Similarity Learning of Graph-Based Image Representations

Similarity learning for graph-based image representations

Ciro de Mauro, Michelangelo Diligenti
Marco Gori, Marco Maggini
Dipartimento di Ingegneria dell'Informazione - Università di Siena
Via Roma, 56 – 53100 Siena ITALY
{demauro,diligmic,marco,maggini}@dii.unisi.it

Abstract
The criteria used to define the similarity among images for their retrieval in visual databases are usually predefined and based on specific local or global perceptual features. In this paper, we propose a novel approach based on neural networks by which the retrieval criterion is derived on the basis of learning from examples. In particular, the proposed approach is based on a graph-based image representation that denotes the relationships among regions in the image and on recursive neural networks which can process Directed Ordered Acyclic Graphs (DOAGs). The graph-based representation combines structural and sub-symbolic features of the image, while recursive neural networks can discover the optimal representation for searching the image database. Some preliminary experiments on artificial images are reported which clearly indicate that the proposed approach is very promising.

1. Introduction

In the last few years many efforts have been spent to efficient and effective devise engines capable of searching images in large multimedia databases. The established techniques are based on global or local perceptual features of the images. The most used features are color [8], texture [4, 10, 7] and shape [3, 5, 6, 11]. Visual Information Retrieval Systems (VIRS) represent images with fixed-length real vectors containing a set of parameters describing some of these features. Moreover, the retrieval task is usually based on a similarity criterion which is usually predefined by the particular choice of the feature vector. Some systems allow the combination of multiple criteria among a set of predefined ones. In general an user may want to adapt the search to his/her own criteria of similarity among images, and may also want to define different criteria. From this point of view, machine learning algorithms can be the basis for solutions in which the user can define the search criterion by providing a set of examples. In this case the system could also tune its performance during the searches using the feedbacks of the user.

In this paper, we propose an algorithm to learn the concept of similarity from examples. In order to use a rich description of the images, we propose a structured representation which maintains both the perceptual features and the structural relationships among the parts of the image. Each image is splitted into elementary pieces (small regions with common features), and the relationships among these elementary regions are explicitly coded as a graph. In particular, we consider the relationships defined by the mutual spatial position of the regions. The data structures that can be used to describe images in this framework are Directed Ordered Acyclic Graphs. The graphs representing the images can have very different sizes. In order to obtain a fixed-length and low-dimensional vector representation, the DOAGs corresponding to the images can be processed by *Recursive Neural Networks* [2]. In order to face the task of learning the search criteria for visual retrieval, we define a novel learning scheme in which RNNs are trained in order to learn the similarity among images, i.e. in order to map DOAGs representing similar images into near vectors.

This paper is organized as follows. In the next section we describe how the graph-based representation of the images is obtained using a region-growing segmentation technique. In section (3.) we present the Recursive Neural Network model and the extension of the training algorithm to deal with the learning of similarity constraints among pairs of graphs. In section (4.) we report a set of complete experimental results on a retrieval task for artificial images. Finally, in section (5.) we draw some conclusions.

2. Graph-based image representation

We are interested in describing the patterns with a representation which takes into account both their structure and the sub-symbolic information. In particular, an image can be represented by its component regions and their relationships. The image can be splitted into homogeneous regions by an image segmentation based on perceptual features. The regions can be determined by a *region growing algorithm*. This algorithm is initialized by assigning each pixel to a different region. Every region is described by a set of features: average color, color variance, area, perimeter, and bounding box. Then an iterative process is started to merge homogeneous regions. At each step k, each region tries to absorb the surrounding regions by comparing the values of its features to those of the neighbor regions, using a logical predicate P. For example in the experiments described further, P depends on a non-linear score function which compares the features of a pair of regions:

$$P(R_i^k \cup R_j^k) = (min(A(R_i^k), A(R_j^k))^{\alpha} d_c(R_i^k, R_j^k) < \theta_k) \qquad (1)$$

where R_i^k and R_j^k are the two adjacent regions that are evaluated, $A(R)$ is the normalized area of the region R, $d_c(R, S)$ is the distance between the average colors of the regions R and S, α is a fixed parameter and θ_k is a threshold that decreases with k till it reaches a minimum. This function merges two regions on the basis of the similarity of their colors and also penalizes the presence of small regions. Using this function a set of pairs of regions to be merged is selected at each step of the algorithm. Then, the features of the new regions are calculated. This process is iterated till no pair of regions satisfies the predicate P.

Once an image has been segmented, it is subdivided into N regions each described by a real valued vector of features. The structural information related to the relationships among these regions can be represented by a graph. In particular, two connected regions R_1, R_2 are *adjacent* if for each pixel $a \in R_1$ and $b \in R_2$, there exists a path connecting a and b, entirely lying into $R_1 \cup R_2$. In order to represent this type of structural information, the *Region Adjacency Graph* (RAG) can be extracted from the segmented image by (see fig.1):

1. associating a node to each region. It is possible to store the real vector of the features of the region as the label of the node.
2. linking the nodes associated to adjacent regions. The edge added in this step is not oriented.

The generic relationship of adjacency between regions could be expanded into many different (and more specific) relationships: horizontal, vertical, diagonal, included-in, or centered-into adjacency. In this case, we would obtain a graph with labelled edges. Anyway we did not explore this possibility at this moment but we consider this possibility for further developments.

Then, it is possible to transform the RAG into a *Directed Ordered Acyclic Graph* (DOAG). This transformation requires to define a direction and an ordering for the edges in the graph. This can be obtained by applying the following steps:

1. a starting region (root node) is chosen;
2. an ordering rule among the adjacent regions is selected. For example the adjacent regions can be ordered by scanning the region boundary in a clockwise direction starting from the vertical axis;
3. the graph is constructed recursively using a breadth-first visit of the nodes starting from the root node. When a new node a is visited, the edges from

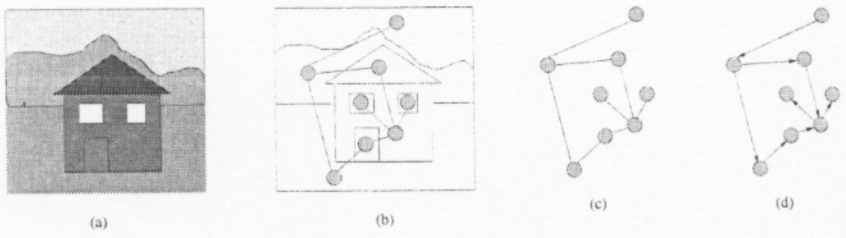

Fig. 1. (a) an artificial image; (b) the image is segmented, using a segmentation algorithm. One node is associated to each connected region and the graph is constructed linking the nodes associated to adjacent regions; (c) the region adjacency graph (RAG); (d) the RAG is transformed into a DOAG.

a to the nodes b_k which have not been already visited are considered. The direction of these edges is chosen to be from a to b_k. Moreover the order of these arcs is defined using the ordering established by the previous rule.

The need to derive a DOAG from the RAG which is directly obtained from the image is motivated by the fact that processing undirected graphs is significantly more difficult. In particular, the framework for adaptive graph processing based on neural networks described in the following section is only defined for DOAGs.

3. Learning similarities with Recursive Neural Networks

We consider a computational model to process data represented by Directed Ordered Acyclic Graphs (DOAGs) which is based on neural units [9, 2]. In particular, this model realizes mappings from graphs to n-dimensional vectors using a recursive *Frontier to Root* computation.
The input graph is processed by a system characterized by a state vector $\mathbf{x} \in \mathbb{R}^n$ whose dynamics is defined by a *state transition function*

$$\mathbf{x}(v) = \phi(\mathbf{x}(q_1^{-1}v), \ldots, \mathbf{x}(q_o^{-1}v), L(v)), \tag{2}$$

where v is a generic node in the graph, $L(v) \in \mathbb{R}^m$ is the label attached to the node v, and $q_i^{-1}v$ specifies the i-th child of the node v. Hence, this function computes the state of the node v given the states $\mathbf{x}(q_i^{-1}v)$ of its descendants and the attribute vector $L(v)$ stored in the node. This definition requires each node to have the same number of children o. When a child is not specified, a given initial state $\mathbf{x}_0$ is used for the missing link (*nil pointer*). In particular, the state of the leaves $\mathbf{x}(v_l)$ of the input graph is computed as:

$$\mathbf{x}(v_l) = \phi(\mathbf{x}_0, \ldots, \mathbf{x}_0, L(v_l)). \tag{3}$$

The nodes are processed following their topological order. This guarantees that the state of each child of the node v is already available when $\mathbf{x}(v)$ is computed using eq. (2). Hence, the computation starts from the frontier nodes since their state is simply obtained from the label and the initial state $\mathbf{x}_0$ as shown in eq. (3). Then, equation (2) is applied recursively to the nodes following their topological order.

The state vector $\mathbf{x}(v)$ *encodes* a representation of some properties of the subgraph rooted at the node v. In fact, if we consider a labelled DOAG $g = l(g_1, \ldots, g_k)$, where l is the label of the root node[1] S and $g_1, \ldots, g_k$ is the ordered list of the subgraphs attached to the root node, the encoding function $enc : DOAG \to \mathbb{R}^n$, which computes the vectorial representation of the input graph, can be defined as $enc(g) = \mathbf{x}(S)$ and recursively computed as $enc(l(g_1, \ldots, g_k)) := \phi(enc(g_1), \ldots, enc(g_k), \mathsf{NIL}^{(o-k)}, l)$, where the $o-k$ empty links (NIL) are encoded by the initial state $\mathbf{x}_0$.

The function $\phi()$ defining the recursive computation can be implemented using a multilayer perceptron (MLP) having $m + o \times n$ inputs and n outputs. In this case the function is parametrized by a vector θ_ϕ collecting the weights of the connections in the MLP. This *encoder network* N_{enc} is applied recursively following equation (2) to compute the state vector for each node. This computational scheme defines a Recursive Neural Network. The complete computation performed on the input graph can be viewed as an *encoding network* whose topology results from the *unfolding* of the recursive neural network on the input graph. An example is shown in figure 2. In this case the recursive neural network is a single-layer perceptron with 2 outputs and it allows a maximum out-degree $o = 3$. When processing the input graph shown on the left, we obtain the encoding network depicted on the right. Note that all the subnetworks in the encoding network are copies of recursive neural network, i.e. they share the same set of weights θ_ϕ.

If the encoder network is a single-layer feedforward network, the dependence of the state of the node v from the states of its children $q_r^{-1}v$ is expressed by the *pointer matrices* $A_r \in \mathbb{R}^{n,n}$, $r = 1, \ldots o$. Similarly, the label vectors attached to the nodes are propagated by a weight matrix $B \in \mathbb{R}^{n,m}$. Hence, the vector of the parameters of the recursive neural network is $\theta_\phi \doteq \{A_1, \ldots, A_o, B\}$, and the state is updated according to the recursive equation

[1] The DOAGs we consider have only one root node, i.e. a node with no ancestors. This does not limit the computational model we introduce since any DOAG can always be transformed into a DOAG which satisfies this property [9].

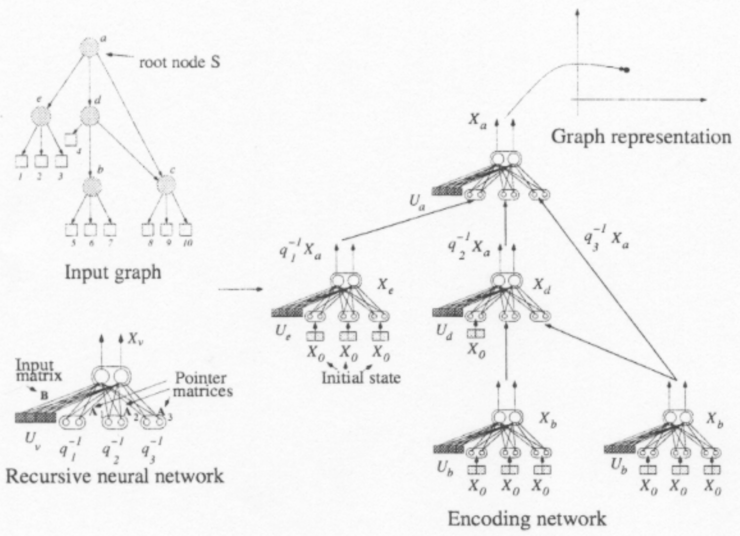

Fig. 2. The input DOAG and the recursive neural network shown on the left yield the encoding network depicted on the right. Note that $o = 3$ (graph out-degree), and that the nil pointers are represented by the initial state $\mathbf{x}_0$. The nodes must be processed according to their *topological sorting*, e.g. e, c, d, b, a.

$$\mathbf{x}(v) = \boldsymbol{\sigma}\left(\sum_{k=1}^{o} A_k \cdot \mathbf{x}(q_k^{-1} v) + B \cdot L(v)\right), \tag{4}$$

where $\boldsymbol{\sigma}$ is a vectorial sigmoidal function.
The parameters of the neural network can be learned in order to develop a mapping which yields vectorial representations for the input graphs satisfying a set of *similarity constraints*. This framework is different with respect to the usual learning in supervised neural networks when a target output is specified for each pattern in the learning set. The aim of *similarity learning* is to tune the mapping between inputs and outputs in order to satisfy topological relationships between pairs of inputs. Given a differentiable distance measure $d(x_1, x_2)$ in the state space, we can assign topological constraints which define the similarity or the dissimilarity between two input graphs. Two graphs g_1, g_2 are *similar* with degree $s \in [0, 1]$ provided that $d(enc(g_1), enc(g_2)) \leq (1 - s)D_s$. Analogously two graphs g_1, g_2 are *dissimilar* with degree $d >= 0$ provided that $d(enc(g_1), enc(g_2)) > (1 + d)D_d$. The reference distances D_s and D_d can be properly chosen by considering the size of the subset where the state of the recursive neural network lies and the number of different clusters defined by

the dissimilarities. For example when using sigmoidal units in the output layer of the MLP, the state vector is inside the hypercube $(0,1)^n$ and D_d can be defined as a fraction of the hypercube diagonal.
The learning set contains a list of pairs for which a similarity or dissimilarity is specified: thus an example is a triple (g_1, g_2, s) or (g_1, g_2, d). The learning process is based on the optimization of a cost function which penalizes those mappings which violate the given similarity constraints. The possible choices for the cost functions in the case of a similarity triple (g_1, g_2, s) and in the case of a dissimilarity (g_1, g_2, d) are respectively

$$\begin{array}{l} E(g_1, g_2, s) = H(d(enc(g_1), enc(g_2)) - (1-s)D_s) \\ E(g_1, g_2, d) = H((1+d)D_d - d(enc(g_1), enc(g_2))), \end{array} \quad (5)$$

where $H(x)$ yields x^2 if $x > 0$ and 0 otherwise. The complete cost function is obtained by summing the contribution of all the triples in the learning set. This function depends on the vector of the MLP weights θ_ϕ and can be optimized using the usual gradient descent scheme. The gradient can be computed by a modified version of the Backpropagation Through Structure (BPTS) algorithm [9] which takes into account that each term in the cost function combines the contribution of a pair of examples.

4. Experimental results

We designed a set of experiments to perform a preliminary evaluation of both the structured representation of images and the similarity learning scheme. In order to provide a systematic and quantitative analysis we used a test set composed of artificial images generated by a set of *attributed plex grammars*. Plex grammars [1] allow us to compose terminal symbols with more general relations than simple concatenation. The images are composed by combining a set of predefined blocks which can be connected together at given attachment points. The productions of the grammar define how the blocks are joined together and the attachment points of the composed object. Further, attributed plex grammars associate a quantitative information represented by a vector of attribute values to each terminal symbol (i.e. length, color, texture parameters, shape parameters, orientation, etc.).
In the experiments we used three different grammars to generate three different classes of images: houses, ships and traffic policemen (see figure 3). By randomly choosing the productions in the grammar and the attribute vector associated to each block, we generated a test set composed of 450 different images (150 images per class).

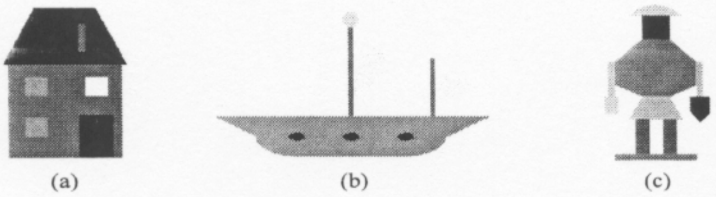

Fig. 3. Example of images generated by the plex grammars: a house (a), a ship (b), and a traffic policeman (c).

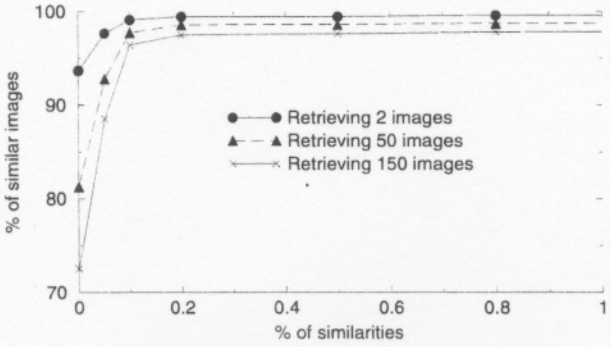

Fig. 4. Average retrieval precision of the proposed algorithm with respect to the percentage of similarities used for training. The results are for different numbers of retrieved images.

In order to build the learning set, we randomly chose pairs of images defining a similarity triple if they were from the same class or a dissimilarity triple if they were from different classes. The number of examples was varied between 0% and 20% of the total number of possible pairs of images. For each set of similarities a recursive neural network with two state units was trained for a maximum number of 600 epochs. Then the retrieval precision was evaluated by choosing a test image and by counting the number of the retrieved images belonging to the same class of the selected image when considering the N nearest vectors. The average retrieval precision was obtained by varying the test image among all the images in the database. Finally the measure was repeated for values of N among $\{2,5,20,50,100,150\}$. When $N = 150$, an exact algorithm would retrieve all the images in the test set belonging to the same class of the

	2	5	20	50	100	150
0	93.7	90.4	85.4	81.3	77.4	72.5
0.2	99.0	98.6	98.6	98.6	98.5	97.5
5	99.2	98.9	98.9	98.9	98.8	98.2
20	99.4	99.3	99.3	99.3	99.2	98.7

Table 1. Average retrieval precision of the algorithm for different percentages of similarities in the learning set (rows) and different numbers of retrieved images (columns).

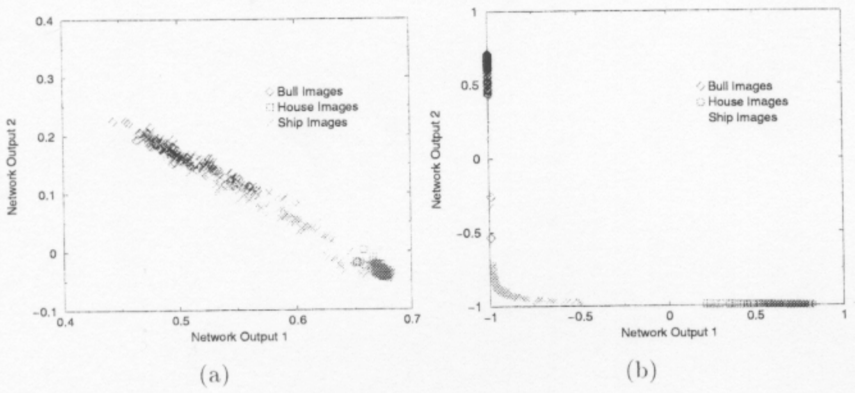

Fig. 5. Plot of the vectors generated by a recursive neural network with 2 output units when processing the graphs in the test set before (a) and after (b) training.

selected image. The results are reported in figure 4. A random (not trained) neural network retrieves between 93.7% ($N = 2$) and 72.5% ($N = 150$) images of the same class (see table 1), but the performance is effectively improved just inserting a small fraction of the total number of possible similarity triples. For example, when using only 0.2% of the pairs, the precision is between 99.0% ($N = 2$) and 97.5% ($N = 150$). A further increase in the number of examples does not improve significantly the performance. The experiments show that the recursive network can be trained to define a mapping able to separate almost optimally the patterns in the dataset, even if few similarities are provided. In figure 5 it is shown that using random weights the patterns of a class are often clustered together with patterns belonging to different classes. After the training, 3 separate clusters can be identified, collecting almost all (and only) patterns of the same class.

5. Conclusions

In this paper, we have proposed a representation of segmented images based on Directed Ordered Acyclic Graphs. DOAGs can represent both the features of each region and the relationships among the regions. The DOAG representation can be further processed with Recurrent Neural Networks in order to obtain a fixed size vector which can be used to define topological relations between the images. The network can be trained to extract the concept of similarity between images from a set of examples. Even when very few similarity constraints are used, the system shows good retrieval and generalization performance. The results reported in this paper are preliminary but they show the promising properties of the proposed scheme. We expect this technique to be an useful framework for developing an image retrieval system whose performance can be tuned using the feedback from an user who can define his/her own criterion in searching the database by providing a set of examples. These examples can consist of only a very small fraction of all possible image pairs.

References

[1] J. Feder, Plex languages, *Information Sciences*, (3), pp. 225–241, 1971.

[2] P. Frasconi, M. Gori, and A. Sperduti, A general framework for adaptive processing of data structures, *IEEE Transactions on Neural Networks*, 9(5), September 1998.

[3] R. C. Gonzalez and P. Wintz, *Digital Image Processing, 2nd Edition*, Addison-Wesley, 1987.

[4] R. M. Haralick, K. Shanmugam, and I. Dinstein, Texture features for image classification, *IEEE Trans. on Sys, Man, and Cyb*, 3(6), 1973.

[5] B. Jhne, *Digital Image Processing - Concepts, Algorithms, and Scientific Applications, 4th Edition*, Springer, 1997.

[6] E. Loupias and N. Sebe, Wavelets-based salient points for image retrieval, *Research Report RR 99.11, RFV-INSA Lyon*, 1999.

[7] J. R. Smith and S. F. Chang, Transformation features for texture classification and discrimination in large image databases, In *Proc. IEEE Int. Conf. on Image Proc.*, 1994.

[8] J. R. Smith and S. F. Chang, Automated binary texture feature sets for image retrieval, In *Proc. ICASSP*, 1996.

[9] A. Sperduti and A. Starita, Supervised Neural Networks for the Classification of Structures, *IEEE Transactions on Neural Networks*, 8(3), 1997.

[10] H. Tamura, S. Mori, and T. Yamawaki, Texture feature corresponding to visual perception, *IEEE Trans. on Sys, Man, and Cyb*, 8(6), 1978.

[11] C. T. Zahn and R. Z. Roskies, Fourier descriptors for plane closed curves, *IEEE Trans. on Computers*, c-21(3), 1972.