Adaptive, Evolving, Hybrid Connectionist Systems for Image Pattern Recognition*

Nikola K. Kasabov[#], Steven I. Israel[*], and Brendon J. Woodford[#]

[#]Department of Information Science, [*]Department of Surveying
University of Otago
PO Box 56, Dunedin, New Zealand.
{*E-mail:*}{*nkasabov,bjwoodford*}*@infoscience.otago.ac.nz, israel@albers.otago.ac.nz*

Abstract. The chapter presents a new methodology for building adaptive, incremental learning systems for image pattern classification. The systems are based on dynamically evolving fuzzy neural networks that are neural architectures to realise connectionist learning, fuzzy logic inference, and case-based reasoning. The methodology and the architecture are applied on two sets of real data - one of satellite image data, and the other of fruit image data. The proposed method and architecture encourage fast learning, life-long learning and on-line learning when the system operates in a changing environment of image data.

Keywords: image classification, evolving fuzzy neural networks, case-based reasoning.

1 Approaches to Pattern Recognition

Most image pattern recognition tasks involve six major sub-tasks: image segmentation, target identification, attribute selection, sampling, discriminant function generation, and evaluation. This chapter focuses upon a novel approach for generating discriminant functions that minimise the limitations of conventional algorithms. Specifically, this new approach is capable of processing high dimensional feature vectors, computationally efficient, and suitable for evaluating both static and dynamic input streams.

Perlovsky [1] defined three classes of pattern recognition algorithms based on how they divide the feature space into acceptance regions: discriminating surfaces, nearest neighbours, and model based approaches. Pattern recognition algorithms can also be viewed by how they apply knowledge: model based, rule based, and example based. Model based algorithms include all deterministic and statistical algorithms where discriminant functions are generated by prior knowledge of the acceptance region's structure of the feature space. Many of these algorithms fall into Perlovsky's definition of a nearest neighbour algorithm. For these algorithms, a small number of examples are required to estimate the parameters. Another advantage is that, when the

* To appear in *Soft Computing for Image Processing* by S.K. Pal, A. Ghosh, and M.K. Kundu (eds.), Physica-Verlag, 1999

data fits the assumed model, an estimate of the mapping precision can be determined prior to analysis. However, there are costs associated with the model based algorithms. The data must support the model and the entire population must be fully visible, available and representative. In general, these algorithms are unadaptable because the boundary between two acceptance regions is not supposed to change. Rule based algorithms apply knowledge without requiring any inductive examples. They divide the feature space by defining boundaries between acceptance regions. Inferences are generated based upon expert opinion. Rule based algorithms are highly adaptable. Acceptance regions are finely tuned to minimise mapping error and are noise invariant. However, the drawbacks are severe. Often the exact mapping from the inputs to the outputs is unknown. The generation of rules requires a large amount of direct operator invention. Because rules form discriminating surfaces, new rule sets are required when the data distribution is not static or output labels of extracted attributes change.

Example based approaches minimise the limitations of model based and rule based algorithms. Example based algorithms do not contain any prior knowledge about the structure of the acceptance regions in the feature space. Often they operate by building discriminating surfaces, which makes them adaptable. The drawbacks are that they require a large number of training examples to estimate a large number of independent parameters.

For sufficiently complex systems the three models are limited by different resources. Model based algorithms are limited by the available computer space since all the examples must be fully visible to the system prior to generating the discriminant functions. Rule based systems are limited by problem complexity, and example based systems are limited by processing power.

In addition to the algorithmic structure, image pattern recognition algorithms are plagued by the dilemma of discriminant function specification versus generalisation. In order to obtain a high degree of similarity discriminant functions must map the inputs to a small feature space distance to their known outputs. For high precision, a high-order discriminant function is required. However, these high-order functions do not generalise well to new data. These two exclusive criteria must be optimized.

One way to utilize the advantages and to overcome the disadvantages of the different methods above is to merge them into one system. That is the topic of this chapter.

2 Merging Approaches into Hybrid Systems

Advances to the image pattern recognition problem have occurred on several fronts. The most notable was Zadeh's [2] idea of fuzzy membership functions This de-convolution of attributes from crisp values to the concept of belonging has been shown to increase the ability of the discriminant functions to generalise and assign more precise labels [3,4].

Perlovsky [1] cited that the argument between prior knowledge and adaptability has continued throughout the history of science. Grossberg [5] defined this as the stability/ plasticity dilemma. Optimum pattern recognition algorithms were identified as requiring a mix of these seemingly exclusive properties. Although no unifying mixture of algorithms has been identified, rules based algorithms have been successfully mixed with example based to produce systems that attempt to minimise the prior knowledge versus stability dilemma. These hybrid systems include fuzzy neural networks [6–10], rule based algorithms with mixed neural networks, and example based algorithms [11–14].

Example based algorithms can be broken down into two basic categories: (1) case based reasoning [15], and (2) connectionist algorithms [16]. These two algorithms process data differently. For case based reasoning, case examples (exemplars) are stored in memory. New examples are compared to existing cases based upon attribute similarity. The discriminant functions do not contain specific parameters to estimate. With connectionist architectures, none of the individual examples are stored. For each class, the similarity among the intraclass training examples and the difference with the interclass training examples define the acceptance region. The former allows for dynamic adaptable training at the cost of huge memory requirement, while the latter is noise tolerant and provides a smooth asymptotic relationship between processing time and mapping precision.

In this research, hybrid fuzzy neural networks are merged with case based reasoning. The new system is capable of dynamic data modelling, case base retrieval of information, and maintain an open architecture where existing knowledge is encoded into the system at any time of its operation.

3 Fuzzy Neural Networks (FuNN) and Evolving Fuzzy Neural Networks (EFuNN)

3.1 Fuzzy Neural Networks FuNNs

Fuzzy neural networks are neural networks that realise a set of fuzzy rules and a fuzzy inference machine in a connectionist way [7,8]. FuNN is a fuzzy neural network introduced in [8] and developed as FuNN/2 in [10]. It is a connectionist feed-forward architecture with five layers of neurons and four layers of connections. The first layer of neurons receives the input information. The second layer calculates the fuzzy membership degrees to which the input values belong to predefined fuzzy membership functions, e.g. small, medium, large. The third layer of neurons represents associations between the input and the output variables, fuzzy rules. The fourth layer calculates the degrees to which output membership functions are matched by the input data, and the fifth layer does defuzzification and calculates exact values for the output variables. A FuNN has features of both a neural network and a fuzzy inference machine. A simple FuNN structure is shown in Fig. 1. The number of neurons in each of the layers can potentially change during operation by growing or shrinking. The number of connections is also modifiable through learning with forgetting, zeroing, pruning and other operations [8,10].

The membership functions (MF) used in FuNN to represent fuzzy values, are of triangular type, the centres of the triangles being attached as weights to the corresponding connections. The MF can be modified through learning that involves changing the centres and the widths of the triangles. Several training algorithms have been developed for FuNN [8,10]:

1. A modified back-propagation (BP) algorithm that does not change the input and the output connections representing MFs;
2. A modified BP algorithm that utilises structural learning with forgetting, i.e. a small forgetting ingredient, e.g. 10^{-5}, is used when the connection weights are updated;
3. A modified BP algorithm that updates both the inner connection layers and the membership layers. This is possible when the derivatives are calculated separately for the two parts of the triangular MF. These are also the non-monotonic activation functions of the neurons in the condition element layer;
4. A genetic algorithm for training; and
5. A combination of any of the methods above used in a different order.

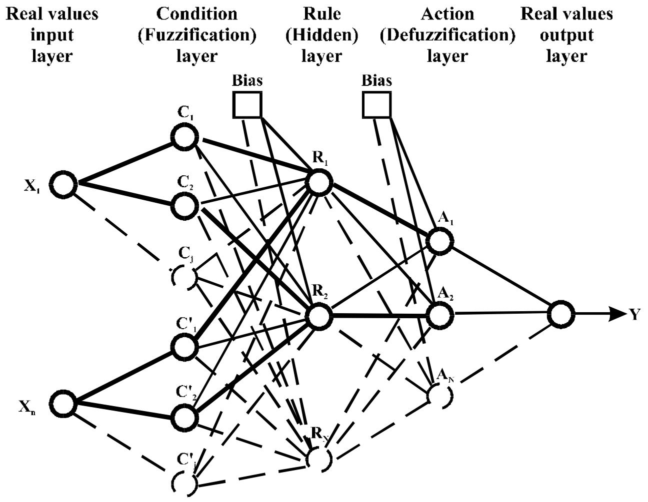

Fig. 1. A FuNN structure of 2 inputs (input variables), 3 fuzzy linguistic terms for each variable (3 membership functions). The number of the rule (case) nodes can vary. Three output membership functions are used for the output variable.

Several algorithms for rule extraction from FuNNs have been developed and applied [8–10]. One of them represents each rule node of a trained FuNN as an IF-THEN fuzzy rule.

FuNNs have several advantages when compared with the traditional connectionist systems, or with the fuzzy systems [8]:

1. They are both statistical and knowledge engineering tools,
2. They are robust to catastrophic forgetting, i.e. when they are further trained on new data, they keep a reasonable memory of the old data,
3. They interpolate and extrapolate well in regions where data is sparse,
4. They accept both real input data and fuzzy input data represented as singletons (centres of the input membership functions).

The above listed features of FuNNs make them universal statistical and knowledge engineering tools. Many applications of FuNNs have been developed and explored so far: pattern recognition and classification; dynamical systems identification and control; modelling chaotic time series and extracting the underlying chaos rules , prediction and decision making [8]. A FuNN simulator is available as part of a hybrid software environment FuzzyCope/3 from http://kel.otago.ac.nz/software.

4 Evolving Fuzzy Neural Networks EFuNNs

4.1 A general description

EFuNNs are FuNN structures that evolve according to the ECOS principles [11–14]. EFuNNs adopt some known techniques from [17–20] but they also introduce new NN techniques, e.g. all nodes in an EFuNN are created during (possibly one-pass) learning. The nodes representing MF (fuzzy label neurons) can be modified during learning. As in FuNN, each input variable is represented here by a group of spatially arranged neurons to represent a fuzzy quantisation of this variable. For example, three neurons can be used to represent "small", "medium" and "large" fuzzy values of the variable. Different membership functions (MF) can be attached to these neurons (Triangular, or Gaussian, etc.). New neurons can evolve in this layer if, for a given input vector, the corresponding variable value does not belong to any of the existing MF to a degree greater than a membership threshold. A new fuzzy input neuron, or an input neuron, can be created during the adaptation phase of an EFuNN.

The EFuNN algorithm, for evolving EFuNNs, has been presented in [13,14]. A new rule node rn is connected (created) and its input and output connection weights are set as follows: W1(rn) = EX; W2(rn) = TE, where TE is the fuzzy output vector for the current fuzzy input vector EX. In case of "one-of-n" EFuNNs, the maximum activation of a rule node is propagated to the next level. Saturated linear functions are used as activation functions of the fuzzy output neurons. In case of "many-of-n" mode, all the activation values of rule (case) nodes, that are above an activation threshold of *Ahtr*, are propagated further in the connectionist structure.

4.2 The EFuNN learning algorithm

Here, the EFuNN evolving algorithm is given as a procedure of consecutive steps [13,14]:

1. Initialise an EFuNN structure with a maximum number of neurons and zero-value connections. Initial connections may be set through inserting fuzzy rules in a FuNN structure. FuNNs allow for insertion of fuzzy rules as an initialization procedure thus allowing for existing information to be used prior to the evolving process (the rule insertion procedure for FuNNs can be applied [8,10]). If initially there are no rule (case) nodes connected to the fuzzy input and fuzzy output neurons with non-zero connections, then*connect* the first node rn=1 to represent the first example EX=x1 and set its input W1(rn) and output W2 (rn) connection weights as follows: it <Connect a new rule node rn to represent an example EX>: W1(rn)=EX;W2(rn) = TE, where TE is the fuzzy output vector for the (fuzzy) example EX.
2. WHILE <*there are examples*> DO
   Enter the current example x_i , EX being the fuzzy input vector (the vector of the degrees to which the input values belong to the input membership functions). If there are new variables that appear in this example and have not been used in previous examples, create new input and/or output nodes with their corresponding membership functions.
3. Find the normalized fuzzy similarity between the new example EX (fuzzy input vector) and the already stored patterns in the case nodes j=1,2..rn: Dj = sum (abs (EX - W1(j))/ 2) / sum (W1(j)).
4. Find the activation of the rule (case) nodes j, j=1:rn. Here radial basis activation function, or a saturated linear one, can be used on the Dj input values i.e. A1 (j) = radbas (Dj), or A1(j) = satlin (1 - Dj).
5. Update the local parameters defined for the rule nodes, e.g. age, average activation as pre-defined.
6. Find all case nodes j with an activation value A1(j) above a sensitivity threshold $Sthr$.
7. If there is no such case node, then <*Connect a new rule node*> using the procedure from step 1.
   ELSE
8. Find the rule node *inda1* that has the maximum activation value. (*maxa1*).
9. (a) in case of one-of-n EFuNNs, propagate the activation *maxa1* of the rule node inda1 to the fuzzy output neurons. Saturated linear functions are used as activation functions of the fuzzy output neurons:
   A2 = satlin (A1(inda1) * W2)

   (b) in case of many-of-n mode, only the activation values of case nodes that are above an activation threshold of *Athr* are propagated to the next neuronal layer.
10. Find the winning fuzzy output neuron *inda2* and its activation *maxa2*.
11. Find the desired winning fuzzy output neuron *indt2* and its value *maxt2*.
12. Calculate the fuzzy output error vector: Err = A2 - TE.
13. IF (*inda*2 is different from *indt*2) or (abs(Err (*inda*2)) > *Errthr*) <*Connect/create a rule node*>
14. Update: (a) the input, and (b) the output connections of rule node k=inda1 as follows:
   (a) Dist=EX-W1(k); W1(k)=W1(k) + $lr1$. Dist, where $lr1$ is the learning rate for the first layer;
   (b) W2(k) = W2 (k) + $lr2$. Err. maxa1, where $lr2$ is the learning rate for the second layer.
15. Prune rule nodes j and their connections that satisfy the following fuzzy pruning rule to a pre-defined level:
   IF (node (j) is OLD) and (average activation A1av(j) is LOW) and (the density of the neighbouring area of neurons is HIGH or MODERATE) and (the sum of the incoming or outgoing connection weights is LOW) and (the neuron is NOT associated with the corresponding "yes" class output nodes (for classification tasks only)) THEN the probability of pruning node (j) is HIGH.

   The above pruning rule is fuzzy and it requires that all fuzzy concepts such as OLD, HIGH, etc., are defined in advance. As a partial case, a fixed value can be used, e.g. a node is old if it has existed during the evolving of a FuNN from more than 60 examples.
16. END of the while loop and the algorithm
17. Repeat steps 2-16 for a second presentation of the same input data or for ECO training if needed.

5 Case Study 1: Environmental Remote Sensing: A Case for Spectral Classification

5.1 Sampling Image Data for the Experiment

A System Pour l'Observation de la Terre (SPOT) satellite image of the Otago Harbour, Dunedin, New Zealand, was used for the classification. The SPOT image has 3 spectral bands sensing the green, red and infrared portions of the electromagnetic spectrum. Ten covertypes, containing intertidal vegetation and substrates, were recorded during a ground reference survey. From the SPOT image, a minimum of three spatially separable reference areas was extracted for each of the ten covertypes. All of the sample pixels for a given covertype were amalgamated and randomly sorted into training and test sets. Typically, remote sensing data provides a large number of examples for each class.

5.2 Natural Confusion Among Classes

The problem with mapping natural systems (inputs) to human determined classes (outputs) is that some confusion may occur. There are 2 major types of confusion: (1) errors of omission, false negative errors, and (2) errors of commission, false positive errors. For the case study problem, considerable confusion exists among classes 3, 4 and 5 (*hisand*, *lowsand* and *lowzost*). To graphically illustrate the confusion among these classes, scatterplots were produced showing the relationship between the inputs and the outputs (Fig.2 and Fig.3).

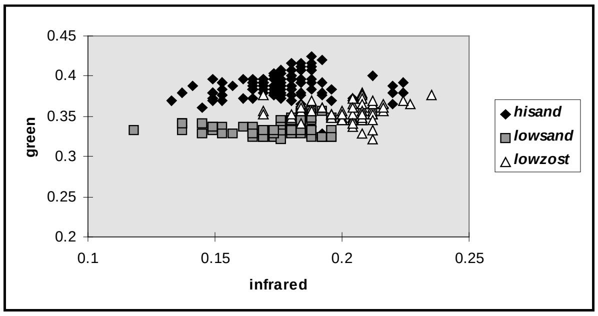

Fig. 2. Scatterplot of 3 Ambiguous Classes (infrared versus green inputs)

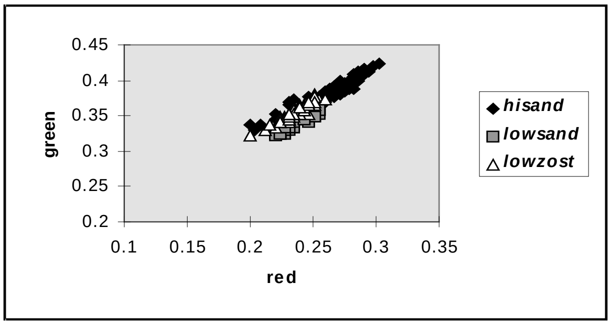

Fig. 3. Scatterplot of 3 Ambiguous Classes (red versus green inputs)

While other classes are readily differentiable, these classes need special attention in our classification system.

To ensure an appropriate network structure for classification, learning and pruning must befinely balanced to ensure sufficient generalisation to untrained data. The parameters that limit the creation of rule nodes or initiate pruning and thereby improve generalisation are age, sensitivity, error threshold, and pruning rate. As the age threshold increases, the network retains what it has learned over a longer time. Pruning is less likely to occur, but the network will be less likely to regenerate information that it has already processed, in other words, it is less likely to reproduce a rule node that has previously been pruned. The pruning rate is a weighting parameter applied to pruning rule.

Sensitivity and error threshold are directly related to the generation of new rule nodes. As the sensitivity between input patterns increases, the network is more likely to create new rule nodes. As the error threshold between the actual output and the calculated output reduces, the network is again more likely to require additional rule nodes. Both of these parameters tend to force the network into overspecification.

The learning rate influences training and overestimating what a node has learned. The learning rate tends to overestimate what a node has learned. As such, as the learning rate increases, the nodes will saturate faster than expected and tend to create larger networks that reduce the generalisation capabilities.

In order to compare the usefulness of the EFuNN to either the Bayes optimum classifier (Fig.4) or existing fuzzy neural networks (Fig.5), an additional preprocessing step was required. Originally, the training data were randomly sorted. The existing fuzzy neural networks randomly selected the training samples with a uniform distribution. The current EFuNN does not contain this function, so the training data were randomly sorted so that the age parameter was not a function of output class. To obtain the output memberships, the outputs needed to be unscrambled.

5.3 Experiments

The experiments associated with EFuNN were designed initially to replicate the performance of a conventional FuNN while highlighting its improved speed. Later experiments were performed to demonstrate the EFuNN's capabilities to improve mapping performance. It will be shown that future research will develop techniques to minimise data manipulation.

The initial EFuNN experiment was performed with conservative values for the thresholds and learning rates. In this manner, the system was constrained to operate as a conventional FuNN with one exception, the data was trained for a single iteration. Sensitivity, error threshold, learning and forgetting were assigned to 0.95, 0.001, 0.05, and 0.01 respectively. The age was assigned to the size of the entire dataset so that all examples contributed evenly during training.

In an attempt to improve generalization, forgetting and learning rates were eliminated. The sensitivity was reduced and the error tolerance was increased. An additional experiment was performed to demonstrate the characteristics of increased specification. To increase specification the learning rate was applied with a small forgetting. Finally, the last experiment looked at incorporating a volatility element by reducing the age parameter to two time positions. Each conditional training philosophy was applied to each class trained separately.

The initial test classification accuracy for EFuNN (kappa =0.80; Fig.6) was identical to the FuNN (kappa = 0.80; Fig.5) and slightly worse than the MLC (kappa = 0.84; Fig4). The training accuracy was slightly higher. It is interesting to note that the number of rule nodes for the EFuNN were considerably larger (279 to 10) than the FuNN. When the learning rate(lr) and pruning rate (pr) were set to zero, the network generalised better with ten percent of the rule nodes assigned. The classification accuracy improved (kappa = 0.82, Fig.7). However, when learning and forgetting were applied to the initial conditions mapping precision decreased (kappa = 0.57; Fig.8). The age parameter added considerable volatility to the analysis as reduced age made the network for *lowsand* unstable. However, when applied to the *hisand* and *lowzost* networks, mapping error was maintained (kappa = 0.82; Fig.9).

	training data results					
	random sort maximum likelihood classifier					
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**118**	0	4		122	97
lowsand	0	**80**	2		82	98
lowzost	8	3	**72**		83	87
				270		
sum	126	83	74		**283**	
percent	94	96	97			**95.41**
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**58**	0	4		62	94
lowsand	0	**34**	0		34	100
lowzost	4	7	**34**		45	76
				126		
sums	62	41	38		**141**	
percent	94	83	89			**89.36**

Fig. 4. MLC (Bayes rule); kappa = 0.84

	training data results					
random sort (3) 15-10-2 fuzzy neural networks in parallel						
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**116**	0	1		117	99
lowsand	0	**80**	2		82	98
lowzost	10	3	**75**		88	85
				271		
sum	126	83	78		**287**	
percent	92	96	96			**94.43**
iterations	*200*	*200*	*300*			
	test data	results				
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**53**	0	1		54	98
lowsand	0	**33**	0		33	100
lowzost	9	8	**37**		54	69
				123		
sums	62	41	38		**141**	
percent	85	80	97			**87.23**

Fig. 5. FuNN without learning techniques; kappa = 0.80

	training data results					
(3) 15-x-5 evolving fuzzy neural networks structure						
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**120**	0	5		125	96
lowsand	1	**82**	1		84	98
lowzost	5	1	**72**		78	92
				274		
sum	126	83	78		**287**	
percent	95	99	92			**95.47**
rules	*279*	*279*	*279*			
sthr=0.95	errthr=.001	lr=0.05	prune=0.1	fgr=0.01		
	test data	results				
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**58**	0	6		64	91
lowsand	0	**35**	2		37	95
lowzost	4	6	**30**		40	75
				123		
sums	62	41	38		**141**	
percent	94	85	79			**87.23**

Fig. 6. Initial EFuNN for three confused landcover classes; kappa 0.80

	training data results					
(3) 15-x-5 evovling fnn structure lr=.0, fr=.0 sth=.5 errthr=.5						
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**124**	0	4		128	97
lowsand	0	**82**	0		82	100
lowzost	2	1	**74**		77	96
				280		
sum	126	83	78		**287**	
percent	98	99	95			**97.56**
rules	*37*	*23*	*39*			
	test data	results				
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**57**	0	8		65	88
lowsand	0	**39**	2		41	95
lowzost	5	2	**28**		35	80
				124		
sums	62	41	38		**141**	
percent	92	95	74			**87.94**

Fig. 7. Optimised EFuNN without learning, forgetting and lower thresholds; kappa = 0.82

	training data results					
(3) 15-x-5 EFuNN structure lr=.1, fr=.1, sthr =.95, ethr = .05						
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**103**	8	1		112	92
lowsand	2	**32**	0		34	94
lowzost	21	43	**77**		141	55
				212		
sum	126	83	78		**287**	
percent	82	39	99			**73.87**
rules	*250*	*249*	*250*			
	test data	results				
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**48**	1	0		49	98
lowsand	1	**14**	0		15	93
lowzost	13	26	**38**		77	49
				100		
sums	62	41	38		**141**	
percent	77	34	100			**70.92**

Fig. 8. EFuNN with leaning and forgetting; kappa = 0.57

	training data results		prune=.5	old =2	classes 1 and 3 only	
(3) 15-x-5 evolving fnn structure lr=.0, fr=.0 sth=.5 errthr=.5						
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**100**	0	1		101	99
lowsand	1	**82**	0		83	99
lowzost	25	1	**77**		103	75
				259		
sum	126	83	78		**287**	
percent	79	99	99			**90.24**
rules	*123*	*23*	*130*			
	test data	results				
	hisand	**lowsand**	**lowzost**		sums	percent
hisand	**48**	0	0		48	100
lowsand	1	**39**	1		41	95
lowzost	13	2	**37**		52	71
				124		
sums	62	41	38		**141**	
percent	77	95	97			**87.94**

Fig. 9. EFuNN with lower thresholds, lower age threshold and learning with forgetting; kappa = 0.82

5.4 Discussion and Future Research

The important point gained by evolving systems is that comparable mapping accuracies can be obtained with a single iteration, reducing the computational burden. The FuNN for the same three classes required a total of 700 iterations while the EFuNN required three. The structure of the EFuNN is also optimized to reduce the computational burden because not all nodes are recomputed for each training example.

Other experiments will allow connections to cross between EFuNNs to force training to occur in parallel. These networks also have the capability to incorporate additional attributes and outputs into the existing network structure. This is important when new information, such as new imagery or additional spectral bands become available. Likewise the analyst is able to identify new output classes to better distinguish among the data.

6 Case Study 2: Fruit Quality Assurance: Based on Image Analysis

6.1 Introduction

The application of neuro-fuzzy techniques for object recognition has been extensively studied [21,22]. One area where these techniques have rarely been applied is in the area of horticultural research, specifically for the analysis of damage to pip fruit in orchards in order to identify which pest caused the damage. The solution to these tasks could become part of a larger computer based system to allow the user to make more informed decisions to improve the quality of the fruit produced.

Each insect or insect group has specific characteristics that allow it to be identified through the damage to fruit and/or leaves. Once the insect has been successfully identified, the appropriate treatment can be applied. Examples of the type of damage are presented below. All the images were in colour, taken at different orientations, lighting conditions, and sometime contained more than one piece of fruit on the tree. Furthermore the damage to the fruit itself was of varying size and shape. There were a total of 90 images taken, displaying the damage of three types of pests (Fig.10,11, and 12).

Successful analysis of the fruit damage requires a technique that copes with the differences in the images and still extracts the relevant features to allow positive identification of the pest. Using Daubechies wavelets for image analysis and comparison has proven to be a successful technique in the analysis of natural images [23,24]. This is because they can characterise the colour variations over the spatial extent of the image that can provide semantically meaningful image analysis. The output of the wavelet analysis could then become input to a Fuzzy Neural Network (FuNN) or Evolving Fuzzy Neural Network (EFuNN).

6.2 Sampling Image Data for the Experiment

To generate a dataset to train an EFuNN or FuNN, the three band RGB image data was converted to Hue/Saturation/Insensity (HSI) representation. Then a 4-layer 2D fast wavelet transform was computed on the intensity component of each image. Extracting a sub-matrix of size 16x16 from each intensity component resulted in a vector of 256 attributes. The lower frequency bands normally represent object configuration in the images and the higher frequency bands represent texture and local colour variation.

6.3 Architecture of the FuNN Classification System

The entire classification system was comprised of 5 FuNNs to reflect the five different types of damage that could be expected:

NN-alm-l Neural network to classify appleleaf curling midge leaf damage.
NN-alm-f Neural network to classify appleleaf curling midge fruit damage.
NN-cm Neural network to classify colding moth damage.
NN-lr-l Neural network to classify leafroller leaf damage.
NN-lr-f neural network to classify leafroller fruit damage.

The architecture of each FuNN had 256 inputs, 1792 condition nodes, (7 membership functions per input) 50 rule nodes, two action nodes, and 1 output. 67 images were used as the training dataset and 23 images were

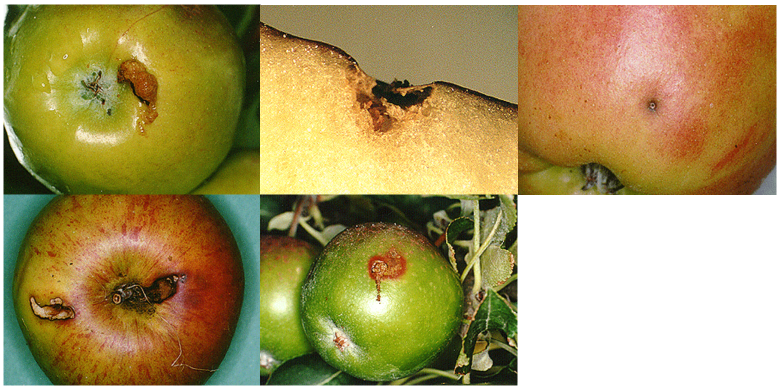

Fig. 10. Examples of codling moth damage

Fig. 11. Examples of appleleaf curling midge damage

Fig. 12. Examples of leafroller damage

used to test the classification system. The reason for the small number of images used in the experiment is due to the unavailability of electronically stored images of pest damage. However it was assumed that the variation of the input data was sufficient enough to accommodate the different types of damage expected. Each FuNN in the classification system was trained with all 67 images and the output value for the output node was changed depending on what each network was required to learn. For example the FuNN-alm-l was trained to give a 1 in position 1 of the output vector for any image that had appleleaf curling midge leaf damage and 0 in position 1 of the output vector for all the rest of the images.

After presenting the image data to each FuNN in the classification system 1000 times, the entire system was tested on the 23 test images. Results of the confusion matrix are shown in Fig.13.

training data - 5 fuzzified networks 256-1792-50-2-1 in parallel								
maximum of 1000 iterations or 0.00001 error.								
	alm-l	**alm-f**	**cm**	**lr-l**	**lr-f**		sums	percent
am-l	**9**	0	0	0	0		9	100
alm-f	0	**5**	0	0	0		5	100
cm	0	0	**22**	0	0		22	100
lr-l	0	0	0	**16**	0		16	100
lr-f	0	0	0	0	**15**		15	100
						67		
sum	9	5	22	16	15		**67**	
percent	100	100	100	100	100			**100**
iterations	*1000*	*1000*	*1000*	*1000*	*1000*			
test data								
	alm-l	**alm-f**	**cm**	**lr-l**	**lr-f**		sums	percent
am-l	**1**	0	0	0	0		1	100
alm-f	0	**1**	0	0	0		1	100
cm	2	1	**3**	2	2		10	30
lr-l	0	0	2	**1**	0		3	33
lr-f	0	0	4	2	**2**		8	25
						8		
sum	3	2	9	5	4		**23**	
percent	33	50	33	20	50			**34.78**

Fig. 13. FuNN with learning and forgetting; kappa=0.10

Recalling the FuNN on the training data resulted in 100% classification. However when the 23 test images were tested on the FuNNs there were only slightly more than a third correctly classified (34.78%). Fruit or leaf damage was correctly identified but the kind of pest inflicting the damage was not. Fine tuning of the parameters for the FuNNs or increasing the number of membership functions to account for the subtle differences in damage especially from appleleaf curling midge and leafroller warrants further investigation.

6.4 Architecture of the EFuNN Classification System

A logical next step was to train a set of 5 EFuNNs on the same image data, and compare the results to that of the FuNNs. The experiment associated with EFuNN was designed to replicate the performance of a conventional FuNN while highlighting its improved speed and demonstrate the EFuNN's capabilities to improve classification performance. The same set of 67 images was used on a set of five EFuNNs with parameters of Sthr=0.95 and Errthr=0.01. The EFuNN was trained for one epoch. The number of rule nodes generated (rn) after training was. EFuNN-alm-l: rn=61, EFuNN-alm-f rn=61, EFuNN-cm: rn=61, EFuNN-lr-l: rn=62, and EFuNN-lr-f: rn=61. The results of the confusion matrix are presented in Fig.14.

training data								
5 eFunns	lr=0.0 pune=0.1 errth=0.01 sthr=0.95 fr=0.							
	alm-l	**alm-f**	**cm**	**lr-l**	**lr-f**		sums	percent
am-l	**9**	0	0	0	0		9	100
alm-f	0	**5**	0	0	0		5	100
cm	0	0	**22**	0	0		22	100
lr-l	0	0	0	**16**	0		16	100
lr-f	0	0	0	0	**15**		15	100
						67		
sum	9	5	22	16	15		**67**	
percent	100	100	100	100	100			**100.00**
rule nodes	*61*	*61*	*61*	*62*	*61*			
test data								
	alm-l	**alm-f**	**cm**	**lr-l**	**lr-f**		sums	percent
am-l	**0**	0	0	1	0		1	0
alm-f	0	**1**	0	0	0		1	100
cm	1	1	**6**	1	2		11	55
lr-l	1	0	2	**2**	0		5	40
lr-f	1	0	1	1	**2**		5	40
						11		
sum	3	2	9	5	4		**23**	
percent	0	50	67	40	50			**47.83**

Fig. 14. EFuNN with learning and forgetting; kappa=0.45

6.5 Discussion and Future Research

It appears that the EFuNNs (48%) are marginally better at identifying what pest has caused the damage to the fruit than the FuNNs (35%). Computing the Kappa coefficient for both the FuNN and EFuNN confusion matrices substantiates this with results of 0.10 for the FuNN and 0.27 for the EFuNN. Yet under a Z test at 95% the results are not statistically significant.

7 Conclusion

This chapter presents a methodology that allows for incremental, adaptive, fast learning of images for their classification. The concept of evolving connectionist systems is used and applied on two case study data sets - satellite images, and image data of fruit.

Acknowledgements

This work was partially supported by the research grants UOO808 and HR0809 funded by the Foundation of Research, Science and Technology of New Zealand.

References

1. L. I. Perlovsky, "Computational Concepts in Classification: Neural Networks, Statistical Pattern Recognition, and Model-Based Vision," *Journal of Mathematical Imaging and Vision*, vol. **4**, no. 1, pp. 81–110, 1994.
2. L. Zadeh, "Fuzzy Sets," *Information and Control*, vol. **8**, pp. 338–353, 1965.
3. G. M. Foody, "Sharpening Fuzzy Classification Output to Refine the Representation of Sub-Pixel Land Cover Distribution," *International Journal of Remote Sensing*, vol. **19**, no. 13, pp. 2593–2599, 1998.
4. S. K. Pal and S. Mitra, "Multilayer Perceptron, Fuzzy Sets, and Classification," *IEEE Transactions on Neural Networks*, vol. **3**, no. 5, pp. 683–697, 1992.
5. S. Grossberg, *Studies of Mind and Brain*. Reidel, Boston: MA, first ed., 1982.
6. M. M. Gupta and D. H. Rao, "On the Principles of Fuzzy Neural Networks, Fuzzy Sets and Systems," *International Journal of Remote Sensing*, vol. **61**, no. 1, pp. 1–18, 1994.
7. R. Jang, "ANFIS: Adaptive Network-Based Fuzzy Inference System," *IEEE Trans. on Syst., Man, Cybernetics*, vol. **23**, no. 3, pp. 665–685, 1993.
8. N. Kasabov, *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*. MIT Press, Cambridge: MA, first ed., 1996.
9. N. Kasabov, "Learning fuzzy rules and approximate reasoning in fuzzy neural networks and hybrid systems," *Fuzzy Sets and Systems*, vol. **82**, no. 2, pp. 2–20, 1996.
10. N. Kasabov, J. S. Kim, M. Watts, and A. Gray, "FuNN/2- A Fuzzy Neural Network Architecture for Adaptive Learning and Knowledge Acquisition," *Information Sciences - Applications*, vol. **101**, no. 3-4, pp. 155–175, 1996.

11. N. Kasabov, "ECOS: A Framework For Evolving Connectionist Systems And The Eco Learning Paradigm," in *Proc. of ICONIP'98*, pp. 1232–1236, 1998.
12. N. Kasabov, *Evolving Connectionist And Fuzzy Connectionist System For On-Line Decision Making And Control, in: Soft Computing in Engineering Design and Manufacturing*. Springer-Verlag, first ed., 1999.
13. N. Kasabov, *Evolving Connectionist and Fuzzy-Connectionist Systems: Theory and Applications for Adaptive, On-line Intelligent Systems*, vol. Neuro-fuzzy Tools and Techniques for Intelligent Systems, N. Kasabov and R. Kozma (eds). Springer-Verlag, first ed., 1999.
14. N. Kasabov, "Evolving Fuzzy Neural Networks - Algorithms, Applications and Biological Motivation," in *Proc. of Iizuka'98, Iizuka, Japan*, pp. 271–274, 1998.
15. B. Jose, B. P. Singh, S. Venkataraman, and R. Krishnan, "Vector Based Image Matching for Indexing in Case Based Reasoning Systems," in *4th German Workshop on Case-based Reasoning-System Development and Evaluation*, pp. 1–7, 1996.
16. T. Kohonen, "An Introduction to Neural Computing," *Neural Networks*, vol. **1**, no. 1, pp. 3–16, 1988.
17. S. Amari and N. Kasabov, *Brain-like computing and intelligent information systems*. Springer Verlag, first ed., 1997.
18. G. Carpenter and S. Grossberg, *Pattern Recognition By Self Organizing Neural Networks*. MIT Press, Cambridge: MA, first ed., 1991.
19. T. Kohonen, "The Self-Organizing Map," *Proceedings of the IEEE*, vol. **78**, no. 9, pp. 1464–1497, 1990.
20. T. Kohonen, *Self-Organizing Maps*. Springer-Verlag, second ed., 1997.
21. A. Hojjatoleslami, L. Sardo, and J. Kittler, "An RBF Based Classifier for the Detection of Microcalcifications in Mammograms with Outlier Rejection Capability," in *Proceedings of the 1997 International Conference in Neural Networks (ICNN'97)*, pp. 1379–1384, 1997.
22. L. Shen and H. Fu, "Principal Component based BDNN for Face Recognition," in *Proceedings of the 1997 International Conference in Neural Networks (ICNN '97*, pp. 1368–1372, 1997.
23. J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei, "Applying Wavelets in Image Database Retrieval," technical report, Stanford University, November 1996.
24. J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei, "Wavelet-Based Image Indexing Techniques with Partial Sketch Retrieval Capability," in *Proceedings of the Fourth Forum on Research and Technology Advances in Digital Libraries (ADL'97)*, pp. 1–9, IEEE Press, May 1997.