Jussi Pakkanen and Jukka Iivarinen, A Novel Self-Organizing Neural Network for Defect Image Classification. In Proceedings of the International Joint Conference on Neural Networks, pages 2553-2558, Budapest, Hungary, July 25-29 2004.

A Novel Self-Organizing Neural Network for Defect Image Classification

Jussi Pakkanen and Jukka Iivarinen
Helsinki University of Technology
Laboratory of Computer and Information Science
P.O. Box 5400, FIN-02015 HUT, Finland
E-mail: {jussi.pakkanen, jukka.iivarinen}@hut.fi

***Abstract*— In this paper a novel self-organizing neural network called the Evolving Tree is applied to classification of defect images. The Evolving Tree resembels the Self-Organizing Map (SOM) but it has several advantages over the SOM. Experiments present a comparison between a normal SOM, a supervised SOM, and the Evolving Tree algorithm for classification of defect images that are taken from a real web inspection system. The MPEG-7 standard feature descriptors are applied. The results show that the Evolving Tree provides better classification accuracies and reduced computational costs over the normal SOMs.**

I. Introduction

Almost all manufacturing plants today use some kind of automated quality control systems. Examples include plants manufacturing metal sheets and pipes, nonwoven and so on. Our research focuses on defect images taken from a running paper web using an automated inspection system. These images show different kinds of defects. Some of them are harmless cosmetic errors, while other defects, such as large holes, may be very serious. Furthermore, one inspection system may produce up to thousands of defect images in a day, so there's a need for automatic and efficient tools.

We are currently applying a content-based image retrieval system called PicSOM[1] to handle the management and retrieval of these large defect databases [2], [3]. PicSOM uses tree-structured self-organizing maps (TS-SOMs)[4] as its main indexing engine. The use of a tree-structure noticeably speeds up training and searching of SOMs, but the map size must still be given in advance. In this paper we describe a novel self-organizing neural network called the Evolving Tree [5] that could be used to replace the normal SOM (or the TS-SOM) in various data analysis applications. The Evolving Tree is a growing, hierarchical network that resembles the normal SOM. In the following we will first describe the Evolving Tree algorithm in detail, and then conduct some comparative experiments with the normal SOMs.

II. The Evolving Tree

As already stated, we have used the Self-Organising Map (SOM) [6] in our previous work. The SOM is an established data analysis tool, but it has some non-optimal features, such as slowness when using very large maps. In our research we have compared the performance of the SOM with a new neural network system called the Evolving Tree first described in [5].

The Evolving Tree algorithm starts by taking a single node, called the *root node*, and training this node as if it were a single unit SOM. Training continues until the node has been the best matching unit some pre-determined amount of times.

At this point we split the node. This means that we create some chosen amount of new nodes and mark them as the children of the split node. We now have a tree structure with some amount of leaf nodes and one trunk node. All further manipulation operations are only done on the leaf nodes. Once created a trunk node remains totally static, its only task is to maintain connections between other trunk and leaf nodes in the tree.

It should be noted that our method is not the only way to decide when to split nodes. Arbitrarily complex decision rules could be used for this task. These rules could, for example, examine the densities of data and prototype vectors in the immediate vicinity of the BMU. For the purposes of these experiments we have chosen this simple rule for clarity and because it produces quite good results.

1) Calculating the best matching unit and tree distances: Now we come to the interesting parts of the algorithm: how to find the BMU in a tree and how to train the tree structure. To illustrate this process we now assume to have a larger tree structure, which can be seen in Figure 1. Finding the BMU is a top-down process. We start with the root node. Then we examine its children and find the node whose prototype vector is closest to the training vector. If that node is a leaf node, then it is the best matching unit. If not, its childen are examined and the closest one of them is chosen. This is repeated until a leaf node is found. Thus the Evolving Tree's trunk nodes work as a hierarchical search tree for the leaf nodes. This method of finding the BMU is very similar to the one used in TS-SOM.

When the BMU has been determined, it is time to update the leaf nodes towards the prototype vector. Regular SOM uses the following formula to update node m_i towards the training vector $x(t)$ [6]:

$$m_i(t+1) = m_i(t) + h_{ci}(t) \cdot (x(t) - m_i(t)). \qquad (1)$$

Here $h_{ci}(t)$ defines a so-called neighborhood function. A common choice is the gaussian neighborhood

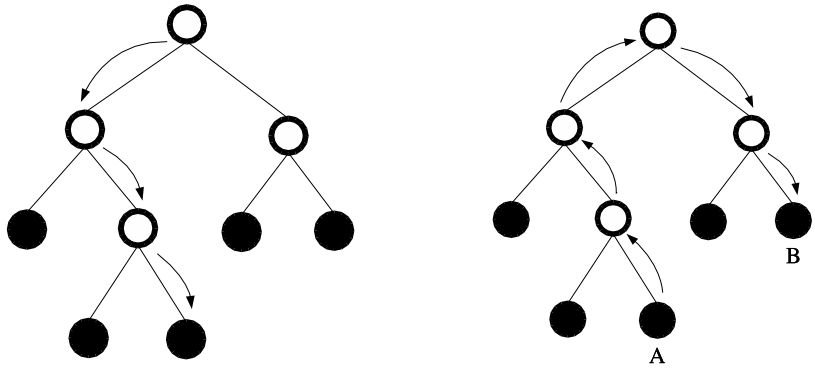

Fig. 1. The left image shows how the BMU is found. The right image demonstrates how the tree distance between two nodes is calculated.

$$h_{ci} = \alpha(t) \cdot \exp\left(\frac{\|r_c - r_i\|^2}{2\sigma^2(t)}\right). \quad (2)$$

The parameters α, which defines the learning rate, and σ, which gives the width of the gaussian kernel work identically in SOM and the Evolving Tree. Calculating the norm $\|r_c - r_i\|$ is not as simple. On the SOM it is the grid distance between the node to be updated and the best matching unit. This is easy to calculate, because SOM is symmetric, static and has regular structure. None of these properties necessarily applies for the Evolving Tree.

There is, however, a simple way to calculate an equivalent distance value. The principle is shown in Figure 1. The basic idea is to calculate how many "hops" must be done to get from the BMU to the desired node. The exact distance is the amount of hops minus one. One is subtracted because we are only interested in distances between leaf nodes. The closest possible leaf nodes have a common parent, so it takes two hops to get from one leaf to another. We want these "closest neighbors" to have a distance of one, which can be achieved by subtracting one from the amount of hops. In the example shown in Figure 1 it takes five hops to get from node *A* to node *B*, so the tree distance between these nodes is four.

It should be noted that our choice of distance is, in fact, sensible. The way the Evolving Tree is formed ensures that there is one, and only one, path between any two given leaf nodes (and also trunk nodes). Therefore the distance function is unique and symmetric. Since the path is unique, it must also be the shortest path. These properties prove that the function is sensible mathematically. It is also suitable intuitively, since we can assume that different branches of the tree grow to different areas of the data space. Therefore distances inside a branch are smaller than between different branches.

2) Adaptation do data: Figure 2 shows how the Evolving Tree fits very strongly to data. In this case the training data consisted of 1000 artificially created two-dimensional data vectors. The figure shows the locations of the leaf nodes at the end of the training. The shape of the original data cloud can be easily seem from the picture. It should also be noted that there are virtually no nodes in those areas of the data space that don't have data vectors.

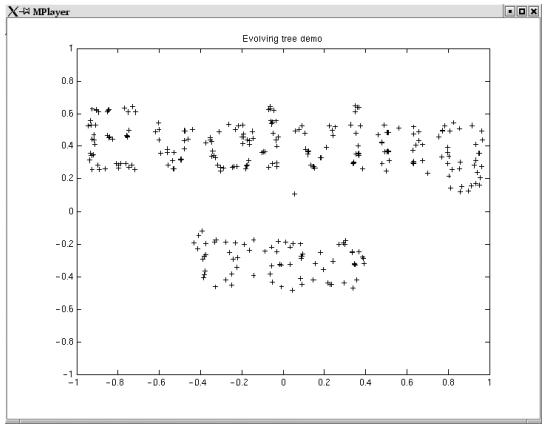

Fig. 2. The Evolving Tree adapting to a 2D data set.

III. Experiments

A comparison between a normal SOM, a supervised SOM, and our Evolving Tree algorithm was conducted in order to see how the Evolving Tree compares with the other two methods. We had a set of classified defect images as our test set and we extracted some standardized feature descriptors from them. Then the defect images were classified using these features and the above mentioned methods. We realize that none of

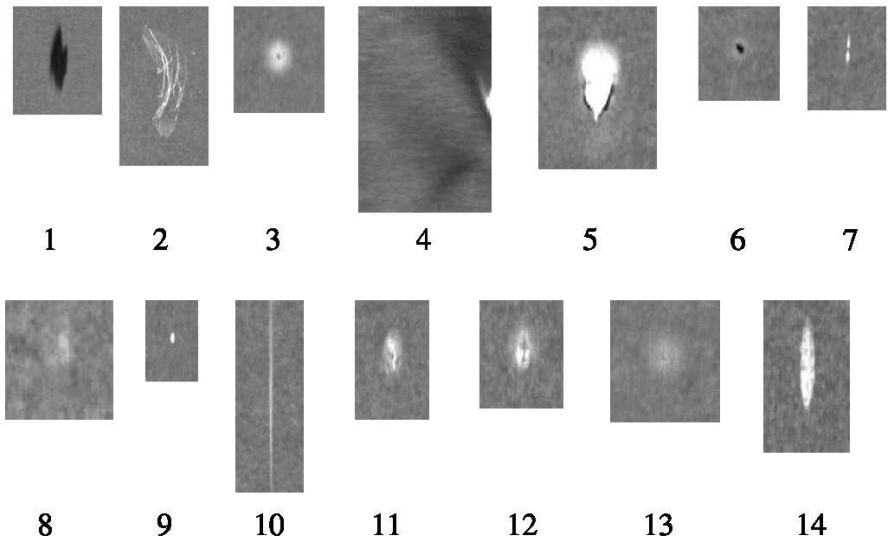

Fig. 3. Example images from each class of the paper defect database.

these methods is a really good classifier, but we think that the comparison is fair since what we want to show is that the Evolving Tree is at least as good as the normal SOMs are.

For SOM we used an well known software package called *SOM Toolbox*[1]. The system has heuristic functions, which automaticlly deduce correct training parameters. Since the Evolving Tree is a new system, choosing the parameters was not a straightforward task. There are two especially crucial parameters. The first is *splitting threshold*, meaning how many times a unit must be the BMU before splitting. The other one is *splitting factor*, which tells how many children a node is split. Experiments indicated that values of 50 and 3, respectively, gave good results. Eight epochs were used to train the trees.

A. Defect Image Database

The experiments were conducted with a set of paper defect images. All the images were obtained from a real, online paper web inspection system. The images have different kinds of defects and their sizes vary according to the size of a defect. Classification of defects is based on the cause and type of a defect, and different classes can therefore contain images that are visually dissimilar in many aspects. The images have 256 gray levels and they are automatically segmented. Thus each image has a gray level image and a binary segmentation mask image that indicates defect areas in the image. The images with the defect segmentation masks were provided by our industrial partner ABB Oy.

There are 1308 defect images in our test set. They are pre-classified into 14 different classes. Most classes had approximately one hundred images, three classes had 63 images and class number 12 had only 27 images. Example images from each class are depicted in Figure 3. It shows how many classes are very similar, especially classes 11 and 12. The images are in fact so hard to classify that even a professional paper maker might not be able to perform a correct classification.

[1] `http://www.cis.hut.fi/projects/somtoolbox/`

B. MPEG-7 Feature Descriptors

The MPEG-7 standard, ISO/IEC 15938, formally named "Multimedia Content Description Interface" [7], [8], [9], provides standardized descriptions of streamed or stored images or video, to be used in searching, identifying, filtering and browsing images or video in various applications. The standard defines several still image descriptors. Based on our earlier results with paper defect images [3], the following three descriptors were clearly ranked as the best ones and thus they are also used in these experiments:

- *Color Structure (CS)* slides a structuring element over the image. The numbers of positions where the element contains each particular color are stored and used as a descriptor.
- *Edge Histogram (EH)* calculates the amount of vertical, horizontal, 45 degree, 135 degree and non-directional edges in 16 sub-images of the picture, resulting in a total of 80 histogram bins.
- *Homogeneous Texture (HT*) filters the image with a bank of orientation and scale tuned filters that are modeled using Gabor functions. The first and second moments of the energy in the frequency domain in the corresponding sub-bands are then used as the components of the texture descriptor.

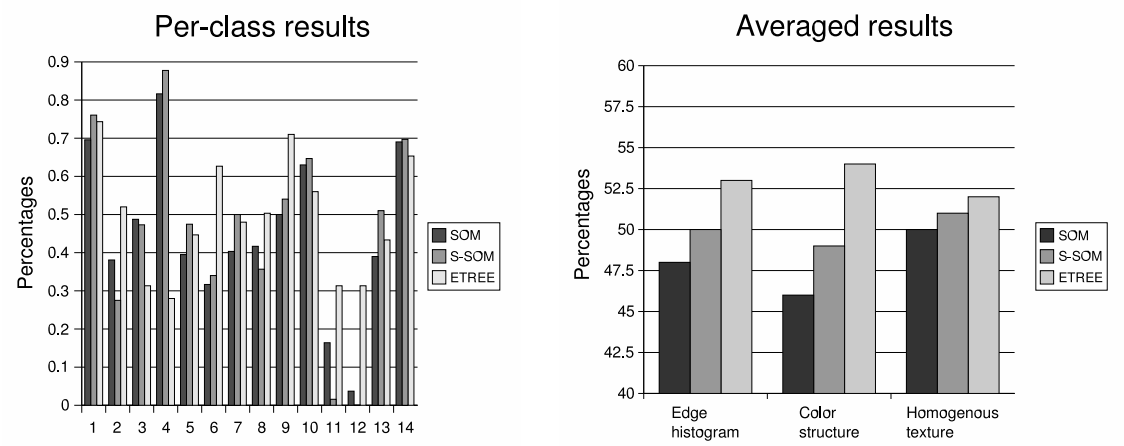

Fig. 4. The left graph shows the results averaged over features while the left one is averaged over classes. Note the difference in scale.

IV. Results

The classification results can be seen in Figure 4. The first figure shows the classification results averaged over the three features. The overall results of the Evolving Tree seem fairly consistent with the SOM percentages. The only significant drop-off is class #4. The Evolving Tree makes up for this poor performance by maintaining a steady classification rate on the very difficult classes 11 and 12. This kind of good performance on the hard portions of the data space is very desirable.

The second graph in Figure 4 gives a better overall view of the classification experiments. It clearly shows that the Evolving Tree perfoms slightly better than the supervised SOM, which is better than the regular SOM.

Another important aspect is the complexity of the data structures used in the training. The system used maps of size 19×9, having a total of 171 nodes. The Evolving Tree's size is not constant. Due to the splitting rule, the amount of leaf nodes depend on many different things, including the ordering of the training vectors. In our experiments the trees had approximately 240 leaf nodes.

This quite large difference in complexity may seem like a lot, but it can be explained by the architectural differences between SOM and Evolving Tree. The trunk nodes of the Evolving Tree maintain a very efficient search tree to the leaf nodes at all times. Searches on the tree can be performed in $O(\log n)$ time compared to $O(n)$ of the regular SOM. This structure makes the Evolving Tree handle complexity very well.

V. Conclusions

The experimental results seem to indicate that the Evolving Tree can be used as a replacement for SOM in some cases. The average classification results are consistently better than is achieved with either normal SOM or supervised SOM. These results, combined with the tree shape of the network indicate that the Evolving Tree could be useful in large, high-dimensional data analysis problems.

Acknowledgment

The authors would like to thank Mr J. Rauhamaa (ABB Oy) and Prof. E. Oja (Helsinki University of Technology) for their help and comments. The financial support of the Technology Development Centre of Finland (TEKES's grant 40120/03) and ABB Oy is gratefully acknowledged.

References

[1] J. Laaksonen, M. Koskela, S. Laakso, and E. Oja, "PicSOM - content-based image retrieval with self-organizing maps," *Pattern Recognition Letters*, vol. 21, no. 13-14, pp. 1199–1207, 2000.

[2] J. Iivarinen, J. Pakkanen, and J. Rauhamaa, "Content-based image retrieval in surface inspection," in *Proceedings of 7th International Conference on Control, Automation, Robotics and Vision*, Singapore, Dec. 3–6 2002, pp. 24–28.

[3] J. Pakkanen, A. Ilvesmäki, and J. Iivarinen, "Defect image classification and retrieval with MPEG-7 descriptors," in *Proceedings of the 13th Scandinavian Conference on Image Analysis*, ser. LNCS 2749, J. Bigun and T. Gustavsson, Eds. Göteborg, Sweden: Springer-Verlag, June 29–July 2 2003, pp. 349–355.

[4] P. Koikkalainen and E. Oja, "Self-organizing hierarchical feature maps," in *Proceedings of 1990 International Joint Conference on Neural Networks*, vol. II, San Diego, CA, 1990, pp. 279–284.

[5] J. Pakkanen, "The evolving tree, a new kind of self-organizing neural network," in *Proceedings of the workshop on Self-Organizing Maps '03*, Kitakyushu, Japan, Sept. 11–14 2003, pp. 311–316.

[6] T. Kohonen, *Self-Organizing Maps*. Berlin: Springer-Verlag, 1995.

[7] B. S. Manjunath, J.-R. Ohm, V. V. Vasudevan, and A. Yamada, "Color and texture descriptors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, June 2001.

[8] MPEG-7, "MPEG-7 multimedia content description interface – part 3 visual," ISO/IEC JTC1/SC29/WG11 W3703, 2001.

[9] ——, "MPEG-7 visual part of the experimentation model (version 9.0)," ISO/IEC JTC1/SC29/WG11 N3914, 2001.