Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.5, No.2, 2014
www.iiste.org

Image Content Analysis Using Neural Networks and Genetic Algorithms

Reyadh S. Naoum[1], Mudhafar M. Al-Jarrah[1*], Minan K. Mohammed[2], Marwan R. Shaker[3],
[1]Faculty of Information Technology, Middle East University, Jordan
[2]Emirates College of Technology, United Arab Emirates
[3]Department of Computer Science, Lincoln University, United Kingdom
*E-mail of the corresponding author: maljarrah@meu.edu.jo

Abstract
The analysis of digital images for content discovery is a process of identifying and classifying patterns and sub-images that can lead to recognizing contents of the processed image. The image content analysis system presented in this paper aims to provide the machine with the capability to simulate in some sense, a similar capability in human beings. The developed system consists of three levels. In the low level, image clustering is performed to extract features of the input data and to reduce dimensionality of the feature space. Classification of the scene images are carried out using a single layer neural network, trained through Kohonen's self-organizing algorithm, with conscience function, to produce a set of equi-probable weights vector. The intermediate level consists of two parts. In the first part an image is partitioned into homogeneous regions with respect to the connectivity property between pixels, which is an important concept used in establishing boundaries of objects and component regions in an image. For each component, connected components can be determined by a process of component labeling. In the second part, feature extraction process is performed to capture significant properties of objects present in the image. In the high level; extracted features and relations of each region in the image are matched against the stored object models using the genetic algorithm approach. The implemented system is used in the analysis and recognition of colored images that represent natural scenes.

Keywords: genetic algorithms, neural networks, image segmentation, clustering, image content analysis.

1. Introduction

An architecture for object recognition based on the integration of neural networks and genetic algorithms techniques is presented. The proposed architecture combines the advantages of the two paradigms: field adaptability and the ability to learn from examples, in the low level, is realized by a neural network [1], while the high level analysis is carried out using a genetic algorithm (GA), a process that is distinguished from other search and optimization techniques by the fact that it uses a population of many individuals (initial points), rather than a single one (initial point) to solve a certain problem [2].

Most object recognition strategies work by first detecting some features in the point image, then object models are fitted to these features to test if the object is present. If there is more than one possible object type, each model object will be fitted in turn, and the object model with the best match will be recognized with this search strategy. The recognition time increases linearly with the number of possible object types, after the constant time for initial feature detection [1].

2. System Design Architecture

The system is designed using a 3-tier architecture, in which the image analysis techniques are grouped into three basic areas [3]. These areas are:

I. Low - level processing: The system starts with a clustering process to identify the different region classes that may be present in the image.

II. Intermediate–level processing: Deals with the task of extracting and characterizing components or regions in an image resulting from the low-level process.

III. High-level processing: Involves recognition and interpretation.

These three levels have no definitive boundaries; they do provide a useful framework for categorizing the various processes that are inherent components of an autonomous image analysis system.

3. Clustering Using Artificial Neural Network

The clustering of similar patterns of data, without supervision, is realized through the use of artificial neural networks. Being un-supervised, the process does not use target information in the training set. Note that no mention has been made here of class labels. A key technique used in training nets in this way is concerned with

establishing the most responsive node to any pattern. One way of doing this, is simply to search the net for the largest activity. This, however, displaces the responsibility for this process onto some kind of supervisory mechanism that is not an integral part of the net [4]. Kohonen's learning law [5] is of this type, the neural network architecture in which this law operates is called the Kohonen's layer.
Implementation of the artificial neural network for clustering in the proposed system is based on the design reported by the authors in [6], which presents design structures and details of the implementation.

4. Image Segmentation

4.1 Connected Component labeling

Segmentation of an image into objects or regions and the assigning of unique labels to disjoint regions are fundamental operations in image interpretation. The label assigned to the regions implies assigning the same label to all pixels within spatially connected pixels, with no two disconnected sets having the same label. The process of partitioning the image into structures having more or less homogeneous properties is referred to as connected component labeling [7].
An image is partitioned into homogeneous regions with respect to one or more uniformity properties (such as intensity levels, textural measures, and range of values) that roughly correspond to objects surfaces, or parts of objects in the scene represented by that image. The property used in this work is the connectivity between pixels, which is an important concept used in establishing boundaries of objects and components of regions in an image. To establish connectivity of two pixels, we need to determine if the two pixels are adjacent in some sense, and if their gray levels satisfy a specified measure of similarity (e.g. if they are equal).
Component labeling is used to determine connected components for each component. The process involves assigning labels to pixels in the image such that adjacent pixels are given the same label. There are various definitions of adjacency. In the proposed system, 8-adjancency neighbors are assumed. That is, for each pixel P, its two horizontal neighbors (N3, N7), and two vertical neighbors (N1, N5), plus its four diagonal neighbors (N2, N4, N6, and N8) are considered adjacent to P, as shown in Figure 1. One method that allows labeling of images at pixel rate is based on a fixed-size local window (3 x 3), which includes the previously labeled line. It should be noted that in a labeling algorithm that considers the labels of line (y-1) while labeling line y, labels appearing in line (y-1) but not in line y will never be used again in lines m>y.

Connected component labeling is carried out in three phases as outlined below:
- **Phase one:** Scan the image pixels from left to right and from top to bottom. For every visited pixel, if its color value is equal (within a threshold) to the color value of one of the neighbors N1, N2, N3 and N4 shown in Figure 1, then the label assigned to the neighbor is also assigned to the visited pixel. This process continues until the entire image is scanned.
- **Phase two:** The process described in phase one might assign different labels to neighboring pixels having the same color value due to the topology of the various regions in the image (connected components). To equate these labels, the image is again scanned from left to right and from top to bottom. For each visited pixel, the neighbors N5, N6, N7, and N8 shown in Figure 1 are checked in order. For each neighbor, if its color value is equal (within threshold) to that of the visited pixel then their corresponding labels are flagged as equal in the equivalence labels list. This process is applied to all the pixels of the image.
- **Phase three:** At the end of phase two, we have a labeled image and a list of equivalent labels. In this phase, the scanned image and all pixels having equivalent labels in the equivalent list are merged and assigned one unique label, and the equivalent list is updated. At the end of this phase we have a connected components map of the original image [7].

N2	N3	N4
N1	P	N5
N8	N7	N6

Figure.1: 8- neighbors of a pixel

4.2 Merging Small Regions

Initial image segmentation is of great importance in image understanding, since all subsequent procedures are strongly dependent on it. The output of the segmentation process is improved by using techniques to remove small regions that may be present in the image, because the regions created by the connected components algorithm may not necessarily correspond to a human partition of the image according to similarity in color value. To deal with the problem of small regions and holes, that are generated, a threshold on the minimum size expected in the image is used [8]. This threshold depends on the type of objects expected in the image. The merge can follow one of two options below:

I. Small regions are merged with the largest neighboring region.
II. A small region may be merged with a neighboring region having closest average intensity level.

The first option is chosen arbitrarily, any of the other could equally be used. The first option is more suitable for our system than the second option, because the second option may cause the size of the new region, resulting from the merge, to be smaller than the threshold value set on the size of smallest region accepted by the system.
An adjacency matrix is created for the regions such that element $a_{i,j}$ equals to 1 if region i is adjacent to region j and 0 otherwise. A list storing equivalent labels is also created and initialized such that each region starts with its own label. The list of regions is scanned. If a region i has size which is lower than the threshold, then a region with label j adjacent to i and having the largest size of all neighboring regions is selected from the adjacency matrix. Then, both regions are assigned the label j in the equivalence list and the process is terminated with regard to region i.
The process is repeated for all regions with size below the threshold value, that have not yet been merged with other regions. When region i is merged with region j, the adjacency matrix is updated accordingly such that regions adjacent to either i or j become adjacent to the newly merged region. The size of the new merged region is also updated. The final task is to update the connected component map. This is done by assigning to each set of equivalent labels a new unique label in connected components map and then replaced by its equivalent new label.
Having extracted the connected components in the image using segmentation techniques, the next step is to extract the properties of these components and the relationships among them. These properties and relations make the matching process possible.

5. Feature Selection

If we desire a system to distinguish objects of different types, we must first decide which descriptive parameters of the objects will be measured. The particular parameters that are measured are called the features. Proper selection of the features is important because they will be used to distinguish the object [10]. To describe the individual parts of an object as well as their arrangements, the following unary and binary constraints on the primitives of an object are defined:

5.1 Unary Constraints

The following quantitative constraints are defined on the primitives. For quantitative constraints, the value is specified either as a range between two limits or as one value [10]. The quantitative constraints that are used in our system are:

5.1.1 Area

The area of a particular labeled component n is computed as the number of pixels having the label n. The image is scanned once and appropriate counters of the components are incremented depending on the label of the pixel [3].

5.1.2 Perimeter and boundaries

The perimeter of a region is the length of its boundaries. Perimeter computation is not a simple task because perimeter computation depends on the object's shape. Computing the boundary as the grid-boundary length (sum of vertical and horizontal edges) is equal to the length of the circumscribed rectangle; no matter what shape the component inside the rectangle is [11]. Proffitt and Rosen recommended a correction factor of $\pi/4$ which nullifies the average error but leaves a standard deviation as large as 12% of the error [12]. Using a polygonal of the boundary, they proposed a corner-counting rule to minimize the large error deviation. The corner-counting rule to approximate the perimeter by using the formula:

$$P = \alpha n - \beta k \quad (1)$$

where n is the number of horizontal and vertical links and k is the number of corners. Proffitt and Rosen searched for values of the coefficients α and β, which would nullify the average error and minimize the deviation for all possible slopes of the segment on the average. As a result, they arrived at $\alpha = 0.984$, $\beta = 0.278$. They reported that these values give an average error equal to zero and a minimal standard deviation of about 2.3%. The above formula is used in computing the perimeter of the regions in the image. The number of horizontal and vertical edges are calculated by scanning the image once and incrementing the entries of an edge matrix such that, whenever a vertical or horizontal edge is encountered between region i and region j, the entry edge (i,j) and edge (j,i) are incremented by one. The total number of edges for a region n is calculated by summing the element of the edge matrix in raw n. The number of corners is counted by scanning the component map using a (2x2) overlapping square in both directions. The center of such square defines a possible corner. Figure 2 shows four possible corner configurations. A corner matrix is defined such that element (i, j) denotes the number of corners along their coming boundary. Considering Figure 2, element (i, j) of the matrix will be incremented by 1 if:

$i \neq j$ and $i \neq k$, or
$i \neq j$ and $j \neq l$

On the other hand, element (k, l) is incremented by 1 if:

$k \neq l$ and $i \neq k$ and $i = l$, or
$k \neq l$ and $j \neq l$ and $j = k$

The total corners of a particular region are the summation of all the elements in the raw and column of that region. The number of corners along a common boundary between region i and region j is the sum of two entries: corner (i, j) and corner (j, i). The length of the boundary between two adjacent regions is found by extracting the number of vertical and horizontal edges between them from the edge matrix and the number of corners they share from the corner matrix and applying the perimeter formula [10].
Area and length of perimeter are often used either as components of more complete descriptions or as initial sorting parameters for deciding which figures in a scene should be processed first. Together, they are used to compute the thinness ratio (compactness) [3].

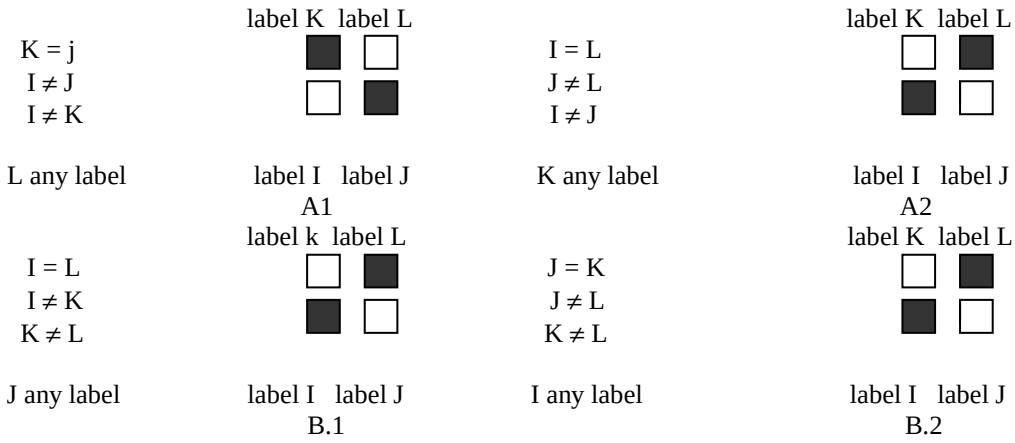


Figure 2: (2x2) squares showing possible corners

5.1.3 Compactness

This parameter measures the radial symmetry of a region. It is measured as the ratio of the area to the perimeter squared [12]. This measure has the greatest possible value of $\pi/4$ for circular region. The value of this measure is normalized to range between 1 for a circular region and 0 for a line. The compactness value is useful in discriminating between classes of simple geometric shapes such as triangles, squares and ellipses. The compactness or thinness ratio (T) defined for a region of area A and perimeter P:

$$T = 4\pi * (A / P^2) \qquad (2)$$

The compactness is dimensionless, hence it depends only on the shape (but not the scale) of the region. For each region (label) in the segmented image, the compactness, together with the color and the position of this region are recorded in a special vector for every region (label), this vector contains the fields: color, position, and compactness. The vectors represent the feature vector of each region [11].

5.2 Binary Constraints

The binary constraints defined on the primitives to describe their structural organization can be grouped either as connection type constraints, or as positional constraints. In this work, only the connection type constraints are needed. The connection type constraint takes into consideration the type of connection between two regions, an example of this is the binary relation adjacent-to, used in this work. Region R1 is adjacent-to region R2 if P1 and P2 are connected and there is no positional restriction on the connection. If there is any type of connectivity (4,8 connectivity) between two regions R1, R2, an adjacent-to relation between the two regions is created such that adjacent-to (R1, R2) is stored in the list of adjacent relations [3]. The features that have been computed for each region of the image are: area, compactness, adjacent-to.

Having extracted the symbolic description of an image, recognition becomes a matter of labeling the primitives of the description with their identifications as objects. The properties of objects, and the relations, which hold between them, imply corresponding properties and relations between the respective image primitives to which they are mapped. These projected properties and relations between the respective image primitives to whom they are mapped constrain the possible labeling of primitives for the image description with object labels.

The approach taken in this work is to match image and object primitives at an intermediate level of description. Object models are described in terms of their constituent and the structural relations among the parts that are preserved by the projection process.

6. Recognition and Interpretation Using Genetic Algorithms

6.1 Representation (Encoding) and Description

Since the genetic algorithms approach is used in the proposed system for recognition and interpretation, we find that the binary representation is the suitable way to represent the feature vector of each region and of the knowledge base. In order to convert the features vector of each region to binary representation, 24 bits will be needed to represent the color value, 6 bits to represent the compactness value, and 2 bits to represent the position of the region that is one of 3 locations. Thus, the length of binary feature vector of each region is (24 + 6 + 2 = 32 bits).

6.2 Genetic Algorithms Components and Tools Requirement

6.2.1 Genetic Algorithms Encoding and Evaluating Function for Regions Recognition.

The main issues in applying GAs to any problem are selecting an appropriate representation and adequate evaluation (fitness) function. The representation of the features is trivial in that a simple binary representation has proved to be very effective [13].

In order to reach the optimal solution to the problem of region recognition, the distance between the region feature vector and the knowledge base feature vector must be minimized. Thus, the suitable fitness function is:

$$\text{Fitness} = \sum_{i=0}^{n} \text{XOR (ith bit of the region vector, ith bit of the object model)}$$

where n is length of the string.

6.2.2. The Crossover Operator Used in the Recognition Process

Simple crossover is used in the recognition phase and the crossover position is randomly selected for each pair of strings.

6.2.3 The Mutation Operator Used in the Recognition Process

Simple mutation is used in the recognition phase and the mutation position is randomly selected for each pair of strings, but it is necessary to check if the crossover position is the same as the mutation position. If the crossover position is the same as the mutation position, the mutation will not be applied.

6.2.4. Genetic Algorithms Procedure

Given a clearly defined problem to be solved and a bit string representation for candidate solutions, GA works as follows:

Let P be a population of N chromosomes (individuals of P). Let P(0) be the initial population, randomly generated, and P(t) the population at time t. The main loop of GA involves the generation of a new population P(t+1) using the existing P(t). This way, the population P(t+1) is created by means of a reproduction operator that gives higher reproduction probability to higher fitted individuals. The overall effect of GAs is to move the population P towards areas of the solution space with higher values and higher computational speed, using GA search directed by the fitness function. This direction is not based on whole chromosome, but on their parts that are strongly related to high values of the fitness function; these parts are called building blocks [13]. The genetic algorithm processes, at each cycle, a number of building blocks proportional to the number of individuals of the population. GA is therefore useful for problems where an optimal solution may be obtained as a composition of a collection of building blocks.

We can state the GA steps as follows:

1. Select an initial population P^k where k =1 of a given size m. This selection is made randomly from the set G^n gene pool. For our system the set $G^n = \{0,1\}$ because binary coding is used.
2. Evaluate each chromosome (string) in population P^k in terms of its fitness function f(x).
3. If a stopping criterion is not met go to step 4 otherwise stop.
4. Generate a new population P_n^k from the given population P^k by some procedure of natural selection. (A procedure called "deterministic sampling" can be used for this purpose).
5. Produce a population of new chromosomes, P^{k+1} , by operating on P_n^k. Operations that are involved in this step attempt to simulate genetic operation observed in biological systems. They include:
   - Simple crossover
   - Simple mutation
6. Replace P_n^k; increase k by 1 and go to step 2.

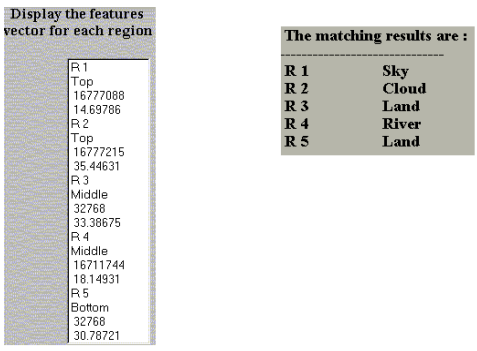
Each iteration of this process is called a generation. The entire set of generations is called a “run”. At the end of a run there are often one highly fit chromosome in the population. Since randomness plays a large role in each run, two runs with different random-number seeds will generally produce different behaviors.

7. Experiment and discussion of results

The steps of the image analysis system described in this paper have been applied to the following image to test its adequacy.

Figure (3.a) shows the image before applying the system, while Figure (3.b) shows the clustered image. The segmented image before merging small regions is shown in Figure (3.c) and the segmented image after merging small regions with size less than 25 pixels are shown in Fig (3.d). It can be seen that the segmented image before and after merging small regions are the same; i.e. the image has no regions to be merged. The unary features of each region are shown in Fig (3.e) and the recognition results shown in Fig (3.f).

(a) (b) (c) (d)



(e) (f)

Figure 5: Image Analysis Steps

8. Conclusion and future work

This paper has presented a content analysis system that is based on a combination of neural networks and genetic algorithms methods. The obtained results show the effectiveness of using the genetic algorithms approach in the recognition and interpretation of sub-images (content) of an image. Further experimental work is needed for the comparison between the neural networks / genetic algorithms approach and the neural networks / semantic networks approach, in terms of recognition effectiveness and computational work.

References

[1] Heideman, G., Lucke, D., & Ritter, H. (1996). A Neural 3-D Object Recognition Architecture Using Optimized Gabor Filters", in Proceedings of 13th International Conference on Pattern Recognition, Vienna, IEEE computer Society Press.

[2] Mitchell, Melanie (1998). An introduction to Genetic Algorithms, MIT Press.

[3] Gonzales, R. C., Woods, R. E. (2008). Digital Image Processing, Third Edition, Prentice-Hall**.**

[4] Chekassky, V., Mulier, F. (1998). Learning From Data, Wiley & Sons.

[5] Kohonen, T. (2001). Self-Organizing Maps. Third extended edition. Springer.

[6] Naoum, Reyadh S., Al-Jarrah, Mudhafar M., Mohammed, Menan K., & Shaker, Marwan R. (2013). A Hybrid Image Content Analysis System Using Semantic and Neural Networks, International Journal of Academic Research, Vol. 5, No. 4, July 2013.

[7] Samet, H. (1984). The Quadtree and Related Hierarchical Data Structures, Computing Surveys, Vol. 16, No. 2, pp. 187-260.

[8] PARK, H. S., RA, J. B. (1999). Efficient Image Segmentation Preserving Semantic Object Shapes, IEICE Trans. Fundamentals, 1Vol. 82-A, No. 6, June 1999.

[9] Doucherty, E. R., Giardina, C. R. (1988). Mathematical Methods for Artificial Intelligence and Autonomous System, Prentice-Hall.

[10] Castleman, K. R. (1979). Digital Image Processing, Prentice- Hall.

[11] Kupa, Z. (1982). More about Area and Perimeters of Quantized Objects, Computer Vision, Graphics and Image Processing, Vol. 22, pp. 268-276.

[12] Proffitt, D., Rosen, D. (1979). Metrication Errors and coding efficiency of chain-encoding schemes for the representation of lines and edges, Computer Graphics and Image Processing, Vol. 4, Issue 10, pp. 318-332.

[13] Vafalo, H. De Jong, K. (1993). Improving a Rule Learning System Using Genetic Algorithms, Machine Learning; A multi-strategy approach, Michalski R. S. and Tecuci G., San Maico, CA: Morgan Kaufmann.