On the Application of Genetic Probabilistic Neural Networks and Cellular Neural Networks in Precision Agriculture

Babatunde Oluleye[1*], Armstrong Leisa[1], Leng[1] Jinsong , and Diepeveen Dean[2]

[1] School of Computer and Security Science, Edith Cowan University, WA, Australia.
[*]*Corresponding author email: hezecomp {at} yahoo.com*

[2]Department of Agriculture and Food, WA, Australia.

ABSTRACT— ***This article details the effect of Gaussian smoothing parameter (spread) on the performance of Probabilistic Neural Networks (PNN). Two (2) different Genetic Algorithms (GAs) were used to optimize the PNN spread in order to avoid under and over fitting. In this work there is a novel combination of Cellular Neural Networks (CNN), Probabilistic Neural Networks (PNN) and GA to address the present challenges on automatic identification of plant species. Such problems include misclassification species of plants that are similar in shapes and image segmentation speed. In this work, GA was used in both feature selection and PNN parameter optimization. The GA developed herein improved the performance of the PNN. This work serves as a framework for building image classification or pattern recognition system.***

Keywords— Probabilistic Neural Networks, Cellular Neural Networks, Genetic Algorithm, Feature Extraction

1. INTRODUCTION

In agricultural domain, the issue of precision agriculture is becoming more popular since it is economical in terms of time and money and also enforces environmental protection [24]. According to [5], there are at least 250,000 to 270,000 plant species around the world. The traditional recognition of these plant species is carried out by manual matching of the plants features, relating to components of the plant, such as leaves, flowers, and bark [16]. The manual approach to plant classification is tedious and tiring. As a result of this, attempts to automate this process have been made using features of plants extracted from images as input parameters to various classifier systems [9] & [17]. In this paper, a technique to argument already existing techniques of plant identification system is described. The main contribution of this paper is to increase the classification speed and accuracy of the existing systems by incorporating Cellular Neural Networks for image segmentation, genetically optimized PNN and use image descriptors with strong discriminating capability. In summary, this paper employ Genetic Algorithm (GA) to optimize PNN parameter and then combination of this classification model is fed with Zernike Moments and Fourier Descriptors.

2. CELLULAR NEURAL NETWORKS

Cellular Neural Networks (CNN) are variants of ANNs having neighbourhood communication as the main distinguishing feature [7]. CNN (an electrical circuit shown in Figure 1) was invented in 1988 by Chua and his graduate student Yang at the Department of Electrical Engineering and Computer Sciences, university of California, Berkeley [6]. A standard CNN topological structure is made up of an $M \times N$ or rectangular array of cells $C(i, j)$ (or dynamic components) with Cartesian coordinates $(i, j), i = 1(1)M,\ j = 1(1)N$ as shown in [13] and [1] (see Figure 3). CNN is a hybrid model, sharing features from both Cellular Automata and Artificial Neural Networks [13]. The circuit structure and element values of all cells of a CNN are homogenous. Equation 2.0 governing the behaviour of a CNN cell circuit is a dynamical system (or Ordinary Differential Equation (ODE)) derived from evolution laws and circuit theory as shown in Figures 1 and 2. The output of the circuit (also called the amplifier), is given in Equation 2.1

$$\frac{dx_{ij}(t)}{dt} = -x_{ij}(t) + \sum_{(k,l)\in N(i,j)} A(i, j : k, l) \cdot y_{kl}(t) + \sum_{(k,l)\in N(i,j)} B(i, j : k, l) \cdot u_{kl}(t) + z(i, j : k, l) \quad \text{.....(2.0)}$$

where

$x_{ij} \Rightarrow$ the internal state of cell (i,j)

$y_{kl} \Rightarrow$ the output of cell (k,l)

$u_{kl} \Rightarrow$ the input to cell (k,l)

$z_{ij} \Rightarrow$ the threshold of cell (i,j)

$C(i,j) \Rightarrow$ the set of cells in the CNN

$S_r(i,j) \Rightarrow$ the set of cells in the sphere of influence of cell (i,j)

$A(i,j;\ k,l)$ and $B(i,j;\ k,l)$ are functional, which are the **feedback** and **input synaptic** operators respectively

The expression for the output $y_{ij}(t)$ is

$$y_{ij}(t) = f(x_{ij}(t)) = \tfrac{1}{2}(|x_{ij}(t)+1| - |x_{ij}(t)-1|) \qquad (2.1)$$

which is similar to the nonlinear function. Other functions of x are also possible since this is the activation function [7]. It is note-worthy,by the reason of voltages measured across the circuit, that the constraint $|x_{ij}(0)| \le 1, 1 \le i \le M;\ 1 \le j \le N$, *and* $|u_{ij}(0)| \le 1, \quad 1 \le i \le M;\ 1 \le j \le N$ are fundamentally imposed on the internal state of neuron (i, j) and input of neuron (i, j).

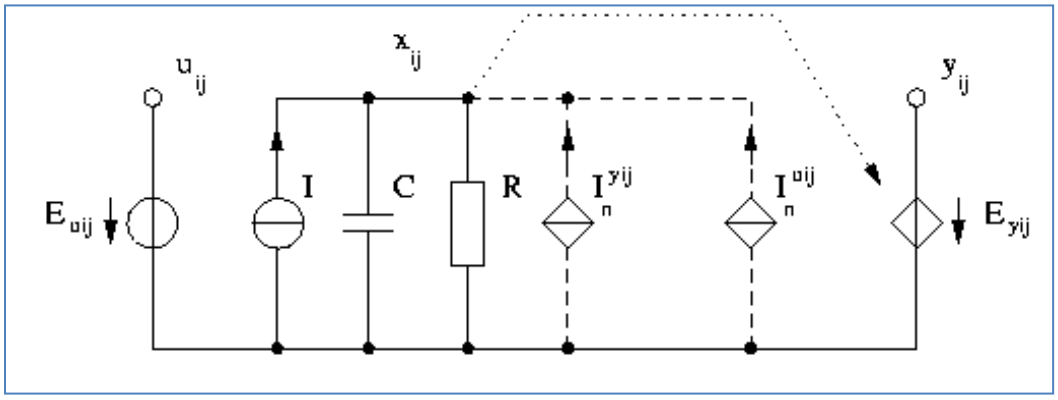

Figure 1: Circuit Representing Cellular Neural Networks ([1,6,7,13])

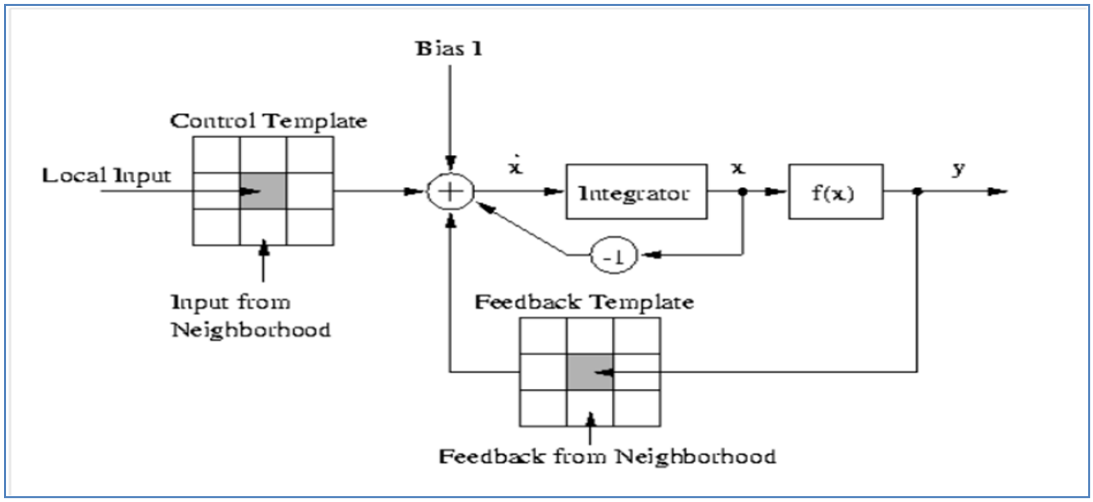

Figure 2: The Block Diagram of a CNN Cell ([1,6,7,13])

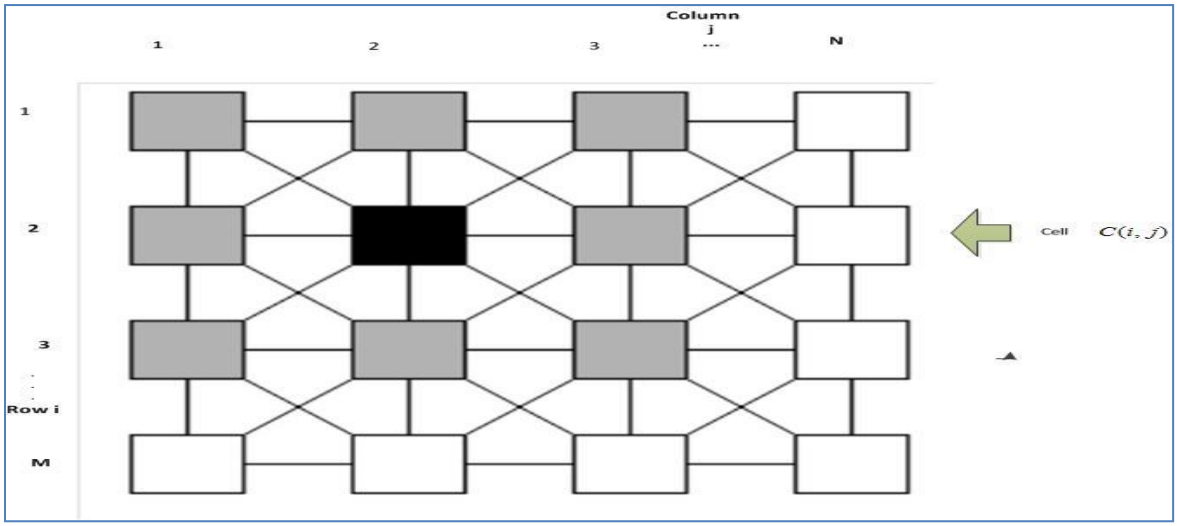

Figure 3: CNN square grid ([1,6,7,13])

The original equation developed by [7] is given as

$$C\frac{dx_{ij}(t)}{dt} = -\frac{1}{R}x_{ij}(t) + \sum_{(k,l)\in N_r(i,j)} A(i,j:k,l)\cdot y_{kl}(t) + \sum_{(k,l)\in N_r(i,j)} B(i,j:k,l)\cdot u_{kl}(t) + I(i,j:k,l) \qquad (2.2)$$

where

$I \Rightarrow$ Independent voltage source, $C \Rightarrow$ Linear Capacitor, $R_x \Rightarrow$ Linear Resistor. It was later adapted to general ordinary differential equations (ODE) in equation **(2.0).**

3. THE FLAVIA DATASET

The source of images of leaves used in this study is images of leaves found in the Flavia dataset which is publicly available [25]. The Flavia dataset is a constrained set of leaf images taken against a white background and without any stem present. The species in the dataset have a varying number of instances as shown [2]. The dataset has 1907 images of 32 species of plants. For this study, the dataset was divided into two disjoint sets, each of which contains 1587 images and 320 images for both training and test set respectively.

4. FEATURES GENERATED FROM THE FLAVIA DATASET

4.1 Image Pre-Processing

The images found in [25] are first pre-processed. The original colored images are converted to grayscale images using the formular in Equation 4.1. One output of this conversion is shown in Figure 4.

$$f_1 = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.114\mathrm{B} \qquad (4.1)$$

4.2 Image segmentation using CNN

Next to image pre-processing, we employ CNN edge detection templates in Equation 4.2 These templates are the matrix coefficient of the systems of equations in 2.0 & 2.1. The outputs of the segmentation stage are finally passed on to feature extraction modules using FDs and ZM.

$$A=\begin{pmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{pmatrix},\quad B=\begin{pmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{pmatrix},\quad I=-1 \qquad (4.2)$$

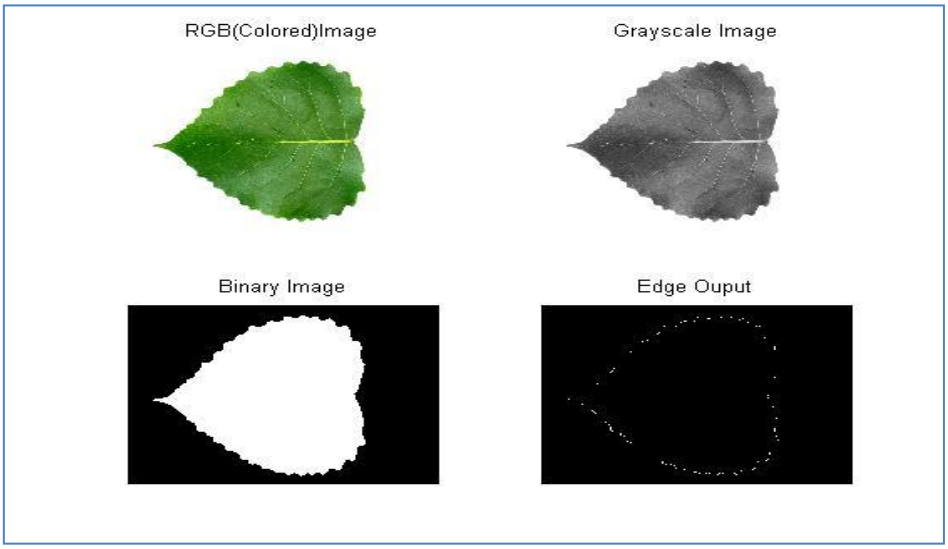

Figure 4: Image pre-processing and segmentation using CNN

4.3 Feature extraction: Fourier Descriptors and Zernike Moments

Fourier descriptors (FD) methods have been traditionally used for shape recognition and are part of general methods used in encoding various shape signatures [4, 10, 23]. The Fourier Transform (FT) and its inverse are described in [14, 12] respectively by formulas in Equations 4.3 and 4.4

$$F(\alpha_1,\alpha_2)=\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty} f(x,y)e^{-i\alpha_1 x}e^{-i\alpha_2 y} \qquad (4.3)$$

Where α_1 *and* α_2 are frequency variables of the image pixels representing the periods of F. The original image of the plant's leaf can re-constructed by using the inverse FT given as Equation 4.4

$$f(x,y)=\frac{1}{4\pi^2}\int_{\alpha_1=-\pi}^{\pi}\int_{\alpha_2=-\pi}^{\pi} F(\alpha_1,\alpha_2)e^{i\alpha_1 x}e^{i\alpha_2 y}\,dxdy \qquad (4.4)$$

It's assumed that a 2D image is given or generated from 3D image through appropriate color-to-grayscale conversion methods such as found in Equation 4.1. The 2D image generated can then be represented as $f(x_i,y_i),\ \ i=1(1)M,\ \ j=1(1)N,\ \ where,\ \ M,N\in Z^+$. A grayscale image is produced from using the formular in Equation 4.1. The edge output from which the boundary pixels are extracted is shown in Figure 4.

4.4 Computation of Fourier Descriptors

The FD is used in describing the boundary of shapes in 2D images using FT. The steps needed to compute FDs in this work are as given in the subsections following.

1. **Boundary Parametization**:

The boundaries of the image are given as the set $\{(x_1,y_1),(x_2,y_2),(x_3,y_3),...,(x_N,y_N)\}$ containing N ordered points/pixels.

2. Boundary Tracing:
Any arbitrary point, say, (x_0, y_0) is chosen as the starting point and the travesal (tracing) of all the boundaries to the left and right of point (x_0, y_0). A complete list of boundary pixels will thus be obtained after this.

3. Complex representation of boundary points:
The boundary pixels set $(x_i, y_j),\ \ i = 0(1)N,\ j = 0(1)N$ is then represented as complex variables
$(x_n + jy_m), n = 0(1)N, m = 1(1)N,\ j^2 = -1.$

3. Application of Fourier Transform:
Next to complex representation above, appropriate Fourier Transform such as FFT or DFT ([15]) is applied to it. The coefficient thus obtained is called **Fourier Descriptors (FDs).**

4. Invariant FDs:
Let the Fourier descriptors be given as $|FD_1|, |FD_2|, |FD_3|, ..., |FD_N|$. To achieve invariant properties with respect to rotation, scaling, and translation, the following steps are performed:

(a) Set $|FD_1| = 0$

(b) Divide the remaining FDs by the first of the remaining FDs. This implies that

$$|FD_2| \to \frac{|FD_2|}{|FD_2|}$$

$$|FD_3| \to \frac{|FD_3|}{|FD_2|}$$

.
.
.

$$|FD_N| \to \frac{|FD_N|}{|FD_2|}$$

4.5 Zernike moments

The Zernike moment can be defined as a set of complete complex orthogonal basis functions that are square integrable and that are defined over the unit disk. If we are given the ordered pair (n, m), which represents the order of the Zernike function, then the ZMI , according to [2] ZM can be defined mathematically as Equation 4.5

$$V_{nm}(\rho, \theta) = R_{nm}(\rho) e^{im\theta}, \quad for \quad \rho \le 1 \tag{4.5}$$

where

$$\rho = \sqrt{x^2 + y^2}, \theta = \arctan(\frac{y}{x}), \tag{4.6}$$

are the image pixel radial vector and angle between the it and x-axis respectively.

$$R_{nm}(\rho) = \sum_{a=0}^{(n-|m|)/2} (-1)^a \frac{(n-a)!}{a!(\frac{n+|m|}{2} - a)!(\frac{n-|m|}{2} - a)!} \rho^{n-2a} \tag{4.7}$$

The detailed about ZM and its computation can be found in the companion paper [2].

4.6 Complete Feature set

The complete feature set for this work comprises of 10 ZMs and 10 FDs, making 20 features all together. The GA employed in this work (also used by the companion paper [3]) was used to reduce the entire feature set to new dataset containg only 8 feature set which is more economical for the concerned image classification system shown in Figure 8.

5.0 PROBABILISTIC NEURAL NETWORK (PNN) and GENETIC ALGORITHM (GA)

5.1 Probabilistic Neural Networks (PNN)

Probabilistic Neural Network (PNN) is a feed forward Neural Network that uses kernel methods for density estimation in a multi-category problem and which was introduced by D.F Specht [2, 20]. PNN is an approximation to Bayes classifier The detailed and pseudocode about PNN can be seen in the companion paper [2]. The functionalities of a PNN depend on the standard deviation of the underlying Gaussian distribution. This parameter, commonly called PNN spread or smoothing parameter is a determinant of the receptive width of the Gaussian window for the pdf of the training set. The PNN spread, when it is too small, can cause the PNN to overfit (be very selective), since each training data point will have too much influence and when it is too large, can cause the PNN to be under selective. Striking the balance between underfitting and overfitting is the main rational for tunning the PNN spread. In this section, the focus is on how to use Genetic Algorithm (GA) to optimize the smoothing parameter (spread) of the PNN Classifier to further improve the classification accuracy of the PNN. The parameters setting for the GA are shown in Table 1. The detailed explanation on GA can be found also in the companion paper [3].

5.2 Genetic Algorithm (GA)

GA is a class of optimization technique that imitate the natural evolution process human biological genetics. GA involves several functions which are iteratively invoked to manipulate initial population of chromosomes (solution candidates) to generate new population [3]. The basic functions (operations) involved in the GA used herein are defined and explained in the following subsections.

1. **Chromosome encoding**: Direct decimal encoding is employed to represent the smoothing parameter. To this end, a set of real random numbers are generated in the GA to represent the initial population. The dimension of the initial population is PopSize **x** GenomeLength, where GenomeLength = 1, PopSize = 50 for GA1 configuration and 100 for GA2.

2. **Fitness evaluation**: Each chromosome is evaluated using classification error from the PNN Classifier itself. This is shown in Algorithm 1. The actual classification error is computed from the confusion matrix generated from the PNN Classifier using the Training Set, Test Set, ClassInformation and the PNN Spread.

3. **Selection mechanisms**: The aim of selection mechanism in GA is to make sure the population (solution candidates) is being constantly improved over all fitness values. The selection mechanism helps the GA in discarding bad spread values and keeping only the best individuals. The employed scheme herein is tournament selection of size 2, where two chromosomes are selected from the population after the Elite kids are taken out and the best of the two chromosomes, (using fitness ranking), is selected. Tournament selection is performed iteratively until the new population is filled up. Tournament scheme was used for both GAs in this work.

4. **Genetic operators**: For GA1-PNN uniform mutation and heuristic crossover were used as genetic operators while for GA2-PNN (our implementation), gaussian mutation and arithmetic crossover were used. Both mutation operators (uniform and gaussian) perturb each chromosome by adding a random number from the appropriate or associated distribution to each parent from the tournament selection. The parameter for the GA mutation operator are shown in Table 1. The heuristic crossover on the other hand, returns a child chromosome lying on the straight line containing the two parents chromosomes. It uses fitness values of the two parent chromosomes to determine the direction of the search. Thus the offsprings produced by the heuristic crossover is:

Child1 = $P_2 + RD * (P_1 - P_2)$
Child2 = P_1
where RD is a random number between 0 and 1 and P_1, P_2 are the two parent chromosomes
(P_1 is the best parent, and P_2 is the worst). Arithmetic crossover also linearly combines two parent chromosomes to produce new offsprings but according to the following equations:
Child1 = $aP_1 + (1-a)P_2$

Child2 = $(1\text{-}a)P_1 + aP_2$

5. **GA stopping criteria**:
The two stopping criteria used for these GA are: (1) maximum number of generation and (2) number of GA iteration. These are already listed in Table 1.

Algorithm 1 Fitness Function Evaluation
1: **procedure** FITFUNCTION()
2: **Parameters:** TrainingSet, TestSet and the ClassInformation.
3: **Input** Spread from the GAPopulationFunction.
4: PNNClassify(TrainingSet, TestSet, ClassInformation, Spread) $\rightarrow$ FitnessValue.
5: **Output** FitnessValue.
6: **end procedure**

Algorithm 2 Pseudocode for GAPNNSpreadOptimizer
1: **procedure** GAPNNSPREADOPTIMIZER(PNNPARAMETERS, GAPARAMETERS)
2: **Generate** Initial population of PNN Spread.
3: **Set** GA()Parameters as shown in Table 1.
4: **Set** Counter = 1
5: **Do**
6: Simulate GA()
7: Output best chromosome $\rightarrow$ **Spread** and store its value in a variable **ListOfChromosomes**.
8: Counter = Counter + 1
9: **Until** Counter = M, where M = Number of GA simulation needed.
10: **Output** $\rightarrow$ **ListOfChromosomes**.
11: **end procedure**

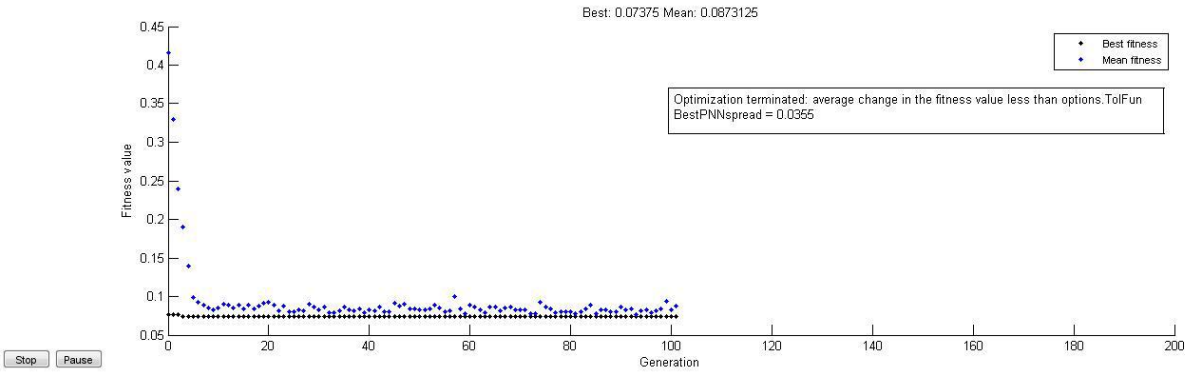

Figure 5: GA Optimization for PNN

Algorithm 3 Pseudocode for PNNAccuracy versus PNNSpread Plot
1: **procedure** PERFORMANCE(TRAININGSET,TESTSET,CLASSINFORMATION,LISTOFCHROMOSOMES)
2: **Input:** TrainingSet, TestSet, ClassInformation, ListOfChromosomes.
3: **Set** M = length(ListofChromosomes).
4: **Set** Counter = 1
5: **Do**
6: spread = ListOfChromosomes(Counter)
7: Accuracy(Counter) = PNNClassify(TrainingSet, TestSet, ClassInformation, spread)
8: Counter = Counter + 1
9: **Until** Counter = M.
10: **Graph** $\rightarrow$ Plot(ListOfChromosomes, Accuracy).
11: **end procedure**

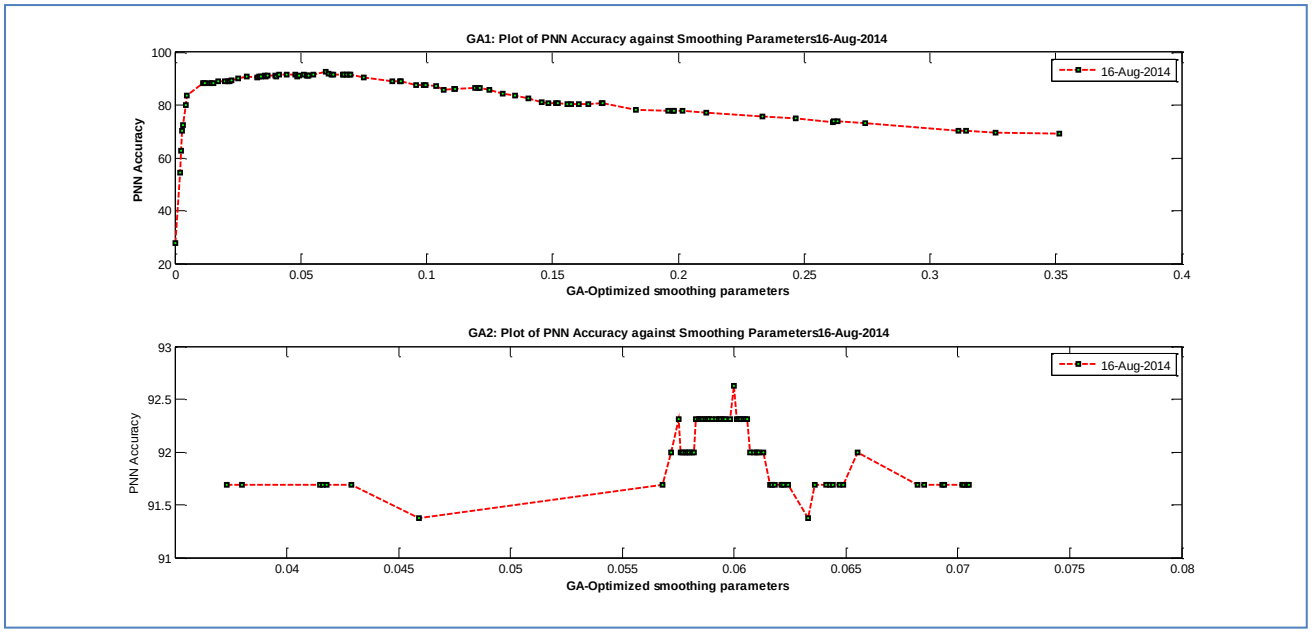

Figure 6: Variation of PNN Accuracy with GA-Based Smoothing Parameters

Table 1: Parameter configuration for the GAs

GA 1Parameter	Value	GA2 Parameter	Value
Population size	50	Population size	50
Genomelength	1	Genomelength	1
Population type	Real	Population type	Real
Fitness Function	PNN-Based Classification Error	Fitness Function	PNN-Based Classification Error
Number of Generations	100	Number of Generations	100
Crossover	Heuristic Crossover	Crossover	Arithmetic Crossover
Crossover fraction	0.8	Crossover fraction	0.8
Mutation	Uniform Mutation	Mutation	Uniform Mutation
Mutation fraction	0.01	Mutation fraction	0.01
Selection scheme	Tournament of size 2	Selection scheme	Tournament of size 2
Elite count	2	Elite count	2

6.0 DESIGN OF IMAGE CLASSIFICATION SYSTEM

The steps involved in the design of the classification system shown in Figure 8 are described in Figure 7. These steps are pretty much the same with most image classification systems. The novelty of this study is seen in the Image segmentation module and the optimization module for the PNN classifier. The entire implementation (including the GUI) was done using MATLAB 2013a.

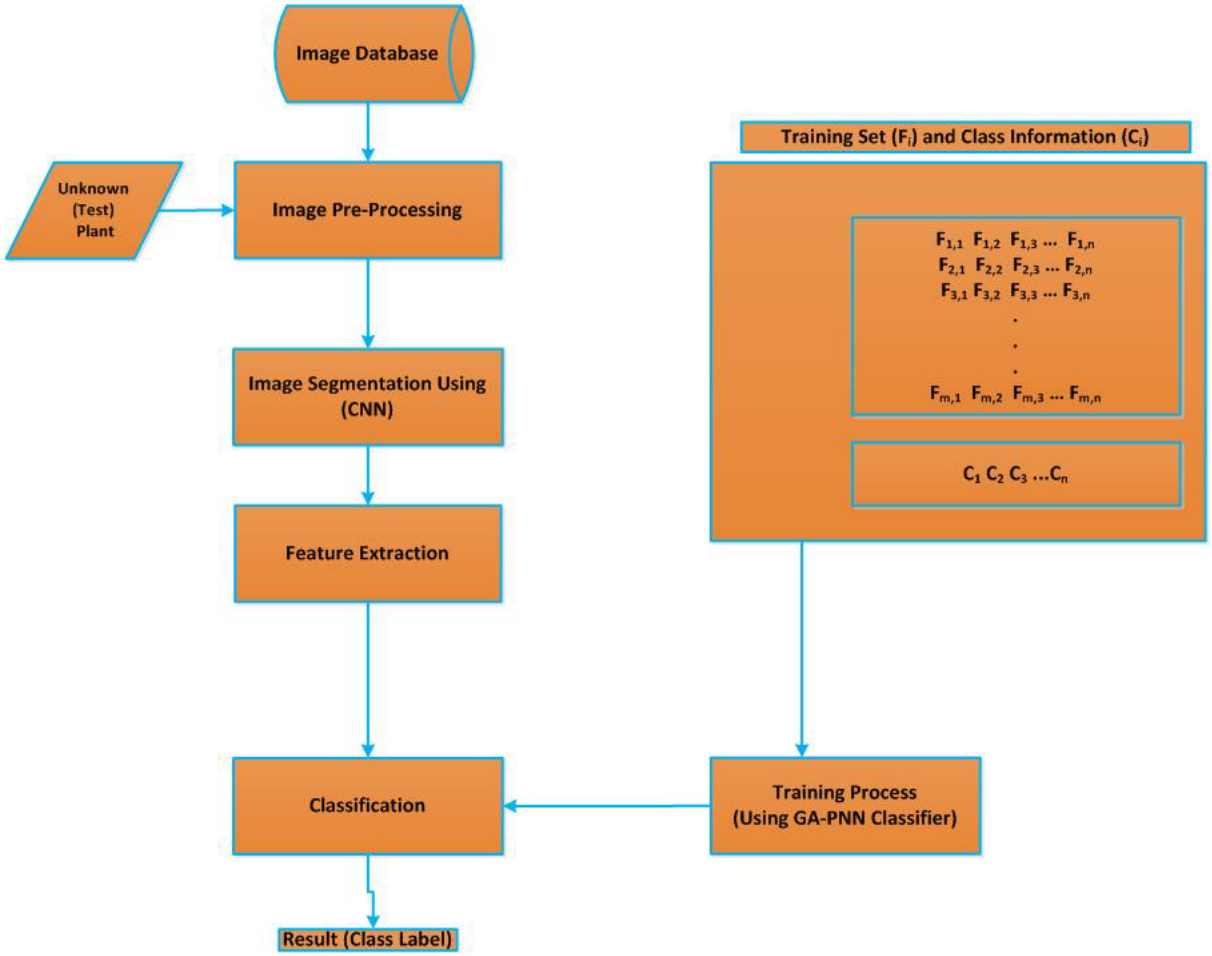

Figure 7: Learning system based on PNN Classifier

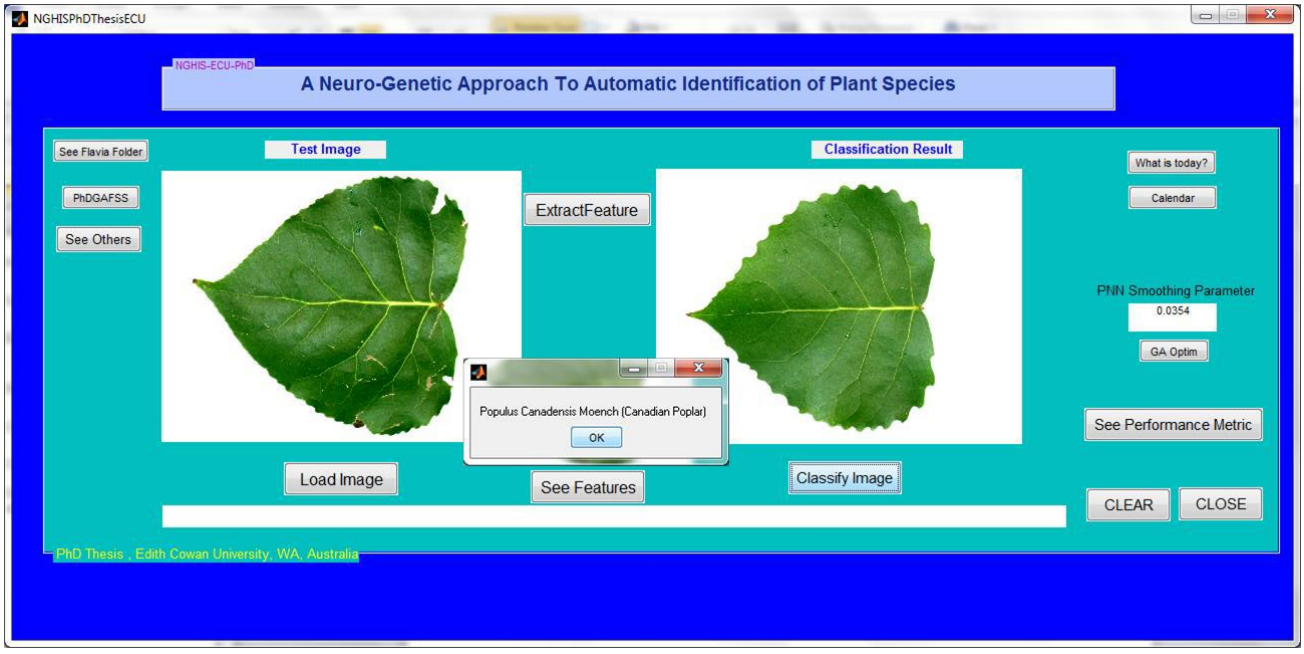

Figure 8: Classification result using an unknown plant specie as a test image

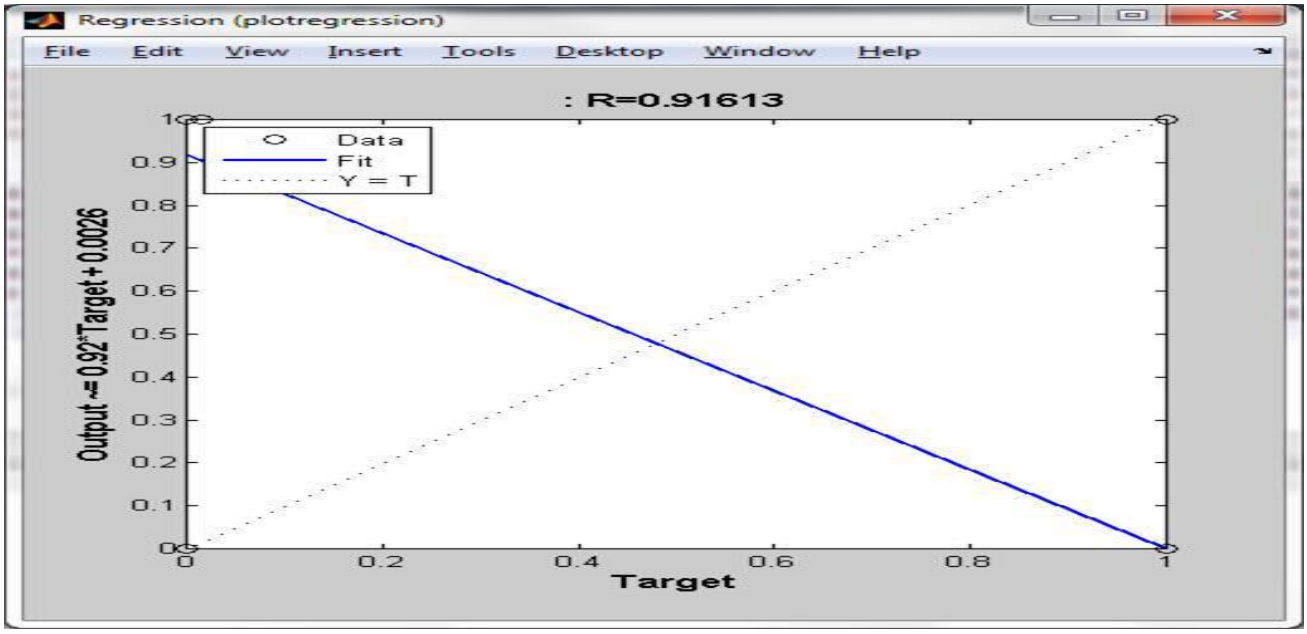

Figure 9: Regression of Predicted Species on the Actual Species

7: EXPERIMENTAL VALIDATION

The approach used in validating the PNN Classifier is the k-Fold Cross Validation (k-Fold CV) with k = 10. Generally, a cross validation (CV) is a method of partitioning the feature space into training and testing sets . The detailed of the CV algorithm is shown in the companion papers [2, 3]. Herein, the models are fitted using training set, while the fitted models are validated through testing set by measuring the error predicted. The training set and testing set are both disjoint to ensure that the testing set for evaluating the model (in our case, PNN Classifier), are not used in fitting the model [8].

8: RESULTS AND OBSERVATION

8.1 Genetically optimized PNN

The two GA approaches for the optimization of PNN reported different PNN spread. This difference is traceable to the difference in the mode of implementation of the two GAs. The MATLAB Toolbox GA (GA1) reported PNN spread value as **0.0354** while our GA (GA2) implementation gave **0.060**. The best classification accuracy was associated with GA implementation. The accuracies for GA1 and GA2 are respectively **92.01% and 92.62%.**

8.2 Image Classification System

The training set consists 8 features from 1587 images while the test set consists of 8 features from 320 images. GA was used to reduce the original feature space from 20 to 8. There are10 samples per each species in the test set. The system in this work was built to be more robust and with improved classification accuracy. CNN segmentation method helped in improving the speed of the entire system. Some of the species in found in ([2, 3]) are very similar in shape and as a result of this, some of them were wrongly classified as belonging to another species. Among such are those of Class 16 (Osmanthus Fragrans), Class 19 (Cinnamomium Japonicum), and Class 26 (Mangletia Fordiana Olive). The shapes of leaves belonging to Class 8, Class 10, and Class 4 are similar. The PNN classifier wrongly classified 1 instances belonging to Class 1 as 2 instances of class 21. The species in Class 9 were wrongly classified as species in Classes 4, 6 and 10. The effect of the PNN spread (smoothing parameter) on the performance of the PNN classifier is also demonstrated in Figure 6, where the classification accuracy of the PNN varies with the values of the spread. A spread value of 0 is not prefarable to avoid overfitting and to show the true classification ability of the PNN. Also, the PNN becomes approximately equal to kNN (k Nearest Neighbor) in functionality when the spread is taken as zero. As part of our contribution, GA was used to optimize the PNN spread used for this work . To validate the system shown in Figure 8, a **10-fold CV** was used to partition the feature space into training data and test data. Further metrics used for the PNN here are regression curve, recall (sensitivity) and precision. The regression plot (linear regression of the target with respect to the output of the classifier) for the 32 species is shown in Figure 9. A good classifier should have the R coefficient in the regression curve close to value 1. The equation in the regression plot is given as **y = 0.92x+0.0026**, where **y** = output, **x** = target, indicates that a good accuracy since the value of R is **0.92** (very close to 1). The confusion value (the fraction of species misclassified) for the PNN classifier was **0.0750**. This implies 24 out of 320 instances were wrongly classified. The average values for False Negative Rate (FNR), False Positive Rate, True Positive Rate (TPR),

True Negative Rate (TNR) for all the classified species of plants are respectively given as {**0.0026, 0.0699, 0.9301, 0.9974**}. Again these values indicate the classification strength of our system.

9: CONCLUSION AND FUTURE DIRECTION

The main challenge normally encountered in the use of PNN for classification is the choice of PNN spread. This spread is the standard deviation for the underlying pdf of the given training set. Choose this value arbitrarily is not good to avoid under and over fitting. It's quite interesting to observe the GAs bringing out the best spread that will not cause the PNN to overfit or underfit. The plot of PNN accuracy against the PNN spread for the two approaches are shown in Figure 6. The peak accuracy for all the two plots were based on the spread value 0.0354 and 0.060 obtained by the two GAs. The spread reported by MATLAB GA Toolbox (GA1) were 0.0354 with accuracy 92.01% while our GA implementation slightly improved accuracy to 92.62%. Further improvement to this work could be in the area of using another classifier and / or more discriminating features than ZMs and FDs. Images of fruits and flowers of the plants may also be amalgamated to the database to avoid misclassifications.

10. REFERENCES

[1] H O Babatunde, C O Akanbi, O G Fadare, A A Eludire, O B Aluko, and O G Egbedokun. On numerical simulation of a boundary-valued neuronal model. World J of Engineering and Pure and Applied Sci, WJEPAS, 20(2):20–25, 2012.

[2] Oluleye Babatunde, L. Armstrong, J Leng, and D Diepeveen. Zernike moments and genetic algorithm: Tutorial and application. British Journal of Mathematics and Computer Science, 4(15):2217–2236, 2014.

[3] Babatunde Oluleye, Armstrong Leisa, Leng Jinsong, and Diepeveen Dean (2014). A Genetic Algorithm-Based Feature Selection. International Journal of Electronics and Communication and Computer Engineering, IJECCE 5(4); 889-905

[4] T Z Charles and Z R Ralph. Fourier descriptors for plane closed curves. IEEE Transactions on Computers, C-21(3):269–281, 1972.

[5] Pornpanomchai Chomtip, Kuakiatngam Chawin, Supapathranon Pitchayuk, and Siriwisesokul Nititat. Leaf and flower recognition system (e-botanist). IACSIT International Journal of Engineering and Technology, 3(4):10–15, 2011.

[6] L O Chua and T Roska. Cellular neural networks and vision computing. Cambridge University Press, 2002.

[7] L O Chua and L Yang. Cellular neural networks: theory and applications. IEEE Trans on Circuits and System, 35(10):1257–1272, 1988.

[8] B Clarke, E Fokoue, and H Zhang. Principles and theory for data mining and machine learning. Springer Series in Statistics;
http://www.amazon.com/Principles-Machine-Learning-Springer-Statistics/dp/0387981349:Page 798, 2009.

[9] James S Cope, David Corney, Jonathan Y Clark, and Paul W Remagnino. Plant species identification using digital morphometrics: A review. Digital Imaging Research Centre, Kingston University,London,UK and Department of Computing, University of Surrey, Guildford Surrey, UK, pages 1–21, 2011.

[10] Z Dengsheng and Lu Guojun. A comparative study on shape retrieval using fourier descriptors with different shape signatures. Gippsland School of Computing and Information Technology,Monash University,Australia, 2000.

[11] R Ercsey, M Maria, Z Nda, and T Roska. Statistical physics on cellular neural network computers. Physica D: Nonlinear Phenomena, 237(9):2051–2068, 2008.

[12] J. B. Fourier. The Analytical Theory of Heat. The Universal Press, 1878.

[13] B Hezekiah, A T Akinwale, and O Folorunso. A cellular neural networks- based model for edge detection. Journal of Information and Computing Science, 5(1):003–010, 2010.

[14] MathWorks. Matlab neural network toolbox documentation. MathWorks. Inc. [Online]. Available:, 2007.

[15] Mathworks. Signal processing toobox (discrete fourier transform (dft)). MathWorks. Inc, 2009.

[16] K Meeta, K Mrunali, P Shubhada, P Prajakta, and B Neha. Survey on techniques for plant leaf classification. International Journal of Modern Engineering Research (IJMER), 1(2):538–544, 2012.

[17] K K Pahalawatta. A plant identification system using both global and local features of plant leaves. MSc Thesis at the department of Computer Science and Software Engineering, University of Canterbury, New Zealand, pages 1–127, 2008.

[18] Emanuel Parzen. On estimation of a probability density function and mode. Annals of Mathematical Statistics, Vol 33, Issue 3:1065–1076, 1962.

[19] T Roska, A Zarandy, and C Rekeczky. Cellular neural networks. CRC Press LLC, 2003.

[20] D F Specht. Probabilistic neural networks for classification, mapping, or associative memory. IEEE International Conference on Neural Networks, 1(2):525–532, 1988.

[21] M.R Teague. Image analysis via the general theory of moments. J. Optical Soc. Am. 70, page 920-930, 1980.

[22] Arif Thawar, Krekor Zyad Shaaban, Lala, and Baba Sami. Object classification via geometric, zernike and legendre moments. Journal of Theoretical and Applied Information Technology, Vol 7. No 1:31–37, 2009.
[23] Karrels Tyler. Fourier descriptors: Properties and utility in leaf classification. ECE 533 Fall 2006

[24] B.; Stefan L.; Karsten B. & Andreas Z. Urilch, W.; Peter. Plant species classication using a 3d lidar sensor and machine learning. Ninth International Conference on Machine Learning and Applications, pages 339–345, 2010.

[25] Stephen Gang Wu, Forrest Sheng Bao, Eric You Xu, Yu-Xuan Wang, Yi-Fan Chang, and Qiao-Liang Xiang. A leaf recognition algorithm for plant classification using probabilistic neural network. IEEE 7th International Symposium on Signal Processing and Information Technology, Cario, Egypt; ArXiv 0707.4289 v1 [CS.AI], 2007.

[26] Yonqing Xin, Pawlak Miroslaw, and X L. Simon. Image reconstruction with polar zernike moments. ICARPR 2005, LNCS 3687, pages 394–403, 2005.