# A Decade of Kasabov's Evolving Connectionist Systems: A Review

Michael J. Watts, *Member, IEEE*

***Abstract*—Evolving connectionist Systems (ECoS) are a family of constructive artificial neural network algorithms that were first proposed by Kasabov in 1998, where 'evolving' in this context means "changing over time", rather than evolving through simulated evolution. A decade on, the number of ECoS algorithms, and the problems to which they have been applied, have multiplied. This paper reviews the current state-of-the-art in the field of ECoS networks via a substantial literature review. It reviews (1) the motivations for ECoS, (2) the major ECoS algorithms in use, (3) previously existing constructive algorithms that are similar to ECoS, (4) empirical evaluations of ECoS networks over benchmark data sets, (5) applications of ECoS to real-world problems. The paper ends with some suggestions of future directions of research into ECoS networks.**

***Index Terms*—Survey, Connectionism and neural nets, Knowledge acquisition**

## I. Introduction

As of 2008, ten years have passed since Evolving Connectionist Systems (ECoS) [67]–[71], [73], [74], [76] artificial neural networks (ANN) were first proposed by Kasabov [64]–[66]. As the variety of ECoS networks and their applications have increased over this period, and the circle of researchers using them has expanded outside of Kasabov's own group, now is an appropriate time to take stock of what has been done with these networks.

It is perhaps unfortunate that Kasabov chose the term "evolving" to describe his ANN. While it seems that for many the term "evolving" evokes thoughts of evolutionary computation, ECoS are not evolutionary algorithms. ECoS networks do not use the mechanisms of evolutionary computation, such as fitness-based selection, reproduction and mutation. Instead, as far as ECoS networks are concerned, the word "evolving" has the much broader meaning of change through time.

The genesis of ECoS ANN lay in the requirements of Intelligent Information Systems (IIS) [66], [72], [77]. IIS are information processing systems that deal with information in an intelligent way, that is, they deal with information in a way similar to that of a human domain expert. Seven general, major requirements for intelligent systems were enumerated in [66]. These requirements are:

1) Fast learning from a large amount of data.
2) Real-time, incremental adaptation to new data.
3) An open structure, where new features (either inputs or outputs) can be added.
4) Able to reasonably keep track of and retrieve data that has been previously seen.
5) Continuous improvement throughout the lifetime of the system.
6) Able to analyse and explain themselves, through for example rule extraction.
7) Able to represent spatial and temporal elements of data.

School of Biological Sciences, University of Sydney, NSW 2006, Australia.

Evolving Connectionist Systems are a family of constructive ANN algorithms that were developed to fulfil these seven requirements. ECoS is a class of ANN architectures with a learning algorithm that modifies the structure of the network as training examples are presented. Although the seminal ECoS architecture was the Evolving Fuzzy Neural Network (EFuNN, see Subsection II-B), several other architectures have been developed that utilise the ECoS algorithm. These include the minimalist Simple Evolving Connectionist System (SECoS, Subsection II-C), Evolving Self-Organising Maps (ESOM, Subsection II-D), and the Dynamic Evolving Neural-Fuzzy Inference System (DENFIS, Subsection II-E).

ECoS was designed around the following principles [66]:

1) The ECoS training algorithm is designed with the intention that all the algorithm can learn from the data is learned in the first training pass (one-pass learning). Additional exposure to the training data is not necessary.
2) ECoS are intended to be used in an on-line learning application. This means that new data will be constantly and continuously coming into the system, and that this data must be learned by the network without forgetting the old. The general ECoS architecture and learning algorithm allows an ECoS network to accommodate this new data without a catastrophic forgetting of the old.
3) The manner in which neurons are added to an ECoS means that some training examples are stored, initially, verbatim within the structure of the network. These examples are then either modified (refined) by exposure to further examples, or, depending upon the training parameters used, remain the same and can be later retrieved.

The advantages of ECoS are that they avoid the problems associated with traditional connectionist structures such as MLP [66], [77]: They are hard to over-train, due to the constructive nature of their learning algorithm; they learn quickly, as the learning algorithm is a one-pass algorithm, that is, it requires only a single presentation of the data set; and they are far more resistant to catastrophic forgetting than most other models, as new training data is represented by adding new neurons, rather than accommodating the additional data in the existing neurons. ECoS networks also have several advantages over other constructive algorithms. Firstly, they are not limited to a particular application domain, they can be applied to both classification and function approximation. Secondly, they do not require multiple presentations of the training data set, as is the case with some of the constructive algorithms in existence, such as RAN [137] and GAL [11]. Finally, they are able to continue learning and are not restricted to learning a single training set as some other constructive algorithms such as RAN and Cascade Correlation [35] are.

For the purposes of this paper, an ANN is considered to be an ECoS network if it fulfils the following criteria:

1) It is a constructive network, where the addition of neurons is determined by the novelty of individual training examples
2) When a neuron is first added to the network, it explicitly represents the training example that caused it to be added.
3) It is capable of training over multiple data sets, with only a single pass over each set.
4) It is explicitly derived from the principles of ECoS, as laid down by Kasabov in [66], [77].

This paper reviews the current state-of-the-art for ECoS networks. In Section II it presents the major ECoS algorithms, surveys techniques that have been developed for extracting rules from ECoS networks and algorithms for optimising the performance of ECoS. Section III compares ECoS to selected other constructive algorithms and Section IV reviews empirical evaluations of ECoS networks. Section V surveys the application of ECoS networks to a wide variety of problems and Section VI suggests areas of future research. Due to the large number of acronyms used in this paper, a list of acronyms is appended.

## II. ECoS Algorithms

This section reviews the extant ECoS algorithms. Firstly, the general ECoS architecture and training algorithm are presented in Subsection II-A. The major members of the ECoS family are then described in Subsections II-B to II-E, where an algorithm is considered to be major if it has significant differences to other ECoS algorithms, or the algorithm has been widely applied. Minor ECoS algorithms are surveyed in Subsection II-F, where an algorithm is considered to be minor if it does not significantly differ from other ECoS algorithms. Methods of optimising ECoS networks are reviewed in Subsection II-G. Algorithms for extracting fuzzy rules from ECoS networks are described in Subsection II-H, and methods for adding additional outputs to EFuNN and SECoS are described in Subsection II-I.

### A. *ECoS Architecture and Learning*

The first ECoS network was EFuNN (Subsection II-B), from which a generalised constructive ANN architecture and training algorithm was derived.

An ECoS network is a multiple neuron layer, constructive artificial neural network. An ECoS network will always have at least one ‘evolving’ neuron layer. This is the constructive layer, the layer that will grow and adapt itself to the incoming data, and is the layer with which the learning algorithm is most concerned. The meaning of the connections leading into this layer, the activation of this layer’s neurons and the forward propagation algorithms of the evolving layer all differ from

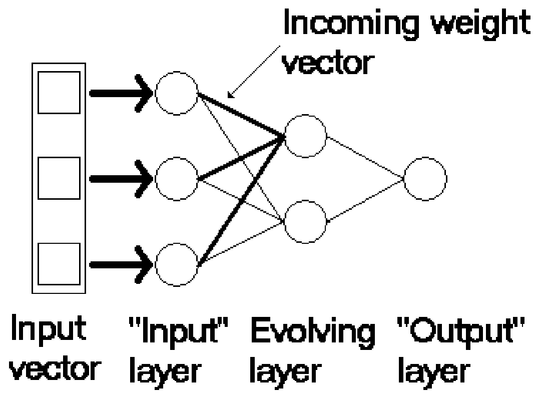

Fig. 1. General ECoS architecture

those of classical connectionist systems such as MLP. For the purposes of this paper, the term ‘input layer’ refers to the neuron layer immediately preceding the evolving layer, while the term ‘output layer’ means the neuron layer immediately following the evolving layer. This is irrespective of whether or not these layers are the actual input or output layers of the network proper. For example, in Figure 1, which shows a generic ECoS structure, the “input layer” could be the input layer proper of the network (as with SECoS networks, Subsection II-C) or it could be a neuron layer that processes the actual input values for presentation to the evolving layer (as with EFuNN networks, Subsection II-B). By the same token, the “output layer” could be the actual output layer of the network (as it is with SECoS) or a neuron layer that further processes the outputs of the evolving layer (as with EFuNN). The connection layers from the input neuron layer to the evolving neuron layer and from the evolving layer to the output neuron layer, are fully connected.

The activation $A_n$ of an evolving layer neuron $n$ is determined by Equation 1.

$$A_n = 1 - D_n \qquad (1)$$

where $D_n$ is the distance between the input vector and the incoming weight vector for that neuron. Since ECoS networks are fully connected, it is possible to measure the distance between the current input vector and the incoming weight vector of each evolving-layer neuron. Although the distance can be measured in any way that is appropriate for the inputs, this distance function must return a value in the range of zero to unity. For this reason, most ECoS algorithms assume that the input data will be normalised, as it is far easier to formulate a distance function that produces output in the desired range if it is normalised to the range zero to unity.

Whereas most ANN propagate the activation of each neuron from one layer to the next, ECoS evolving layers propagate their activation by one of two alternative strategies. The first of these strategies, entitled *OneOfN* propagation, involves only propagating the activation of the most highly activated (“winning”) neuron. The second strategy, *ManyOfN* propagates the activation values of those neurons with an activation value greater than the activation threshold $A_{thr}$.

```
for each input vector $\mathbf{I}$ and its associated desired output vector $\mathbf{O}_d$ do
  Propagate $\mathbf{I}$ through the network
  Find the most activated evolving layer neuron $j$ and its activation $A_j$
  if $A_j < S_{thr}$ then
    Add a neuron
  else
    Find the errors between $\mathbf{O_d}$ and the output activations $A_o$
    if $|\mathbf{O}_d - A_o| > E_{thr}$ then
      Add a neuron
    else
      Update the connections to the winning evolving layer neuron $j$
    end if
  end if
end for
```

Fig. 2. ECoS learning algorithm

The ECoS learning algorithm is based on accommodating new training examples within the evolving layer, by either modifying the weight values of the connections attached to the evolving layer neurons, or by adding a new neuron to that layer. The algorithm employed is described in Figure 2. The addition of neurons to the evolving layer is driven by the novelty of the current training example: if the current example is particularly novel (it is not adequately represented by the existing neurons) then a new neuron will be added. Four parameters are involved in this algorithm: the sensitivity threshold $S_{thr}$, the error threshold $E_{thr}$, and the two learning rates $\eta_1$ and $\eta_2$. The sensitivity threshold and error threshold both control the addition of neurons and when a neuron is added, its incoming connection weight vector is set to the input vector **I**, and its outgoing weight vector is set to the desired output vector $\mathbf{O}_d$. The sensitivity and error thresholds are measures of the novelty of the current example. As can be seen in Figure 2, if the current example causes a low activation (that is, it is novel with respect to the existing neurons) then the sensitivity threshold will cause a neuron to be added that represents that example. If the example does not trigger the addition of a neuron via the sensitivity threshold, but the output generated by that example results in an output error that is greater than the error threshold (that is, it had a novel output), then a neuron will be added.

The weights of the connections from each input $i$ to the winning neuron $j$ are modified according to Equation 2.

$$W_{i,j}(t+1) = W_{i,j}(t) + \eta_1(I_i - W_{i,j}(t)) \tag{2}$$

where:
$W_{i,j}(t)$ is the connection weight from input $i$ to $j$ at time $t$
$I_i$ is the $i$th component of the input vector **I**

The weights from neuron $j$ to output $o$ are modified according to Equation 3.

$$W_{j,o}(t+1) = W_{j,o}(t) + \eta_2 A_j E_o \tag{3}$$

where:
$W_{j,o}(t)$ is the connection weight from $j$ to output $o$ at time $t$
$A_j$ is the activation of $j$
$E_o$ is the signed error at $o$, as measured according to Equation 4.

$$E_o = O_o - A_o \tag{4}$$

where:
$O_o$ is the desired activation value of output $o$
$A_o$ is the actual activation of $o$.

This is essentially the perceptron learning rule. From this it becomes apparent that in [65] and subsequent publications [91], [172] the terms $O_d$ and $A_o$ above were incorrectly reversed.

Since this algorithm deals with each training example as it is seen, the way in which the network learns will be affected by the order in which examples are presented. While this is not a concern for applications that are continuously learning, it is for situations where the network is learning from an existing set of data. Although no literature has come to light that examined the effect of changing the order of training examples, it was considered in designing the optimisation algorithm of [167], as described in Subsection II-G.

## B. *Evolving Fuzzy Neural Networks*

EFuNN was the first ECoS network described [65], [66] and is an application of the ECoS principles to the Fuzzy Neural Network (FuNN) [85]. It is a five neuron layer feed forward network (Figure 3), where each layer performs a specific function. The first neuron layer is the input layer. The second layer is the condition layer. Each neuron in this layer represents a single triangular fuzzy membership function (MF) [182] attached to a particular input, and performs fuzzification of the input values based on that MF. This layer is not fully connected to the input layer, as each condition neuron is connected to a single input neuron, that is, each input neuron is connected to its own subset of condition neurons. The weight of the connection between the condition neuron and its input defines the centre of the condition neuron's MF, where the lower and upper bounds of the MF are defined as the centres of the neighbouring MF. The activation function for a condition neuron $c$, which was based on triangular membership functions in [65], [66], is defined by Equation 5.

$$A_c = \begin{cases} 1 - \frac{I_i - W_{i,c}}{W_{i,c+1} - W_{i,c}}, & W_{i,c} < I_i < W_{i,c+1} \\ 1 - \frac{W_{i,c} - I_i}{W_{i,c} - W_{i,c-1}}, & W_{i,c-1} < I_i < W_{i,c} \\ 1, & W_{i,c} = I_i \\ 0, & \texttt{otherwise} \end{cases} \tag{5}$$

where:
$A_c$ is the activation of the condition node $c$
$W_{i,c}$ is the connection weight defining the centre of the MF attached to condition neuron $c$
$W_{i,c-1}$ is the connection weight defining the centre of the MF to the left of $c$
$W_{i,c+1}$ is the connection weight defining the centre of the

MF to the right of $c$
$\mathbf{I}$ is the input vector
$W_{i,c}$ is the connection weight from input node $i$ to condition node $c$.

The third layer of neurons is the evolving layer, which is also referred to as the rule layer. The distance measure used in this layer, which is the distance between the fuzzified input vector and the weight vector, is described in Equation 6.

$$D_n = \frac{\frac{1}{2}\left(\sum_{i=1}^{c} |I_i - W_{i,n}|\right)}{\sum_{i=1}^{c} W_{i,n}} \tag{6}$$

where:
$c$ is the number of condition neurons (fuzzy inputs)
$\mathbf{I}$ is the fuzzified input vector
$\mathbf{W}$ is the Condition to Rule layer weight matrix.

The fourth layer of neurons is the action layer: neurons in this layer represent fuzzy membership functions attached to the output neurons. This layer is similar to the input layer, in that each action neuron is connected only to the output neuron with which its membership function is associated. Also, the value of the connection weight connecting the action neuron to its output defines the centre of the action neuron's membership function. The activation function of the action layer neurons is a simple saturated linear function. The final neuron layer is the output layer. This calculates crisp output values from the fuzzy output values produced by the action layer neurons. The output layer performs centre of gravity defuzzification over the action layer activations to produce a crisp output. This value is calculated according to Equation 7

$$A_o = \frac{\sum_{a=i}^{m} W_{o,a} A_a}{\sum A_a} \tag{7}$$

where:
$A_o$ is the activation of the output node $o$
$A_a$ is the activation of action node $a$
$m$ is the number of action neurons attached to $o$
$W_{o,a}$ is the value of the connection weight from action node $a$ to output $o$

Figure 3 shows an idealised EFuNN with three input neurons. Two MF are attached to the first input neuron, three to the second, and two to the third. There are three rule neurons and two outputs, with two MF attached to each output.

EFuNN have been applied to a large number of applications, as discussed in Section V, and have also been found to be useful as member of ensembles [118], [172].

## C. Simple Evolving Connectionist Systems

The Simple Evolving Connectionist System (SECoS) was proposed as a minimalist implementation of the ECoS algorithm [161], [165], that is, it is an architecture that has the minimum number of neuron layers necessary to learn data. Alternatively, SECoS can be viewed as a minimalist EFuNN, with the fuzzification and defuzzification components being removed. Lacking the fuzzification and defuzzification mechanisms of EFuNN, the SECoS model was created for several reasons. Firstly, they are intended as a simpler alternative to EFuNN. Since they lack the fuzzification and defuzzification structures of EFuNN, SECoS are much simpler to implement. Having fewer connection matrices and a smaller number of neurons, there is much less processing involved in simulating a SECoS network. They are also much easier to understand and analyse: while EFuNN expands the dimensionality of the input and output spaces with its fuzzy logic elements, SECoS deals with the input and output space 'as is'. Therefore, rather than dealing with a fuzzy problem space, SECoS deals with the problem space directly. Each neuron that is added to the network during training represents a point in the problem space, rather than a point in the expanded fuzzy problem space. Secondly, for some situations, fuzzified inputs are not only unnecessary but harmful to performance, as they lead to an increase in the dimensionality of the input space and hence an increase in the number of evolving layer neurons. Binary data sets are particularly vulnerable to this, as fuzzification does nothing but increase the dimensionality of the input data. By removing the fuzzification and defuzzification capabilities, the adaptation advantages of EFuNN are retained while eliminating the disadvantages of fuzzification, specifically by eliminating the need to select the number and parameters of the input and output membership functions. For most applications, SECoS are able to model the training data with fewer neurons in the evolving layer than an equivalent EFuNN [159].

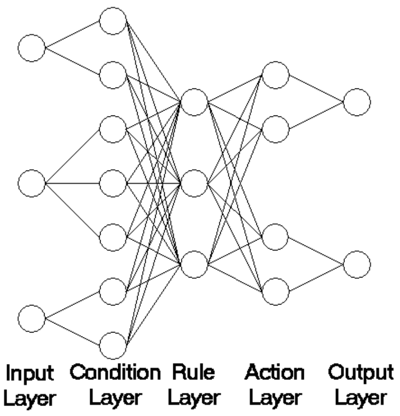

Fig. 3. EFuNN architecture

A SECoS network consists of three layers of neurons: The input layer, with linear transfer functions; The evolving layer; And an output layer with a simple saturated linear activation function. The distance measure used in the evolving layer is the normalised Manhattan distance, as shown in Equation 8:

$$D_n = \frac{\sum_{i=1}^{c} \mid I_i - W_{i,n} \mid}{\sum_{i=1}^{c} \mid I_i + W_{i,n} \mid} \tag{8}$$

where:
$c$ is the number of input neurons in the SECoS
$I$ is the input vector
$W$ is the input to evolving layer weight matrix.

The normalised Euclidean distance can also be used, as defined by Equation 9.

$$D_n = \frac{\sqrt{\sum_{i=1}^{c}(I_i - W_{i,n})^2}}{\sqrt{c}} \tag{9}$$

There are two layers of connections in the SECoS model. The first connects the input neuron layer to the evolving layer. The weight values here represent the coordinates of the point in input space each evolving layer neuron represents. The second layer of connections connects the evolving layer to the output neuron layer. The weights in this layer represent the output values associated with the input examples.

The learning algorithm is the same as that described in Subsection II-A and as used by EFuNN. However in SECoS the input vector $\mathbf{I}$ is the actual crisp input vector, while the desired outputs $\mathbf{O}_d$ vector is the crisp target output vector.

## D. Evolving Self-Organising Maps

The previously presented algorithms, EFuNN and SECoS, both follow the general ECoS architecture and training algorithm outlined in Subsection II-A. While it is still an ECoS network, the Evolving Self-Organising Map (ESOM) [30]–[32] deviates from the general algorithms in order to implement unsupervised learning. As in a conventional Self-Organising Map (SOM) [96] the ESOM has two layers of neurons, the input layer and the map layer, and weighted connections from the input to map neurons. It is the map layer that evolves in this model. Neurons also have weighted connections to their two immediate neighbours. The activation $A_n$ of each map neuron $n$ from the input vector $\mathbf{I}$ is calculated according to Equation 10.

$$A_n = exp\left(\frac{-2||I_i - W_{i,n}||^2}{\epsilon^2}\right) \tag{10}$$

where $\epsilon$ is a radial.

The ESOM learning algorithm is based on the concept of dynamically forming spatial clusters, and is presented in Figure 4.

An optional additional step to the learning algorithm is to prune the weakest connections after a certain number of examples have been presented, and also to prune neurons that have had all of their connections pruned. Since ESOM does not perform vector quantisation as a conventional SOM does, Sammon projection [144] must be used to visualise the clusters that are formed.

```
for each input vector I do
  Find the set of neurons $N$ such that $a_i \in N > S_{thr}$
  if $N = \emptyset$ then
    Insert a neuron $w$
    Connect $w$ to its two nearest neighbours $k$
    Set $N = (w, k)$
  else
    Set $w$ to the most activated neuron
    Update connection weights from input to map neurons
    $h \in N$ according to $\Delta W_{i,h} = \gamma a_h (I_i - W_{i,h})$
  end if
  Set connections from $w$ to $k$ according to $W_{n,k} = \frac{a_n a_k}{max(a_n, a_k)}$
end for
```

where $\gamma$ is the learning rate parameter.

Fig. 4. ESOM learning algorithm

## E. Dynamic Evolving Neural-Fuzzy Inference Systems

The Dynamic Evolving Neural-Fuzzy Inference System (DENFIS) is an application of the ECoS principles to an ANN that implements a Takagi-Sugeno fuzzy inference system [152]. DENFIS was first described in [86], [147] and was more completely described in [87].

DENFIS heavily utilises the so-called Evolving Clustering Method (ECM) [149]. This is based on the concept of dynamically adding and modifying the clusters as new data is presented, where the modification to the clusters affects both the position of the clusters and the size of the cluster, in terms of a radius parameter associated with each cluster that determines the boundaries of that cluster. ECM has only one parameter, which drives the addition of clusters, known as the distance threshold $D_{thr}$. When new clusters are added, their centres are set to equal the example that triggered their creation, and the radius $R$ of a new cluster is initially set to zero. $R$ grows as more vectors are allocated to the cluster. Due to the mechanism by which $R$ is updated, it cannot exceed $D_{thr}$. The ECM algorithm is shown in Figure 5.

When cluster $a$ is updated, its centre is shifted closer to $\mathbf{I}_n$ and its radius $R_a(t+1)$ is set according to Equation 11

$$R_a(t+1) = \frac{S_{i,a}}{2} \tag{11}$$

The new centre of $a$, $C_a(t+1)$ is set so that its distance is on the line between $C_a(t)$ and $\mathbf{I}_n$ at a distance of $R_a(t+1)$.

Although ECM appears to be a useful clustering method in and of itself, its primary function is to support the inference of fuzzy rules from data in DENFIS. This is done in two phases, firstly forming the antecedents, followed by the consequent functions. The antecedents are formulated by finding which combination of input membership functions (MF) activate the most highly for the centre of the cluster, that is, the values represented by the cluster centre are fuzzified by the input MF set and the winning, most highly activated, MF are taken as the antecedents for that rule. This is very similar in concept to the way in which fuzzy rules are extracted from EFuNN and SECoS (Subsections II-H1 and II-H2). The consequent

```
Create the first cluster centre $C_0$ from the first example $\mathbf{I}_0$
for each subsequent vector $\mathbf{I}_n$ do
  Find the minimum distance $D_{min}$ between $\mathbf{I}_n$ and each cluster centre $C_n$
  if $D_{min}$ is less than any cluster radius then
    Add $\mathbf{I}_n$ to the nearest cluster
  else
    Find the cluster $a$ with minimum value of $S_{i,j}$, where $S_{i,j} = D_{i,j} + R_i$, $D_{i,j}$ is the distance between the cluster centre and vector $j$, and $R_i$ is the radius of cluster $i$
    if $S_{i,a} > 2D_{thr}$ then
      Create a new cluster
    else
      Update $a$
    end if
  end if
end for
```

Fig. 5. ECM algorithm

functions are then found using a Least Means Estimation process over the examples within the cluster. Thus, each cluster is used as the basis of a single rule. Clustering and reformulation of the rules is performed whenever a new training example is presented to the network. For any input vector **I** the output of the DENFIS in calculated as the combined output of the most strongly activated $m$ rules. There is no adjustment of the MF during training.

### *F. Other ECoS Algorithms*

The previous sections have reviewed the major ECoS algorithms, that is, those algorithms that are significantly different to one another, or that are more widely used. This subsection briefly reviews ECoS algorithms that are less frequently found in the literature.

Nominal-scale Evolving Connectionist Systems (NECoS) were introduced in [164]. These extend the ECoS algorithm to allow it to deal with nominal-scale data [151]. In a NECoS network, the connections from the input to evolving layer represent nominal-scale class labels, and the distance measure is a simple similarity measure. The learning algorithm was modified so that the labels would change during training to reflect the more common classes seen for each input variable.

Temporal extensions to ECoS, based on the Jordan-Elman Simple Recurrent Network (SRN) [34], [102], have also been proposed [90]. These extensions add a second evolving layer (analogous to the context layer in SRN) to the ECoS, which is solely connected to the main evolving layer. Connections between the context layer and the evolving layer were modified by Hebbian learning.

The Evolving Clustering Method for Classification (ECMC) was described in [148]. ECMC is ECM with class labels, that is, each cluster has a class label associated with it. Unknown examples are classified according to which cluster they are assigned to, that is, they are classified according to which class prototype they are closest to.

The Evolving Classifier Function (ECF) [77] uses a similar algorithm to ECMC. Parallels to EFuNN and SECoS are obvious and expected, although there are still differences between the ECMC and SECoS learning algorithms. The addition of fuzzy membership functions to ECM yielded the Evolving Fuzzy Clustering Method (EFCM ) [140].

The Evolving Fuzzy Inference System (EFIS) [133] is a derivative of DENFIS that includes attributes of the Hybrid Fuzzy Inference System HyFIS [95], and utilises an enhanced version of ECM called ECMm (although the origins of the acronym ECMm are not clearly laid out). Another algorithm derived from DENFIS is the Dynamic Evolving Computation System (DECS) which was proposed in [23]. DECS is quite similar to DENFIS, but includes a genetic algorithm to evolve and tune the rules.

A Weighted EFuNN (WEFuNN) was reported in [21]. In this model, each input has a weighting factor associated with it, and a weighted Euclidean distance function is used in the evolving layer. An exponential, as opposed to saturated linear, transfer function was used for the evolving layer neurons and the $k$-nearest evolving layer neurons were allowed to propagate their activations to the action layer (see the $m$-of-$n$ activation strategy in Subsection II-A). When evaluated over a case study problem of laying out printed circuit boards, WEFuNN was more accurate than EFuNN.

### *G. ECoS Optimisation Algorithms*

Optimisation of ECoS is taken to mean the creation of an ECoS network that fulfils the following criteria. The network should:

- Exhibit good memorisation of the training data
- Exhibit good generalisation to data it has not previously experienced
- Be parsimonious, that is, be of the smallest size that can fulfil the previous two criteria.

ECoS optimisation algorithms fall into two broad groups. The first is based on combining, or aggregating, evolving layer neurons. The second is based on evolutionary algorithms.

*1) Neuron Aggregation:* Evolving layer neuron aggregation is the process of combining several adjacent neurons into one neuron that represents all of the previous exemplars for that spatial region. During the aggregation process, the distance between the incoming and outgoing weight vectors of two neurons is calculated. If the distances are below specified thresholds, the two neurons are either aggregated together, or added to a set of neurons that are all aggregated into one [159], [165]. The rationale behind aggregation is to reduce the size of the evolving layer of the ECoS, while retaining the knowledge stored within the connections to each neuron. Connection weights of the resulting neuron are set to the arithmetic mean of the weights of the aggregated neurons. Two kinds of aggregation have been developed: online, which takes place during training; and offline, which takes place when training over a data set is complete.

Online aggregation is carried out when neuron connection weights are modified. After the weight changes have been applied, the incoming and outgoing distances between the modified neuron and its immediate neighbours are measured. If both distances are below the aggregation thresholds, then the neurons are aggregated together. There is no need to measure the distance between any other neurons, as only one neuron at a time is ever modified under the canonical ECoS training algorithm. Note that this algorithm assumes that new neurons are allocated spatially, that is, new neurons are inserted next to the existing neurons they are closest to in space.

Online aggregation has the advantage of modifying the network as training is under way: there is no need to halt training at any point to perform a global (offline) aggregation, and there is no need to examine every neuron in the evolving layer as there is with offline aggregation.

The offline aggregation strategy exhaustively compares each neuron to every other neuron, which requires $\frac{1}{2}n(n-1)$ comparisons for $n$ neurons. This strategy is very thorough: all neurons that are close together will be aggregated together, no matter where they are in the evolving layer. Offline aggregation requires that training be halted before optimisation can be carried out. However, experiments have shown [159] that it is able to reduce the size of the target network more effectively than online aggregation.

*2) Evolutionary Methods:* The majority of the optimisation methods developed thus far for ECoS involve some form of evolutionary computation (EC) [37]. Types of EC used include genetic algorithms (GA) [27], [55], [58], [124] and evolution strategies (ES) [15], [57].

Although EA have been used in many ways to optimise ANN (see for example the excellent review by Yao [181]) the applications to ECoS have been largely limited to the optimisation of the training parameters of the ECoS. This optimisation is done in either an on-line manner (that is, as training was under way) or in an off-line or batch mode (where the parameters were optimised with respect to a specific data set).

Evolution strategies were used to optimise the parameters on-line in both [166] and [20]. While [166] used a simple (1+1) ES to optimise all four basic training parameters for each training example, [20] used a $(\mu, \lambda)$ ES to optimise the learning rates only, with respect to a sliding window of data where the window was moved though the training data stream. A similar windowing-of-data approach was used in [89], although in this case a GA was used instead of an ES. The GA optimised all learning parameters and the fitness was evaluated as a function of the error over the training data window.

Genetic algorithms were used for off-line optimisation of both training parameters and the order of training examples in [167], [159]. The justification for optimising the order of the examples was that the overall size and performance of the ECoS network is determined by the values of the training parameters and the order in which examples are presented. The fitness function in this case was based on minimising both the error over the provided data set, and the overall change in the size of the evolving layer of the ECoS. A similar approach was taken in [119], although the order of the training examples was not optimised. A GA was also used for off-line optimisation of the training parameters in [88], where the fitness function was based entirely on the error over the training data set. A co-evolutionary GA was used to evolve ensembles of EFuNN in [122], where data was first clustered, and each cluster split into a training and a testing data set. The GA optimised an individual EFuNN over each training and testing set, where the optimisation was with respect to the accuracy and size of the network. The overall result was that the ensemble performed better than a single EFuNN trained over the entire data set. A clustering method similar to ECM was used in [120], [121], where the same GA-based ensemble approach was used.

An unusual approach to off-line optimisation was reported in [14]. In this work, a GA was used to optimise the fuzzy rules and membership functions that were extracted from a trained EFuNN. The modified rules and MF were then re-inserted into the EFuNN for testing. Thus, the GA was really optimising a fuzzy system, although the close-coupling between EFuNN and their associated rules and MF made it an useful approach.

## *H. Knowledge Discovery with ECoS Networks*

Rule extraction from ANN is the process of formulating, from a trained artificial neural network (ANN), a set of symbolic rules that mimic the behaviour of the ANN [13], [63], [125]. Six motivations for extracting rules from ANN are given in [13]: Provision of a user explanation capability, that is, elucidating what the ANN has learned; Extension of ANN systems to safety-critical problem domains, by increasing confidence in the ANN via explanation; Software verification and debugging of ANN components in software systems, by elucidating the internal state of the ANN; Improving the generalisation of ANN solutions, by providing a means to predict where the ANN may fail (via examination of the rules); Data exploration and the induction of scientific theories, by extracting rules from an ANN trained over a data set for which the processes are unknown; Knowledge acquisition for symbolic AI systems, by providing a method of automatically acquiring rules.

Since ECoS learn by partitioning the input space into regions, and fuzzy rules can be visualised as methods of associating regions of input space with consequents, the fit between the two models is quite natural [75], [78]. Rule extraction is simply a matter of mapping the two sets of regions together and by so doing finding the antecedents and consequents.

*1) Fuzzy Rule Extraction from EFuNN:* The Rule Extraction from EFuNN (RE-EFuNN) algorithm for extracting Zadeh-Mamdani fuzzy rules [113] as described in [75], [91], is presented in Figure 6.

*2) Fuzzy Rule Extraction from SECoS:* Although SECoS do not have fuzzy logic elements within their structure, it is possible to extract fuzzy rules from them using external MF [162]. The rationale behind this approach was that there is no practical difference between the fuzzy exemplars in an EFuNN, where those exemplars have been fuzzified by the EFuNN internal MF, and using external MF to fuzzify the crisp exemplars stored within a SECoS. The SECoS Fuzzy Rule Extraction algorithm (SECoS-FRE) for extracting Zadeh-Mamdani fuzzy rules from trained SECoS networks is as in Figure 7.

```
for each evolving layer neuron $h$ do
  Create a new rule $r$
  for each input neuron $i$ do
    Find the condition neuron $c$ with the largest weight $W_{c,h}$
    Add an antecedent to $r$ of the form "$i$ is $c$ $W_{c,h}$" where $W_{c,h}$ is the confidence factor for that antecedent
  end for
  for each output neuron $o$ do
    Find the action neuron $a$ with the largest weight $W_{h,a}$
    Add a consequent to $r$ of the form "$o$ is $a$ $W_{h,a}$" where $W_{h,a}$ is the confidence factor for that consequent
  end for
end for
```

Fig. 6. EFuNN fuzzy rule extraction algorithm

```
for each evolving layer neuron $h$ do
  Create a new rule $r$
  for each input neuron $i$ do
    Find the MF $\mu$ associated with $i$ that activates the most strongly for $W_{i,h}$
    Add an antecedent to $r$ of the form "$i$ is $\mu$ $\mu(W_{i,h})$", where $\mu(W_{i,h})$ is the confidence factor for the antecedent
  end for
  for each output neuron $o$ do
    Find the MF $\mu$ associated with $o$ that activates the most strongly for $W_{h,o}$
    Add a consequent to $r$ of the form "$o$ is $\mu$ $\mu(W_{h,o})$", where $\mu(W_{h,o})$ is the confidence factor for the consequent
  end for
end for
```

Fig. 7. SECoS fuzzy rule extraction algorithm

Functionally, this algorithm is equivalent to the RE-EFuNN algorithm. The RE-EFuNN algorithm chooses antecedent MF based on the highest magnitude weights from the condition to rule neurons, which are really crisp exemplar values that have been fuzzified by the EFuNNs internal MF. The SECoS-FRE algorithm chooses antecedent MF based on the fuzzified values of the weights, which while representing crisp exemplars, are fuzzified using the provided external MF.

The advantage of this algorithm is that, since the membership functions are not an integral part of the network, the number of MF, their type and their parameters can all be optimised before the rule extraction process is carried out. If the rules extracted with a particular set of MF are not optimal, then the MF can be changed and fresh rules generated, without altering the SECoS. This fact was exploited in [163], where evolutionary programming (EP) [37] was used to optimise sets of MF used to extract rules from SECoS.

### *I. Output Space Expansion in ECoS*

ECoS networks are intended to be used in an online, lifelong learning situation. In such situations, it is entirely possible that new target classes will be introduced that must be handled by the existing system. In these cases, there are three possible solutions. Firstly, the existing system can be thrown away and a new network created from scratch. This is not satisfactory, for two reasons: Firstly, the amount of time required may be significant. Secondly, the data that has been seen and must be accommodated by the existing network may not be available for retraining.

The second option is to retain the existing network and create a new network specifically to handle the new class. This avoids the problem of training time, but the problem of missing data remains: if the new network is to handle the new class, then it must be trained on negative examples as well as positive examples.

The third option is to modify the existing network to accommodate the new class. This has the advantage of only requiring additional training on the new class, obviating both the time and data availability problems of the previous two options. For conventional ANN such as MLP this is a very difficult thing to do, but ECoS are inherently suited to this problem; whereas neurons in MLP learn with respect to the entire input space (global learning), neurons in ECoS learn only with respect to a small patch of the problem space (local learning).

The method of adding new outputs to ECoS networks is referred to as output space expansion, because each class added increases the dimensionality of the output space. Algorithms for adding output classes have been developed for both EFuNN and SECoS.

*1) EFuNN Output Expansion:* Expansion of the output space of EFuNN was introduced in [53]. Addition of a new output neuron effects the output neuron layer, the action layer and the rule to action layer connections. When a new output and its action neurons are inserted into the EFuNN, the connections from the existing rule neurons to the new action neurons are set to the fuzzified value of the crisp output zero, using the fuzzy membership functions defined for the new action neurons. This has the effect of making all existing rule neurons represent negative examples for the new output, that is, if any of the existing rule neurons fire, then the new output will be inactive by default. The network is then further trained on examples of the new class, allowing new rule neurons to be constructed to represent the class.

*2) SECoS Output Expansion:* As befits the simplicity of the SECoS model, the algorithm for the addition of new output classes to a SECoS network is also simple. A new output neuron is inserted into the network, and the connection weights from the evolving layer to the new output are set to zero. This again has the effect of making all existing examples negative by default. This approach was used in [52] to expand the vocabulary of a spoken word recognition system.

## III. Similarities of ECoS to Other Constructive ANN

There are many other constructive neural network algorithms apart from ECoS in existence [103]. While older constructive algorithms such as Cascade Correlation [35], Tiling

[115] or Upstart [40] have very little similarity to ECoS, a few algorithms have sufficient features in common that some comments on their similarities and differences are informative. This comparison is necessarily qualitative, as there seems to be no published empirical results comparing ECoS networks to any of the algorithms discussed here.

The Resource Allocating Network (RAN) [137] has several common elements with ECoS. The addition of new neurons in both RAN and ECoS are based upon the novelty of each training example and the network error over each training example and the neurons that are added will themselves represent these examples. In cases where a neuron is not added, then the parameters (connection weights or neuron parameters) are adjusted, in such a way as to optimise performance of the network over the current training example. Finally, while RAN and ECoS both use a distance-based function to calculate the activation values of the growing neuron layer, the activation of the output neurons in RAN is based on a more conventional multiply and sum operation. Differences between the algorithms are principally related to the complexity of the algorithm and its intended means of application. Firstly, RAN uses Gaussian functions to explicitly represent a region of input space, where the region is defined by the parameters of Gaussian functions in the growing layer of the network. Conversely, each neuron in the evolving layer of an ECoS network defines a point in input space, where the point is defined by the connection weight vector of that neuron. ECoS neurons do define regions in the input space, but they do so implicitly rather than explicitly. Secondly, RAN performs an exponential post-processing on the output values of the inputs, and has a "bias" function attached to the output layer, which is adjusted to perform the function mapping. RAN is therefore a more complex system, as it has more parameters to adjust and requires more complex calculations. Finally, ECoS training is based on the idea of a one-pass, continuous, life-long learning algorithm, whereas RAN is not. Specifically, RAN has a "resolution" parameter that determines how finely the RAN matches the function being learned, which decays as learning progresses. This means that RAN is unlikely to be useful in life-long learning applications.

As with RAN, at first glance the Zero Instruction Set Computer ZISC [145] looks very similar to ECoS. Both activate their neurons based on the distance between input vectors and neurons and both divide the input space into regions. There, however, the similarities end. Whereas an ECoS network will always have one neuron that activates for any particular example, with ZISC it is possible that no neuron at all will activate for an example. Alternatively, several neurons may activate in response to the same example. This will require some form of conflict resolution strategy, but at present this situation is dealt with by simply labelling the example as unidentifiable. The final difference between ECoS and ZISC is that ZISC is intended for classification purposes only.

The ECoS algorithm, and SECoS in particular, is most similar to the Grow and Learn (GAL) constructive algorithm [11]. Both use a distance based activation function, and deal with the input data one example at a time. The addition of neurons in both algorithms is driven by the error of the network over each training example, and when a neuron is added, its incoming connection weights are set to the input vector of the current training example. The "fine-tuning" of incoming connection weights in GAL is identical to the learning rule used in the first layer of adjustable connections in EFuNN and SECoS, where the intention of both is to modify the exemplar represented by the neuron into a prototype that represents several examples in a cluster. Interestingly, it is explicitly stated that "a large number of iterations will be necessary" [11, pg407] , while ECoS is touted as a one-pass algorithm. The differences between the two are quite informative, however. Firstly, GAL was designed for classification applications only [103], even though extensions were suggested [11] that would allow it to learn function approximation problems. Because of this restriction, the connection weights in the hidden to output connection layer are used only as class labels. This means that there is no learning in the second layer, as a class either exists, or not. Although there is an equivalent to the error threshold parameter from ECoS training, there is no equivalent to the sensitivity threshold parameter. This may be because of the restriction to applications to classification.

A comparison of ESOM and the Growing Self-Organising Map (GSOM) [10] is also informative. Firstly, GSOM performs vector quantisation, as the original SOM does, whereas ESOM does not. This obviates the need to perform a Sammon projection on the trained network when visualisation is required. Secondly, the GSOM is generally initialised with several neurons (usually four) before the start of training, whereas ESOM has no neurons initially. The criteria for adding neurons is also different: ESOM adds a neuron immediately if the error is unacceptable, while GSOM accumulates error over multiple passes over the training set. Finally, GSOM adds neurons only at the edges of the map, while ESOM can add neurons to any part of the output map.

A comparison of ECoS with the Growing Cell Structure (GCS) Network algorithm [41], which has a large number of similarities with ECoS. Both algorithms are a winner-take-all kind of network, where the activation of neurons is based on the distance between the neuron and the current input vector. GCS networks partition the input space into Voronoi regions, as ECoS do [159], and the winning neuron during training is adjusted to be spatially nearer to the current example. Both algorithms allow for continuous, life long learning. The differences between the two algorithms are quite significant, however. Firstly, neurons in a GCS are connected together by 'edges'. Signal counters are attached to each neuron, and these counters are used to measure the performance and importance of each neuron. Rather than adding neurons when an example requires them, neurons are added after a set number of examples has been presented. Also, new neurons do not represent training examples: the connections of new neurons are set to the means of the two parent neurons, and the neurons are inserted with the goal of optimising the partitioning of the input space, rather than optimising the representation of the input data. Although both ECoS and GCS adjust the weights of the winning neuron, GCS will adjust the weights of the winners neighbours also. This means that GCS is not a local learning algorithm, while ECoS is purely local. GCS are also more computationally complex than ECoS, as a large number of calculations must be made at each step, such as updating and tracking signal counters and calculating the local resources of each neuron.

The Growing-And-Pruning Radial Basis Function (GAP)-RBF network [61] also has some similarities with ECoS. (GAP)-RBF adds and prunes neurons based on their 'significance' to the network, where significance is calculated by an approximation of the effect of the neuron over several training examples. Neurons are only added if their significance exceeds a threshold value, whereas existing neurons are pruned if their significance is less than that threshold. The addition of neurons is thus similar to the way in which the error threshold drives neuron addition to ECoS: neurons are added to ECoS if they will reduce the error of the network. Also, in those cases where a neuron is not added to (GAP)-RBF, the network parameters are adjusted, analogous to the way in which the connection weights in ECoS are changed. The authors of [61] also used a piecewise linear approximation of the Gaussian functions that are used in RBF networks, which substantially reduced the computational complexity of the algorithm. The algorithm is still more complex than ECoS, however, and the use of radial basis functions sets it apart from ECoS, which uses simple distance-based functions. Finally, the significance of neurons in (GAP)-RBF is calculated over several examples, whereas ECoS is a purely local-learning algorithm that considers only one example at a time.

While there are many more constructive algorithms in existence, the algorithms described above are the ones most similar to ECoS. An important future step is the performance of a rigorous, quantitative comparison between these algorithms and the appropriate ECoS algorithms.

## IV. Empirical Evaluations of ECoS over Benchmark Data Sets

Although, there is a relatively large number of publications in existence that evaluate ECoS networks over a wide range of applications (see Section V), only a few have presented results over well-established benchmark data sets. These data sets include: iris classification [36]; Box-Jenkins Gas Furnace [19] time-series data set; Mackey-Glass chaotic time-series function [24]; and the waste water flow problem [92].

The authors of [5] reviewed a variety of algorithms over the Mackey-Glass data set and found that EFuNN, while not the most accurate, was still fairly accurate and was the fastest of the algorithms they examined.

The model "nonlinear autoregressive moving average with exogenous inputs" (NARMAX) [46] was compared with DENFIS over Mackey-Glass time-series. NARMAX was found to be more accurate.

In [134] EFuNN was compared with the proposed algorithm Adaptive Resource Allocating Neural Fuzzy Inference System (ARANFIS) over iris classification and Mackey-Glass time series. ARANFIS was found to be more accurate.

In [1] and [2] an algorithm hybridising evolutionary and local learning was compared to EFuNN over the gas furnace, Mackey-Glass and waste water flow data sets. This algorithm Meta Learning Evolutionary Artificial Neural Network (MLEANN) [2] was shown to be more accurate than EFuNN, although EFuNN was faster.

The work reported in [127], [128] compared EFuNN with the model ANFIS [62] over a handwriting recognition problem, and found that EFuNN was faster to train, more accurate and better at adapting to new data. However, it also found that EFuNN was slower to recall and needed more memory, due to its larger network size.

A review of methods of evolving Takagi-Sugeno fuzzy rules was presented in [138], [139]. The authors compared EFuNN and DENFIS over the Mackey-Glass function. While DENFIS had fewer rules and was more accurate than EFuNN, neither DENFIS nor EFuNN was the most accurate model evaluated.

A more rigorous evaluation of ECoS networks over iris, gas furnace, and Mackey-Glass data sets was carried out in [159]. In these experiments, ten-fold cross validation was used, where the data sets were split into two training sets and a single testing set. The second data set was used to evaluate the ability of the tested networks to adapt to new data after training over the first training set had finished. EFuNN networks were compared to backpropagation-trained FuNN, and SECoS were compared to backpropagation-trained MLP. In all cases, the ability of the networks to learn, generalise and adapt were evaluated.

The results showed that for each of the data sets, SECoS and EFuNN were able to learn and generalise with an accuracy that was not significantly different to either MLP or FuNN. Furthermore, while MLP and FuNN uniformly exhibited significant levels of forgetting after further training on the second training set there were no significant levels of forgetting exhibited by either SECoS or EFuNN. All statistical tests of significance were done to the 99% level of confidence.

## V. ECoS Applications

### A. Speech Processing

Automatic speech recognition (ASR) [111] is a challenging problem and ANN have been fruitfully applied to it many times in the past [12], [18], [39], [54], [56], [59], [98], [104], [107], [110], [136], [141], [143], [146], [153]–[155], [157], [158]. ECoS networks are well suited to applications in ASR [94], as they can more effectively adapt to changes in pronunciation and speakers (which is the major source of variation in ASR) than other ANN.

In [69] EFuNNs were applied to spoken phoneme recognition. EFuNNs were again applied to this problem in [160], and were shown to be both accurate and highly adaptable. The efficiency of SECoS networks in this application area was demonstrated in [165], where a comparison with MLP showed that they were faster training, more adaptable and more accurate than MLP.

Speech recognition via whole word recognition has also been done using ECoS networks. EFuNNs were used in [53], while SECoS were adopted in [52], for speech-control of a robot, and in [50], for control of a home-automation system. SECoS were again used to recognise whole-words in [51].

The goal of this paper was to test a speech de-noising method in an in-car environment. A noise-cancellation approach was also demonstrated in [83], where the speech recognition was done by EFuNN. In each case, ECoS networks were able to accurately recognise the words, and the adaptation capability of ECoS, through further training and output space expansion, made them far more useful for this task than alternative ANN.

Finally, in [105] ECM was used to cluster English and Māori words together, based on their acoustic properties. These "bilingual acoustic clusters" were intended to highlight similarities and differences between the two languages.

## B. Bio-Medical Applications of ECoS

Bioinformatics and medicine are strongly related areas where ANN have been widely applied [16], [25], [178]. ECoS networks have been applied to several different problems in this area [79], [116].

Prediction of RNA initiation sites, that is, sites on RNA strands where protein transcription beings, was accomplished in [42]. Comparisons between EFuNN and MLP were carried out, and EFuNN was found to be able to detect unique RNA patterns that other methods could not. EFuNN was used to model gene regulatory networks in [80]. The data for the model were gene expression levels captured from micro-arrays. Useful rules describing the development of the cells through time were discovered.

Using EFuNN, the work in [43] identified cancer tissues from gene expression in micro arrays and used rule extraction to identify cancer genes. No comparison with other algorithms was done, but this was not an issue because knowledge discovery via rule extraction was the goal of the work. Useful rules were extracted from the trained EFuNN.

Clinical and micro array genetic data was combined using EFuNN in [44], [45] to generate patient prognoses, where the outcomes were either that the patient would die, or the patient would recover. A multi-modular approach was used, where clinical data was modelled by one module and genetic data by another. The reported accuracies of EFuNN were higher than those reported by earlier researchers (87.5% as opposed to 77.6%).

Another application of EFuNN was presented in [106]. In this paper, an EFuNN was used to classify the stimulus received by a subject based on the electrical activity of the brain, as read by an EEG. Although the purpose of the paper was to demonstrate an Independent Component Analysis (ICA) based approach to denoising the EEG data, EFuNN was found to be quite accurate.

The work reported in [114] modelled kidney function from blood levels of the metabolic by-product creatinine. The motivation of this work was that being able to accurately model the function of a patient's kidneys allows the commencement of dialysis treatment to be optimally timed, which improves the patient's long-term prognosis. In this application, DENFIS was found to be more accurate than the algebraic formulae that had been previously used.

The relationship between the setting of a ventilator machine used to support patient breathing and patient blood oxygen levels was investigated using several fuzzy neural network models in [112], including EFuNN and DENFIS. When the results of EFuNN and DENFIS were compared with other fuzzy neural models such as ANFIS, it was shown that while both EFuNN and DENFIS had low errors, they were not the lowest reported in the study (a method based on merging fuzzy sets according to their Hebbian importance, followed by tuning fuzzy membership functions using the Least-Mean-Squares algorithm, yielded the greatest accuracy). The number of rules produced by EFuNN, however, was one of the largest.

## C. Image Processing

Although ECoS networks in general could be expected to lend themselves well to image processing problems, especially problems involving image recognition, EFuNN has so far been used to the exclusion of all other ECoS algorithms in this area.

The problem investigated in [84], [93] was to classify vegetation and ground cover images taken from orbit by SPOT satellites: EFuNN was compared to FuNN in this work. It was found that EFuNN was more accurate and faster training than FuNN for this problem.

Classifying motion vector patterns, or the changes from one frame to the next in an MPEG video stream was the application in [100]. The task was to classify a frame from a compressed video stream as one of six classes: static; panning; zooming; object motion; tracking; and dissolve. This work compared the performance of EFuNN against the LVQ algorithm [97], as well as analysing the effect of varying the number of membership functions on performance.

The performance of MLP and EFuNN were compared in [150], where the application was the classification of images of textures. EFuNN was found to be more accurate and to have a much lower computational load than MLP. The lower computational load presumably came about from EFuNNs faster training algorithm.

Horticultural applications were the subject of [33], [174], [175], [177]. The problem dealt with here was classification of pest damage on apple tree leaves. This involved presenting full colour images to an EFuNN that then had to identify which of three insect pests - Codling Moth, Leafroller and Appleleaf Curling Midge - had caused the damage in the image. EFuNN were found to be more accurate at classifying images than k-means, MLP and SVM. The ability to extract fuzzy rules was also considered to be an advantage in this application area

EFuNN were used to classify images of handwritten digits in [129], where an extensive exploration of the parameters of the EFuNN and the problem itself was carried out. Accuracies ranged from 94-99%, showing how effective EFuNN was for this problem.

## D. Applications in Information Technology

With the exception of [180], which dealt with robot signal control, applications of ECoS networks in information technology (IT) have focussed entirely on network management applications. EFuNN was used in [29] predicting the availability of resources in a heterogeneous network for Voice over IP (VoIP) applications, while ESOM was used in the same paper to assist in visualising the network quality. A comparison of the accuracy of the EFuNN-based approach with a MLP-based approach found that while MLP generalised better, EFuNN adapted much faster to the changing dynamics of a real-life network. EFuNN was again used in [60], where the application was to optimise a Bluetooth routing protocol. It was found that EFuNN reduced the number of useless packets in the piconet and improved the reply time between devices within the piconet. EFuNN was used to predict places to hand-off service in a Multi-Protocol Label Switching (MPLS) network in [49], where EFuNN was found to give superior network quality.

The remaining IT applications of ECoS were in network security. In [3], [4] EFuNN were used for network intrusion detection, where they achieved a detection accuracy of 100%. Incoming network packets were analysed by EFuNN in [22] to determine whether or not the packets represented an attack. A comparison with MLP showed that the rapid adaptation capability of EFuNN made it the better choice for this application. The work reported in [108], [109] used ESOM to detect anomalous network activity to attempt to detect network intrusions, and [132] used ECM and EFIS, a hybrid of HyFIS [95] and DENFIS to profile network traffic to detect network anomalies. Again, the adaptivity of ECoS networks were a major advantage. Finally, [179] used an improved ESOM algorithm to cluster together multiple network intrusion alerts, thus reducing the number of alerts the operator had to deal with. The improvement to ESOM was an improved method of selecting the initial connection weights.

### *E. Economics*

Economic and financial data is a rich application area for connectionist systems [171]. Despite this, relatively few applications in economics have been investigated using ECoS networks.

In [82] the task was to predict the New Zealand stock market index the SE40. Comparison of the results of EFuNN with a FuNN showed that EFuNN was both more accurate and more adaptable than FuNN. EFuNN were also applied to predicting the SE40 in [156], although no comparison with other models was presented. The NASDAQ-100 stock index was modelled in [7], where an EFuNN was used to predict whether or not stocks were going to go up or down. The results were described in the paper as 'promising', although no comparison with other predictive methods was reported.

In [30] an ESOM was used to generate a world macroeconomic map, which clustered together countries of similar economic performance. EFuNN and ESOM were used in [81] to perform a risk analysis of the European monetary union, with the results showing that EFuNN in particular was useful for this application.

### *F. Other Applications*

In this subsection, other applications of ECoS networks are presented.

The work presented in [6] compared MLP and EFuNN for predicting electricity demand in the Australian state of Victoria, finding that EFuNN was more accurate. In follow-up work [17], the predictive power of linear genetic programming was also investigated, but EFuNN was still found to be superior.

EFuNN was used in [8], [9] to predict rainfall in the southern Indian state of Kerala. Comparisons with MLP trained via conjugate gradient and backpropagation showed that EFuNN was more accurate and much faster. A similar application, modelling rainfall in Switzerland, was presented in [173], [176], where the goal was to learn more about the rainfall phenomenon by extracting fuzzy rules from EFuNN.

EFuNN was again used in [28] for on-line evaluation of trainee performance in a virtual reality training task. A comparison with MLP showed that EFuNN was marginally more accurate (98.6% compared with 95.6%).

The problem of detecting abnormal behaviour by the occupants of an intelligent environment was addressed in [142], where they used a SECoS network to integrate the data from 18 different sensors in a single room. While the accuracy of SECoS was equivalent to other methods that were investigated, the fast training speed and adaptability of SECoS made it more useful.

Time-delay neural networks and EFuNN were used to classify the outputs of an artificial nose in [184], where it was shown that EFuNN was faster and more accurate.

A method of identifying people by combining speech and image information, using ECoS, was presented in [185] which used ECF to identify speakers and a modified ZISC [145] to identify faces.

An entry in the RoboCupRescue competition was described in [183], where EFuNN was used to integrate sensors on a rescue robot and determine whether or not a building was on fire. Fuzzy rule extraction was heavily used in this application.

[168]–[170] used SECoS networks to predict the abundance of aphids in Canterbury, New Zealand, from climate variables. The superior training speed of SECoS made it possible to investigate which of the climate variables were more significant to aphid abundance, by iteratively removing variables from the data set and examining the effects on prediction accuracy.

EFuNN and ANFIS were compared over the problem of modelling the social effects of anti-smoking legislation on young smokers in [135]. The results showed that while both algorithms were capable of modelling the problem, and that EFuNN was faster, ANFIS was more accurate.

## VI. Future Directions

### *A. Input Significance*

Many methods have been proposed for determining the importance of each of the input neurons of an MLP [47], [48], [117], [130]. These methods are useful for identifying redundant variables in the data set and for data mining, where the identification of important variables can be useful in determining what the MLP has learned. So far, however, no work has come to light in objectively determining the significance of ECoS inputs. Such a technique would be very useful for ECoS, as apart from the advantages of removing redundant variables, the greater learning speed of ECoS would allow an input significance analysis technique to contribute greatly to data mining with ECoS.

Although a method using incremental principal components analysis was reported in [131] for selecting inputs for ECM / DENFIS, it is unclear if this method is applicable to other ECoS algorithms such as EFuNN. A method for determining input significance by analysing the connection weights in an ECoS network is therefore considered to be an important avenue of future research.

### *B. Optimisation*

A number of methods for optimising ECoS networks using EC have been discussed in this review. All of these methods used single-objective algorithms, whereas multi-objective algorithms [38] may be more suitable, as the goal is to not only improve the performance of the ECoS but also to reduce the size (or rate of growth) of the network. However, EC is in many ways unsatisfactory as an optimisation technique, for a number of reasons. Firstly, EC algorithms are costly in terms of the computing power required. This conflicts with the motivation of ECoS as a fast, on-line algorithm. Secondly, EC-based algorithms optimise the ECoS to a specific set of data, which is unsuitable for life-long learning. While non-evolutionary algorithms exist for automatically adjusting the parameters of back-propagation training [126], such algorithms have not yet been developed for ECoS. Plainly, if such a technique were possible, it would be very advantageous, as the automatic selection of ECoS training parameters, as training were under way, would be a significant advantage, as it would allow the rapid training advantages of ECoS to be retained. It seems likely that a formalisation of ECoS, such as is discussed in the following subsection, would point towards methods of automatically determining at least the two learning rate parameters.

### *C. Formalisation*

Traditional ANN are supported by a large body of theory [26], [99], [101], [123]. This body of theory describes: How the ANN training algorithms behave, given the settings of their training parameters; How the training algorithms allow the network to capture knowledge; And how this knowledge is represented by the ANN.

This theory assists the neural network practitioner in both applying these algorithms and in optimising and extending them. A theoretical basis is also useful in assisting the acceptance of a new algorithm: other researchers are more likely to utilise a new algorithm if its theoretical grounding is known. More importantly, a formalisation would almost certainly point to methods of addressing the previous two issues in this section.

It is for these reasons that a theoretical basis to ECoS is desirable. Any theory, or formalisation, that describes the ECoS algorithm must cover two distinct aspects. Firstly, the behaviour, or state, of the network at any time $t$. Secondly, the way in which the state of the network changes as it trains, which includes the effect each training parameter has on the changes made to the ECoS by the training algorithm.

Kasabov [66], [69], [77] suggests a formalisation based on hyper-spheres, where there is one hyper-sphere in input space and one hyper-sphere in output space for each evolving layer neuron. The radius of the input hyper-sphere is defined by the sensitivity threshold parameter $S_{thr}$, and any example that falls within this hyper-sphere will not cause a new neuron to be added to the network.

This theory, however, does not describe the effects of the other training parameters. Although experimental results [160] show that the parameters have different effects upon the behaviour of an ECoS network, this theory does not tackle parameter optimisation.

While an attempt at a formalisation based on Voronoi regions was made in [159], this was incomplete in that it was restricted to uniformly distributed data and did not suggest any ways in which the training parameters could be optimised. A complete, testable formalisation of ECoS networks is therefore required as a matter of urgency.

## VII. Conclusion

The first ten years of evolving connectionist systems have shown that they are fast and efficient learning algorithms that are able to adapt to new data without forgetting the old. These years have also seen the creation of an exciting variety of algorithms. Each of these algorithms has certain advantages or disadvantages, and areas of application. While it does not seem likely that the basic algorithms will significantly change, it is likely that further refinements and improvements to these algorithms will be developed in the future. The real-world effectiveness of these algorithms has also been shown, by the wide variety of applications to which they have been applied. It is also likely that more applications will be fruitfully addressed with the use of ECoS networks. The next decade is sure to see interesting developments in both algorithms and applications.

## Appendix
### List of Abbreviations and Acronyms

ANFIS – Adaptive Neural Fuzzy Inference System
ANN – Artificial Neural Networks
ARANFIS – Adaptive Resource Allocating Neural Fuzzy Inference System
ASR – Automatic Speech Recognition
DENFIS – Dynamic Evolving Neuro-Fuzzy Inference System
DECS – Dynamic Evolving Computation System
ECF – Evolving Clustering Function
ECM – Evolving Clustering Method
ECMC – Evolving Clustering Method for Classification
ECoS – Evolving Connectionist System
EFCM – Evolving Fuzzy Clustering Method
EFIS – Evolving Fuzzy Inference System
EFuNN – Evolving Fuzzy Neural Network
ES – Evolution Strategy
ESOM – Evolving Self-Organising Map
FuNN – Fuzzy Neural Network
GA – Genetic Algorithm
GAL – Grow and Learn Network
(GAP)-RBF – (Growing and Pruning) Radial Basis Function

GSOM – Growing Self-Organising Map
HyFIS – Hybrid Fuzzy Inference System
ICA – Independent Component Analysis
IIS – Intelligent Information Systems
MF – Membership Function
MLEANN – Meta Learning Evolutionary Artificial Neural Network
MPLS – Multi-Protocol Label Switching
NARMAX – nonlinear autoregressive moving average with exogenous inputs
NECoS – Nominal-scale Evolving Connectionist System
RAN – Resource Allocating Network
RBF – Radial Basis Function
RE-EFuNN – Rule Extraction from EFuNN
SECoS – Simple Evolving Connectionist System
SECoS-FRE – SECoS Fuzzy Rule Extraction
SOM – Self Organising Map
SRN – Simple Recurrent Network
TDNN – Time Delay Neural Network
VoIP – Voice over IP
WEFuNN – Weighted EFuNN
ZISC – Zero Instruction Set Computer

## References

[1] A. Abraham. “Optimization of evolutionary neural networks using hybrid learning algorithms”. In *Proceedings of IJCNN 2002*, pages 2797–2802, 2002.

[2] A. Abraham. “Meta-learning evolutionary artificial neural networks”. *Neurocomputing*, 56:1–38, 2003.

[3] A. Abraham and L. Jain. “Soft computing models for network intrusion detection systems”. In Saman Halgamuge and Lipo Wang, editors, *Soft Computing in Knowledge Discovery: Methods and Applications*, chapter 16. Springer Verlag Germany, 2004.

[4] A. Abraham, R. Jain, and S.Y. Han. “SCIDS: A soft computing intrusion detection system”. In *IWDC 2004, LNCS 3326*, pages 252–257, 2004.

[5] A. Abraham and B. Nath. “Hybrid intelligent systems design: A review of a decade of research”. Technical report, School of Computing and Information Technology, Monash University, Australia, 2000.

[6] A. Abraham and B. Nath. “A neuro-fuzzy approach for modelling electricity demand in victoria”. *Applied Soft Computing*, 1:127138, 2001.

[7] A. Abraham, B.. Nath, and P.K. Mahanti. “Hybrid intelligent systems for stock market analysis”. In *Lecture Notes in Computer Science*, volume 2074, pages 337–345. Springer-Verlag, Berlin Heidelberg New York, 2001.

[8] A. Abraham, N.S. Philip, and J.K. Babu. “Will we have a wet summer? soft computing models for long-term rainfall forecasting”. In *Soft Computing Models for Long-term RainfallForecasting, In Proceedings of 15th European Simulation Conference ESM 2001, Prague, June 2001*, 2001.

[9] A. Abraham, D. Steinberg, and N.S. Philip. “Rainfall forecasting using soft computing models and multivariate adaptive regression splines”. *IEEE SMC Transactions: Special Issue on Fusion of Soft Computing and Hard Computing in Industrial Applications*, 1:1–6, February 2001.

[10] D. Alahakoon, S.K. Halgamuge, and B. Srinivasan. “Dynamic self-organizing maps with controlled growth for knowledge discovery”. *IEEE Transactions on Neural Networks*, 11(3):601–614, 2000.

[11] E. Alpaydin. “GAL: Networks that grow when they learn and shrink when they forget”. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(1):391–414, 1994.

[12] S. Anderson, J. Merrill, and R.T Port. “Dynamic speech categorization with recurrent networks”. In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 398–396. Morgan Kaufmann, 1988.

[13] R. Andrews, J. Diederich, and Alan B. Tickle. “Survey and critique of techniques for extracting rules from trained artificial neural networks”. *Knowledge-Based Systems*, 8(6):373–389, December 1995.

[14] R. Arnott. “Application of a genetic algorithm to optimise EFuNN rule sets”. Technical report, Department of Information Science, University of Otago, New Zealand, 2000.

[15] T. Bäck, F. Hoffmeister, and H-P. Schwefel. “A survey of evolution strategies”. In Richard K Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, 1991.

[16] P. Baldi and S. Brunak. *“Bioinformatics: The Machine Learning Approach”*. MIT Press, 1998.

[17] M. Bhattacharya, A. Abraham, and B. Nath. “A linear genetic programming approach for modeling electricity demand prediction in victoria”. In *Hybrid Information Systems, Advances in Soft Computing*, pages 379–394. Physica Verlag, Springer Verlag, Germany, 2002.

[18] H. Bourland and C.J. Wellekens. “Multiplayer perceptrons and automatic speech recognition”. In *IEEE First Annual Conference on Neural Networks*, volume IV, pages 407–416, San Diego, June 1987.

[19] G. E.P. Box and G. M. Jenkins. *“Time Series Analysis forecasting and control”*. Holden-Day, 1970.

[20] Z. Chan and N. Kasabov. “Evolutionary computation for on-line and off-line parameter tuning of evolving fuzzy neural networks”. *International Journal of Computational Intelligence and Applications*, 4:309–319, 2004.

[21] P-C. Chang, C-H. Liu, C-H. Yeh, and S-H. Chen. “The development of a weighted evolving fuzzy neural network”. In D.-S. Huang, K. Li, and G.W. Irwin, editors, *Proceedings of ICIC 2006*, pages 212–221, 2006.

[22] S. Chavan, N D. Khusbu Shah, S. Mukherjee, A. Abraham, and S. Sanyal. “Adaptive neuro-fuzzy intrusion detection systems”. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC04)*, pages 70–75, 2004.

[23] Y.M. Chen and C-Y. Lin. “Dynamic parameter optimization of evolutionary computation for on-line prediction of time series with changing dynamics”. *Applied Soft Computing*, 7:1170–1176, 2007.

[24] R.S. Crowder. “Predicting the Mackey-Glass timeseries with cascade-correlation learning”. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1990 Connectionist Models Summer School*, pages 117–123, Carnegie Mellon Univ., 1990.

[25] J.A. Cruz and D.S. Wishart. “Applications of machine learning in cancer prediction and prognosis”. *Cancer Informatics*, 2:59–78, 2006.

[26] G. Cybenko. “Approximation by superpositions of sigmoidal function”. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[27] L. Davis, editor. *“Handbook of Genetic Algorithms”*. International Thomson Computer Press, 1996.

[28] R.M. de Moraes and L. dos Santos Machado. “Evaluation system based on EFuNN for on-line training evaluation in virtual reality”. In *CIARP 2005, LNCS 3773*, 2005.

[29] R. del Hoyo-Alonso, P. Pilar Fernández-de Alarcón, J-J. Navamuel-Castillo, N. J. Medrano-Marqués, B. Martin-del Brio, J. Fernández-Navajas, and D. Abadía-Gallego. “Neural networks for QoS network management”. In *IWANN 2007, LNCS 4507*, pages 887–894, 2007.

[30] D. Deng and N. Kasabov. “Evolving self-organizing map and its application in generating a world macroeconomic map”. In N. Kasabov and K. Ko, editors, *Emerging Knowledge Engineering and Connectionist-based Systems (Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop “Future directions for intelligent systems and information sciences”, Dunedin, 22-23 November 1999)*, pages 7–12. University of Otago Press, 1999.

[31] D Deng and N. Kasabov. “ESOM: An algorithm to evolve self-organizing maps from on-line data streams”. In *International Joint Conference on Neural Networks (IJCNN'2000), Como, Italy*, volume 6, pages 3–8, 2000.

[32] D. Deng and N. Kasabov. “On-line pattern analysis by evolving self-organizing maps”. In *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES2001)*, pages 46–51, 2001.

[33] R. Elder. “Image classifying of fruit pest damage”. Technical report, Department of Information Science, University of Otago, Dunedin, 2000.

[34] J. L. Elman. “Finding structure in time”. *Cognitive Science*, 14:179–211, 1990.

[35] S. E. Fahlman and C. Lebiere. “The cascade-correlation learning architecture”. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, pages 524–532. Morgan Kaufman Publishers, 1990.

[36] R.A. Fisher. “The use of multiple measurements in taxonomic problems”. *Annals of Eugenics*, 7:179–188, 1936.

[37] D. B. Fogel. *"Evolutionary Computation : Toward a New Philosophy of Machine Intelligence"*. IEEE, 2nd edition, August 1999.

[38] C.M. Fonseca and P.J. Fleming. "An overview of evolutionary algorithms in multiobjective optimization". *Evolutionary Computation*, 3:1–16, 1995.

[39] Michael A. Franzini. "Learning to recognize spoken words: A study in connectionist speech recognition". In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer*, pages 407–416. Morgan Kaufmann, 1988.

[40] M. Frean. "The upstart algorithm: A method for constructing and training feedforward neural networks". *Neural Computation*, 2(2):198–209, 1990.

[41] B. Fritzke. "A growing neural gas network learns topologies". In G. Tesauro, D. Tourezky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. The MIT Press, 1995.

[42] M. Futschik, M. Schreiber, C.M. Brown, and N. Kasabov. "Comparative studies of neural network models for mRNA analysis". In *ISMB 1999, Heidelberg*, page 43, 1999.

[43] M.E. Futschik, A. Reeve, and N. Kasabov. "Evolving connectionist systems for knowledge discovery from gene expression data of cancer tissue". *Artificial Intelligence in Medicine*, 28:165–189, 2003.

[44] M.E. Futschik, M. Sullivan, A. Reeve, and N. Kasabov. "Modular decision system and information integration for improved disease outcome prediction". In *Proceedings of the ECCB'2003*, pages 13–15, 2003.

[45] M.E. Futschik, M. Sullivan, A. Reeve, and N. Kasabov. "Prediction of clinical behaviour and treatment for cancers". *OMJ Applied Bioinformatics*, 2(3):S53–58, 2003.

[46] Y. Gao and M.J. Er. "NARMAX time series model prediction: feedforward and recurrent fuzzy neural networks approaches". *Fuzzy Sets and Systems*, 150:331–350, 2005.

[47] G. D Garson. "Interpreting neural-network connection weights". *AI Expert*, 6(7):47–51, 1991.

[48] M. Gevrey, I. Dimopoulos, and S. Lek. "Review and comparison of methods to study the contribution of variables in artificial neural network models". *Ecological modelling*, 160:249–264, 2003.

[49] B.S. Ghahfarokhi and N. Movahhedinia. "QoS provisioning by EFuNNs-based handoff planning in cellular MPLS networks". *Computer Communications*, 30:2676–2685, 2007.

[50] A. Ghobakhlou and N. Kasabov. "A methodology for adaptive speech recognition systems and a development environment". In *Proc. of Artificial Neural Networks and Neural Information Processing ICANN/ICONIP 2003 International Conference, Istanbul, Turkey, June 2003*, pages 316–319, 2003.

[51] A. Ghobakhlou and R. Kilgour. "In-vehicle noise and enhanced speech intelligibility". In Nikhil R. Pal, Nikola Kasabov, Rajani K. Mudi, Srimanta Pal, and Swapan K. Parui, editors, *Neural Information Processing: 11th International Conference, ICONIP 2004*, pages 375–380, 2004.

[52] A. Ghobakhlou and R. Seesink. "An interactive multi modal system for mobile robotic control". In *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES2001)*, pages 93–99, 2001.

[53] A. Ghobakhlou, M. Watts, and N. Kasabov. "On-line expansion of output space in evolving fuzzy neural networks". In *Proceedings ICONIP 2000, Taejon, Korea, November, 2000*, volume 2, pages 891–896, 2000.

[54] A. Glaeser. "Modular neural networks for low-complex phoneme recognition". In *Proceedings of ICSLP'98*, pages 1303–1306, 1998.

[55] D. E. Goldberg. *"Genetic Algorithms in Search, Optimisation and Machine Learning"*. Addison-Wesley, 1989.

[56] S.J. Haskey and S. Datta. "A comparative study of OCON and MLP architectures for phoneme recognition". In *Proceedings of ICSLP 98*, 1998.

[57] F. Hoffmeister and T. Bäck. "Genetic algorithms and evolution strategies: Similarities and differences". In H-P Schwefel and R. Manner, editors, *Parallel Problem Solving from Nature*, pages 455–469. Springer Verlag, 1991.

[58] J. H. Holland. *"Adaptation in Natural and Artificial Systems"*. MIT Press, 1975.

[59] T. Homma, L. E. Atlas, and R. J. Marks. "An artificial neural network for spatio-temporal bipolar patterns: Application to phoneme classification". In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 380–387. Morgan Kaufmann, 1988.

[60] C-J. Huang, W-K. Lai, S-Y Hsiao, H-Y Liu, and R-L Luo. "A bluetooth routing protocol using evolving fuzzy neural networks". *International Journal of Wireless Information Networks*, 11(3):131–146, 2004.

[61] G-B. Huang, P. Saratch, and N. Sundararajan. "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks". *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 34(6):2284–2292, 2004.

[62] J-S. R. Jang. "ANFIS: Adaptive-network-based fuzzy inference system". *IEEE Transactions on Systems, Man and Cybernetics*, 23:665–684, May/June 1993.

[63] Y. Jin and B. Sendhoff. "Extracting interpretable fuzzy rules from RBF networks". *Neural Processing Letters*, 17(2):149–164, 2003.

[64] N. Kasabov. "ECOS: Evolving connectionist systems and the ECO learning paradigm". In Shiro Usui and Takashi Omori, editors, *ICONIP'98 Proceedings*, volume 2, pages 1232–1235, 1998.

[65] N. Kasabov. "Evolving fuzzy neural networks - algorithms, applications and biological motivation". In Takeshi Yamakawa and Gen Matsumoto, editors, *Methodologies for the Conception, Design and Application of Soft Computing*, volume 1, pages 271–274. World Scientific, 1998.

[66] N. Kasabov. "The ECOS framework and the ECO learning method for evolving connectionist systems". *Journal of Advanced Computational Intelligence*, 2(6):195–202, 1998.

[67] N. Kasabov. "Evolving connectionist and fuzzy connectionist systems - theory and applications for adaptive, on-line intelligent systems". In N. Kasabov and R.Kozma, editors, *Neuro-Fuzzy Techniques for Intelligent Information Systems*, pages 111–146. Heidelberg, Physica Verlag, 1999.

[68] N. Kasabov. "Evolving connectionist and fuzzy connectionist systems for on-line adaptive decision making and control". In R. Roy, T. Furuhashi, and P.K. Chawdhry, editors, *Advances in Soft Computing - Engineering Design and Manufacturing*. Springer-Verlag, London Limited, 1999.

[69] N. Kasabov. "Evolving connectionist systems: A theory and a case study on adaptive speech recognition". In *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, volume 5, pages 3002–3007, 1999.

[70] N. Kasabov. "Evolving connectionist systems for on-line, knowledge-based learning: Principles and applications". Technical Report TR99/02, Department of Information Science, University of Otago, 1999.

[71] N. Kasabov. "Evolving connectionist systems - a symbiosis of learning and evolution". In *Proceedings of ICONIP'2000, November 14-18, 2000, Taejon, Korea*, pages 676–680, 2000.

[72] N. Kasabov. "Brain-like functions in evolving connectionist systems for on-line, knowledge-based learning". In T. Kitamura, editor, *What should be Computed to Understand and Model Brain Function*, volume 3 of *FLSI Soft Computing Series*, pages 77–113. World Scientific, 2001.

[73] N. Kasabov. "Evolving connectionist systems, the brain and the genes". In *Proceedings of ICONIP 2001, Shanghai, China*, 2001.

[74] N. Kasabov. "Evolving fuzzy neural networks for supervised/unsupervised on-line, knowledge-based learning". *IEEE Transactions of Systems, Man and Cybernetics, Part B Cybernetics*, 31(6):902–918, 2001.

[75] N. Kasabov. "On-line learning, reasoning, rule extraction and aggregation in locally optimised evolving fuzzy neural networks". *Neurocomputing*, 41:25–41, 2001.

[76] N. Kasabov. "Evolving connectionist-based decision support systems". In X.Yu and J.Kacprzyk, editors, *Applied Decision Support with Soft Computing, series: Studies in Fuzziness and Soft Computing*, volume 124, pages 86–98. Springer, 2003.

[77] N. Kasabov. *"Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines"*. Springer, 2003.

[78] N. Kasabov. "Adaptation and interaction in dynamical systems: Modelling and rule discovery through evolving connectionist systems". *Applied Soft Computing*, 6(3):307–322, 2006.

[79] N. Kasabov. "Global, local and personalised modeling and pattern discovery in bioinformatics: An integrated approach". *Pattern Recognition Letters*, 28:673–685, 2007.

[80] N. Kasabov and D. Dimitrov. "A method for gene regulatory network modelling with the use of evolving connectionist systems". In *Proc. of ICONIP2002 - International Conference on Neuro-InformationProcessing, Singapore, November 2002*, pages 596–601. IEEE Press, 2002.

[81] N. Kasabov, L. Erzegovezi, M. Fedrizzi, A. Beber, and D. Deng. "Hybrid intelligent decision support systems and applications for risk

analysis and prediction of evolving economic clusters in europe". In N. Kasabov, editor, *Future directions for intelligent information systems and information sciences*, pages 347–372. Springer Verlag, 2000.

[82] N. Kasabov and M. Fedrizzi. "Fuzzy neural networks and evolving connectionist systems for intelligent decision making". In *Proceedings of the Eighth International Fuzzy Systems Association World Congress, Taiwan, , August 17-20*, pages 30–35, 1999.

[83] N. Kasabov and G. Iliev. "A methodology and a system for adaptive speech recognition in a noisy environment based on adaptive noise cancellation and evolving fuzzy neural networks". In H. Bunke and A. Kandel, editors, *Neuro-Fuzzy Pattern Recognition*, pages 179–203. World Scientific, 2000.

[84] N. Kasabov, S. Israel, and B.J. Woodford. "Methodology and evolving connectionist architecture for image pattern recognition". In Ghosh Pal and Kundu, editors, *Soft Computing and Image Processing*, pages 318–336. Heidelberg, Physica-Verlag (Springer Verlag), 1999.

[85] N. Kasabov, J. Kim, M. Watts, and A. Gray. "FuNN/2 - a fuzzy neural network architecture for adaptive learning and knowledge acquisition in multi-modular distributed environments". *Information Sciences - Applications*, 101(3-4):155–175, 1997.

[86] N. Kasabov and Q. Song. "Dynamic evolving fuzzy neural networks with 'm-out-of-n' activation nodes for on-line adaptive systems". Technical Report TR99/04, Department of Information Science, University of Otago, 1999.

[87] N. Kasabov and Q. Song. "DENFIS: Dynamic evolving neural-fuzzy inference systems". *IEEE Transactions on Fuzzy Systems*, 10(2):144–154, April 2002.

[88] N. Kasabov and Q. Song. "GA-parameter optimisation of evolving connectionist systems for classification and a case study from bioinformatics". In *Proc. of ICONIP2002 - International Conference on Neuro-Information Processing, Singapore, November 2002*, pages 602–605. IEEE Press, 2002.

[89] N. Kasabov, Q. Song, and I. Nishikawa. "Evolutionary computation for dynamic parameter optimisation of evolving connectionist systems for on-line prediction of time series with changing dynamics". In *Proceedings, IJCNN'2003, Portland, Oregon, July 2003*, pages 438–443, 2003.

[90] N. Kasabov and M. Watts. "Spatial-temporal adaptation in evolving fuzzy neural networks for on-line adaptive phoneme recognition". Technical Report TR99/03, Department of Information Science, University of Otago, 1999.

[91] N. Kasabov and B. Woodford. "Rule insertion and rule extraction from evolving fuzzy neural networks: Algorithms and applications for building adaptive, intelligent expert systems". In *IEEE International Fuzzy Systems Conference*, pages 1406–1411, 1999.

[92] Nikola K. Kasabov. *"Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering"*. MIT Press, 1996.

[93] N.K. Kasabov, S.A. Israel, and B.J. Woodford. "The application of hybrid connectionist systems to image classification". In S.K. Pal, A. Ghosh, and M.K. Kundu, editors, *Soft Computing for Image Processing*, pages 318–336. Springer-Verlag Berlin/Heidelberg, 2000.

[94] R.I. Kilgour. *"Evolving Systems for Connectionist-Based Speech Recognition"*. PhD thesis, University of Otago, 2003.

[95] J. Kim and N. Kasabov. "HyFIS: Adaptive neuro-fuzzy system and their application to non-linear dynamical systems". *Neural Networks*, 12(9):1301–1319, 1999.

[96] T. Kohonen. "The self-organizing map". *Proceedings of the IEEE*, 78(9):1464–1479, September 1990.

[97] T. Kohonen. *"Self-Organizing Maps"*. Springer, second edition, 1997.

[98] T. Koizumi, M. Mori, S. Taniguchi, and M. Maruya. "Recurrent neural networks for phoneme recognition". In *Proceedings of ICSLP'96*, volume 1, pages 326–329, 1996.

[99] A.N. Kolmogorov. "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition". *Dokl. Akad. Nauk. USSR*, 114:953–956, 1957. (in Russian).

[100] I. Koprinska and N. Kasabov. "An application of evolving fuzzy neural network for compressed video parsing". In *ICONIP/ANZIIS/ANNES'99 Workshop, Dunedin, New Zealand, November 22-24*, pages 96–102, 1999.

[101] B. Kosko. *"Neural Networks and Expert Systems: A Dynamical Systems Approach to Machine Intelligence"*. Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[102] S. C. Kremer. "On the computational power of Elman-style recurrent networks". *IEEE Transactions on Neural Networks*, 6(4):1000–1004, 1995.

[103] T-Y. Kwok and D-Y. Yeung. "Constructive algorithms for structure learning in feedforward neural networks for regression problems". *IEEE Transactions on Neural Networks*, 8(3):630–645, 1999.

[104] S. Lawrence, A. C. Tsoi, and A. D. Back. "The gamma MLP for speech phoneme recognition". In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 785–791. MIT Press, 1996.

[105] M. Laws, R. Kilgour, and N. Kasabov. "Modelling the emergence of bilingual acoustic clusters: a preliminary case study". *Information Sciences*, 156:85–107, 2003.

[106] C. S. Leichter, A. Cichocki, and N. Kasabov. "Independent component analysis and evolving fuzzy neural networks for the classification of single trial EEG data". In *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES2001)*, pages 100–105, 2001.

[107] H. C. Leung, J. R. Glass, M.I S. Philips, and V. W. Zue. "Phonetic classification and recognition using the multi-layer perceptron". In *Advances in Neural Information Processing*, pages 248–254, 1990.

[108] Y. Liao, V.R. Vemuri, and A. Pasos. "A general framework for adaptive anomaly detection with evolving connectionist systems". In *SIAM International Conference on Data Mining, Lake Buena Vista, FL, April 22-24, 2004*, pages 437–441, 2004.

[109] Y. Liao, V.R. Vemuri, and A. Pasos. "Adaptive anomaly detection with evolving connectionist systems". *Journal of Network and Computer Applications*, 30:60–80, 2007.

[110] R. P. Lippmann. "Review of neural networks for speech recognition". *Neural Computation*, 1:1–38, 1989.

[111] R. P. Lippmann. "Speech recognition by machines and humans". *Speech Communications*, 22:1–15, 1997.

[112] F. Liu, G.S. Ng, C. Quek, and T.F. Loh. "Artificial ventilation modeling using neuro-fuzzy hybrid system". In *2006 International Joint Conference on Neural Networks, Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, July 16-21, 2006*, pages 5166–5171, 2006.

[113] E. Mamdani. "Advances in linguistic synthesis of fuzzy controllers". *International journal of Man-Machine Studies*, 8(6):669–678, 1976.

[114] M.R. Marshall, Q. Song, T.M. Ma, S.G. Macdonell, and N.K. Kasabov. "Evolving connectionist system versus algebraic formulas for prediction of renal function from serum creatinine". *Kidney International*, 67:1944–1954, 2005.

[115] M. Mézard and J.-P. Nadal. "Learning in feedforward layered networks: the tiling algorithm". *Journal of Physics A*, 22:2191–2203, 1989.

[116] M. Middlemiss. "Intelligent information systems for online analysis and modelling of biological data". Master's thesis, Department of Information Science, University of Otago, 2001.

[117] L.K. Milne. "Feature selection with neural networks with contribution measures". In *Proceedings of the Australian Conference on Artificial Intelligence AI'95, Canberra*, 1995.

[118] F. L. Minku and Xin Yao. "On-line negative correlation learning". In *Proceedings of IJCNN 2008*, pages 1375–1382, 2008.

[119] F.L Minku and T.B. Ludermir. "Evolutionary strategies and genetic algorithms for dynamic parameter optimization of evolving fuzzy neural networks". In *Proceedings of CEC2005 Edinburgh, Scotland, September, 2005*, volume 3, pages 1951–1958, 2005.

[120] F.L. Minku and T.B. Ludermir. "EFuNN ensembles construction using a clustering method and a coevolutionary multi-objective genetic algorithm". In *13th International Conference on Neural Information Processing, ICONIP2006 Hong Kong, October 2006*, pages 884–891, 2006.

[121] F.L. Minku and T.B. Ludermir. "EFuNN ensembles construction using CONE with multi-objective GA". In *IX Brazilian Symposium on Artificial Neural Networks, SBRN'2006, Ribeiro Preto, Brazil, October 2006*, pages 48–53, 2006.

[122] F.L. Minku and T.B. Ludermir. "EFuNNs ensembles construction using a clustering method and a coevolutionary genetic algorithm". In *Proceedings of CEC 2006, Vancouver, BC, Canada, July 16-21, 2006*, pages 5548–5555, 2006.

[123] M. L. Minsky and S. A. Papert. *"Perceptrons"*. MIT Press, 1969.

[124] M. Mitchell. *"An Introduction to Genetic Algorithms"*. MIT Press, 1996.

[125] S. Mitra and Y. Hayashi. "Neuro-fuzzy rule generation: Survey in soft computing framework". *IEEE Transactions on Neural Networks*, 11(3):748–768, May 2000.

[126] M. Moreira and E. Fiesler. "Neural networks with adaptive learning rate and momentum terms". Technical Report 95-04, Institut Dalle D'Intelligence Artificielle Perceptive, 1995.

[127] T. Murali, N. Sriskanthan, and G.S. Ng. "Comparative analysis of the two fuzzy neural systems ANFIS and EFuNN for the classification of handwritten digits". In *The Seventh IEEE International Symposium on Consumer Electronics, Sydney, Australia, December 3-5, 2003*, 2003.

[128] G.S. Ng, S. Erdogan, D. Shi, and A. Wahab. "Insight of fuzzy neural systems in the application of handwritten digits classification". *International Journal of Image and Graphics*, 4:511–532, 2006.

[129] G.S. Ng, T. Murali, D. Shi, and A. Wahab. "Application of EFuNN for the classification of handwritten digits". *International Journal of Computers, Systems and Signals*, 5(2):27–35, 2004.

[130] J.D. Olden and D.A. Jackson. "Illuminating the "black box": a randomization approach for understanding variable contributions in artificial neural networks". *Ecological Modelling*, 154:135–150, 2002.

[131] S. Ozawa, S. Pang, and N. Kasabov. "On-line feature selection for adaptive evolving connectionist systems". *International Journal of Innovative Computing, Information and Control*, 2(1):181–192, February 2006.

[132] M.F Pasaha, R. Budiarto, and M. Syukur. "Connectionist model for distributed adaptive network anomaly detection system". In *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005*, pages 3915–3920, 2005.

[133] M.F. Pasha, R. Budiarto, M. Syukur, and M. Yamada. "EFIS: Evolvable-neural-based fuzzy inference system and its application for adaptive networks anomaly detection". In *ICMLC 2005*, pages 662–671, 2005.

[134] M. Pertselakis, N. Tsapatsoulis, S. Kollias, and A. Stafylopatis. "An adaptive resource allocating neural fuzzy inference system". In *Proceedings of the IEEE Intelligent Systems Application to Power Systems (ISAP'03), Lemnos, Greece, August 2003*, 2001.

[135] S. Petrovic-Lazarevic, K. Coghill, and A. Abraham. "Neuro-fuzzy modelling in support of knowledge management in social regulation of access to cigarettes by minors". *Knowledge-Based- Systems*, 17:57–60, 2004.

[136] N. Pican, D. Fohr, and J.-F. Mari. "HMMs and OWE neural network for continuous speech recognition". In *Proceedings of ICSLP*, 1996.

[137] J. Platt. "A resource-allocating network for function interpolation". *Neural Computation*, 3(2):213–225, 1991.

[138] J.V. Ramos. "Evolving Takagi-Sugeno fuzzy models for data mining". In *Proc. of the Workshop on Soft Computing and Complex Systems, SoftComplex'03, Coimbra, Portugal, June 2003*, pages 144–154, 2003.

[139] J.V. Ramos and A. Dourado. "Evolving Takagi-Sugeno fuzzy models". Technical report, Centre for Informatics and Systems Adaptive Computation Group, September 2003.

[140] V. Ravi, E. R. Srinivas, and N. K. Kasabov. "On-line evolving fuzzy clustering". In *International Conference on Computational Intelligence and Multimedia Applications, 2007.*, volume 1, pages 347–351, 13-15 Dec. 2007.

[141] S. Renals and R. Rohwer. "Phoneme classification experiments using radial basis functions". In *Proceedings of International Joint Conference on Neural Networks - IJCNN, Washington, D.C.*, volume I, pages 461–467, 1989.

[142] F. Rivera-Illingworth, V. Callaghan, and H. Hagras. "A neural network agent based approach to activity detection in AmI environments". In *IEE International Workshop, Intelligent Environments 2005 (IE05), Colchester, UK, 28-29th June 05*, 2005.

[143] T. Robinson and F. Fallside. "Phoneme recognition from the TIMIT database using recurrent error propagation networks". Technical report, Cambridge University, Engineering Department, March 1990.

[144] J.W. Sammon. "A nonlinear mapping for data structure analysis". *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[145] Silicon Recognition, Inc. *"ZISC Zero Instruction Set Computer"*, version 4.2 edition, June 2002. http://www.silirec.com.

[146] M. Sima, V. Croitoru, and D. Burileanu. "Performance analysis on speech recognition using neural networks". In *Proceedings of the International Conference and Development and Application Systems, Suceava, Romania*, pages 259–266, 1998.

[147] Q. Song and N. Kasabov. "Dynamic evolving neuro-fuzzy inference system (DENFIS): On-line learning and application for time-series prediction". In *Proceedings of the 6th International Conference on Soft Computing, October 1-4, 2000, Fuzzy Logic Systems Institute, Iizuka, Japan*, pages 696–702, 2000.

[148] Q. Song and N. Kasabov. "Weighted data normalization and feature selection for evolving connectionist systems". In *Proc. of the Eight Australian and New Zealand Intelligent Information Systems Conference, Sydney, Australia Dec. 2003*, pages 285–290, 2003.

[149] Q. Song and N. Kasasbov. "ECM, a novel on-line, evolving clustering method and its applications". In *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES2001)*, pages 87–92, 2001.

[150] G. Sorwar, A. Abraham, and L.S. Dooley. "Texture classification based on DCT and soft computing". In *Fuzzy Systems, 2001. The 10th IEEE International Conference on*, volume 3, pages 545– 548, 2001.

[151] S.S. Stevens. "On the theory of scales of measurement". *Science*, 103:677–680, 1946.

[152] T. Takagi and M. Sugeno. "Fuzzy identification of systems and its applications to modeling and control". *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-15:116–132, January / February 1985.

[153] R. Togneri, D. Forrokhi, Y. Zhang, and Y. Attikiouzel. "A comparison of the LBG, MLP, SOM and GMM algorithms for vector quantisation and clustering analysis". In *Proceedings Fourth Australian International Conference of Speech Science and Technology, Brisbane, Australia*, pages 173–177, 1992.

[154] A. Waibel, T. Hanazawa, G.E. Hinton, K. Shikano, and J.K. Lang. "Phoneme recognition: neural networks versus hidden Markov models". In *Proceedings ICASSP, New York, NY*, pages 107–110, 1988.

[155] A. Waibel, T.I Hanazawa, G. Hinton, and K. Shikano. "Phoneme recognition using time-delay neural networks". *IEEE Transactions of Acoustics, Speech and Signal Processing*, 37(3):328–339, 1989.

[156] X. Wang. ""On-line" time series prediction system—EFuNN-T". In *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES2001)*, pages 82–86, 2001.

[157] R. L. Watrous, B. Ladendorf, and G. Kuhn. "Complete gradient optimization of a recurrent network applied to /b/,/d/,/g/ discrimination". *Journal of the Acoustical Society of America*, 87(3):1301–1309, March 1990.

[158] R.L. Watrous and L. Shastri. "Learning phonetic features using connectionist networks: An experiment in speech recognition". In *Proceedings of the 10th International Conference on Artificial Intelligence*, pages 351–354, 1987.

[159] M. Watts. *"Evolving connectionist systems: characterisation, simplification, formalisation, explanation and optimisation"*. PhD thesis, University of Otago, New Zealand, 2004.

[160] M.J. Watts. "An investigation of the properties of evolving fuzzy neural networks". In *Proceedings of ICONIP'99, November 1999, Perth, Australia*, pages 217–221, 1999.

[161] M.J. Watts. "Evolving connectionist systems for biochemical applications". In N. Kasabov and K. Ko, editors, *Emerging Knowledge Engineering and Connectionist-based Systems (Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop "Future directions for intelligent systems and information sciences", Dunedin, 22-23 November 1999)*, pages 147–151. University of Otago Press, 1999.

[162] M.J. Watts. "Fuzzy rule extraction from simple evolving connectionist systems". *International Journal of Computational Intelligence and Applications, Special Issue on Neuro-Computing and Hybrid Methods for Evolving Intelligence*, 4(3):299–308, 2004.

[163] M.J. Watts. "ANN rule extraction using evolutionary programmed fuzzy membership functions". *International Journal of Information Technology: Special Issue on Evolutionary Algorithm and Advanced Learning System*, 11(10):45–53, 2005.

[164] M.J. Watts. "Nominal-scale evolving connectionist systems". In *Proceedings of 2006 International Joint Conference on Neural Networks (IJCNN 2006), Vancouver, Canada*, pages 4057–4061, 2006.

[165] M.J. Watts and N. Kasabov. "Simple evolving connectionist systems and experiments on isolated phoneme recognition". In *Proceedings of the first IEEE conference on evolutionary computation and neural networks, San Antonio, May 2000*, pages 232–239. IEEE Press, 2000.

[166] M.J. Watts and N.K. Kasabov. "Dynamic optimisation of evolving connectionist system training parameters by pseudo-evolution strategy". In *Proceedings of the Congress on Evolutionary Computation (CEC) 2001. Seoul, Korea*, pages 1335–1342, 2001.

[167] M.J. Watts and N.K. Kasabov. "Evolutionary optimisation of evolving connectionist systems". In *Proceedings of the Congress on Evolutionary Computation (CEC) 2002*, pages 606–610, 2002.

[168] M.J. Watts and S.P. Worner. "Comparison of multi-layer perceptrons and simple evolving connectionist systems over the Lincoln aphid data set". Technical Report ISBN 978-0-86476-175-9, Bio-Protection and Ecology, Lincoln University, New Zealand, February 2007.

[169] M.J. Watts and S.P. Worner. "Further sensitivity analysis of simple evolving connectionist systems applied to the Lincoln aphid data set". Technical Report ISBN 978-0-86476-177-5, Bio-Protection and Ecology, Lincoln University, New Zealand, February 2007.

[170] M.J. Watts, S.P. Worner, G.O. Lankin, , and D. Teulon. "A comparison of MLP and ECoS networks for the prediction of the flight of aphids in autumn sown wheat crops". In Zeke S.H. Chan Nik Kasabov,

editor, *Proceedings Conference on Neuro-Computing and Evolving Intelligence 2004, NCEI'04, 13-15 December, 2004, AUT Technology Park, 581 Great South Road, Auckland, New Zealand*, 2004.

[171] B. Widrow, D. E. Rumelhart, and M. A. Lehr. "Neural networks: Applications in industry, business and science". *Communications of the ACM*, 37(3):93–105, March 1994.

[172] B. J. Woodford and N. Kasabov. "Ensembles of EFuNNs: An architecture for a multi module classifier". In *The proceedings of FUZZ-IEEE'2001. The 10th IEEE International Conference on Fuzzy Systems, December 2-5 2001, Melbourne, Australia*, pages 441–445, 2001.

[173] B.F. Woodford. "Rule extraction from spatial data using local learning techniques". In *Proceedings of SIRC 2005 - The 17th Annual Colloquium of the Spatial Information Research Centre, Universty of Otago, Dunedin, New Zealand, November 24-25th, 2005*, 2005.

[174] B.J. Woodford. "Comparative analysis of the EFuNN and the Support Vector Machine models for the classification of horticulture data". In *Proceedings of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES2001)*, 2001.

[175] B.J. Woodford. "Evolving neurocomputing systems for horticulture applications". *Journal of Applied Soft Computing*, 8:564–578, 2008.

[176] B.J. Woodford. "Rule extraction from spatial data using a entropy-based fuzzy neural network". In P.A. Whigham, I. Drecki, and A. Moore, editors, *Proceedings of the 20th Annual Colloquium of the Spatial Information Research Centre & GEOCOMP (SIRC'2008), 1-3 September, 2008, The University of Auckland, Auckland, New Zealand.*, pages 55–66. University of Otago Press, 2008.

[177] B.J. Woodford, D. Deng, and G.L. Benwell. "A wavelet-based neuro-fuzzy system for data mining small image sets". In J. Hogan, P. Montague, M. Purvis, and C. Steketee, editors, *Australasian Workshop on Data Mining and Web Intelligence (DMWI2004), 1822 January, 2004, University of Otago, Dunedin, New Zealand.*, pages 139–144. Australian Computer Society, 2004.

[178] C. Wu and J. McLarty. *"Neural Networks and Genome Informatics"*. Elsevier Health Sciences, 2000.

[179] Y. Xiao and C. Han. "Correlating intrusion alerts into attack scenarios based on improved evolving self-organizing maps". *International Journal of Computer Science and Network Security*, 6(6):199–203, June 2006.

[180] K. Yamauchi and M. Sato. "Incremental learning of spatio-temporal patterns with model selection". In *Proceedings of ICANN 2007*, 2007.

[181] X. Yao. "Evolving artificial neural networks". *Proceedings of the IEEE*, 87(9):1423–1447, September 1999.

[182] L. Zadeh. "Fuzzy sets". *Information and Control*, 8:338–353, 1965.

[183] K. Zamanifar, B.S. Ghahfarokhi, H. Shahbazi, and M. Kazemifard. "Bam rescue simulation team description", 2006.

[184] C. Zanchettin and T.B. Ludermir. "Evolving fuzzy neural networks applied to odor recognition in an artificial nose". In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 1, pages 675–680, 2004.

[185] D. Zhang, N. Kasabov, and A. Ghobakhlou. "An adaptive model of person identification combining speech and image information". In *ICARCV 2004, Kunming, China*, pages 413–418, 2004.

**Michael J. Watts** received the degree of Bachelor of Science with First Class Honours in Information Science from the University of Otago in 1996, and his PhD degree, with a focus on artificial neural networks, from the same department in 2004. From 2000 to 2004 he was a senior teaching fellow in the Department of Information Science, University of Otago. From 2004 to 2007 he was a post-doctoral fellow in the National Centre for Advanced Bio-Protection Technologies at Lincoln University, New Zealand, where he developed methods using computational intelligence to predict the invasiveness of insect pests and crop diseases. As of 2008 he is a post-doctoral fellow in the School of Biological Sciences at the University of Sydney, Australia.