Logo Recognition by Recursive Neural Networks

E. Francesconi ◇ P. Frasconi ◇ M. Gori *
S. Marinai ◇ J. Q. Sheng ◇ G. Soda ◇ A. Sperduti •

◇ Dipartimento di Sistemi e Informatica - Università di Firenze - Italy
* Dipartimento di Ingegneria dell'Informazione - Università di Siena - Italy
• Dipartimento di Informatica - Università di Pisa - Italy

In this paper we propose an adaptive model, referred to as Recursive Neural Networks (RRNNs) for logo recognition by explicitly conveying logo item into m-ary tree representation, where symbolic and sub-symbolic information coexist. Each node in the contour-tree is associated with an exterior or interior contour extracted from the logo instance. A feature vector, which includes the perimeter of the contour, the area surrounded and the number of critical points at some pre-determined intervals, is associated with each node. The pattern representation transformed in this way contains the topological structured information of logo and continuous values pertaining to each contour node. Afterwards, the RRNNs are used to learn the logo regularities expressed by contour-trees. The experimental results are reported on 40 real logos with very promising results.

1 Introduction

The adaptive computational scheme of statistical models and artificial neural networks is very well-suited for pattern recognition. It is commonly recognized, however, that these adaptive models, especially neural networks, can hardly deal with highly-structured information. On the other hand, most of the models proposed in the field of syntactic pattern recognition are very-well suited for incorporating pattern grammatical regularities, but, one of their drawbacks is that they are not strongly oriented to face the presence of noise. This scenario seems to be mainly due to the development of computational approaches that are basically oriented to either dealing with symbols or with unstructured information, whilst in the field of pattern recognition one is often looking for a balanced combination of the symbolic and sub-symbolic processing levels. This paper is intended to bridge these two distinct paradigms by applying RRNNs to solving logo recognition problems, a typical task in which the significant presence of noise and the structured information associated with the patterns might require both paradigms be taken into account.

Logo recognition has been investigated extensively as an important pattern recognition task in the last few years. Many of the proposed approaches are based on syntactic approach [1] and statistical model (see e.g. [2] [3]). A connectionist-based approach to logo recognition, has been recently proposed in [4] and extended to deal with spot noises [5]. In these papers, the logo membership is estimated by multilayer perceptrons acting as autoassociators that are fed by a vector of unstructured features. Such an approach relies completely on the supposed capabilities of multilayer perceptrons to learn the logo regularities and face the presence of noise. Although very satisfactory results have been achieved, the effectiveness of this approach is likely to decrease for pattern recognition applications in which there is a strong symbol's component.

On the other hand, in order to unify symbolic and sub-symbolic processing, instead of solely

processing vector of real values, some researchers have recently investigated the feasibility of learning from examples structured data by using recursive artificial neural networks (RRNNs) [6]. The proposed neural model can be straightforwardly used for solving logo recognition problem, while the noise on the logo instance and the structured information derived from logos can be properly incorporated into RRNNs. We propose to make the logo structure explicit by offering the neural networks a representation which is based on tree structures, obtained from logos by using a recent algorithm proposed by Cortelazzo *et al.* [2]. An m-ary tree is created by recursively detecting contours and inclusion relations, so that nodes in odd levels of the tree are associated to exterior contours and even levels are associated to interior contours. Each node v is then labeled by a set of numerical features measured on the object associated to the node and collected in a multivariate variable U_v. Examples of these features are the perimeter of the contour, the area surrounded by the contour, and the number of critical points at some predetermined intervals. Scale invariance is obtained by appropriate normalization of continuous label and rotation invariance is gained by appropriate ordering of the children for a given node.

The paper is organized as follows: in Section 2, the tree representation of logos is presented while in Section 2.3 we describe our approaches of tree pruning and grafting. In Section 3 we describe with more detail the RRNN model and its relevance for logo recognition; In Section 4 we briefly introduce the network architecture used for our experiments for the logo recognition. In Section 5 we report the experimental results on 40 logos and finally some conclusions are drawn in Section 6.

2 Tree Construction

Structural representations of patterns can be gained by region- and contour-based approaches. The region-based approach derives a fixed number of out-degree of tree and contour-based approach constructs a tree by detecting contours and their relationship. We have chosen a recent algorithm proposed by by Cortelazzo *et al.* [2] to construct tree representation of logos. The algorithm creates an m-ary tree by recursively detecting contours and inclusion relations (see examples in Fig. 3), so that nodes in odd levels of the tree are associated to exterior contours and even levels are associated to interior contours. In this paper, the tree obtained by means of the algorithm is called *primary contour-tree*. It represents primary contour items and the topological inclusion relationships among extracted contours. Although the primary tree could be undertaken a graph pruning and grafting step in order to avoid learning very high valence tree, the tree nodes representing significant primitives remain unchanged, main features pertaining to the logo instance are reasonably retained. When a *primary contour-tree* be pruned and grafted, a *secondary contour-tree* is generated by deleting some nodes and links from the *primary contour-tree* and adding new nodes to the tree. The motivations for pruning and grafting *primary tree* are described in Section 2.3 in great detail.

Obviously, each node v in either *primary* or *secondary contour-tree* is labeled by a set of numerical features measured on the object(s) associated to the node(s), collected in a multivariate variable U_v. Appropriate normalization of continuous labels assures the scale-invariance and rotation invariance can be obtained through an ordering operation of children for a given node. In Fig. 2 we depict how logo can be represented by contour tree and their recursive neural models. By investigating the examples, it was noted that the feature vector remains unchanged after normalization of contour perimeters and areas, thus assuring the scale invariance. Furthermore, the ordering of the children according to contour topological characteristics for a node can guarantee the invariance of tree architecture, consequently, rotation-invariance is warranted, otherwise, the order of the children for a given node can be interchanged since the contour-tracing can be operated stochastically in the case of logo rotation.

2.1 Contour Tracing

Contour tracing is a local operation in a binarized image by traversing to one 8-neighbor pixel from a black one (suppose the white is background color). Contour tracing and contour-tree construction follows the algorithm described in [2]. We have used incoming direction for contour pixels in order to memorize the Freeman code instead of traditional outgoing direction. Pixel in the orthogonal direction w.r.t the incoming direction in clockwise order is first visited when scanning the raster image from the upper-left corner to bottom-right corner. The exterior contour is formed with clockwise ordered pixels and interior contour with anti-clockwise ordered pixels. Perimeter of contour is obtained directly in the phase of contour tracing by counting the even direction pixel as 1 and odd direction pixel as $\sqrt{2}$. The area surrounded by the contour is the total number of black pixels inside exterior contour and white pixels inside the interior contour. In the case of open exterior contour, the perimeter is the length of the contour and the area is equal to its length.

2.2 Feature Extraction

Feature values include the perimeter p of contour, the area a surrounded by the contour and the number of critical points at five predetermined intervals. In order to describe the feature extraction and their value computation, we formulate the following notation: From any given logo $\mathcal{L}_i$, N_c contours can be extracted; N_e denotes number of exterior contour, N_t denote the total number of interior contours, thus $N_c = N_e + N_t$. In order to maintain the consistency for the root node r of the tree, a rectangle is virtually applied to the image and corresponding feature vector can still be obtained. So the perimeter for the root is $P_r = (w + h) * 2$ where w and h are width and height of the entangled window respectively, and $A_r = w * h$.

Because the graph node is related closely to the contour, some feature values are derived through the analysis of contour image. A typical characteristic is the critical points with different critical levels along the contour pixels. In order to quantitize properly the features, we have used the Smoothed Compensated Extended Freeman Code (SCEFC) [7] method. Derivative values at points of contour is calculated on the basis of SCEFC. We have set up five intervals with three *threshold* values θ_i ($i = 0, 1, 2$) for derivative values, and the number of peak points whose derivative values fall into corresponding interval are chosen as a feature vector associated with graph node. This vector $\mathcal{D} = (D_i)$ ($i = 0 \ldots 4$) with five elements are unified together with the perimeter P and A, thus a vector can be consolidated as feature pertaining to the graph node. It can be formulated as follows.

$$
\begin{array}{lll}
U_v = & (P_v, A_v, D_v^{(0)}, \ldots, D_v^{(2)}) & \\
\text{where:} & & \\
 & D^{(0)} = \text{No}(p) & \text{where} \quad \Re(p) > \theta_0 \\
 & D^{(i)} = \text{No}(p) & \text{where} \quad \Re(p) \in ([\theta_{i-1}, \theta_i), i = 1, 2, 3. \\
 & D^{(4)} = \text{No}(p) & \text{where} \quad \Re(p) > \theta_4
\end{array}
\tag{1}
$$

$\Re(p)$ is the critical level for the run of points which contains the point p with the maximum value in the contour list. All elements in the *feature vector* are normalized in an appropriate way. The perimeter is normalized with the maximum contour length found in a logo, the area is normalized with the entangled logo area and the number of critical points are normalized with the highest value among the number of critical points at five intervals for all contours extracted

from the logo instance. We denote them using small letters as follows:

$$\begin{aligned} p_v &= \frac{P_v}{\max_{v=0,\ldots,n_c} P_v} \\ a_v &= \frac{A_v}{w*h} \\ d_v^{(i)} &= \frac{D_v^{(i)}}{\max_{v=0,\ldots,n_c;\, j=0,\ldots,4} D_v^{(j)}} \end{aligned} \tag{2}$$

In Fig. 1 an example of logo contour and its derivative value in each point by means of the above-mentioned procedures are depicted.

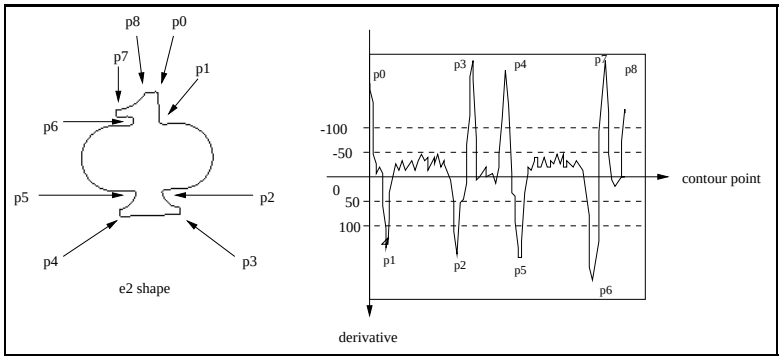

Figure 1: A contour extracted from an instance of logo58 and its derivative trend calculated by mean of SCEFC.

2.3 Tree Pruning and Grafting

The tree pruning and grafting have been used to increase the classification accuracy in machine learning tasks. In our application, they are employed to reduce the valence of tree in case the learning of neural network turns to be slow. High valence of tree can cause saturation of neuron activation and the network training can hardly proceed. On the other hand, very high degree of valence can result in that the RRNNs model has large quantity of frontier units, thus the network dimension increases greatly and much learning time might be wasted.

For logo recognition, small primitives in some logos can be concentrated into a representative virtual contour. When the number [1] of *primary contours*, which are children of a given node, is more than valence predefined, one or more new nodes (grafted node) will be generated automatically in order to reduce the valence of corresponding node. One criterion for whether or not a node will be concentrated is the topological properties of its associated contour. We have chosen directly the dimension of the contour as a criterion. So, the pruning is always followed by grafting operation by a possible concentration process and shifting some nodes as child(ren) of new node(s). The detailed algorithm for the tree pruning and grafting can be coded as follows.

Algorithm 1 Tree Pruning and Grafting Algorithm

```
1   PROCEDURE Tree_Pruning_Grafting(Tree_node node)
2   if Outdegree(node) > VALENCE then
        ▷ Initialize values No_small=No_big=Small[0 . . . Ψ]= Big[0 . . . Ψ]=0
        ▷ Search small and big children for the node
3       for i←0, . . . , Outdegree(node) do
            ▷ Check the dimension of ith associated contour of node
4           if ith node is BIG then
5               Big[No_big++] ← i
6           else
7               Small[No_small++] ← i
```

[1] The quantity of the maximum valence is an experienced value according to the requirement of specific task.

```
8       if No_small > 0 then
9           Create a new Tree_node
10          Search all children of small nodes
11          if Exist children then
12              Create a new Tree_node
13              Calculate the feature vector for newly-created node
14          Delete all small nodes from node
15          Add newly-created node as a child of node
16      No_big ← No_big + 1
17      if No_big > VALENCE then
18          No_new_node ← (No_big - (VALENCE+1)) / (VALENCE-1) + 1
19          for i←0, ..., No_new_node do
20              Create a new Tree_Node
21              Assign consequently at most VALENCE small nodes as its children
    ▷ Recursively call PROCEDURE for the children of node
22  for i←0, ..., Outdegree(node) do
23      node ← ith child of node
24      Tree_Pruning_Grafting(node)
    ▷ End of Algorithm Tree_Pruning_Grafting(node)
```

3 Recursive Neural Networks

In order to fill the gap from symbolic and sub-symbolic processing, instead of processing vector of reals, some researchers have recently investigated the feasibility of learning from examples structured data by using recursive artificial neural networks (RRNNs) [6]. The basic entity that RRNNs are able to adaptively processing is a data structure, mathematically described by a directed ordered acyclic graph (DOAG) with labeled nodes. Each node v in the DOAG is labeled with a variable U_v, which we shall assume to be continuous and multivariate. It is required that the DOAG possess a supersource, i.e., a vertex $s \in V$ such that every vertex in V can be reached by a path starting from s. A simple class which satisfies the above conditions is the class of linear chains, in which nodes v are associated to a serial order on the labels (i.e. v effectively corresponds to a discrete time index). In fact, recurrent neural networks can be thought of as a special case of recursive ANNs that deal with linear chains. RRNNs are a particular class of models operating structural transductions. In a general formulation, a structural transduction is based on the following recursive state space representation:

$$X_v = f(X_{\mathrm{ch}(v)}, U_v) \tag{3}$$

where X_v is a multivariate continuous state variable associated to the generic node v, $\mathrm{ch}(v)$ denotes the ordered set of children of v, and f is a generalized *state transition function*. Computation proceeds following the reverse topological sort of the input data structure U. In order to apply the above equation it is also necessary to specify the state associated to the base step of recursion. This is analogous to the concept of *initial state* X_0 in classic dynamical systems (such as recurrent neural networks) that deal with sequentially ordered data. In the case of DOAGs it is necessary to specify a set of *frontier states*, which are used whenever a child of v is the nil pointer in the input data structure. In the case of structure classification, the learning problem consists of estimating the distribution $P(Y|U)$, where Y is a single categorical variable and U is an input labeled DOAG. In this case output predictions are obtained as a function of the state associated to the supersource s: $Y = g(X_s)$. In the case of RRNNs, both f and g are realized by means of neural units. For example, f and g may be obtained as squashed weighted sums of the vectorial components of their arguments. Weights can be adapted to maximize the likelihood $\prod_{\mathcal{D}} P(Y|U)$, being $\mathcal{D}$ a dataset of data structures whose class is known. The maximum likelihood problem can be easily solved using gradient descent, as a simple extension

of the backpropagation algorithm is available for computing the gradient by error propagation through structure [6].

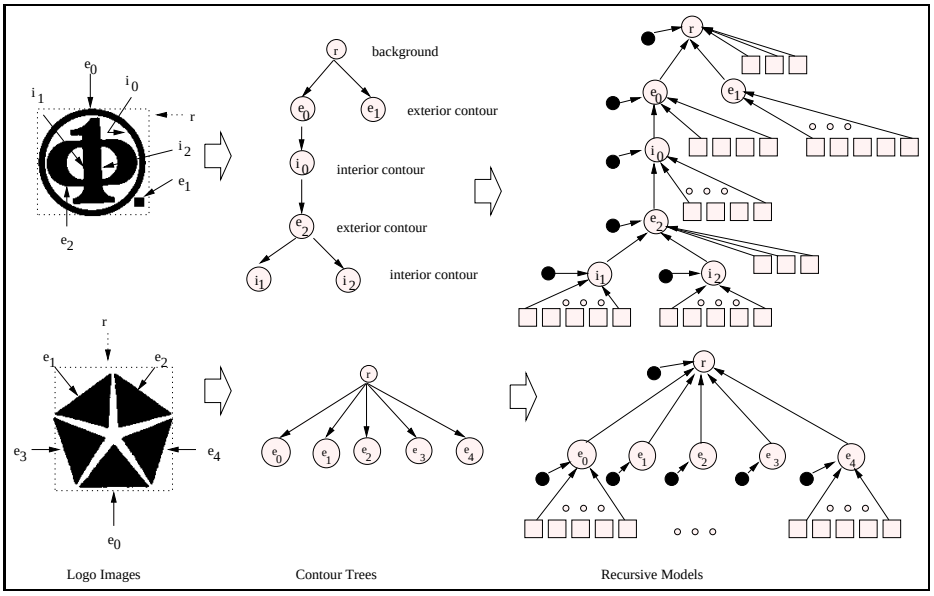

Figure 2: Two logo images are converted to the corresponding tree representations and to their recursive models. Squares indicate null pointer and black circle indicate actual value vectors associated with the corresponding contour.

4 Network Architecture

A MLP-like network architecture with one hidden layer is employed to realize the RRNNs in logo recognition.

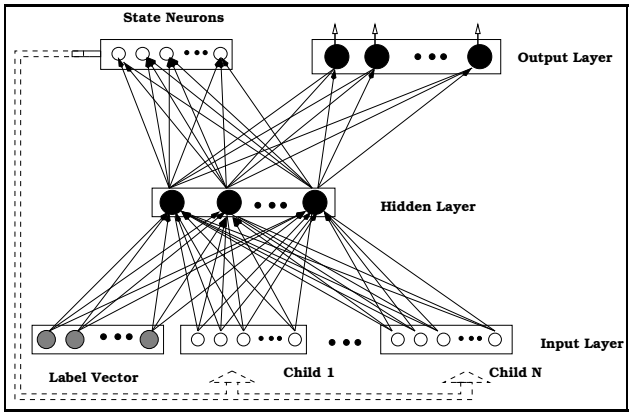

Figure 3: The MLP-like architecture employed for logo recognition

From the figure 3, it is shown that the output neurons are fully connected with the hidden neurons and state neurons are used to memorize the state vectors. In the learning process, the state vector is copied to the input layer according to the topological structure of *secondary contour-trees*.

5 Experimental results

Here we report experiments using BPTS and static MLP on the basis of the same set of images.

5.1 Learning with BPTS

Back-propagation Through Structure can be appropriately applied for logo structure learning. A massive experimentation is carried out in order to know of the optimal architecture on the basis of a variety of class sets. Here we report an experiment based on 40 logo classes picked form a large logo database. By applying *Baird* image defect model [8], there are 200 patterns generated for each class. A total of 8000 structures are created, half of them is used for net learning and the other half for test. Among these noised logos, some graphs derived from contour-tree construction and tree pruning algorithm turn out to be quite different, and continuous labels change too. The network is chosen containing 7 state neurons for the graph node and 25 hidden neurons and 6 neurons in the output layer. In Table 1 shown the network performances for the learning and test set with respect to different epochs.

Epoch	Tss	Threshold	Learn %	Test %	Learn Uncertainty %	Test Uncertainty %
7000	170	0.5	96.95	96.62	0.95	1.42
9400	100	0.5	98.70	98.28	0.38	0.55
10800	80	0.5	98.85	98.45	0.28	0.50
13400	60	0.5	98.98	98.55	0.10	0.25
16400	27	0.5	99.60	98.88	0.08	0.40
20000	18	0.5	99.75	99.15	0.03	0.25

Table 1: Experimental (based on 40 classes with total of 4000 structures), performances reported with respect to different sample epochs for the learning and test set separately.

5.2 Learning with MLP

A MLP-like neural classifier is trained with the same set of images used in the previous section. The image pattern is generated by fitting a frame of $16 * 16$ windows. The network has 256 inputs, 25 hidden neurons and 6 output neurons. The targets are defined in the same way as employed in the BPTS learning network.

Epoch	Tss	Threshold	Learn %	Test %	Learn Uncertainty %	Test Uncertainty %
4000	46	0.5	95.95	91.55	3.82	8.45
5000	23	0.5	99.12	93.70	0.65	4.98
6000	12	0.5	99.78	94.72	0.08	3.88
7000	8	0.5	99.78	95.60	0.08	2.92
8000	5	0.5	99.90	95.88	0	2.75
9000	4	0.5	99.90	96.15	0	2.45
10000	4	0.5	99.90	96.45	0	2.00

Table 2: Experimental (based on 40 classes with total of 4000 patterns), performances reported with respect to different sample epochs for the learning and test set separately.

5.3 Network Paradigms Comparison

By comparing the experimental results obtained using RRNN model and static multilayer perceptron network, it can be observed that the MLP converges faster than RRNN model, but, the generalization performance is unlikely satisfactory, instead, RRNNs can accomplish an expected generalization capability to the test set.

The uncertainty of these models can be also of our interests. With the same threshold, the result tables have shown that the learning by BPTS for RRNN model has higher degree of confidence that MLP-like network.

One explanation about why the RRNNs overperform MLP is that the structure information integrated in the RRNN model assures a better generalization ability in the presence of noises, and it is just because of this that makes the proposed approach robust enough and promising to solve the structured logo recognition problem.

6 Conclusion

In this paper, we have proposed a new approach for the classification of structural logo image by means of recursive neural networks. The experiments show that this adaptive processing paradigm is suitable for recognition of patterns which can be described by DOAGs. This approach could be significantly different from that using traditional artificial neural network to recognize logos and also differ from syntactic pattern recognition approaches based on structure decomposition. The unification between neural processing and structured data representation can integrate the symbolic processing in the neural networks application domain. On the other hand, logo recognition shares many common properties with image recognition. The proposed RRNNs can be integrated into pattern recognition system in an efficient way if only a pattern can be expressed with DOAG.

Acknowledgment: We thank Marco Maggini (DII, Università di Siena) for useful suggestions and his software assistance in the development of BPTS. We also thank the research groups at University of Maryland for making their logo databases available on Internet.

References

[1] D. Doermann, E. Rivlin, and I. Weiss, "Logo recognition using geometric invariants," in *Proceedings of the Second International Conference on Document Analysis and Recognition*, pp. 894–897, 1993.

[2] G. Cortelazzo, G. A. Mian, G. Vezzi, and P. Zamperoni, "Trademark shapes description by string-matching techniques," *Pattern Recognition*, vol. 27, no. 8, pp. 1005–1018, 1994.

[3] K. Hachimura, "Decomposition of hand-printed patterns," *IEEE*, pp. 417–429, 1992.

[4] F. Cesarini, M. Gori, S. Marinai, and G. Soda, "A hybrid system for locating and recognizing low level graphic items," in *Graphics Recognition* (R. Kasturi and K. Tombre, eds.), pp. 135–147, LNCS, Vol. 1072, Springer Verlag, March 1996.

[5] F. Cesarini, E. Francesconi, M. Gori, S. Marinai, J. Sheng, and G. Soda, "A neural based architecture for spot-noisy logo recognition," in *Proceedings of the International Conference on Document Analysis and Recognition*, 1997. to appear.

[6] A. Sperduti and A. Starita, "Supervised neural netwroks for the classification of structures," *IEEE Transaction on Neural Networks*, vol. 8, no. 3, 1996. to appear.

[7] W. McKee and J. Aggarwal, "Computer recognition of partial views of curved objects," *IEEE Trans. Computers*, vol. 26, no. 8, pp. 790 – 800, 1977.

[8] H. S. Baird, "Document image defect models," in *Structured Document Image Analysis (H.S. Baird, H. Bunke and K. Yamamoto eds.*, pp. 546–556, Springer Verlag, 1992.