum_{j=0}^{w-1} \{Y[i,j] - (a_0 + a_1 i + a_2 j)\}^2$$

  where w, h are the width and height of the image and $Y^{h \times w}$ is the brightness intensity matrix.
- Let $Z[i,j] = 1 - (a_0 + a_1 i + a_2 j)$. The image is normalized by computing the arithmetic mean of Y and Z.

Faces Database and Training

- 

MIT CBCL Face Database #1 (http://cbcl.mit.edu/software-datasets/FaceData2.html)
- images format 19×19 PGM
- Faces: 2.901
- Not-faces: 28.121
- faces of different people at varying orientations, brightnesses and scales.
- another DB is the Yale Face Database B (http://cvc.yale.edu/projects/yalefacesB/yalefacesB.html)
- The training has been conducted for 20 epochs
- At each epoch the dataset was randomly partitioned into 80% training set and 20% validation set
- As test set we applied the system to several real images

Application of the Face Neural Filter

- We have a filter that analyses a window in the image of dimension 19×19 and returns a value ψ.
- Approximately if $\psi \to 1$ we have a face, otherwise if $\psi \to -1$ we have a not-face
- The filter is applied to the image at varying scales with progressive scale reduction of 20% (the dataset covered a scale range of 20%)
- At each scale, the filter is applied to each window 19×19 in the image
- It is not efficient! It is possible to train a filter to be tolerant to face shifts within a certain limit, in order to reduce the number of windows to analyze

False Positive and Negative reduction ...

- It is reasonable to assume that faces generate many detections of high intensity in a local neighborhood of the image, unlike false detections
- We attach to each pixel p of the image, 3 fields: face hypothesis intensity $I[p]$, number of detections $\#p$, window size $D[p]$.
- A detection happens on a pixel if it represents the center of the filtering window
- Then we have two thresholds, on the face hypothesis intensity γ and on the number of detections π, that must be reached in order to have a feasible face
- Since we can have multiple detections on a pixel only if they happened at different scales, we consider as face hypothesis intensity $I[p]$ the mean of the detections. For the same rationale we have in $D[p]$ a mean of the filtering window dimensions at varying scales.

. . . False Positive and Negative reduction

- For each pixel p we consider its neighborhood of 9 pixels, eliminating those with $I[p] < \gamma$. The new pixel intensity $I[p]$ is given by the mean of the remaining pixels. The new number of detections consists in the sum of all detections in the neighborhood.
- Then we order the detections in intensity-decreasing order eliminating those with $\#p < \pi$. We consider good the first detection ad eliminate each detection that overlaps with it within a certain tolerance. And we iterate this process.
- The good detections are the faces.

Experimental results

- We conduct a small test on 12 images (#V=num. faces; #NV=num. not-faces; #F=num. filterings)

Im.	#V	#NV	#F	Im.	#V	#NV	#F
1.jpg	33/36	4	1.706.018	2.jpg	03/07	5	603.133
3.jpg	19/22	2	1.137.523	4.jpg	17/31	3	706.913
5.jpg	11/11	0	645.128	6.jpg	01/01	2	379.426
7.jpg	01/01	0	120.208	8.jpg	06/07	0	691.796
9.jpg	07/11	4	1.320.321	10.jpg	17/27	2	1.419.163
11.jpg	00/00	2	760.204	12.jpg	00/00	0	161.063

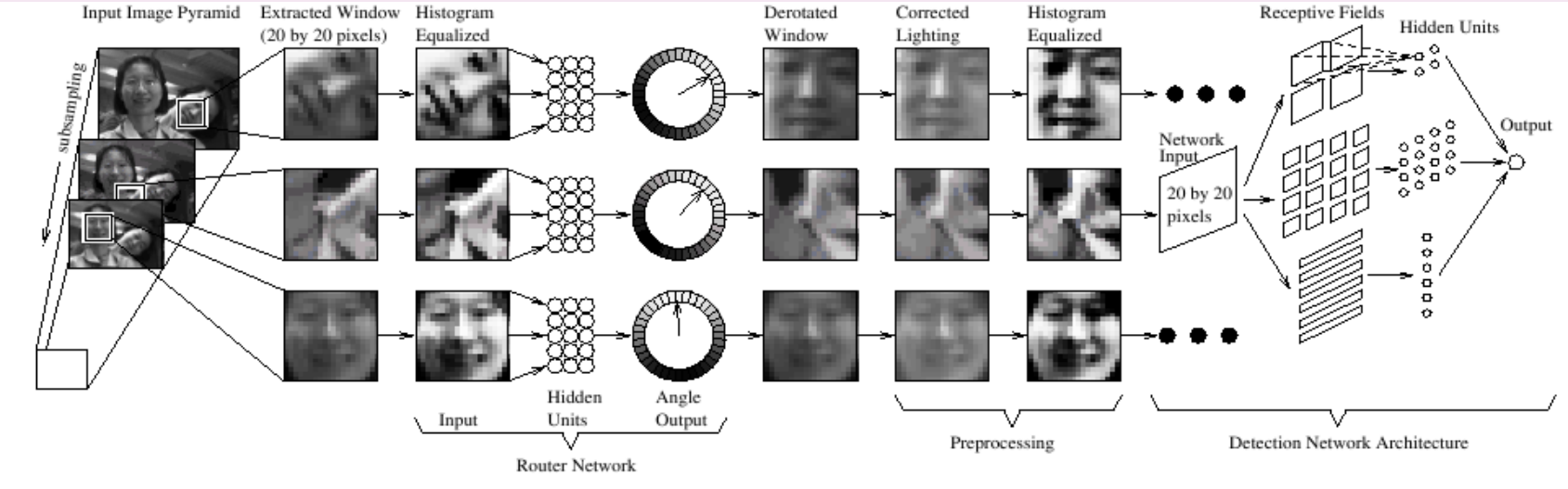
- Faces detected: 74,6%; Wrong detections: 2.5E-6%
- The number of test conducted is too small in order to asses the network performances, however our training step produced a filter that privileges avoiding false positive detections

Conclusions

- The face detection is an active research area, and in the last years there were great advances in developing algorithms that solve this problem, however some of the best methods are still computationally heavy to be used in real-time systems
- The neural networks are a powerful tool to solve pattern recognition problems, and can potentially be applied at each stage of a face recognition system [1]
- Their value is robustness to varying environment conditions and particularly effective
- A drawback is that their design is crucial as well as the parameter tuning and training steps.
- We saw only a particular kind of neural networks for face detection, however there are many others.[4, 7]

Future developments

- The presented system has been extended to work also with rotated faces [11].
- There are multiple neural networks.
  1. A "router" network processes the input to state the face orientation
  2. Depending on the orientation, the image is rotated uprightly
  3. Finally the technique seen in this talk is applied



Bibliography I


R. Chellappa, C. L. Wilson, and S. Sirohey.
Human and machine recognition of faces. a survey.
In *Proc. IEEE*, volume 83, 1995.


T. F. Cootes and C. J. Taylor.
Active shape models – smart snakes.
In *Proc. of British Machine Vision Conference*, pages 266–275, 1992.


L. C. De Silva, K. Aizawa, and M. Hatori.
Detection and tracking of facial feature by using facial feature model and deformable circular template.
IEICE Trans. Inform. Systems, E78-D(9):1195–1207, 1995.


R. Feraud, O. Bernier, and D. Collobert.
A constrained generative model applied to face detection.
Neural Process. Lett., 5:73–81, 1997.

Bibliography II


S. H. Jeng, Y. M. Liao, C. C. Han, M. Y. Chern, and Y. T. Liu.
Facial feature detection using geometrical face model: An efficient approach.
Pattern Recog., 31, 1998.


M. Kass, A. Witkin, and D. Terzopoulos.
Snakes: active contour models.
In *Proc. of 1st Int. Conf. an Computer Vision, London*, 1987.


S. H. Lin, S. Y. Kung, and L. J. Lin.
Face recognition/detection by probabilistic decision-based neural network.
IEEE Trans. Neural Networks, 8:144–132, 1997.


D. Maio and D. Maltoni.
Real-time face location on gray-scale static images.
Pattern Recog., 33:1525–1539, 2000.

Bibliography III


B. Moghaddam and A. Pentland.
Face recognition using view-based and modular eigenspaces.
In *Automatic Systems for the identification of Humans*, volume 2277, 1994.


H. A. Rowley, S. Baluja, and T. Kanade.
Neural network-based face detection.
IEEE Trans. Pattern Anal. March. Intell., 20:23–28, 1998.


H. A. Rowley, S. Baluja, and T. Kanade.
Rotation invariant neural network-based face detection.
In *Proc. IEEE Intl. Conf. on Computer Vision and Pattern Recog.*, pages 38–44, 1998.


T. Sakai, M. Nagao, and T. Kanade.
Computer analysis and classification of photographs of human faces.
In *Proc. First USA – Japan Computer Conference*, page 2.7., 1972.

Bibliography IV


A. Samal and P. A. Iyengat.
Human face detection using silhouttes.
Int. J. Pattern Recog. Atificial Intell., 9(6), 1995.


K. C. Yow and R. Cipolla.
Feature-based human face detection.
Image Vision Comput., 15(9), 1997.


A. L. Yuille, P. W. Hallinan, and D. S. Cohen.
Feature extraction from faces using deformable templates.
Int. J. Comput. Vision, 8:99–111, 1992.