ENCODING PYRAMIDS BY LABELING RAAM

Stefano Lonardi   Alessandro Sperduti   Antonina Starita
Dipartimento di Informatica
Università di Pisa
Corso Italia 40, 56125 Pisa, Italy
e-mail: perso@di.unipi.it

Abstract

In this paper we present preliminary results on the application of Labeling RAAM for encoding pyramids. The LRAAM is a neural network able to develop reduced representations of labeled directed graphs. The capability of such representations to preserve structural similarities is demonstrated on a pyramid. We suggest to exploit this skill in data compression and/or to discover scale affine autosimilarities.

INTRODUCTION

A model frequently used in image analysis is the *quadtree*, a hierarchical data structure based on the principle of recursive decomposition of space [10]. Different instances of quadtree can be obtained depending on the type of data represented, the decomposition principle, and the resolution (variable or not). In this paper, we discuss how a new type of neural network, the Labeling RAAM, seems specially suited to code it. Specifically, we consider a complete quadtree implementing a *pyramid*, a data structure used to represent a multiresolution version of an image using nonoverlapping 2 by 2 blocks of pixels (see Fig. 1). The aim is twofold: to get a compressed representation and/or to discover scale affine redundancy in the image represented by the pyramid. Firstly, we introduce the LRAAM model. Then, preliminary results obtained on pyramids are presented and discussed.

LABELING RAAM

The *Labeling RAAM (LRAAM)* [12, 15, 16, 17] is an extension of the RAAM model [9] which allows one to encode labeled structures. The general structure of the network for an LRAAM is shown in Figure 2.

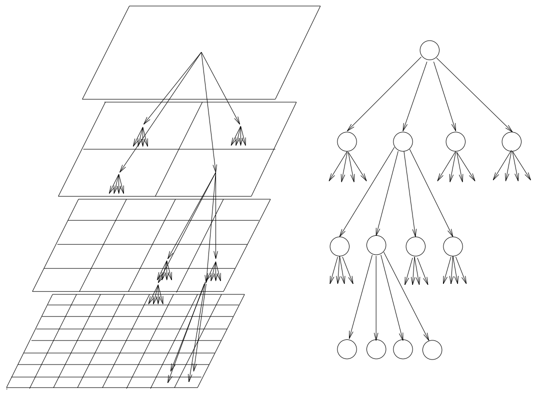


Figure 1: A pyramid implemented by a quadtree.

The network is trained by backpropagation to learn the identity function. The idea is to obtain a compressed representation (hidden layer activation) of a node of a labeled directed graph by allocating a part of the input (output) of the network to represent the label (N_L units) and the rest to represent one or more pointers. This representation is then used as pointer to the node. To allow the recursive use of these compressed representations, the part of the input (output) layer which represents a pointer must be of the same dimension as the hidden layer (N_H units). Thus, a general LRAAM is implemented by a $N_I - N_H - N_I$ feed-forward network, where $N_I = N_L + nN_H$, and n is the number of pointer fields.

Labeled directed graphs can be easily encoded using an LRAAM. Each node of the graph only needs to be represented as a record, with one field for the label and one field for each pointer to a connected node. The pointers only need to be logical pointers, since their actual values will be the patterns of hidden activation of the network. At the beginning of learning, their values are set at random. A graph is represented by a list of these records, and

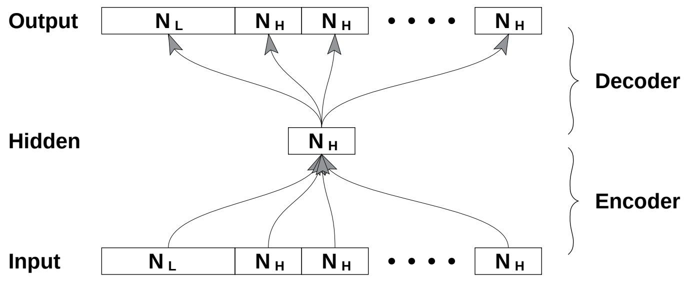


Figure 2: The network for a general Labeling RAAM.

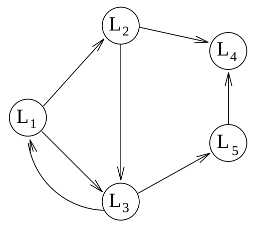


Figure 3: An example of a labeled directed graph.

this list constitutes the initial training set for the LRAAM. During training the representations of the pointers are consistently updated according to the hidden activations. Consequently, the training set is dynamic. For example, the network for the graph shown in figure 3 can be trained as follows:

$$\begin{array}{lclcl}
\text{input} & & \text{hidden} & & \text{output} \\
(L_1\ P_{n_2}(t)\ P_{n_3}(t)) & \rightarrow & P'_{n_1}(t) & \rightarrow & (L''_1(t)\ P''_{n_2}(t)\ P''_{n_3}(t)) \\
(L_2\ P_{n_3}(t)\ P_{n_4}(t)) & \rightarrow & P'_{n_2}(t) & \rightarrow & (L''_2(t)\ P''_{n_3}(t)\ P''_{n_4}(t)) \\
(L_3\ P_{n_1}(t)\ P_{n_5}(t)) & \rightarrow & P'_{n_3}(t) & \rightarrow & (L''_3(t)\ P''_{n_1}(t)\ P''_{n_5}(t)) \\
(L_4\ nil_1(t)\ nil_2(t)) & \rightarrow & P'_{n_4}(t) & \rightarrow & (L''_4(t)\ nil''_1(t)\ nil''_2(t)) \\
(L_5\ P_{n_4}(t)\ nil_3(t)) & \rightarrow & P'_{n_5}(t) & \rightarrow & (L''_5(t)\ P''_{n_4}(t)\ nil''_3(t))
\end{array}$$

where L_i and P_{ni} are the label of and the pointer to the ith node, respectively, and t represents the time, or epoch, of training. At the beginning of training ($t = 1$) the representations for the non-void pointers ($P_{n_i}(1)$) and void pointers ($nil_i(1)$) in the training set are set at random. After each epoch, the representations for the non-void pointers in the training set are updated depending on the hidden activation obtained in the previous epoch for each pattern: $\forall i\ P_{n_i}(t+1) = P'_{n_i}(t)$. The void representations are, on the other hand, copied from the output: $nil_i(t+1) = nil'_i(t)$.

If the backpropagation algorithm converges to zero error, it can be stated that:

$$\begin{array}{cccc}
L_1 = L''_1 & L_2 = L''_2 & L_3 = L''_3 & L_4 = L''_4 \\
L_5 = L''_5 & P_{n_2} = P''_{n_2} & P_{n_3} = P''_{n_3} & P_{n_4} = P''_{n_4} \\
P_{n_5} = P''_{n_5} & nil_1 = nil''_1 & nil_2 = nil''_2 & nil_3 = nil''_3
\end{array}$$

Once the training is complete, the patterns of activation representing pointers can be used to retrieve information. Thus, for example, if the activity of the hidden units of the network is clamped to P_{n1}, the output of the network becomes (L_1,P_{n2},P_{n3}), enabling further retrieval of information by decoding P_{n2} or P_{n3}, and so on. In order to decide whether a pointer is void or not, one bit of the label is allocated for each pointer field to represent the void condition. This convention allows us to avoid a commitment to any predefined representation for the void pointer. Consequently, copying

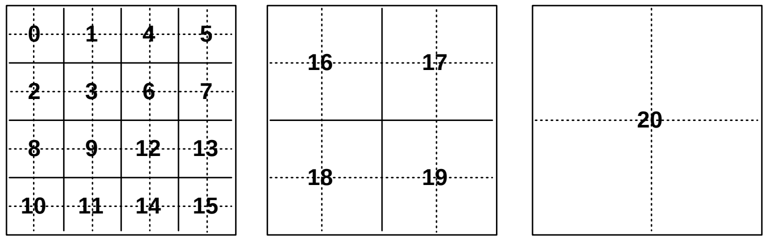


Figure 4: An example of label encoding of a pyramid for an 8×8 image.

the representations for the void pointers from the output of the network to the training set results in a faster training, where multiple representations for the void pointer are developed by the network itself. For more details on this issue the reader is reffered to [15]. Note that multiple labeled directed graphs can be encoded in the same LRAAM.

ENCODING PYRAMIDS

Since a pyramid can be represented as a labeled tree, it can be easily encoded by an LRAAM. The aim in using an LRAAM is to obtain a compact representation of the pyramid where the same pattern at different scales is uniquely represented, i.e., without affine redundancy [1]. The definition of the label for a pattern in the training set of the LRAAM can proceed from the leaf level to the root of the pyramid (see Fig. 4). One drawback of this approach is that the number of patterns in the training set grows exponentially in the dimension of the image. Given a $2^n \times 2^n$ image, if the scale factor is $1/2$, the total number of patterns in the training set is $\sum_{i=0}^{n-1} 4^i$. Thus, it is clear that images of large dimensions are difficult to handle with a single LRAAM. For example, the training set for a 256×256 image would be composed by slightly less than 22,000 patterns. One possible solution to this problem, that we have to verify, is the use of a modular LRAAM, where every module is responsible for the encoding of a given subtree. Another solution is to start learning with a training set representing the lower resolution levels of the pyramid and, after convergence, augmenting the training set to the complete one.

Once the training set has been generated, the LRAAM is trained until it can decode successfully the pyramid. We have observed that, with this stopping criterion, and under the condition of affine redundancy in the image, learning converges rather early. For example, using the encoding scheme shown in figure 5 on an 8×8 version of the Sierpinski triangle, the mean number of epochs employed by the corresponding $12-2-12$ LRAAM was slightly more than 100.

We verified faster learning using the *descending-epsilon* heuristic technique [18]: during the learning phase we maintain a list of the patterns having a decoding error higher than a specified value. The backpropagation is performed only on the patterns of the list: when all patterns are below the threshold, we lower the threshold and resume the backpropagation. The procedure stops when we obtain the perfect decoding.

Analysis

An important property of the LRAAM model is *the capability to develop similar hidden representations for pointers to similar labeled trees.* An example of this capability is given by the hidden activation deviced by the $12-2-12$ LRAAM encoding the Sierpinski triangle. In figure 6 we show, on the left side, the label map obtained by decoding a set of points sampled from the pointer space, and on the right side, the vector fields obtained by transforming the same set of points through the children transformations. A vector field is represented by plotting the sampled points (domain points) and their transformed results (image points) as vectors starting from the domain (dots) and arriving to the image points. Note how the network exploits the same pointer transformation for pointer fields encoding the same set of subtrees. This allows the LRAAM *to decode correctly this image at a resolution higher than the one used in the training set.* This property, however, must be verified on images which are not so regular.

The pointers' dynamics is the subject of figure 7: in this representation the application of the same pointer transformation is repeated until convergence to a fixed point. The gray scale denotes the number of steps to reach the fixed point where darker areas mean a higher number of iterations. Note that the network places the fixed points in the vertices of the $[-1,1]^{N_H}$ space. This property can be explained by probabilistic arguments concerning the stability of the decoding (for a discussion on stability properties of the LRAAM see [16]).

Another example of pyramid (Fig. 8) with the corresponding label map, children vector fields and pointers' dynamics maps are shown in figures 9 and 10.

From our analysis of the decoding of the label and the pointers' dynamics, the fractal approximation developed by the LRAAM seems to have a close relationship with the *hierarchical iterated function system* model [3, 4], where a graph of hierarchical IFS generates the image. In our case, nodes of the graph represent fixed points of the pointers' dynamics. Each node is labeled by the label obtained by decoding the corresponding fixed point. Arcs of the graph represent pointer transformations of the fixed points (see Figure 11).

It must be stressed that the construction of these graphs can be done, in this case, only because any fixed point is actually transformed in another one (or in itself) in just one transformational step. In general, however, this may not be the case. We are currently investigating under which conditions the graph construction scheme holds.

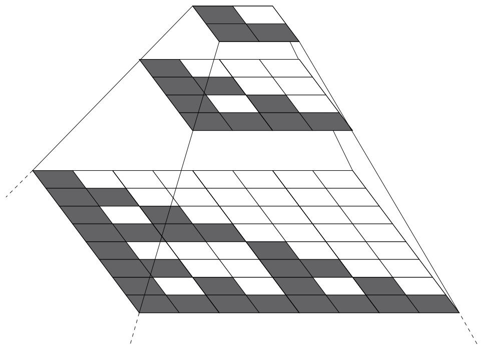
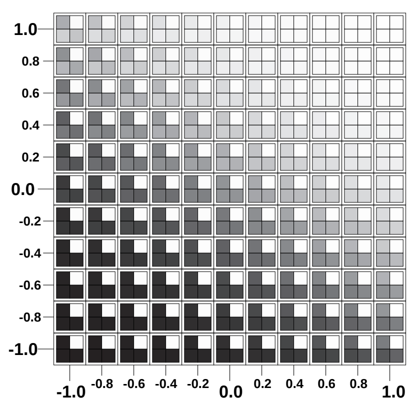


Figure 5: The pyramid coding the Sierpinski triangle.



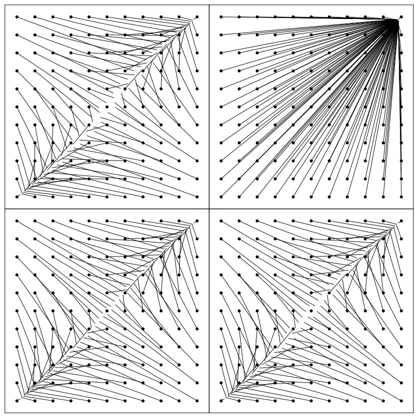


Figure 6: Representations devised by an LRAAM encoding an 8×8 Sierpinski triangle; **left:** label map on the pointer space; **right:** vector fields for the children transformations.

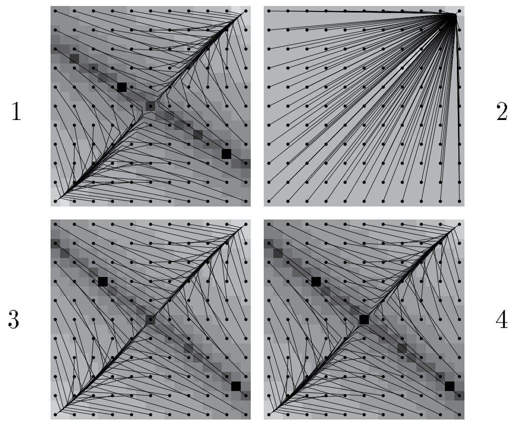


Figure 7: The pointers' dynamics maps for the Sierpinski triangle.

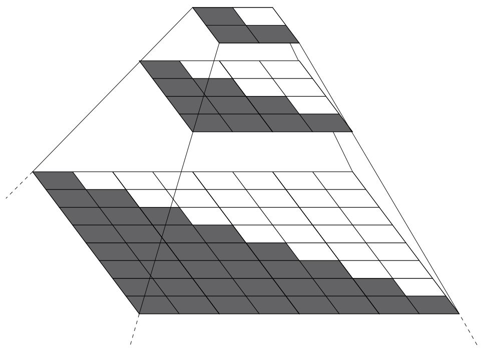


Figure 8: The pyramid coding the simple triangle.

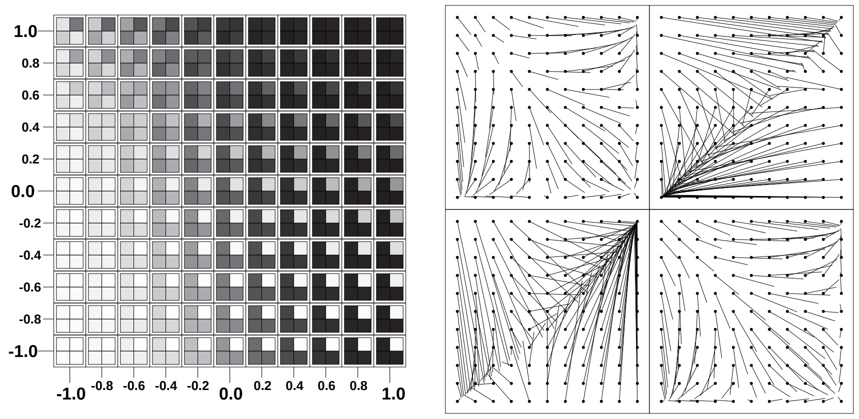


Figure 9: Representations devised by an LRAAM encoding an 8×8 triangle. **left:** label map on the pointer space; **right:** vector fields for the children transformations.

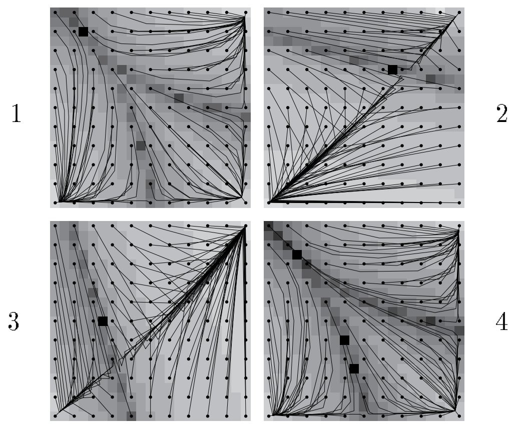


Figure 10: The pointers' dynamics maps for the simple triangle.

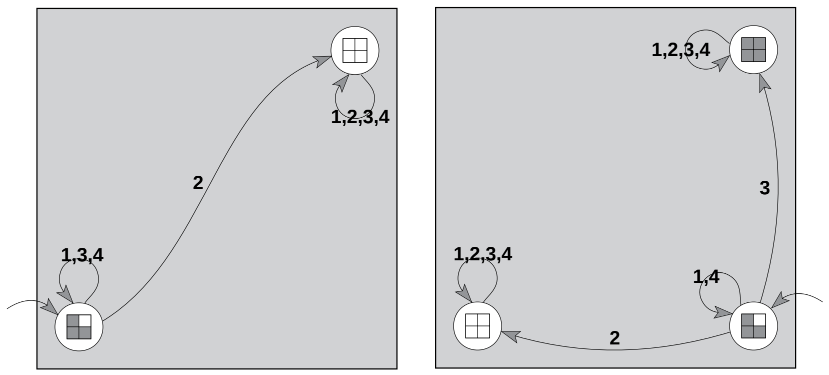


Figure 11: The graphs, derived by the LRAAM, generating the Sierpinski triangle pyramid (left) and the triangle pyramid (right). Each node represents a fixed point in the pointers' dynamics and its label is the label obtained by decoding the fixed point. Arcs represent pointer transformations. Node x is connected to node y by an arc labeled i if $T_i(x) = y$, i.e., if the fixed point associated to x is transformed to y by the pointer transformation $T_i()$.

Applications

There are two possible applications of encoding pyramids by Labeling RAAM:

- *data compression*; a set of pyramids can be described by the set of pointers to the roots plus the decoder part of the LRAAM;
- *affine redundancy discovery*; the likeness among pointers can be used to establish similarities among patterns at different scales, so to device an efficient fractal compression.

Using an LRAAM for data compression requires a careful choice of the number of units in the hidden layer. More units in the hidden layer guarantee a better reconstruction of the image represented by the pyramid, but the compression factor decrease because the number of parameters (weights) grows as a quadratic function. Specifically, given pointers of dimension N_H and label of dimension N_L, we have

$$4N_H^2 + (N_L + 5)N_H + N_L$$

parameters. In fact, the dimension of the hidden layer affects the number of different labels that the LRAAM can encode. Having an insufficient number of hidden units constrains the network to find the minimal approximation with the most used label present in the image at every scale, which is the simplest fractal approximation allowed by the pyramidal representation. We are still working on the evaluation of the best trade-off between quality of the image and data compression. Note that *pruning* techniques like OBD [7], and OBS [6] can be used to reduce the number of weights in the network.

Classification of fractal images by a neural network was explored in the paper [5]. The feed-forward network used in that work, where the pyramid is explicitly represented into the topology of the network, can be considered as a special case of our model. In our case, however, we are not interested in classification, but in discovering scale affine autosimilarity. The ability of the LRAAM to represent similar patterns by similar pointers can be exploited to identify a fractal approximation of the image [1, 2, 11, 8].

CONCLUSIONS

The LRAAM model seems ideal to code a pyramid representing an image with scale affine redundancy. The hidden representations developed by the LRAAM seem to capture redundancies present in the image at different scales. The LRAAM can be used either to compress the pyramid or to discover autosimilarities which can be exploited by a fractal compression method. The compression rate can also be improved by coding the image into an incomplete pyramid using a decomposition principle based on statistical regularities. Incomplete pyramids can be encoded without problems by an LRAAM which can represent every kind of tree.

Besides the present line of research, it is likely that all areas utilizing quadtrees for different applications could use the interesting characteristics of the subsymbolic coding developed by the LRAAM.

Acknowledgments

The authors wish to thank Ahmad Zandi for comments and suggestions.

References

[1] M. F. Barnsley, Fractals Everywhere, Academic Press, 1988.

[2] M. F. Barnsley, L. P. Hurd, Fractal Image Compression, AK Peters Ltd, 1993.

[3] K. Culik, S. Dube, "Rational and Affine Expressions for Image Description", Discrete Applied Mathematics, 41, pp. 85–120, 1993.

[4] K. Culik, S. Dube, "Balancing Order and Chaos in Image Generation", Computer & Graphics, 17(4), pp.465–486, 1993.

[5] B. Freisleben, J. H. Greve, J. Lober, "Recognition of Fractals Images Using a Neural Network", Proceedings of Int. Workshop on Artificial Neural Networks, Lecture notes in Computer Science 686, pp.632–637, Springer Verlag, 1993.

[6] B. Hassibi, D.G. Stork, "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon", Advances in Neural Information Processing Systems, San Mateo: Morgan Kaufmann, Vol. 5, pp.164–171, 1993.

[7] Y. Le Cun, J.S. Denker, S.A. Solla, "Optimal Brain Surgeon", Advances in Neural Information Processing Systems, San Mateo: Morgan Kaufmann, Vol. 2, pp.598–605, 1990.

[8] S. Lonardi, Analisi e sintesi frattale di immagini, Thesis, Computer Science Department, University of Pisa, Italy, 1994.

[9] J. B. Pollack, "Recursive distributed representations", Artificial Intelligence, 46(1-2), pp.77–106, 1990.

[10] H. Samet, "The Quad-Tree and Related Hierarchical Data Structures", ACM Computing Surveys, 16(2), pp.188–260, 1984.

[11] P. Sommaruga, S. Lonardi, "Best Fractal Orthogonal Approximation and Block Coding of Images", submitted to Signal Processing.

[12] A. Sperduti, A. Starita, "An Example of Neural Code: Neural Trees Implemented by LRAAMs", Intl. Conf. on Neural Networks and Genetic Algorithms, Innsbruck 1993, pp.33–39.

[13] A. Sperduti, A. Starita, "Modular Neural Codes Implementing Neural Trees", to appear in 6th Italian Workshop on Parallel Architectures and Neural Networks, 1993.

[14] A. Sperduti, Optimization and Functional Reduced Descriptors in Neural Networks, PhD Thesis, Computer Science Department, University of Pisa, Italy, TD-22/93, 1993.

[15] A. Sperduti, "Labeling RAAM", Technical Report 93-029, International Computer Science Institute, 1993. Accepted for pubblication on *Connection Science*.

[16] A. Sperduti, "On Some Stability Properties of the LRAAM Model", Technical Report 93-031, International Computer Science Institute, 1993.

[17] A. Sperduti, "Encoding of Labeled Graphs by Labeling RAAM", to appear in Advances in Neural Information Processing Systems, San Mateo: Morgan Kaufmann,Vol. 6, 1994.

[18] Y. Yu, R. Simmons, "Descending Epsilon in Backpropagation: a Technique for Better Generalization", Proceedings of IJCNN , San Diego 1990, pp.167–172.