Recursive neural networks for object detection

M. Bianchini, M. Maggini, L. Sarti, F. Scarselli
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Siena
Via Roma, 56
53100 – Siena (Italy)
E-mail: {monica,maggini,sarti,franco}@dii.unisi.it

***Abstract*— In this paper, a new recursive neural network model, able to process directed acyclic graphs with labeled edges, is introduced, in order to address the problem of object detection in images. In fact, the detection is a preliminary step in any object recognition system. The proposed method assumes a graph–based representation of images, that combines both spatial and visual features. In particular, after segmentation, an edge between two nodes stands for the adjacency relationship of two homogeneous regions, the edge label collects information on their relative positions, whereas node labels contain visual and geometric information on each region (area, color, texture, etc.). Such graphs are then processed by the recursive model in order to determine the eventual presence and the position of objects inside the image. Some experiments on face detection, carried out on scenes acquired by an indoor camera, are reported, showing very promising results. The proposed technique is general and can be applied in different object detection systems, since it does not include any a priori knowledge on the particular problem.**

I. Introduction

Recently, there has been a growth of interest in object detection models, which must be inherently robust to the wide variations that are observed in natural images. In fact, the general problem of object detection is a very challenging task, since the object detection system is required to distinguish a particular class of objects from all other objects represented in the images. The difficulty of this task lies in the definition of a general model of the object class to be detected, which has a high inter–class and a low intra–class variability. An object detection system should be able to localize objects which can vary their appearance w.r.t. light conditions, orientation, and dimension. Furthermore, the objects can be partially occluded or can be blent in with the background. Object detection methods can be classified in four main categories [1]:

- Knowledge–based;
- Feature invariant;
- Template matching;
- Appearance–based.

Knowledge–based methods exploit the human knowledge on the searched objects and use some rules in order to describe the object models. Those rules are then used to detect and localize objects that match the predefined models.

Instead, the aim of feature invariant approaches [2], [3] is to define a set of features that are invariant w.r.t. object orientation, light conditions, dimension, etc. Template matching methods store several patterns of objects and describe each pattern by visual and geometrical features. The correlation among an input image and the stored patterns is computed for detecting objects [4]. Finally, in appearance–based methods, machine learning techniques are exploited to learn templates by examples [5], [6], [7].

In this paper we present a new appearance–based method that uses Recursive Neural Networks (RNNs). The novelty of the approach consists in representing images by graphs with labeled edges. Such graphs are then transformed into forests of trees and processed by a new RNN model able to deal with structures with labeled edges. The proposed method does not use any a priori or heuristic information on the object models and can be useful to detect objects under any illumination, orientation, and position.

The paper is organized as follows. In the next section, some notation is introduced. The new RNN model is presented in Section III, while Section IV describes the graph–based representation of images. In Section V some experimental results on a face detection task are reported, and, finally, Section VI collects some conclusions.

II. Notation

Let $G = (V, E, \mathcal{L})$ be a directed graph, where V is the set of nodes, $E \subseteq V \times V$ represents the set of arcs, and $\mathcal{L} : V \rightarrow L_v$ is a labeling function, being $L_v \subset \boldsymbol{R}^m$ a finite set of labels. Given any node $v \in V$, pa[v] is the set of the *parents* of v, while ch[v] represents the set of its *children*. The *outdegree* of v, od[v], is the cardinality of ch[v], and $o = \max_v \text{od}[v]$ is the maximum outdegree. Each node stores a set of domain variables into a *label*. The presence of an edge (v, w) in a labeled graph stands for the existence of a causal link between the labels of v and w. Moreover, for recursive processing, G should have a *supersource*, i.e. a node $s \in V$ with no incoming edges, and from which any other node in V can be reached. The supersource s may eventually be added following the algorithm in [8].

In this paper, we consider the class of Directed Acyclic Graphs (DAGs), where a partial ordering can be defined on E, such that $v \prec w$ if v is connected to w by a direct path. Directed Positional Acyclic Graphs (DPAGs), for which recursive networks were originally defined, are a subclass of DAGs, where an injective function $o_v : \text{ch}[v] \rightarrow \{1, \ldots, o\}$ assigns a position $o_v(c)$ to each child c of a node v. Therefore, a DPAG is represented by the tuple $(V, E, \mathcal{L}, \mathcal{O})$, where $\mathcal{O} =$

$\{o_1, \ldots, o_{|V|}\}$ is the set of functions defining the position of the children for each node. Finally, the definition of the class of DAGs with Labeled Edges (DAGs–LE) requires the introduction of an additional edge labeling function $\mathcal{E}$, such that $G = (V, E, \mathcal{L}, \mathcal{E})$, where $\mathcal{E} : E \rightarrow L_e$, and L_e is a finite subset of $\boldsymbol{R}^k$. The presence of an edge label introduces some semantical contents into the link between two nodes.

III. Recursive Neural Networks for processing DAGs–LE

Recursive neural networks were originally proposed to process DPAGs [9], [10], [8]. In this case, the state transition function f, computed by an RNN, depends on the order of the children of each node, since the state of each child occupies a particular position in the list of the arguments of f. In order to overcome such a limitation, in [11], a weight sharing approach was described, able to relax the order constraint, and to devise a neural network architecture suited for DAGs with a bounded outdegree. In fact, the weight sharing technique cannot be applied to DAGs with a large outdegree o, due to the factorial growth in the network parameters w.r.t. o. Even if the maximum outdegree can be bounded, for instance by pruning those connections that are heuristically classified as less informative, nevertheless some important information may be discarded in such a preprocessing phase. On the other hand, both the ordering constraint and the bound on the maximum outdegree can be removed when considering DAGs–LE [12].

In fact, for DAGs–LE, a state transition function $\tilde{f}$ can be defined which has not a predefined number of arguments and that does not depend on their order. The different contribution of each child depends on the label attached to the corresponding edge. At each node v, the total contribution $\overline{\mathbf{X}}(\mathrm{ch}[v]) \in \boldsymbol{R}^p$ of the state of its children is computed as

$$\overline{\mathbf{X}}(\mathrm{ch}[v]) = \frac{1}{|\mathrm{ch}[v]|}\left(\sum_{i=1}^{|\mathrm{ch}[v]|}\left(\sum_{j=1}^{k}\mathbf{H}_j\mathbf{L}^{(j)}_{(v,\mathrm{ch}_i[v])}\right)\mathbf{X}_{\mathrm{ch}_i[v]}\right) \tag{1}$$

where $\mathbf{L}_{(v,\mathrm{ch}_i[v])} \in \boldsymbol{R}^k$ is the label attached to the edge $(v, \mathrm{ch}_i[v])$, and $\mathbf{H} \in \boldsymbol{R}^{p,n,k}$ is the weight matrix. In particular, $\mathbf{H}_j \in \boldsymbol{R}^{p,n}$ is the j–th layer of matrix $\mathbf{H}$ and $\mathbf{L}^{(j)}_{(v,\mathrm{ch}_i[v])}$ is the j–th component of the edge label. Finally, the state at node v is computed by a two–layer perceptron with linear outputs, as

$$\mathbf{X}_v = \tilde{f}(\overline{\mathbf{X}}(\mathrm{ch}[v]), \mathbf{U}_v, \theta_{\tilde{f}}) = \mathbf{V}\vec{\sigma}(\mathbf{A}\overline{\mathbf{X}}(\mathrm{ch}[v]) + \mathbf{B}\mathbf{U}_v + \mathbf{C}) + \mathbf{D} \tag{2}$$

where $\theta_{\tilde{f}}$ collects $\mathbf{A} \in \boldsymbol{R}^{q,p}$, $\mathbf{B} \in \boldsymbol{R}^{q,m}$, $\mathbf{C} \in \boldsymbol{R}^{q}$, $\mathbf{D} \in \boldsymbol{R}^{n}$, and $\mathbf{V} \in \boldsymbol{R}^{n,q}$, being q the number of hidden units. At the supersource, also an *output* function is evaluated by a feedforward network,

$$\mathbf{Y}_s = g(\mathbf{X}_s, \theta_s) = \mathbf{W}\vec{\sigma}(\mathbf{E}\mathbf{X}_s + \mathbf{F}) + \mathbf{G}$$

with $\mathbf{E} \in \boldsymbol{R}^{q',n}$, $\mathbf{F} \in \boldsymbol{R}^{q'}$, $\mathbf{G} \in \boldsymbol{R}^{r}$, and $\mathbf{W} \in \boldsymbol{R}^{r,q'}$.

Starting from eqs. (1) and (2), an MLP implementation of the recursive network can be derived, simply rewriting eq. (1) as

$$\overline{\mathbf{X}}(\mathrm{ch}[v]) = \frac{1}{|\mathrm{ch}[v]|}\left(\sum_{j=1}^{k}\mathbf{H}_j\left(\sum_{i=1}^{|\mathrm{ch}[v]|}\mathbf{L}^{(j)}_{(v,\mathrm{ch}_i[v])}\mathbf{X}_{\mathrm{ch}_i[v]}\right)\right) \tag{3}$$

Therefore, eq. (3) suggests that the contribution of the children of each node to its state can be computed using a three–layer perceptron with k inputs $\sum_{i=1}^{|\mathrm{ch}[v]|}\mathbf{L}^{(j)}_{(v,\mathrm{ch}_i[v])}\mathbf{X}_{\mathrm{ch}_i[v]}$, $j = 1, \ldots, k$, and with $\mathbf{H}_j$ as input–to–hidden weight matrices. The second hidden layer and the output layer remain unchanged and their contribution to the calculation is described by eq. (2) (see Fig. 1).

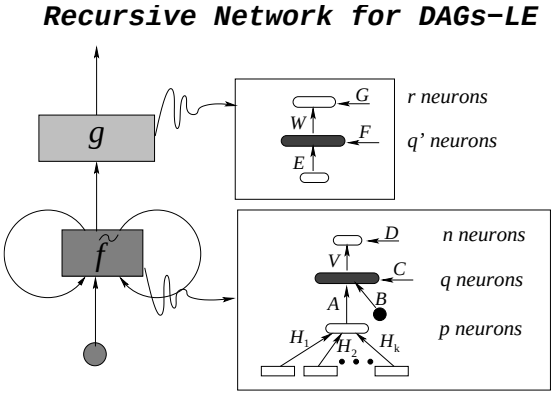

Fig. 1. An MLP implementation of the recursive network. Grey layers are sigmoidal, whereas white layers are linear in both the feedforward networks.

Remark: *In the more general case,* $\overline{\mathbf{X}}(ch[v])$ *may be computed using a nonlinear function* $\phi : \boldsymbol{R}^{(n+k)} \rightarrow \boldsymbol{R}^p$*, depending on a set of parameters* θ_ϕ*:*

$$\overline{\mathbf{X}}(ch[v]) = \frac{1}{|ch[v]|}\left(\sum_{i=1}^{|ch[v]|}\phi(\mathbf{X}_{ch_i[v]}, \mathbf{L}_{(v,ch_i[v])}, \theta_\phi)\right)$$

Moreover, it is worth noting that, even if the edge labels increase the semantical content attached to the links, they can also be used to codify the order relationship. In fact, each DPAG can be represented by a DAG–LE. In particular, in [13], it was proved that for any DPAG G and any standard recursive neural network RNN with a transition function f, there exists a DAG–LE L and an RNN–LE, with a transition function $\tilde{f}$ such that $f(G) = \tilde{f}(L)$. As a direct consequence, the novel recursive architecture maintains the universal approximation capabilities on the set of DPAGs. Finally, since any DAG can be represented with a DPAG, by assigning an arbitrary position to each child, the above approximation property can also be extended to DAGs.

IV. The Graph–based representation of images

Our object detection method assumes a graph-based representation of images. Thus, in order to describe our approach, we need to introduce the preprocessing phase that allow us to represent each image by a DAG–LE.

First of all, each image is segmented in order to obtain a set of distinct regions. Each region has homogeneous content w.r.t. some visual features. The segmentation method we used is based on color information. The effectiveness of color–based segmentation approaches was already shown in [14], [15].

We assume that images are represented in the RGB (Red, Green, Blue) color space. Our segmentation algorithm performs a K–means clustering in this space on the pixels belonging to the images, based on the Euclidean Distance. At the end of the clustering, a region growing procedure is carried out to reduce the number of regions. A suitable choice of the K initial pixels which correspond to the centroids of the initial clusters, and an appropriate region growing policy, allow to obtain an invariant set of regions w.r.t. both rotation and translation. The segmentation process yields a set of regions. Each region can be described by a vector of real valued features which collect geometrical and visual information. On the other hand, the structural information related to the adjacency relationship among regions can be coded by an undirected graph with labeled edges. Strictly speaking, the *Region Adjacency Graph with Labeled Edges* (RAG–LE) is extracted from the segmented image by:

1) Associating a node to each region. The real vector of features represents the node label;
2) Linking the nodes associated to adjacent regions with undirected edges;
3) Attaching a real vector of features to each edge of the graph. The vector describes the mutual position of the two adjacent regions.

In order to set up a learning environment, a target equal to 1 is attached to each node of RAGs–LE that correspond to a part of the object in which we are interested, whereas a target equal to 0 is attached otherwise. In Fig. 2, the RAG–LE extraction is summarized. In this example, we want to localize the "red toy car". The black nodes correspond to parts of the toy car and have target 1, while white nodes correspond to parts of other objects and have target 0.

Our object detection software provides a visual interface to tag the regions that correspond to the object or not (see Fig. 3).

Since the RNN model described in Section III can process only DAGs–LE, each RAG–LE must be transformed into a directed acyclic graph. The transformation procedure takes a RAG–LE R, along with a selected node n, as input, and produces a tree T having n as its root. The method must be repeated for each node of the RAG–LE, or, more practically, for a random set of nodes. It can be proved that the forest of trees built from R is recursive–equivalent to R, that is the RNN behavior is the same either if the network processes R or if it processes the forest of trees [16], [17]. The first step of the procedure is a preprocessing phase that transforms R into a directed RAG–LE G, by assuming that a pair of directed arcs replaces each undirected edge. Each arc in the pair is assigned the same label as the original undirected edge. G is unfolded into T by the following algorithm:

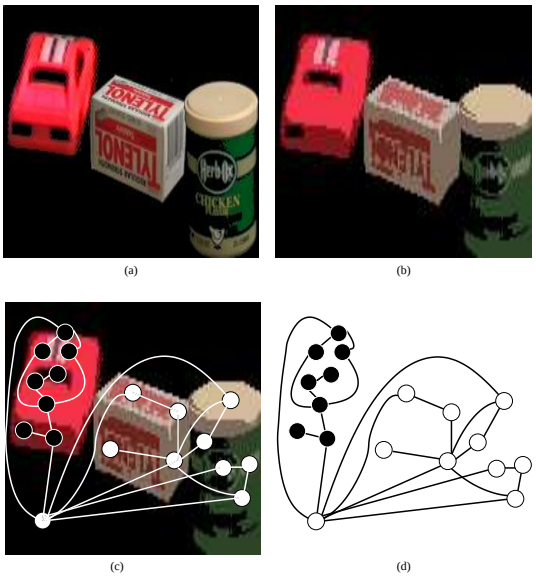

Fig. 2. The original image (a), the segmented image (b), and the extracted RAG–LE (c,d).

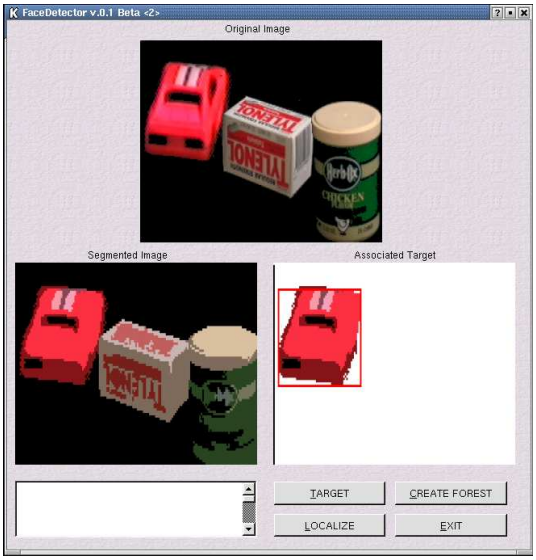

Fig. 3. The visual interface used to create the learning environment.

1) Insert a copy of n in T;
2) Visit G, starting from n, using a breadth–first strategy; for each visited node v, insert a copy of v into T, link v to its parent node preserving the information attached to each arc;
3) Repeat step 2 until a predefined stop criterion is satisfied, and, however, until all arcs have been visited at least once.
4) Attach the target associated to n to the root node of t.

If the breadth–first visit is halted when all the arcs have been visited once, the minimal recursive–equivalent tree is obtained (Minimal Unfolding – see Fig 4(a)). However, different stop criteria can be chosen. For example, each arc can be visited

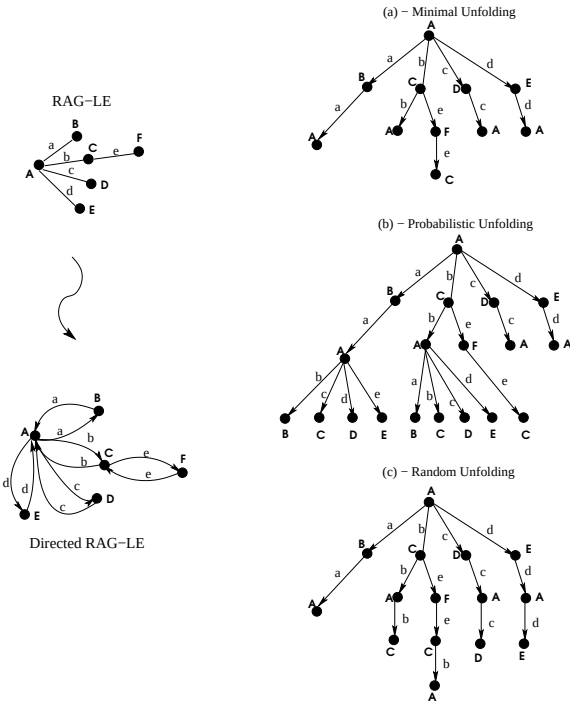

Fig. 4. The transformation from a RAG–LE to a recursive–equivalent tree. The dimension of the recursive–equivalent tree depends on the stop criterion chosen during the unfolding of the directed RAG–LE.

once, then step 2 is repeated until a stochastic variable x becomes true (Probabilistic Unfolding – see Fig. 4(b)). Otherwise, the breadth–first visit can be replaced with a random visit (Random Unfolding – see Fig. 4(c)). Anyway, each arc must be visited at least once, in order to guarantee the recursive–equivalence between R and T.

V. Experimental Results

As a case study, the experimentation performed to evaluate the effectiveness of our approach has been focused on detecting faces in images. However, The proposed method does not exploit any a priori information about the particular object model and, therefore, is independent of the problem at hand. The experimental dataset contains 500 images and 384 faces (each image contains at most one face) and was acquired by an indoor camera, which was placed in front of a door. A person at a time went in through the door and walked until he/she was out of the camera eye. Each image corresponds to a frame of the acquired scene. We are interested only in detecting the face position, whereas no tracking of the faces was performed and no information derived by the movement of the object was exploited. The faces appear in different orientations and dimensions (see Fig. V). The images were divided in three sets: training set, cross–validation set, and test set. Both the training and the cross–validation sets contain 100 images, whereas 300 images (199 faces) constitute the test set.

Each image was segmented, producing a RAG–LE. Each node in the RAG–LE stores a label that describes some visual features (average color, bounding box coordinates, barycenter coordinates, area, perimeter, and momentum). The mutual spatial position of two adjacent regions is represented by the label attached to the corresponding edge. Given a pair of adjacent regions i and j, the label of the edge (i, j) is represented by the vector $[D, A, B, C]$ (see Fig. 6), where:

- D represents the distance between the two barycenters;
- A measures the angle between the two principal inertial axes;
- B is the angle between the intersection of the principal inertial axis of i and the line connecting the barycenters;
- C is the angle between the intersection of the principal inertial axis of j and the line connecting the barycenters.

Subsequently, each RAG–LE was unfolded using the Probabilistic Unfolding Strategy described in Section IV, which was empirically shown in the past to be the more promising one [18]. The directed RAGs–LE were obtained decomposing the labels attached to each edge in the following way:

- the label $[D, A, B]$ was attached to the edge from i to j;
- the label $[D, A, C]$ was attached to the edge from j to i.

Finally, the RNN was trained to predict whether each node in the trees belongs to a face or not. Then, using the trained RNN, faces were localized in a given image, performing the following steps:

1) The image is segmented and the corresponding RAG–LE is built;
2) The RAG–LE is unfolded, producing a forest of trees;
3) Each tree is processed by the trained RNN. The network predicts whether the root node of each tree is a part of a face or not;
4) Adjacent regions predicted as a part of a face are merged together in order to compute the minimum bounding boxes that contain the faces.
5) A face is recognized if the predicted bounding box equals the true bounding box.

Notice that, in the general case of object detection, if the number of distinct objects which must be detected is equal to n, then n RNNs must be trained. Each RNN is trained to localize parts of a single object.

Fig. 5. Variability of the face appearance in the dataset. Faces vary w.r.t. dimension and pose and can be partially occluded. The images used to perform the experimentation were provided by ELSAG S.p.A.; all the images were used strictly for research purpose and are published under license of the reproduced persons.

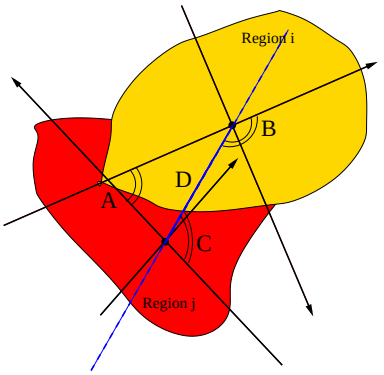

Fig. 6. Features stored into the edge label.

RNN architecture	RNN Accuracy Rate	Recall	Precision
one layer – 5 state neurons	77.29%	85.22%	78.23%
one layer – 10 state neurons	**83.62**%	**90.88**%	**85.33**%
one layer – 15 state neurons	73.22%	79.85%	72.18%

TABLE I

RESULTS OBTAINED BY THE PROPOSED RESULT, VARYING THE RNN ARCHITECTURE

During the experimentation, several recursive neural networks were trained with the aim of determining the best architecture. The performance achieved by some trained neural networks are reported in Table V. The accuracy rate is the number of regions correctly classified (as part of a face or not) divided by the whole number of regions in the test set, while the recall and the precision rate are computed considering detected faces. A face is considered detected if, given the predicted and the corresponding correct bounding box, the ratio computed dividing the intersection of the bounding boxes by their union is smaller than a predefined threshold. In our experiments, this threshold was set to 90%.

Generally, the recall and the precision rate are greater than the accuracy rate. In fact, a face can be correctly localized even if some parts of it are not correctly classified. A one–layer RNN with 10 state neurons yields the best performances.

The obtained results can be furtherly improved considering that some false faces correspond to very small bounding boxes.

Assuming that a face, or in general an object, can not be smaller than a predefined threshold, this kind of erroneous detection can be bounded. However, this kind of post–processing was not introduced, in order to evaluate the generality of our approach.

VI. CONCLUSIONS

In this paper, we proposed a new appearance–based method for detecting objects in images, which are represented by DAGs–LE. This approach is invariant under image translations and rotations, due to the invariance of the graphical representation used. The method localizes the parts that constitute the objects, using the learning capability of a new recursive neural network model, able to deal with DAGs–LE. Furthermore, the detection is performed without any heuristics or a priori information on the specific object model. The experimental results validate the effectiveness of the proposed approach.

ACKNOWLEDGMENT

This research was carried out in collaboration with Elsag S.p.A. – Genoa – Italy.

REFERENCES

[1] M.-H. Yang, J. 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, January 2002.

[2] S. McKenna, Y. Raya, and S. Gong, "Tracking colour objects using adaptive mixture model," *Image and Vision Computing*, vol. 17, no. 3/4, pp. 223–229, 1998.

[3] T.K. Leung, M.C. Burl, and P. Perona, "Probabilistic affi ne invariants for recognition," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 1998, pp. 678–684.

[4] P. Shina, *Processing and Recognizing 3D forms*, Ph.D. thesis, Massachusetts Institute of Technology, 1995.

[5] H. Schneiderman and T. Kanade, "A statistical method for 3d object detection applied to faces and cars," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2000, pp. 746–751.

[6] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Proceedings of the 6th IEEE International Conference on Computer Vision*, 1998, pp. 555–562.

[7] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object recognition," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 696–710, July 1997.

[8] A. Sperduti and A. Starita, "Supervised neural networks for the classifi cation of structures," *IEEE Transactions on Neural Networks*, vol. 8, pp. 429–459, 1997.

[9] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 768–786, September 1998.

[10] A. Küchler and C. Goller, "Inductive learning in symbolic domains using structure–driven recurrent neural networks," in *Advances in Artificial Intelligence*, G. Görz and S. Hölldobler, Eds., pp. 183–197. Springer, Berlin, 1996.

[11] M. Bianchini, M. Gori, and F. Scarselli, "Processing directed acyclic graphs with recursive neural networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1464–1470, 2001.

[12] M. Gori, M. Maggini, and L. Sarti, "A recursive neural network model for processing directed acyclic graphs with labeled edges," in *Proceedings of the International Joint Conference on Neural Networks*, 2003, pp. 1351–1355.

[13] M. Bianchini, M. Maggini, L. Sarti, and F. Scarselli, "Recursive neural networks for processing graphs with labelled edges," in *Proceedings of ESANN 2004*, Bruges (Belgium), April 2004, pp. 325–330.

[14] H. Wu, Q. Chen, and M. Yachida, "An application of fuzzy theory: Face detection," in *International Workshop on Automatic Face and Gesture Recognition*, Zurich (Switzerland), June 26–28 1995, pp. 314–319.

[15] Y. Dai and Y. Nakano, "Extraction of facial images from complex background using color information and sgld matrices," in *International Workshop on Automatic Face and Gesture Recognition*, Zurich (Switzerland), June 26–28 1995, pp. 238–242.

[16] M. Bianchini, M. Gori, and F. Scarselli, "Recursive processing of cyclic graphs," *IEEE International Joint Conference on Neural Networks*, pp. 154–159, 2002.

[17] M. Bianchini, M. Gori, L. Sarti, and F. Scarselli, "Backpropagation through cyclic structures," in *LNAI - AI*IA 2003: Advanced in Artificial Intelligence*, A. Cappelli and F. Turini, Eds., Pisa (Italy), September 2003, pp. 118–129, LNCS–Springer.

[18] M. Bianchini, P. Mazzoni, L. Sarti, and F. Scarselli, "Face spotting in color images using recursive neural networks," in *IAPR - TC3 International Workshop on Artificial Neural Networks in Pattern Recognition*, M. Gori and S. Marinai, Eds., Florence (Italy), September 2003.