Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Image and Signal Processing (CIISP 2007)

Neural Networks Applied for Cork Tiles Image Classification

A. Georgieva, I. Jordanov, T. Rafik

***Abstract*—In this paper a hybrid global optimization method $GLP_\tau S$ is further investigated and applied for feed-forward neural networks supervised learning. The method is initially tested on several benchmark problems and subsequently employed for pattern recognition problem. The proposed technique is used for training Neural Networks (NN) that have to inspect and classify three types of cork tiles images. During the feature extraction phase, statistical textural characteristics are obtained from the tiles' images and then used for training several different NN architectures. Results from the testing phase are discussed and analysed, showing good generalization abilities of the trained NN. Finally, directions of future work are briefly stated.**

***Index Terms* – Supervised neural networks learning, global optimization, image processing, feature extraction, cork tiles classification.**

I. Introduction

Neural Networks are nowadays widely applied in image processing, as predicted more than a decade ago ([1]). They have been used for preprocessing, feature extraction, segmentation, object detection, and pattern recognition. The focus of this paper is on the latter of these applications.

The use of statistical pattern recognition dates from 1950s and, although it is not one of the main topics of image processing research, it provides an important background – especially in the area of automated visual inspection where decisions about the adequacy of the products have to be made constantly [2]. On the other hand, real industrial applications of texture description and recognition are becoming more and more common [1, 3]: in industrial inspection – quality and process control [4, 5]; in medical diagnosis [6, 7, 8]; in defense [9, 10], etc.

This paper further investigates a hybrid stochastic method, called $GLP_\tau S$ [11], which combines the $LP_\tau O$ technique [12], based on LP_τ low-discrepancy sequences, Genetic Algorithms (GA) and Simplex Local Search. We employ this method for supervised Neural Network (NN) learning of pattern recognition problem – automated visual inspection and classification of cork tiles. The texture images are to be classified into three different classes. Initially, feature extraction is performed and the obtained statistical textural characteristics are used for training NN of several architectures.

Before employing the investigated $GLP_\tau S$ method for a pattern recognition problem, we report and discuss results from tests on global optimization of multimodal mathematical functions and from training NNs, applied for benchmark classification and regression tasks.

In the next section, the $GLP_\tau S$ global optimization technique is briefly discussed and results of optimal NN learning from tests on well-known benchmark problems are shown. In Section III, the problem of cork tiles classification is stated and the employed image processing technique is described. Results from the application of NN for tiles classification are subsequently reported, analysed and discussed. Finally, directions for future work and conclusion are given.

II. $GLP_\tau S$ Optimization Technique – Description and Test Results

We have developed global optimization method called $GLP_\tau S$ that uses evolutionary computation based on genetic operators, low-discrepancy sequences of points and heuristic rules to find regions of attraction when searching for a global minimum of defined objective function. Once a region of attraction with a global minimum is located, the Nelder-Mead Simplex local search is used to further refine the solution. The combination of the three techniques (Genetic Algorithms, $LP_\tau O$ and Simplex Search) provides a powerful hybrid heuristic optimization method, which has been tested on a number of benchmark mathematical functions from 30 to 150 dimensions and has been applied for NN training of several classification and prediction tasks [11].

A. $GLP_\tau S$ Optimization Technique

Low-discrepancy sequences of points are deterministically generated uniformly distributed points. *Uniformity* is an important property of a sequence of points which, simply said, guarantees that the points are evenly distributed in the whole domain. When comparing two uniformly distributed sequences, there are ways to specify which one of them is the better one (Fig.1). In such cases, features as *discrepancy* and *dispersion* are used in order to quantify the uniformity of a sequence.

The advantage of the low-discrepancy sequences is that they avoid so called shadow effect, i.e., when projections of several points on the projective planes are coincident. As it

A. Georgieva is with the School of Computing, University of Portsmouth, Portsmouth, PO13HE, UK (e-mail: antoniya.georgieva@port.ac.uk).

I. Jordanov is with the School of Computing, University of Portsmouth, Portsmouthrda, PO1 3HE, UK (e-mail: ivan.jordanov@port.ac.uk).

T. Rafik is with the School of Product and Engineering Design, University of Wales Institute, Cardiff, CF5 2YB, UK (e-mail: trafik@uwic.ac.uk).

can be seen from Fig.1, the projections from the cubic sequence give four different points on the projection plane, each of them repeated twice; whether the LP_τ sequence gives eight different projection points. Therefore, the low-discrepancy sequence would describe a function behavior in any of the projection planes much better than the cubic one, and this advantage increases with the increase of the dimensionality and the number of points. This feature is especially important when the function in hand is weakly dependent on some of the variables and strongly dependent on the rest of them.

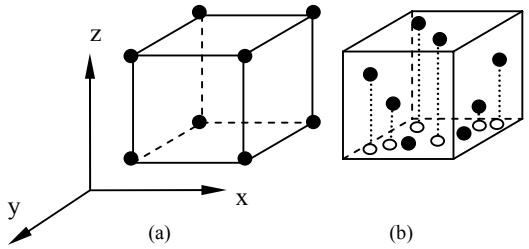

Fig. 1. Two different uniform sequences: (a) Cubic sequence; (b) LP_τ low-discrepancy sequence.

The $LP_\tau O$ technique [12] can be summarized as follows: initially we *seed* Sobol's low-discrepancy points (LP_τ points) in the whole search region, from which we choose the most promising and *seed* new LP_τ points in hyper-cubes with centers in the chosen ones. Then we choose few promising from the new ones and again *seed* in the neighborhood of each one and so on, until a halting condition is satisfied. We combine $LP_\tau O$ and Genetic Algorithms with moderate population size. The GA aims to explore the searched space and improve the initial *seeding* with LP_τ points by applying genetic operators in a few generations. Subsequently, a heuristic-stochastic rule is applied in order to select some of the individuals and to start $LP_\tau O$ search in the neighborhood of each chosen one. Finally, we use a local Simplex Search in order to refine the solution and achieve better accuracy.

Our method combines the effectiveness of GA during the early stages of the search with the advantages of $LP_\tau O$ and the local improvement abilities of Simplex Search. Generally, the technique could be described as follows ([11]):

1. Generate a number *I* of initial LP_τ points.
2. Select *G*, ($G \in I$) points, that correspond to the best function values. Let this be the initial population *p(G)* of the GA.
3. Perform GA until a halting condition is satisfied.
4. From the population *p(G)* of the last GA generation, choose *N* points ($1 < N < G/2$) .
5. Initialize the $LP_\tau O$ search in the neighborhood of each chosen point.
6. After the stopping conditions of the $LP_\tau O$ searches are satisfied, initialize a local Simplex Search in the best point found by all $LP_\tau O$ searches.

For the GA we adopt the selection and mutation operators as given in [13], and for the recombination we use a conventional one-point recombination. The general form of the performed GA is:

a) From the current population *p(G)*, each individual is selected to undergo recombination with probability P_r. If the number of selected individuals is odd, we dispose of the last one selected. All chosen individuals are randomly paired for *mating*. Each pair produces two new individuals by recombination;
b) From the current population *p(G)*, each individual is selected to undergo mutation with probability P_m;
c) Among the *parent G* individuals and those generated by the recombination and mutation, select the best *G* to form the new generation *p(G)*;
d) If the halting condition is not satisfied, repeat from step (a).

Recombination: let $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are the *parents* and *n* is the dimensionality of the problem. Then a random integer *k* between 1 and $n - 1$ is drawn and two *children* $\bar{x}$ and $\bar{y}$ are formed as follows:

$$\begin{aligned} \bar{x} &= (x_1, \ldots, x_k, \bar{x}_{k+1}, \ldots, \bar{x}_n), \\ \bar{y} &= (y_1, \ldots, y_k, \bar{y}_{k+1}, \ldots, \bar{y}_n), \end{aligned} \tag{1}$$

where $\bar{x}_i = x_i + \alpha_i (y_i - x_i)$ and $\bar{y}_i = y_i + \beta_i (x_i - y_i)$ for $i = k + 1, \ldots, n$ where α_i, β_i are randomly drawn numbers in the interval [0, 1]. An illustrative example for the two dimensional case is given in Fig. 2. Two *children* are produced after shifting one of the coordinates of each *parent*.

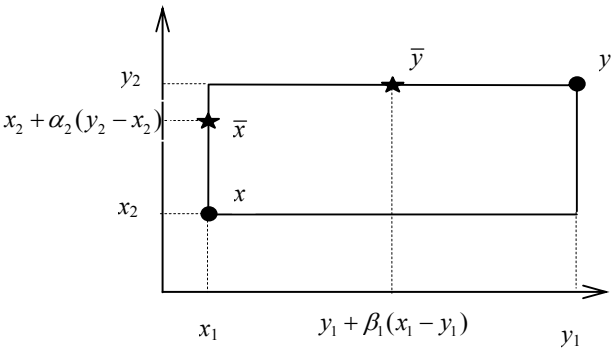

Fig. 2. Recombination in the two dimensional case: *x* and *y* (circles) are the *parents* and $\bar{x}$ and $\bar{y}$ (stars) are the *children*.

Mutation: if an individual is selected for mutation, we draw a random integer $1 \leq k \leq n$. Then, the k-th coordinate of the individual is replaced by a randomly drawn value in the search interval for this coordinate.

Stopping condition: we compute the mean $\bar{f}$ of all function values in $p(G)$. If f^* is the current best function value, found by the previous generations, the stopping condition is considered satisfied if $|\bar{f} - f^*| < 0.001 * |f^*|$. This condition shows that the GA population has converged to a function value that can not be significantly improved any further. If it is not satisfied, the algorithms stops when a predefined number of generations is reached. The adopted population size and maximal number of iterations are smaller than the usually used in conventional GA algorithms. For example, in [11] we adopted population size of 60 for the 30 dimensional problems, opposed to 200 in [13] and 100 in [14]. As it can be expected, at this stage a global minimum can not be found, and the aim is to find regions in which the $LP_\tau O$ will be employed to continue the search.

Combining GA with $LP_\tau O$: to determine the number N of subsequent $LP_\tau O$ searches (see point **4**), we proceed as follows:

Let $p(G)$ is the last generation of individuals, found by the GA run. Firstly, we sort the individuals in a non-descending order and then associate rank r_i to the first half of them by using formula (2):

$$r_i = \frac{f_{\max} - f_i}{f_{\max} - f_{\min}},\ i = 1, ..., G/2. \qquad (2)$$

The rank r_i takes values in the range [0, 1] and is a linear function which decreases with the growth of f_i and vice versa, as f_i decreases, r_i increases. In (2), $f_{\max}$ and $f_{\min}$ are the maximal and minimal values of the population.

The best individual of the population $p(G)$ from the last generation has $r_1 = 1$ and always competes. It is used as a center for a hypercube with side $2R$ in which the $LP_\tau O$ search will start. The parameter R is chosen heuristically after testing and determined as:

$$R = 50/G + \text{int}_{\max} * 0.001. \qquad (3)$$

In (3), the constant $\text{int}_{\max}$ represents the largest of all initial search intervals. Subsequently, the next individual p_i, $i = 2, ..., G/2$ is considered. If any of the Euclidean distances between the individual and the selected ones is greater than 2R (so that there is no overlapping in the $LP_\tau O$ search regions), another $LP_\tau O$ search with probability $P_{LP} * r_i$ will be initiated, where P_{LP} is a user defined probability constant in the interval [0.01, 1].

The Global Optimization Algorithm $LP_\tau O$. Each $LP_\tau O$ search is executed in a hypercube with side $2R$ and a center in the selected point from the last GA generation.

Let n be the dimensionality of the hypercube, then the algorithm can be defined as follows:

1. Generate I^* initial LP_τ points in the hypercube of interest and compute the objective function for them. Arrange the function values in ascending order;
2. Choose $k \leq K$ from the K "best" points for further investigation;
3. In *small* hypercubes, each having as a center one of the chosen in step 2 points, generate new points and calculate the objective function for them;
4. If among the new points no better point is detected or the predefined maximum number of iterations is reached, stop. Otherwise, repeat steps 2-4 with the new points.

Self-adaptation is one of the most important characteristics of the $LP_\tau O$ technique. The method is constructed on the base of heuristic rules (valid for all of the optimized functions) that determine:

- Which k regions (and with what size) to be explored;
- Number of new LP_τ points to be generated in each of the regions (the same algorithm is used in order to choose the initial number of points I^*). The exact number of points is in the range of minimal and maximal user defined values. For 30-dimensional test functions in [11] we chose minimal possible value of $I^*_{\min} = 2^{11}$ and maximal $I^*_{\max} = 2^{13}$, and $K = 20$. For the 100-dimentional test functions, the minimal and maximal values are increased to 2^{13} and 2^{15} correspondingly, and $K = 80$. After the regions of interest are found, it is reasonable to generate more points in those of them, where the function is changing rapidly (a large function change in a small region) and vice versa – less number of points in regions where the function is smooth. To handle this problem, the $LP_\tau O$ technique includes additional heuristic rules, which are given in detail in [12].

All of those rules more or less depend on a parameter R_i^*, which we consider to be the average Euclidean distance between any two neighboring points in the current iteration i. This parameter is a function of the volume of the searched space, the dimensionality n, and the number of points in it. The value of R_i^* decreases with each iteration.

The search always continues in a neighbourhood of the current best point. As new points are generated in the neighbourhood of each core point, the core point itself is still competing in the next pass. Thus, the next best point found in the following pass, would not be worse than the previous one (*descent* property). The algorithm would terminate if the best point has not been improved during the last iteration, or if the predefined number of passes is reached. The method does not guarantee that a global minimum (GM) is found, but guarantees that a GM region of attraction is located (in which a local search can be started). As any other stochastic method we have a probabilistic guarantee that a GM is found ([12]), which means that with the increase of the number of testing points, the probability

of reaching a GM, tends to one.

Combining GA and $LP_\tau O$ with a Simplex Local Search to form $GLP_\tau S$: the Nelder-Mead simplex method for function minimization is a fast local search technique that makes use only of function values and does not require restrictive conditions on the objective function, such as smoothness, Lipschitz continuity, etc. For the parameters (α, β, γ) we adopt the combination of values (1, 0.8, 2). The speed of convergence (measured by the number of function evaluations) depends strongly on these parameters, but mostly, it is influenced by the choice of the initial simplex - its coordinates, form, and size. We chose the initial simplex to have one vertex in the best point found by the $LP_\tau O$ search and another n vertices distanced from it in positive direction along each of its n coordinates with coefficient λ. The parameter λ is chosen to be $\lambda = R$ (given with (3)) and depends on the size of the search region and the GA implementation. The simplex search stops if inequality (4) is satisfied:

$$\frac{1}{n}\sqrt{\sum_{i=1}^{n+1}(f_i-\overline{f})^2} \; < 10^{-9}, \qquad (4)$$

where f_i, $i = 1, \ldots, n+1$ are the function values in the vertices of the current simplex and $\overline{f}$ is the function value in the *centroid* of the simplex. This stopping condition shows how small the simplex has become.

B. Testing $GLP_\tau S$ on multimodal mathematical functions

The $GLP_\tau S$ technique was initially tested on a number of multimodal, multidimensional mathematical functions and its performance was compared with other stochastic methods: Orthogonal Genetic Algorithm with Quantization (OGA/Q, [13]) and Fast Evolutionary Programming (FEP, [14]).

For all tested benchmark functions, the multiple tests (50 runs) of our method converged to mean function values very close to the optimal extrema. The standard deviation also showed low variance of the solution values, indicating a good stability of the results. The number of function evaluations needed to reach an optimal solution (Fig. 3), was reported and compared with other global techniques in [11]. Of course fair comparison based only on the number of function evaluations is not possible, because there is variance in the stopping conditions adopted by the different methods, and also sometimes there are hidden and auxiliary function calculations used for calibration of the methods' parameters. Nevertheless, the overall number of function evaluations can be used as an indicative measure for methods' speed of convergence. Fig. 3 illustrates the average number of function evaluations (from 50 runs), needed by each of the techniques for all 30 and 100-dimensional test functions. The overall comparison with the other two methods demonstrates very competitive, stable and efficient results for our method in terms of both number of function evaluations and mean function values [11].

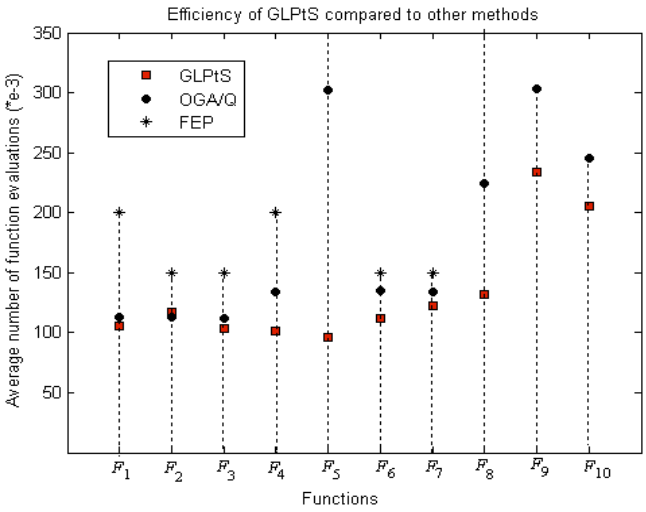

Fig. 3. Test functions – 30 and 100 dimensional, ref. [11].

C. NN Training and Testing on Benchmark Problems

The proposed in [11] algorithm was tested on several well-known benchmark problems with different dimensionalities: some of them classification problems (*Iris* and *Diabetes*); some prediction problems with continuous output (*Servo*); and one regression example. For comparison, a Levenberg-Marquardt Backpropagation algorithm was also performed, using Matlab NN Toolbox.

The investigated NN architectures have static topology in which only the adjacent layers are fully connected. Each unit i in layer l, $0 \le l \le L$, is indexed as i_l, $i_l = 1, 2, \ldots, u_l$, where u_l is the number of units in that layer. The input and target patterns are denoted with $\overline{\boldsymbol{x}}^p$ and $\overline{\boldsymbol{t}}^p$ respectively. For a given pattern p, $p = 1,2,\ldots, P$, the activation of a unit i from the l-th layer, is $a_i^p = g(\overline{\boldsymbol{w}}_i, \overline{\boldsymbol{x}}_{l-1}^p) = \sum_{j=0}^{u_{l-1}} w_{i,j}\, x_j^p$, where $w_{i,j}$ is the connection weight between unit j from the previous layer and unit i from the l-th layer. The bias is $x_0^p = 1$ with corresponding bias weight $w_{i,0}$. The transfer function is a standard sigmoid $f(a_i) = (1+e^{-a_i})^{-1}$, and for a given experiment with P learning samples, the error function is given with $E_p = \frac{1}{2}\sum_{i=1}^{u_l}\left(x_i^p - t_i^p\right)^2$. The network weight vector is an n-dimensional real Euclidean vector $\overline{W}$, whose components are the weights of the network. The $GLP_\tau S$ global optimisation algorithm is employed to minimise the objective function given with (5) and to perform optimal training:

$$F = \frac{1}{P}\sum_{p=1}^{P} E_p(\bar{W}) . \quad (5)$$

Every couple of rows in Table I shows the NN performance for each of the problems, assessing the training with the *Error Function* from equation (5) and evaluating the generalization abilities with the *Mean Test Error*.

TABLE I
OPTIMAL ERRORS FOR THE $GLP_\tau S$ AND BP

Task	Measure	$GLP_\tau S$	BP
Iris	Error Function	0.00097 (0.00056)	0.0091 (0.05)
	Mean Test Error	0.029 (0.073)	0.042 (0.078)
Servo	Error Function	0.0245 (0.005)	0.0474 (0.06)
	Mean Test Error	0.2841 (0.445)	0.4171 (0.5515)
Diabetes	Error Function	0.001 (0.005)	0.0764 (0.07)
	Mean Test Error	0.2619 (0.386)	0.2831 (0.2541)

Table II illustrates the performance of $GLP_\tau S$ for the regression example described in [15]. The test results from 2000 testing samples and 20 independent runs of the experiment show preferable performance of our method.

TABLE II
TEST RESULTS FOR THE $GLP_\tau S$ AND METHODS IN [15]

Method	Average	Max	Min	Std. Dev.
RLS	0.1901	0.2567	0.1553	0.0259
IPRLS	0.1453	0.1674	0.1207	0.0076
TWDRLS	0.1472	0.1711	0.1288	0.0108
$GLP_\tau S$	0.1349	0.1602	0.1184	0.0100

Based on the reported and discussed results, we found the investigated and proposed $GLP_\tau S$ technique to be very competitive and reliable [11]. In this work we investigate further and employ the proposed method for a pattern recognition real world problem – automated visual inspection and classification of cork tiles.

III. CORK TILES CLASSIFICATION PROBLEM

A. Motivation

Natural cork is an environmental-friendly material that completely biodegrades and can be recycled without creating any significant secondary waste. Although wine cork stoppers drive the cork industry [4], floor and wall cork covering still give 20% of the total cork business worldwide [16]. Insulation cork board has been invented in 1892 mainly for use in cold storage areas. It consists of various sizes of cork granules compressed together under a high temperature. It is used for insulation between walls, roofs, floors, around pipes, etc, also as decorative element for walls and ceilings, for soundproofing purposes, etc.

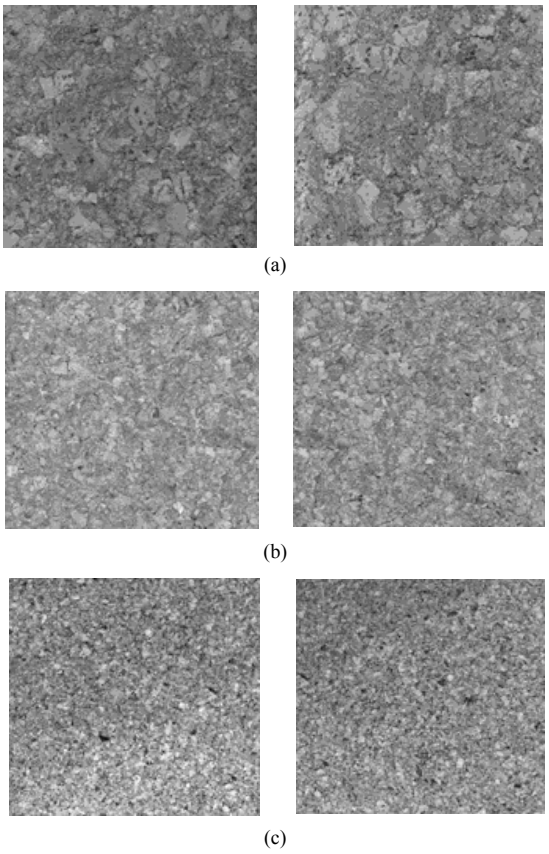

Fig. 4. (a), (b), and (c): Three classes of tiles.

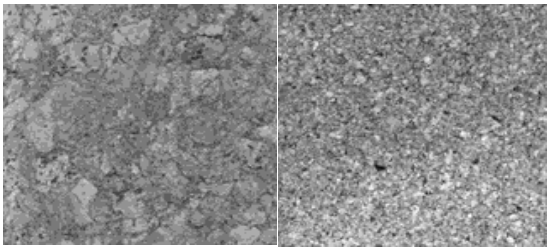

Fig. 5. Tiles of two different classes do not match when put together.

Automated visual inspection has already been applied for quality control of cork stoppers [4, 17]. The image-based inspection systems have high production rates and are based on a line-scan camera and a computer embedded in an industrial sorting machine, capable of acquiring and processing in real-time the surface image of stoppers. Our intention is to use $GLP_\tau S$ global optimization technique for training of NN capable of inspecting and classifying cork

tiles. Initially we consider three types of cork tiles (Fig. 4), which are inspected for their *appearance*. Automated classification of the types of cork board is motivated by the fact that producers would want to have a box containing a number of tiles that belong to the same class of appearance (or quality), since the use of different types (Fig. 5) of tiles would not be aesthetic (the crude tile images are courtesy of Prof. B.Batchelor and Mr. S. Caton from Cardiff University, UK).

B. Texture Features Extraction

Texture is defined as the variation of intensity (or variation of colour) in the image. There are five major categories of features for texture classification: statistical, geometrical, structural, model-based, and signal processing features ([18]). One commonly applied and referenced statistical approach to texture features extraction is the co-occurrence method, introduced by Haralick [19]. The texture features based on these matrixes have been already successfully used for classification of wood, corn, grass, and water ([2]). The texture measures are easily computable and utilize the grey-tone spatial dependencies. All of them are based on the assumption that the texture information of an image is contained in the overall or "average" spatial relationship, which the gray tones in the image have to one another. We used the Matlab Image Processing Toolbox in order to compute the co-occurrence matrix with the default eight grey levels. As commonly accepted by many authors ([18]), we used the nearest neighbour pairs at orientations of 0°, 45°, 90°, and 135°. The Matlab Image Toolbox was used for the computation of the *Contrast, Correlation, Homogeneity*, and *Energy* characteristics in each direction. We considered the mean values of all four directions to be the parameters fed into the NN, and treated the classification to be *rotation invariant* task. Hence, each image was characterized by a four dimensional *feature vector*. Typical examples of the feature vectors for the three classes are given in Table III.

TABLE III
EXAMPLE OF THE FOUR-DIMENSIONAL FEATURE VECTOR FOR EACH CLASS.

Class	Contrast	Correlation	Homogeneity	Energy
1st	0.2381	0.4211	0.8829	0.4547
2nd	0.4795	0.5712	0.7971	0.1814
3rd	0.3034	0.4679	0.8539	0.3467

Fig. 6 and Fig. 7 show how the feature values are distributed for each of the classes and illustrate the mean values and the standard deviations of the four characteristics (16 samples for each class). It is seen from the figures that there is some major overlapping of all features for the first and the third class and the two classes are not linearly separable. For the *Contrast, Homogeneity,* and *Energy* characteristics, the first class contains most of the values of the third class. There is greater difference for the *Correlation* feature, but still the classes are not separable. This makes the task of correctly classifying them very challenging. The second class overlaps with the first one only for the *Correlation* feature (Fig. 7), while it is completely separated from the third class, which makes the correct classification of instances of class two easier. However, future research needs to be done concentrating on the various feature extraction techniques. As mentioned in [1], the search for the optimal set of features that gives the best class separability is a never-ending quest.

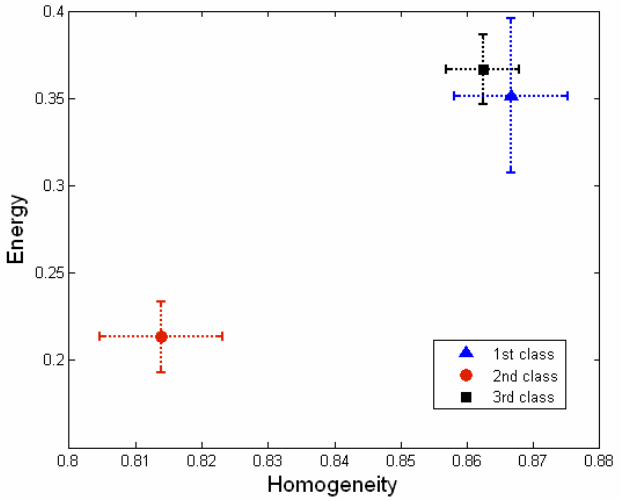

Fig. 6. Distribution of the Contrast and Correlation features for the three classes – mean and standard deviation values.

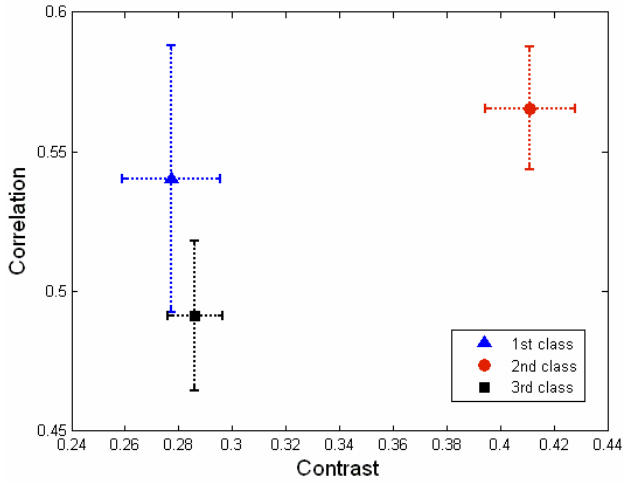

Fig. 7. Distribution of the Homogeneity and Energy features for the three classes – mean and standard deviation values.

C. Experimental Setup and Results

For the Neural Networks learning problem, we used non-overlapping images of three different tile types, adopting size of 150x150 pixels for each sample. The aim was to classify correctly each testing image. From each of the three classes we used 8 samples for training and 8 different ones for testing. Therefore, in total, the NN was trained with 24

samples and tested with another 24. We used the architecture described in Section II (B) where the input layer consists of four neurons. In the hidden layer three different cases were considered with the number of neurons being: 7, 8 and 10 (this implies problem dimension of n = 51, n = 58, and n = 78, respectively). In all cases, the third layer had two neurons and the outputs for the three classes were coded as (0, 0) – for the first, (1, 0) – for the second, and (0, 1) – for the third class.

The investigated NNs were not only with different topology but also two types of transfer functions were employed for the output layer – Heaviside and Sigmoid. In the second case, output values greater than 0.5 were assumed to be 1, and otherwise – 0.

Results from the training and testing of NNs with three different topologies are given in Table IV. For each configuration, the NN was trained and tested 50 independent times and the obtained average results from the simulation are shown in Table IV. The fourth and fifth columns illustrate the classification rate and the mean test error for the cases of *Sigmoidal* transfer function. For the case of *Heaviside* function, mean test error is not applicable, since the error is always 0, 0.5, or 1. These measures evaluate the generalization ability of the corresponding Neural Network. The third column shows the mean train error from equation (5) and the corresponding standard deviation (given in parentheses). It demonstrates and assesses the minimization abilities of the $GLP_{\tau}S$ global optimization technique.

TABLE IV
CORK TILES CLASSIFICTATION WITH TWO DIFFERENT TRANSFER FUNCTIONS IN THE OUTPUT LAYER. TRAINING AND TESTING RESULTS.

Architecture Dimension	Transfer function	Mean Train Error Function (std)	Mean Test Error (std)	Test Success Rate
4-7-2, D = 51	Heav.	0.059 (0.021)	–	74.9%
	Sigm.	0.072 (0.0015)	0.094 (0.076)	72.17%
4-8-2, D = 58	Heav.	0.0633 (0.0117)	–	72.9%
	Sigm.	0.069 (0.0015)	0.092 (0.097)	71.33%
4-10-2, D = 72	Heav.	0.0746 (0.015)	–	72.25%
	Sigm.	0.067 (0.0014)	0.089 (0.067)	71.5%

While natural cork stoppers are manufactured by punching a one-piece cork strip (and may have cracks and insect tunnels), insulation board consists of various sizes of granules compressed together under high temperature [16] and cracks are not likely to appear. Therefore, in [4] the authors are looking mostly for cracks in the cork stoppers, whereas in our case gray density changes and overall appearance are of interest. In [4], the authors classified three different types of stoppers and reported overall success rate between 65% and 79% (depending on different features and on different parts of the stoppers considered – body or tops). Comparing with their data, the results for the three types of cork tiles classification included in Table IV show strongly competitive values.

For all configurations, the test success rate was more than 70%. These are promising results, since this experiment is just in an initial stage of building methodology for automated visual inspection of cork tiles. In all cases the train error function was minimized to values less than 0.072 and the small standard deviation values indicate stable results for all 50 runs. Best minimization results were achieved in the case of 4-7-2 architecture with a *Heaviside* transfer function in the output layer. This was also the case with the best success rate – 74.9%, which gives us an optimal NN configuration with acceptable generalization abilities for the problem at hand.

D. Future Work

Success rate over 70% is a very good rate, considering that only few texture values were used and the data set was very limited. However, we expect that the results could be further improved if the following steps are implemented as future work:

- Higher number of texture features to be examined. We also intend to employ other techniques for feature extraction (e.g., filtering).
- The effect of normalization of the input data to be investigated.
- Different sizes of the processed images to be considered (here we used only 150x150).
- The number of training and testing samples to be increased, as well as to include more types of cork tiles in the classification problem.
- Different cross-validation approaches to be investigated when larger data sets are available.

IV. CONCLUSION

In this work we investigated initial stages of building an automated system for visual control and inspection of cork tiles. Image features were extracted based on co-occurrence matrices. Feed-forward Neural Networks were applied for the classification of texture images (Fig.4). The NN were trained with the $GLP_{\tau}S$ global optimization technique and subsequently their generalization abilities tested with unseen samples. Six different NN architectures were employed for the task and all of them achieved classification success rate over 70%. However, in order to improve these results, a number of system alternations and research steps (listed in section III (D)) could be considered and investigated as a future work.

REFERENCES

[1] M. Egmont-Petersen, D. Ridder, and H. Handels, "Image processing with neural networks – a review", *Pattern recognition*, vol. 35, pp. 2279-2301, 2002.

[2] E.R. Davies, *Machine vision: theory, algorithms, practicalities.* Morgan Kaufmann, 2005.

[3] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, Brooks/Cole Publishing Company, 1999.

[4] A. Costa, H. Pareira, "Decision rules for computer-vision quality classification of wine natural cork stoppers", *American Journal of Enology and Viticulture*, vol. 57, pp. 210-219, 2006.

[5] B.G. Batchelor, P.F. Whelan. (Eds.), *Selected Papers on Industrial Machine Vision Systems*, SPIE MilestoneSeries MS 97, SPIE Optical Engineering Press, 1994.

[6] M. Cenci, C. Nagar, A. Vecchione, "PAPNET assisted primary screening of conventional cervical smears", Anticancer Res., vol. 20, pp. 3887-3889, 2000.

[7] G.M.T. Brake, N. Karssemeijer, and J.H.C.L. Hendriks, "An automated method to discriminate malignant masses from normal tissue in digital mammograms", *Phys. Med. Biol.*, vol. 45, pp. 2843-2857, 2000.

[8] S.K. Lee, C.S. Lo, C.M. Wang et al., "A computer aided design mammography screening system for detection and classification of microcalcifications ", *Int. J. Med. Inform.*, vol. 60, pp 29-57, 2000.

[9] Armed Forces Communications and Electronics Association, DARPA neural network study, AFCEA, Fairfax, 1988.

[10] M.W. Roth, "Survey of neural network technology for automatic target recognition", *IEEE Trans. Neural Networks*, vol. 1, pp. 28-43, 1990.

[11] A. Georgieva and I. Jordanov, "Supervised Neural Network Training with a Hybrid Global Optimization Technique", *Proc. of IEEE WCCI'06 World Congress on Computational Intelligence*, Canada, pp. 6433-6440, 2006.

[12] A. Georgieva and I. Jordanov, "A hybrid method for stochastic global optimization using low-discrepancy sequences of points". *Journal of Global Optimization*, submitted for publication.

[13] Y.W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization". *IEEE Transactions on Evolutionary Computation*, vol. 5, pp. 41-53, 2001.

[14] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster". *IEEE Transactions on Evolutionary Computation*, vol. 3, pp. 82-102, 1999.

[15] C. Leung, A. Tsoi, L.W. Chan, "Two regularizers for recursive least squared algorithms in feedforward multilayered neural networks", *IEEE Trans. Neural Networks*, vol. 12, pp. 1314-1332, 2001.

[16] A WWF Report, *Cork Screwed? Environmental and economic impacts of the cork stoppers market.* WWF/MEDPO, May 2006.

[17] J. Chang, G. Han, J.M. Valverde, N.C. Grisworld at al., "Cork quality classification system using a unified image processing and fuzzy-neural network methodology", *IEEE Trans. Neural Networks*, vol. 8, pp. 964-974, 1997.

[18] T. Randen and J.H. Husøy, "Filtering for texture classification: a comparative study", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 291-310, 1999.

[19] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification", *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, pp. 610-621, 1973.