Available online at www.sciencedirect.com
SCIENCE DIRECT®
Neural Networks 18 (2005) 1040–1050
Neural Networks
www.elsevier.com/locate/neunet

2005 Special Issue

Recursive neural networks for processing graphs with labelled edges: theory and applications

M. Bianchini*, M. Maggini, L. Sarti, F. Scarselli

Dipartimento di Ingegneria dell'Informazione Università degli Studi di Siena Via Roma, 56 53100 - Siena (Italy)

Abstract

In this paper, we introduce a new recursive neural network model able to process directed acyclic graphs with labelled edges. The model uses a state transition function which considers the edge labels and is independent both from the number and the order of the children of each node. The computational capabilities of the new recursive architecture are assessed. Moreover, in order to test the proposed architecture on a practical challenging application, the problem of object detection in images is also addressed. In fact, the localization of target objects is a preliminary step in any recognition system. The proposed technique is general and can be applied in different detection systems, since it does not exploit any a priori knowledge on the particular problem. Some experiments on face detection, carried out on scenes acquired by an indoor camera, are reported, showing very promising results.

1. Introduction

Recursive neural networks (RNNs) are a connectionist model (Frasconi, Gori & Sperduti, 1998; Küchler and Goller, 1996; Sperduti et al., 1997) particularly suited to process Directed Positional Acyclic Graphs (DPAGs). In the case of DPAGs, each arc starting from a node has an assigned position, and any rearrangement of the children of a node corresponds to a different graph. While this assumption is useful in some applications, it sometimes introduces unnatural constraints on the representation. For example, this hypothesis is not suitable when representing a chemical compound or a protein structure, and might not be adequate in several pattern recognition problems.

The concept of invariance or symmetry has emerged as one of the key ideas in many different areas, such as physics, mathematics, psychology, and engineering (Lenz, 1994). From a theoretical point of view, seminal contributions in permutation invariant algebras can be found in (Curtis and Reiner, 1988; Fulton and Harris, 1991). Significant applications are in computer vision (Lenz, 1994; Mundy et al., 1994), in signal theory (Schempp, 1986), and in pattern recognition (Lenz, 1987; Lenz, 1989). Even in the neural network research field, there have been some attempts to construct application-specific architectures, that are permutation invariant w.r.t. the inputs. Invariance is often realized with preprocessing (Lahtinen et al., 2000; Chiu & Tseng, 1997), or via the extraction of features which are invariant under input transformations (Nordström et al., 1994). Otherwise, *ad hoc* network models are used, to preserve the output under particular input transformations (Bianchini et al., 2001; le Cun et al., 1989).

In this paper, we present a novel recursive neural network model, able to process Directed Acyclic Graphs with Labelled Edges (DAGs-LE). At each node of the structure to be learnt, the state transition function is independent both from the number and the order of the children. In fact, the particular contribution of each child is weighed by the label attached to the corresponding edge, that can encode the appropriate information to specify the nature of the relationship between the two nodes. The computational capabilities of RNNs-LE (Recursive Neural Networks for DAGs-LE) are also established, assessing that the universal approximation property is maintained for the classes of DPAGs and DAGs (which can be mapped onto DAGs-LE).

From a practical point of view, RNNs have been widely used in computer vision, f.i. in logo recognition (Diligenti et al., 2001), in similarity retrieval (de Mauro et al., 2003),

* Fax: +39 0577 233602.
E-mail addresses: monica@dii.unisi.it (M. Bianchini), maggini@dii.unisi.it (M. Maggini), sarti@dii.unisi.it (L. Sarti), franco@dii.unisi.it (F. Scarselli).

0893-6080/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2005.07.003

and for object detection (Bianchini et al., 2003b). Moreover, recently, there has been a growth of interest in object detection systems, which must be inherently robust to the wide variations that are observed in real-world 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 the other objects found 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 size. Furthermore, the objects can be partially occluded or can be blended in with the background. In order to test RNNs-LE, we propose to represent images by graphs with labelled edges. Such graphs are then transformed into forests of trees and processed by the new recursive model. 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. Actually, preliminary experiments on face detection show very promising results.

The paper is organized as follows. In the next section, some notation is introduced. The RNN-LE model is presented in Section 3, and Section 4 shows some theoretical results, assessing the computational power of the proposed model (all the proofs are collected in the Appendix). In Section 5, the problem of object detection is introduced and the graph-based representation of images is described, whereas Section 6 collects the experimental results on a face detection task. Finally, Section 7 draws some conclusions.

2. Notation

Let $G=(V,E,\mathcal{L})$ be a directed graph, where V is the set of nodes (or vertexes), $E\subseteq V\times V$ represents the set of arcs (or edges), and $\mathcal{L}: V\rightarrow\mathbb{Q}^m$ is a node labeling function, being $\mathbb{Q}$ the set of rationals. 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 $o=\max_v$ od[v] is the maximum outdegree.

In this paper, we consider the class of Directed Acyclic Graphs (DAGs), where a partial ordering can be defined on E, such that $v<w$ if v is connected to w by a direct path. Directed Positional Acyclic Graphs (DPAGs), for which recursive networks were originally defined, are a subset of DAGs, where an injective function o_v: 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. For this class of graphs, we denote with ch$_i$[v] the i- th child of node v. The definition of the class of DAGs with Labelled Edges (DAGs-LE) requires the introduction of an additional edge labeling function $\mathcal{E}$, such that $G=(V,E,\mathcal{L},\mathcal{E})$, where $\mathcal{E}: E\rightarrow\mathbb{Q}^k$. The presence of an edge label introduces a possible semantical content into the link between two nodes. Finally, we denote with PTREEs and TREEs-LE, respectively, the subsets of DPAGs and DAGs-LE that contain graphs which are trees, i.e. such that each node in the structure (except the root) has just one parent.

3. Recursive neural networks for processing DAGs-LE

Recursive neural networks were originally proposed to process DPAGs (Frasconi et al., 1998; Küchler & Goller, 1996; Sperduti et al., 1997), i.e. to realize functions from the space of directed positional acyclic graphs to an Euclidean space, in which the structures can be appropriately represented in order to solve the classification or approximation problem at hand. Let G be a DPAG. In the following, we shall require G either to be empty or to possess a *supersource*, i.e. a vertex $s\in V$ such that any other vertex of G can be reached by a directed path starting from s. Such a requirement is strictly related to the processing scheme which will be subsequently described[1]

In order to process a graph G, the recursive network is unfolded through the graph structure, producing the *encoding* network (see Fig. 1).

At each node v of the graph, the *state* $\mathbf{X}_v$ is computed by a feedforward network as a function of the node label $\mathbf{U}_v$ and the state of its children:

$$\mathbf{X}_v = f(\mathbf{X}_{\mathrm{ch}[v]}, \mathbf{U}_v, \theta_f), \tag{1}$$

with

$$\mathbf{X}_{\mathrm{ch}[v]} = [\mathbf{X}'_{\mathrm{ch}_1[v]}, \ldots, \mathbf{X}'_{\mathrm{ch}_o[v]}]', \quad o = \max_{v\in V}\{\mathrm{od}[v]\},$$

where $\mathbf{X}_{\mathrm{ch}_i[v]}$ is set to the *frontier state* $\mathbf{X}_0$ if node v lacks of its i-th child, and ‘$'$‘ represents the transpose operator.

At the supersource, an output function is computed by a feedforward network, called the *output* network:

$$\mathbf{Y}_s = g(\mathbf{X}_s, \theta_g).$$

The parameters θ_f and θ_g are connection weights, being θ_f independent of node v[2]. The parametric functions f and g can be implemented by a variety of neural network models.

Following Eq. (1), the state transition function f, computed by the 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 this limitation, in (Bianchini et al., 2001), a weight sharing approach was proposed, to relax the order

[1] Note that if a DPAG does not have a supersource, it is still possible to add an extra vertex s with a minimal number of outgoing edges, such that s is a supersource for the expanded DPAG (Sperduti et al., 1997).

[2] In this case, we say that the recursive neural network is *stationary*.

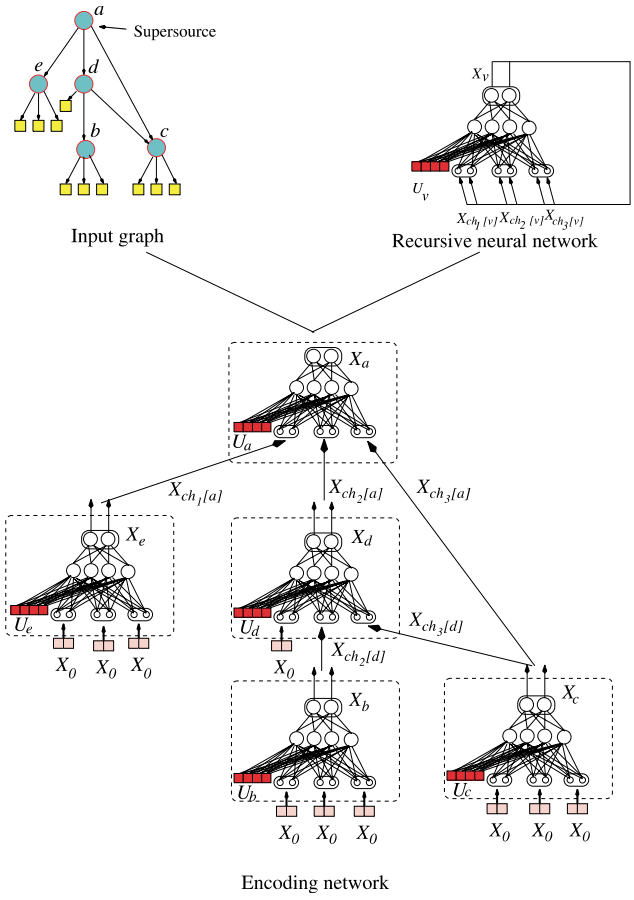

Fig. 1. The *encoding network* associated to a graph. The *recursive network* is un- folded through the structure of the graph.

constraint and to realize a neural network architecture suited for DAGs with a bounded outdegree. Unfortunately, the resulting architecture must have $\mathcal{O}(o!)$ hidden units, being o the maximum outdegree of the structures to be learnt. Thus it cannot be applied to DAGs with a large o. Moreover, even if o can be bounded, for instance by pruning those connections that are heuristically classified as less informative, nevertheless some important information may be discarded in this preprocessing phase. On the other hand, both the ordering constraint and the bound on the maximum outdegree can be removed when considering DAGs-LE (Gori et al., 2003).

For DAGs-LE, a state transition function 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 $\bar{\mathbf{X}}(\mathrm{ch}[v]) \in \mathbb{R}^p$ of the children is computed as

$$\bar{\mathbf{X}}(\mathrm{ch}[v]) = \sum_{i=1}^{|\mathrm{ch}[v]|} \phi(\mathbf{X}_{\mathrm{ch}_i[v]}, \mathbf{L}_{(v,\mathrm{ch}_i[v])}, \theta_\phi) \tag{2}$$

where $\mathbf{L}_{(v,\mathrm{ch}_i[v])} \in \mathbb{R}^k$ is the label attached to the edge $(v, \mathrm{ch}_i[v])$ and the *edge-weighting function* $\phi : \mathbb{R}^{(n+k)} \to \mathbb{R}^p$ is a nonlinear function parameterized by θ_ϕ. Then, the state at node v is computed combining $\bar{\mathbf{X}}(\mathrm{ch}[v])$ and the node label $\mathbf{U}_v$ by a parametric function $\tilde{f}$, as

$$\mathbf{X}_v = f(\mathbf{X}_{\mathrm{ch}[v]}, \mathbf{U}_v, \theta_f) \doteq \tilde{f}(\bar{\mathbf{X}}(\mathrm{ch}[v]), \mathbf{U}_v, \theta_{\tilde{f}}). \tag{3}$$

With this approach, the transition function f can be applied to nodes with any number of children and is independent of the order of the children.

The parametric functions ϕ, $\tilde{f}$ and g, involved in the recursive network, can be implemented by feedforward neural networks. For example, ϕ an be computed by a two-layer perceptron with linear outputs, as

$$\phi(\mathbf{X}_{\mathrm{ch}_i[v]}, \mathbf{L}_{(v,\mathrm{ch}_i[v])}, \theta_\phi) = \mathbf{V}\vec{\sigma}(\mathbf{A}\mathbf{X}_{\mathrm{ch}_i[v]} + \mathbf{B}\mathbf{L}_{(v,\mathrm{ch}_i[v])} + \mathbf{C}) + \mathbf{D},$$

where θ_ϕ collects $\mathbf{A} \in \mathbb{R}^{q,n}$, $\mathbf{B} \in \mathbb{R}^{q,k}$, $\mathbf{C} \in \mathbb{R}^q$, $\mathbf{D} \in \mathbb{R}^p$, and $\mathbf{V} \in \mathbb{R}^{p,q}$, being q the number of hidden neurons and $\vec{\sigma}$ a sigmoidal vectorial function.

On the other hand, the function ϕ an be also realized by an *ad hoc* model. In the following, we will consider the solution originally proposed in (Gori et al., 2003), where ϕ is realized as

$$\bar{\mathbf{X}}(\mathrm{ch}[v]) = \sum_{i=1}^{|\mathrm{ch}[v]|} \left(\sum_{j=1}^{k} \mathbf{H}_j \mathbf{L}^{(j)}_{(v,\mathrm{ch}_i[v])} \right) \mathbf{X}_{\mathrm{ch}_i[v]}, \tag{4}$$

being $\mathbf{H} \in \mathbb{R}^{p,n,k}$ the edge-weight matrix. In particular, $\mathbf{H}_j \in \mathbb{R}^{p,n}$ 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.

4. Theoretical results

RNNs have been proved to be able to approximate in probability any function on PTREEs (Hammer, 1999a). RNNs-LE differ from RNNs because the model of the transition function (Eq. (2) and (3)) is less general. Apart from this limitation, in the following we will prove that RNNs-LE can approximate any function both on PTREEs and TREEs-LE.

Thus, let us denote by p the function that takes a DPAG as input and returns the corresponding DAG-LE. The following theorem shows that RNNs-LE can approximate any function on PTREEs provided that the trees are preprocessed using p. Such a result holds even if the RNN-LE is implemented by the edge- weighting function of Eq. (4)

Theorem 1. *Given a measurable function* t: *PTREEs* $\to \mathbb{R}^r$, *a probability measure* P *on PTREEs, and any real* $\varepsilon > 0$, *there is a function* $\tilde{h}$, *realized by an RNN-LE whose activation function is defined as in* (3) and (4), *such that*

$$P(||t(G) - \tilde{h}(p(G))|| \geq \varepsilon) \leq \varepsilon.$$

Proof. See Appendix A.

On the other hand, RNNs-LE which implement the edge weighting function (2) and (3) can approximate any function on TREEs-LE.

Theorem 2. *For any measurable function t: $TREEs-LE \rightarrow \mathbb{R}$, any probability measure P on TREEs-LE, any real $\varepsilon > 0$, there exists a function h implemented by a RNN-LE such that*

$$P(||t(T) - h(T)|| \geq \varepsilon) \leq \varepsilon \tag{5}$$

holds and the states attached to nodes are reals, i.e. $n = 1$. Moreover, the functions $g, \tilde{f}, \phi$ the RNN-LE can be realized by two-layer (one hidden layer) perceptrons.

Proof. See Appendix B.

Apart from its theoretical properties, the proposed recursive model is particularly tailored to address problems involving graph-structured information, where the symbolic portion of data is not simply related to the entities represented by the nodes, but instead to their mutual relationships encoded in the graph arcs.

In the following, we describe an application to object detection in images, which supports the use of RNNs-LE for tasks where the information is appropriately represented by DAGs-LE.

5. The RNN-LE model for detecting objects in images

Object detection methods can be classified in four main categories (Yang et al., 2002):

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

Knowledge-based methods exploit the human knowledge on the objects to be detected and use some rules in order to describe their models. These rules are used to localize the objects that match the predefined models. Instead, the aim of feature invariant approaches (McKenna et al., 1998; Leung et al., 1998) is to define a set of features that are invariant w.r.t. object orientation, dimension, and light conditions. 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 (Shina, 1995). Finally, in appearance-based methods, machine learning techniques are exploited to learn templates from examples (Schneiderman & Kanade, 2000; Papageorgiou et al., 1998; Moghaddam & Pentland, 1997). In the following, we present a new appearance-based method that uses RNNs-LE. The proposed approach exploits a representation of images using graphs with labelled edges (see, f.i. (Ambauen et al., 2003)). Such graphs are then transformed into forests of trees and processed by an *ad hoc* RNN-LE.

5.1. The Graph-based representation of images

The proposed object detection method assumes a graph-based representation of images, which are originally encoded in the RGB (Red, Green, Blue) color space.

In the preprocessing phase, 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 algorithm used in the experiments is based on a K-means clustering of the pixels in the image, described by color and texture features. At the end of the clustering phase, 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 of the original image. After the segmentation process, 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 labelled edges. Strictly speaking, the *Region Adjacency Graph with Labelled Edges* (RAG-LE) is extracted from the segmented image by:

1 Associating a node to each region, where 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 the RAG-LE that corresponds to a part of the object in which we are interested, otherwise a target equal to 0 is specified. In Fig. 2, the RAG-LE extraction is sketched. 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 the white nodes correspond to parts of different objects and have target 0.

Since RNNs can process only Directed Positional Acyclic Graphs (DPAGs), each RAG-LE must be transformed into a directed graph. The transformation procedure considers a RAG-LE R, along with a selected node n, as inputs, 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 contains the same information as R (Bianchini et al., 2003a). In fact, each tree in the forest is recursive-equivalent to R, but only if R has unique labels, an hypothesis that holds with probability one,

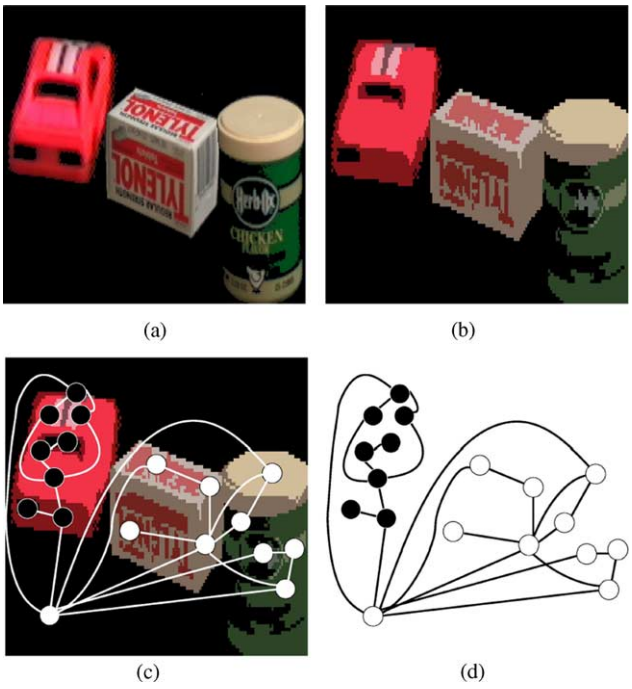

Fig. 2. The original image (a), the segmented image (b), and the extracted RAG-LE (c, d).

being the labels represented by attributes corresponding to real-valued features (e.g. area, perimeter, momentum, etc.).

The first step of the procedure is a preprocessing phase that transforms R into a directed RAG-LE G (see Fig. 3), by assuming that a pair of directed edges replaces each undirected one, thus preserving the symmetrical relationship between the nodes. The label attached to each undirected edge is attached to both the directed arcs. G is then unfolded to T by the following algorithm:

1 Insert a copy of n into 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 preserving the information attached to the edge;
3 Continue the visit until a predefined stop criterion is satisfied, and, however, until all the arcs have been visited (at least) once.

At the end of the algorithm, the target associated to n in R is attached to the root node of T. According to (Bianchini et al., 2003a), the above unfolding strategy produces a recursive-equivalent tree that holds the same information contained in R. On the basis of the selected stop criterion, 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. 3(a)). However, other stop criteria can be applied. For example, each edge can be visited once; then the visit proceeds starting from each leaf node $v \in T$, if a stochastic variable x_v is true, adding all the children of v to T (Probabilistic

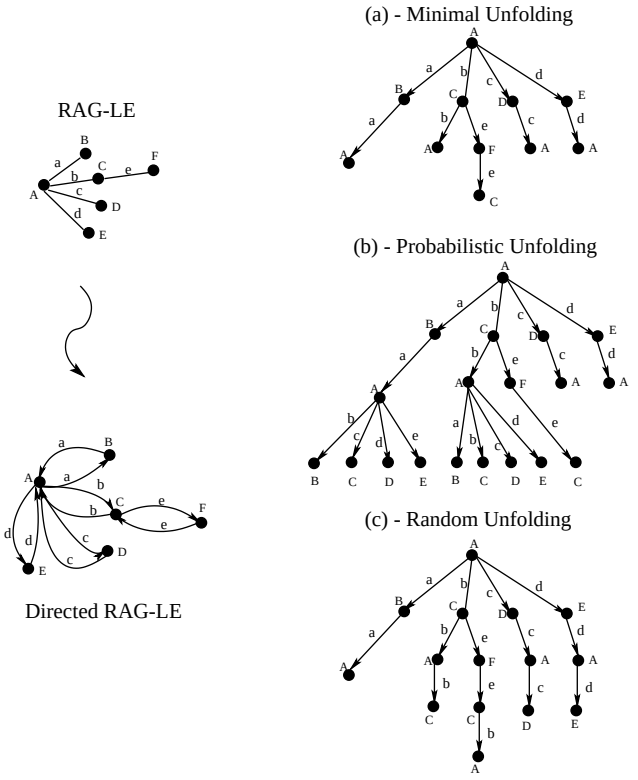

Fig. 3. The transformation from a RAG-LE to a recursive-equivalent tree. The dimension of the recursive-equivalent tree depends on the stop criterion selected during the unfolding of the directed RAG-LE.

Unfolding—see Fig. 3(b)). Otherwise, we can replace the breadth-first visit with a random visit of the graph (Random Unfolding — see Fig 3(c)). In this case, starting from the current node v, the visit can proceed or not depending on a set of stochastic variables $x_{v_1}, \ldots, x_{v_o}$, one for each arc outcoming from v. The probability of visiting a given arc is uniform over the whole graphical structure. Anyway, each edge must be visited at least once in order to guarantee the recursive-equivalence between R and T. In (Bianchini et al., in press), the probabilistic unfolding was experimentally proved to outperform the other methods.

6. Experimental results

The evaluation of the new recursive model was performed on the task of face detection 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 348 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

Fig. 4. 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.

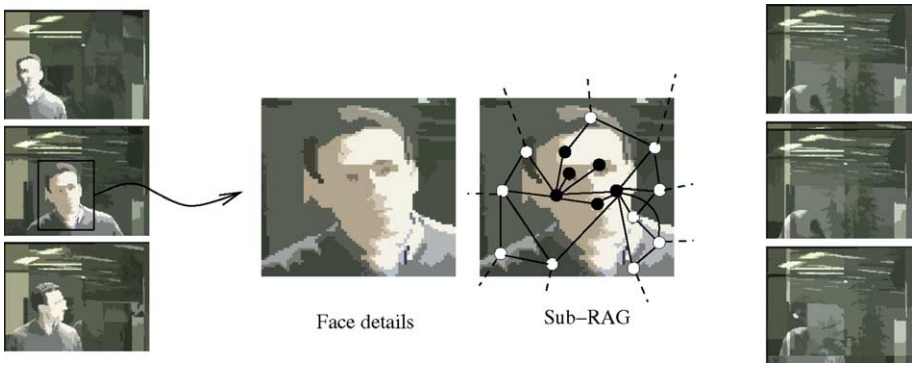

Fig. 5. Some examples of segmented images.

was performed, and no information derived from the movement of the object was exploited. The faces appear in different orientations, dimensions, and positions (see Fig. 4). 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 —see Fig. 5. The obtained RAGs-LE collect about 70 nodes on average, and each face, if it is present, is represented by about 15 nodes. 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 5, which was empirically shown to be the most effective

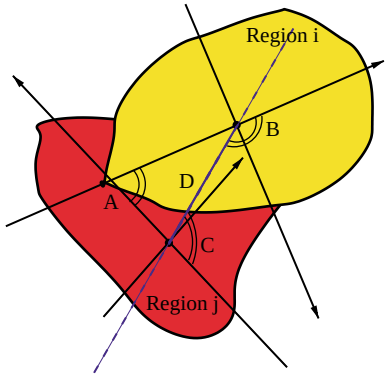

Fig. 6. Features stored into the edge label.

(Bianchini et al., 2003). 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.

In order to obtain balanced training and cross-validation sets, each RAG-LE, corresponding to a training or a validation image, was unfolded starting the breadth-first visit from all the nodes belonging to a part of a face and from a randomly chosen set of nodes which do not belong to parts of faces. We assume that the number of nodes corresponding to parts of faces is smaller than the number of the other kind of nodes, and this assumption is always true considering our dataset. On the contrary, the test set is obtained performing the Probabilistic Unfolding for all the nodes belonging to each RAG-LE. In fact, in order to locate faces, the trained RNN-LE must be able to predict whether each node in the recursive-equivalent trees represents a part of a face or not.

Several RNNs-LE were trained in order to determine how the implementation of the function ϕ (see Section 3) affects the detection results. In fact, the function ϕ can be implemented using a feedforward neural network (neural ϕ-see Eq. (2)) or using an *ad hoc* model (linear ϕ). In the neural case, ϕ can be obtained considering a two-layer feedforward neural network with sigmoidal hidden units and linear output units, while in the linear case a three dimensional weight matrix can be considered (see Eq. (4)). In the following, a comparison between the neural and the linear model is performed. The RNN-LE architecture was varied in order to determine the best architecture and to investigate how ϕ should be implemented. In particular, six distinct RNNs-LE were trained considering 5, 10, and 15 state neurons, both in the neural and in the linear case. Moreover, the hidden layer of the feedforward neural networks used to implement the neural ϕ collects a number of neurons equal to the number of the state neurons. Each RNN-LE was trained several times using the BPTS algorithm (Sperduti et al., 1997), exploiting both momentum and an adaptive learning rate strategy, and the results reported in Table 1 represent the average accuracy obtained on the test set.

Table 1
Accuracy Rate of the trained RNNs-LE using the unbalanced test set

RNN-LE	Face	Not Face	Global
Architecture	Accuracy	Accuracy	Accuracy
Linearϕ - 5 state	0.9201	0.7890	0.8225
Linearϕ - 10 state	0.8630	0.6909	0.7426
Linearϕ - 15 state	0.8795	0.6650	0.7292
Neuralϕ - 5 state	0.8996	0.8568	0.8696
Neuralϕ - 10 state	0.9067	0.8341	0.8558
Neuralϕ - 15 state	0.8087	0.7938	0.7982

The reported results show that RNNs-LE in which ϕ is implemented using a feedforward neural network yield a better performance. As a matter of fact, all the tested RNN-LE architectures succeed in their learning task showing how the proposed model is able to generalize, even if the training is performed on perfectly balanced data sets.

The accuracy rates reported in Table 1 are computed dividing the number of trees (regions) correctly classified by the total number of trees (regions) and can be considered as the portions of faces correctly detected.

In a practical application, a postprocessing procedure uses the prediction provided by a trained RNN-LE in order to compute the exact location of faces. In fact, adjacent regions predicted as parts of a face are merged together in order to compute the minimum bounding boxes that surround the face.

Since the focus of this paper is mainly on the RNN-LE model, instead of the object detection problem, the postprocessing procedure was not carried out. More details on the postprocessing phase can be found in (Bianchini et al., 2003b; Bianchini et al., 2004), where the achieved results show that the accuracy in the detection of the bounding boxes is generally greater than the accuracy reached by the RNN-LE on the single regions. Actually, the correct bounding boxes can be computed even if some regions belonging to faces are not correctly classified.

7. Conclusions

In this paper, a novel recursive neural network model, able to process DAGs with labelled edges, was presented. The computational capabilities of the new recursive model were discussed, assessing that it is an universal approximator with respect to the classes of DPAGs and DAGs. Moreover, the new model was exploited in order to define a new appearance-based method for object detection in images, which are represented by DAGs-LE. This approach is invariant under image translation and rotation, due to the invariance of the graphical representation used. Furthermore, the proposed object detection technique does not exploit any a priori or heuristic information on the specific object in which we are interested. The experimentation carried out to evaluate the method shows very promising results.

Appendix

A.1. Proof of Theorem 1

In Section 3, Eqs. (3) and (4) describe how to compute the state of a node v, given the contribution of its children, independently of their number and order. In the following lemmas, the equivalence between RNNs on DPAGs and RNNs-LE on DAGs-LE is established.

Notice that, at each node v of the input graph, the feedforward network which computes the state transition function f of Eq. (4) can be realized by a three-layer perceptron, having $\mathbf{H}_j$, $j=1,\ldots,k$, as input-to-hidden weight matrices. In fact, Eq. (4) may be rewritten as

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

which shows how the contribution of the children of each node to its state can be computed using a three-layer perceptron with kn 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 recursive neural network for processing graphs with labelled edges (RNN-LE) is depicted in Fig. A.1.

Even if the edge labels increase the semantical content attached to the links, it is worth studying how such labels can be used to encode the position of the children.

Lemma 1. *Each DPAG can be represented by a DAG-LE.*

Proof. The assertion follows straightforwardly by observing that the position of the i-th child can be encoded by the label attached to the link. In particular, for each node in the DPAG, a corresponding node exists in the DAG-LE, and the edge labels must have dimension o (being o the maximum outdegree). Moreover, for each node of the DPAG the corresponding node in the DAG-LE has o outcoming edges, with labels having all but one zero entries (one-hot code of the position). An entry equal to one stands for the presence of the child in the corresponding position (in this case, the labels form the Euclidean basis of $\mathbb{R}^o$).

Lemma 2. Let p: DPAG$\rightarrow$DAG-LE the function that realizes the preprocessing described in Lemma 1. For any DPAG G and any recursive neural network that computes η, there exists an RNN-LE, implementing $\tilde{\eta}$, such that $\eta(G)=\tilde{\eta}(p(G))$.

Proof. Let us consider the RNN model defined in (1) in which, at each node v, the state transition function f is computed using a multi-layer perceptron, with sigmoidal hidden neurons and linear output neurons, in which the matrices $\mathbf{A}_k$, $k=1,\ldots,o$ are the weights between the inputs and the first hidden layer in the state transition network (Bianchini et al., 2001). Then, referring to Fig. A.1, the proof follows directly by choosing $\mathbf{A}\cdot\mathbf{H}_k=\mathbf{A}_k$ in the RNN-LE, and assuming $\mathbf{X}_0=\mathbf{0}$.

□

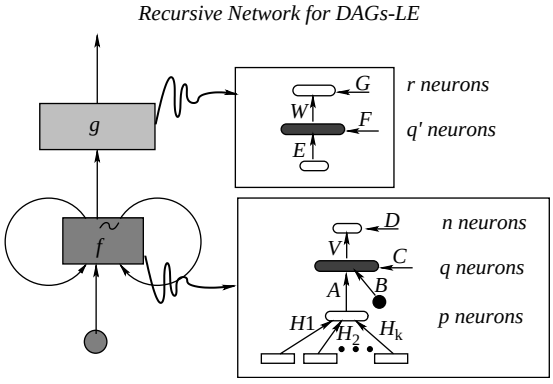

Fig. A.1. An MLP implementation of the recursive network. Grey layers are sigmoidal, whereas white layers are linear in both the feedforward networks.

Proof of Theorem 1. The assertion follows by Lemma 2 and by the results on universal approximation capabilities of standard RNNs reported in (Hammer, 1999a; Hammer and Sperschneider, 1997; Hammer, 1999b).

A.2. Proof of Theorem 2

The approach used for this proof is similar to that proposed in (Hammer, 1999a). The proof will be carried out by two steps: first, we will reduce the problem to the task of defining an encoding function that maps TREEs-LE onto real numbers; then, we will specify the encoding function and we will prove that it can be implemented by an RNN-LE.

A.2.1. Problem reduction

Each tree in TREEs-LE is completely defined by its structure and its labels. Since both the sets of the possible structures and the set of the admissible labels are enumerable, then also the class of TREEs-LE is enumerable. Thus, by using $P(\text{TREEs-LE})=1$ and the enumerability of TREEs-LE, there exists a finite subset $Q=\{T_1,\ldots,T_k\}\subseteq$ TREEs-LE such that $P(Q)>1-\varepsilon$, for any $\varepsilon>0$.

Let us denote by $\mathrm{Sub}(T,n)$ the subtree of T whose root is the node n. The main idea supporting our proof is that, while processing a tree T, the state $\mathbf{X}_n$ contains a representation of $\mathrm{Sub}(T,n)$, i.e. $\mathbf{X}_n=\mathrm{Cod}(\mathrm{Sub}(T,n))$ for some encoding function Cod. In this way, the state at the root s of T stores an encoding of the whole tree T, i.e. $\mathbf{X}_s=\mathrm{Cod}(T)$.

On the other hand, provided that Cod maps different trees in Q into different encodings, i.e. $\mathrm{Cod}(T_i)\neq\mathrm{Cod}(T_j)$ if $i\neq j$, we can always find a function $\hat{g}$ such that $\hat{g}(\mathbf{X}_s)=\hat{g}(\mathrm{Cod}(T_i))=t(T_i)$, being t the target function. Moreover, $\hat{g}$ can be approximated up to any degree of precision by a function g realized by a two-layer perceptron (Hornik et al., 1989; Cybenko, 1989; Funahashi, 1989)

$$\max_{i=1,\ldots,k} |t(T_i)-g(\mathrm{Cod}(T_i))| < \varepsilon. \tag{A1}$$

This implies that

$$P(||t(T)-g(\mathrm{Cod}(T))|| \geq \varepsilon) \leq \varepsilon,$$

because $P(\text{TREEs-LE}\backslash Q)\leq\varepsilon$ by definition of Q.

Thus, it remains to prove that the encoding Cod exists and it can be implemented by a transition function f, whose recursive application, from the leaves to the root of T, constructs $\mathbf{X}_s=\mathrm{Cod}(T)$. In this case, the thesis follows,

when the transition function of the RNN-LE is f and the output function is g.

A.2.2. Encoding function

In the following, first a function Cod is defined that maps each subtree Sub(T, n) into a real number, where $T \in Q$ and n is a node of T. Then, we will prove that Cod is injective, i.e. it assigns different real numbers to different inputs. Finally, we will define the functions $\tilde{f}$ and ϕ of Eqs. (2) and (3), respectively.

Definition of Cod

Function Cod is specified by three steps:

1 Let us consider the set Q^L of the labels which are attached to a node or an edge of a tree in Q. Since, Q^L is finite, we can define an injective function (an encoding) Cl: $Q^L \to \mathbb{N}$ which assigns to each label a different positive integer.
2 Moreover, also the set Q^T of the pairs (Sub(T, u), $\mathbf{L}_{(n,u)}$) is finite, where $T \in Q$, n is a node of T, and u one of its children. Thus, we can define an injective function (an encoding) Ct: $Q^T \to \mathbb{N}$ such that it assigns a different positive integer to each pair in Q^T.
3 Finally, Cod is defined as

$$\mathrm{Cod}(\mathrm{Sub}(T,n)) = N^{-1}\mathrm{Cl}(U_n) + \sum_{i=1}^{|\mathrm{ch}[n]|} N^{\mathrm{Ct}(\mathrm{Sub}(T,\mathrm{ch}_i[n]),\mathbf{L}_{(n,\mathrm{ch}_i[n])})} \quad \text{(A2)}$$

where N is a parameter. For our purpose, N can assume any value, provided that it is an integer and fulfills

$$N > \max_n(|\mathrm{ch}[n]|), \quad N > \max_n(\mathrm{Cl}(U_n)),$$

where the computation of $\max_n$ takes into account only the nodes n contained in Q.

It is worth noting that such an encoding function produces a real number, where the fractional part represents U_n and the integer part represents the set of pairs $\{(\mathrm{Sub}(T,\mathrm{ch}_i[n]),\mathbf{L}_{(n,\mathrm{ch}_i[n])}) | 1 \le i \le \mathrm{ch}[n]\}$. In fact, the sum in the right hand of Eq. (B.2) is a polynomial in N and it can be represented as $p(N) = \sum_{j=0}^{M} \alpha_j N^j$, where M is the maximal value which can be assumed by the term $\mathrm{Ct}(\mathrm{Sub}(T,\mathrm{ch}_i[n]),\mathbf{L}_{(n,\mathrm{ch}_i[n])})$. For each j, α_j is equal to the number of pairs $(\mathrm{Sub}(T,\mathrm{ch}_i[n]),\mathbf{L}_{(n,\mathrm{ch}_i[n])})$, that are mapped on j by Ct. In other words, each coefficient of the polynomial counts the number of occurrences of a given pair[3].

[3] In order to clarify the idea underlying the coding implemented by the sum in Eq. (B.2), let us use the base N representation of the integers. Since $\alpha_j \le \max_n(\mathrm{ch}[n]) < N$, each α_j can be represented by a single digit $\bar{\alpha}_j$. Thus, the following string $\bar{\alpha}_M \ldots \bar{\alpha}_1 \bar{\alpha}_0$ is a representation of the value of the sum in Eq. (A.2).

Cod is injective

Let us assume that

$$\mathrm{Cod}(\mathrm{Sub}(T_1,n_1)) = \mathrm{Cod}(\mathrm{Sub}(T_2,n_2)) \quad \text{(A3)}$$

holds, where $T_1 \in Q$, $T_2 \in Q$ and n_1, n_2 are nodes of T_1, T_2, respectively. We will show that Eq. (A.3) implies Sub(T_1, n_1) = Sub(T_2, n_2).

According to the previous discussion on Cod, there exist two polynomials p_1, p_2, such that

$$N^{-1}\mathrm{Cl}(U_{n_1}) + p_1(N) = \mathrm{Cod}(\mathrm{Sub}(T_1,n_1)) = \mathrm{Cod}(\mathrm{Sub}(T_2,n_2)) = N^{-1}\mathrm{Cl}(U_{n_2}) + p_2(N).$$

It follows that

$$\mathrm{Cl}(U_{n_1}) = \mathrm{Cl}(U_{n_2}), \quad \text{(A4)}$$

$$p_1(N) = p_2(N), \quad \text{(A.5)}$$

because $p_1(N)$ and $p_2(N)$ are integers, and both the terms $N^{-1}\mathrm{Cl}(U_{n_1})$ and $N^{-1}\mathrm{Cl}(U_{n_2})$ are smaller than 1.

Eq. (A.4) implies $U_{n_1} = U_{n_2}$. On the other hand, Eq. (A.5) implies that either N is a root of p_1–p_2 or the two polynomials are equal. However, the former case can be excluded by the following reasoning. Let $\beta_0,\ldots,\beta_M$ be the coefficients of polynomial p_1–p_2 and let β_k be the largest non–null coefficient. Since, β_j is the difference of the corresponding coefficients of p_1 and p_2, then $|\beta_j| \le N-1$ and N is not a root because

$$|p_1(N) - p_2(N)| = \left|\sum_{j=0}^{M} \beta_j N^j\right| \ge |\beta_k| N^k - \left|\sum_{j=0}^{k-1} \beta_j N^j\right| \ge N^k - (N-1)\left(\sum_{j=0}^{k-1} N^j\right) = 1.$$

Thus, the polynomials p_1 and p_2 are equal. On the other hand, $p_1 = p_2$ implies that also the two sets $S_1 = \{(\mathrm{Sub}(T_1, \mathrm{ch}_i[n_1]), \mathbf{L}_{(n_1,\mathrm{ch}_i[n_1])}) | 1 \le i \le |\mathrm{ch}[n_1]|\}$ and $S_2 = \{(\mathrm{Sub}(T_2, \mathrm{ch}_i[n_2]), \mathbf{L}_{(n_2,\mathrm{ch}_i[n_2])}) | 1 \le i \le |\mathrm{ch}[n_2]|\}$ are equal. Notice that the set S_1 (S_2) contains all the information included in the tree Sub(T_1, n_1) (Sub(T_2, n_2)) except for the label of the root U_{n_1}(U_{n_2}). Then, $S_1 = S_2$, along with the previously deduced equality $U_{n_1} = U_{n_2}$, implies Sub(T_1, n_2) = Sub(T_1, n_2).

Definition of $\tilde{f}$ and ϕ

Since Cod is injective, there exists a decoding function Dec such that Sub(T, n) = Dec(Cod(Sub(T, n))). Then, ϕ and

$\tilde{f}$ can be defined as

$$\phi(\mathbf{X}_{\mathrm{ch}_i[n]}, \mathbf{L}_{(n,\mathrm{ch}_i[n])}) = N^{\mathrm{Ct}(\mathrm{Dec}(\mathbf{X}_{\mathrm{ch}_i[n]}),\mathbf{L}_{(n,\mathbf{ch}_i[n])})},$$

$$\tilde{f}(\bar{\mathbf{X}}(\mathrm{ch}[n]), U_n) = N^{-1}\mathrm{Cl}(U_n) + \bar{\mathbf{X}}(\mathrm{ch}[n]).$$

By those functions, the application of the RNN-LE on a tree T produces the states $\mathbf{X}_n = \mathrm{Cod}(\mathrm{Sub}(T, n))$. This can be easily proved by induction on the depth d of the tree $\mathrm{Sub}(T, n)$.

Base: Consider all the trees $\mathrm{Sub}(T, n)$, whose depth is $d=1$.

Since the depth is $d=1$, n is a leaf and has no children. It immediately follows that

$$\mathbf{X}_n = N^{-1}\mathrm{Cl}(U_n) = \mathrm{Cod}(\mathrm{Sub}(T, n)).$$

Induction: Assume that $\mathbf{X}_u = \mathrm{Cod}(\mathrm{Sub}(T, u))$ holds for d, any $T \in Q$ and any node u.

Let $\mathrm{Sub}(T, n)$ be a tree whose depth is $d+1$. Since the children of n are on the lower level of the tree, the depth of each $\mathrm{Sub}(T, \mathrm{ch}_i[n])$ is at most d. By applying the induction hypothesis,

$$\mathrm{Dec}(\mathbf{X}_{\mathrm{ch}_i[n]}) = \mathrm{Dec}(\mathrm{Cod}(\mathrm{Sub}(T, \mathrm{ch}_i[n]))) = \mathrm{Sub}(T, \mathrm{ch}_i[n]),$$

for any i, $1 \le i \le |\mathrm{ch}[n]|$. As a consequence,

$$\begin{aligned}\mathbf{X}_n &= N^{-1}\mathrm{Cl}(U_n) + \sum_{i=1}^{|\mathrm{ch}[n]|} N^{\mathrm{Ct}(\mathrm{Dec}(\mathbf{X}_{\mathrm{ch}_i[n]}),\mathbf{L}_{(n,\mathrm{ch}_i[n])})} \\ &= N^{-1}\mathrm{Cl}(U_n) + \sum_{i=1}^{|\mathrm{ch}[n]|} N^{\mathrm{Ct}(\mathrm{Sub}(T,\mathrm{ch}_i[n]),\mathbf{L}_{(n,\mathbf{ch}_i[n])})} \\ &= \mathrm{Cod}(\mathrm{Sub}(T, n))\end{aligned}$$

Thus, we have defined a transition function that encodes $\mathrm{Sub}(T, n)$ into the state $\mathbf{X}_n$. Since, by the approximation properties of feedforward neural networks (Hornik et al., 1989; Cybenko, 1989; Funahashi, 1989), $\tilde{f}$ and ϕ can be approximated by two layer perceptrons, the theorem has been proved.

References

Ambauen, R., Fischer, S., & Bunke, H. (2003). Graph edit distance with node splitting and merging, and its application to diatom identification. In E. R. Hancock, & M. Vento (Eds.), *Lecture notes in computer science 2726* (pp. 95–106). York: Springer.

Bianchini, M., Gori, M., & Scarselli, F. (2001). Processing directed acyclic graphs with recursive neural networks. *IEEE Transactions oil Neural Networks*, *12*(6), 1464–1470.

Bianchini, M., Gori, M., Sarti, L., & Scarselli, F. (2003a). Backpropagation through cyclic structures. In A. Cappelli, & F. Turini (Eds.), *LNAI - AI*IA 2003: Advances in artificial intelligence* (pp. 118–129). Pisa, Italy: LNCS-Springer.

Bianchini M., Maggini M., Sarti L., & Scarselli F. (in press). Recursive neural networks learn to localize faces. *Pattern recognition letters.*

Bianchini, M., Mazzoni, P., Sarti, L., & Scarselli, F. (2003b). Face spotting in color images using recursive neural networks. In M. Gori, & S. Marinai (Eds.), *IAPR - TC3 International workshop on artificial neural networks in pattern recognition,* Florence, Italy, September.

Bianchini M., Maggini M., Sarti L., & Scarselli F. (2004). Recursive neural networks for object detection. In *Proceedings of the IEEE international joint conference on neural networks (IJCNN 2004)* (pp. 1911–1915).

Chiu, H. P., & Tseng, D. C. (1997). Invariant handwritten Chinese character recognition using fuzzy min-max neural networks. *Pattern Recognition*, *18*(5), 481–491.

Curtis, C. W., & Reiner, L. (1988). *Representation theory of finite groups and associative algebras*. New York: Wiley.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *3*, 303–314.

de Mauro, C., Diligenti, M., Gori, M., & Maggini, M. (2003). Similarity learning for graph based image representation. *Special issue of Pattern Recognition Letters*, *24*(8), 1115–1122.

Diligenti, M., Gori, M., Maggini, M., & Martinelli, E. (2001). Adaptive graphical pattern recognition for the classification of company logos. *Pattern Recognition*, *34*, 2049–2061.

Frasconi, P., Gori, M., & Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, *9*(5), 768–786.

Fulton W., & Harris J. (1991). Representation theory. In *Graduate texts in mathematics*. New York: Springer-Verlag.

Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, *2*, 183–192.

Gori M., Maggini M., & Sarti L. (2003). A recursive neural network model for processing directed acyclic graphs with labeled edges. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN 2003)* (pp. 1351–1355).

Hammer, B., & Sperschneider, V. (1997). Neural networks can approximate mappings on structured objects. In P. P. Wang (Ed.), *International conference on computational intelligence and neural networks'97*, 211–214.

Hammer B. (1999a). Approximation capabilities of folding networks. In *Proceedings of ESANN '99* Bruges, Belgium, April (pp. 33–38).

Hammer, B. (1999b). *Learning with recurrent neural networks*. PhD thesis, Fachbereich Mathematik/Informatik, Universität Osnabrück.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359–366.

Küchler, A., & Goller, C. (1996). Inductive learning in symbolic domains using structure-driven recurrent neural networks. In G. Görz, & S. Hölldobler (Eds.), *Advances in artificial intelligence* (pp. 183–197). Berlin: Springer.

Lahtinen J., Martinsen T., & Lampinen J. (2000). Improved rotational invariance for statistical inverse in electrical impedance tomography. In *International joint conference on neural networks*, Como, Italy (pp. 154–158).

le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*, 541–551.

Lenz, R. (1987). Optimal filters for the detection of linear patterns in 2-D and higher dimensional images. *Pattern Recognition*, *20*(2), 163–172.

Lenz R. (1989). A group theoretical approach to filter design. In *International conference on acoustic, speech, and signal processing* (pp. 1594–1597).

Lenz, R. (1994). Group theoretical transforms in image processing. *Current Topics in Pattern Recognition Research*, *1*, 83–106.

Leung T.K., Burl M.C., & Perona P. (1998). Probabilistic affine invariants for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 678–684).

McKenna, S., Raya, Y., & Gong, S. (1998). Tracking colour objects using adaptive mixture model. *Image and Vision Computing*, *17*(3/4), 223–229.

Moghaddam, B., & Pentland, A. (1997). Probabilistic visual learning for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(7), 696–710.

Mundy J.L., Zissermann A., & Forsyth D. (1994). Applications of invariance in computer vision. In *Lecture notes in computer science*. New York: Springer-Verlag.

Nordström, E., Gällmo, O., Asplund, L., Gustafsson, M., & Eriksson, B. (1994). Neural networks for admission control in an ATM network. In L. F. Niklasson, & M. B. Bóden (Eds.), *Connectionism in a broad perspective* (pp. 239–250). Chichester, UK: Ellis Horwood.

Papageorgiou C., Oren M., & Poggio T. (1998). A general framework for object detection. In *Proceedings of the 6th IEEE international conference on computer vision* (pp. 555–562).

Schempp W. (1986). Harmonic analysis of the Heisenberg nilpotent Lie group, with applications to signal theory. In *Pitman research notes in mathematics*. Harlow: Longman.

Schneiderman H., & Kanade T. (2000). A statistical method for 3D object detection applied to faces and cars. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 746–751).

Shina P. (1995). *Processing and Recognizing 3D forms*. PhD thesis, Massachusetts Institute of Technology.

Sperduti, A., & Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, *8*, 429–459.

Yang, M.-H., Kriegman, J., & Ahuja, N. (2002). Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(1), 34–58.