Connectionist Systems for Rapid Adaptive Learning: A Comparative Analysis on Speech Recognition

George Coghill[2], David Zhang[1], Akbar Ghobakhlou[1] and Nikola Kasabov[1]
[1]Knowledge Engineering and Discovery Institute
Auckland University of Technology, PO Box 92006, Auckland, New Zealand
[2]University of Auckland, Private Bag 92019, Auckland
akbar@aut.ac.nz, g.coghill@auckland.ac.nz, www.kedri.info

Abstract
In may real time applications a system needs to be trained quickly on a small amount of new data and then generalise on unseen data. This is a challenging task for neural networks especially when applied to difficult problems such as adaptive speech recognition. An experimental comparison between three recently introduced on-line adaptive connectionist paradigms on adaptive phoneme recognition task is presented in the paper. The models used are: the Evolving Classifying Function (ECF), the Zero Instruction Set Computer (ZISC) and the Deterministic Adaptive Random Access Memory Network (DARN). An MLP network, although slow and not an incremental learning model, was used to provide a benchmark for comparison. The results show that ECF adapts faster to new data, while ZISC and DARN require mode data.

Keywords: adaptive systems; evolving systems; connectionist systems; deterministic adaptive random memory.

1. Introduction
In this paper we investigate the behaviour of three recently introduced networks that are intended for use in applications where incremental and fast learning are desired. We have chosen to base our experiments on speech data, taken from the Otago Speech Corpus [7]. We felt that this data set posed a difficult problem and to demonstrate this fact, we performed the first set of experiments using a multi-layer perceptron (MLP) network. This network is neither incremental in its learning, nor fast (requiring many passes through the data), but it should give some sort of benchmark against which we may assess the other networks.

We begin by describing the properties of the speech data set and the following section describes results obtained using the MLP network. The next three sections briefly describe the principles of the ECF [5,6], ZISC [8] and DARN [2]. Section 7 concludes with the results and a discussion of the implications.

2. Experimental Speech Data
The phonemes provided [8], are obtained from 6 NZ English speakers (3 males and 3 females). The list of 10 NZ English vowels and their corresponding codes is shown in Table 1.

The wave format data was read in as 16-bit integers. The sample rate was 22.050 kHz. A Hamming Window was used to create a matrix of 512 samples, equating to approximately 23.2 ms of speech. Each window was subsequently transformed into 25 mel-cepstrum coefficients. The log energy was also included to generate 26 input features for every frame. Each window overlaps the previous by 50%.

Table 1: New Zealand English vowels (monophthongs)

Phoneme Number	Symbol	Example (word)	Example (phonetic)
1	ɪ	get	gɪt
2	ɛ	cat	kɛt
3	ɐ	hut	hɐt
4	ɔ	hot	hɔt
5	ʊ	put	pʊt
6	i:	see	si:
7	ɐ:	father	fɐ:ðə
8	o:	sort	so:t
9	ø:	bird	bø:d
10	ʉ:	too	tʉ

The phoneme data is divided into two sets, A and B. The data in set A is again divided into two, one for training and one for testing as follows:
- Training set A contains 4665 samples obtained from 4 speakers (2 males and 2 females).
- Testing set A contains 2503 samples obtained from new instances of the training set A.

The data in set B is divided into two, a testing set and an adaptation set, which is used to demonstrate the incremental learning capability of these networks:
- Testing set B (1 male and 1 female) contains 1427 samples, which are obtained from 2 new speakers.
- Adaptation set contains 1427 samples of training values for data set B.

The ZISC neurocomputing chip only accepts up to 64 8-bit valued input vectors. The data sets were therefore scaled to lie between 0 and 255 to accomplish this requirement. This same data was presented to the ECF and DARN networks and, after input normalisation, also used for the MLP network.

3. Off-line learning in MLP

Although the MLP is slow to learn, requiring many passes (250 in this case), it did generalize well. The network used here had a single hidden layer of size 60 and a learning rate of 0.4. Column 2, in table 2, shows the network's ability to recognise its own training set. The results for test set A, give some idea of the difficulty of the task. The results for test set B are even poorer, mainly because there are no samples of the speakers for set B in the original training set. The final column shows the results of adding more training samples (an adaptation set) to the original training set. This, as expected, somewhat improves the performance for test set B. Note, however, that to achieve this last set of results, the adaptation set must be added to the original training set and the network completely retrained, because incremental learning is not an option for the MLP network.

Phonemes (table 1)	Training set	Test set A	Test set B	Adaptation set
1	96	82	36	59
2	87	68	17	63
3	97	76	50	67
4	91	77	27	55
5	96	73	28	75
6	77	33	3	33
7	89	77	7	63
8	94	21	18	24
9	86	61	19	61
10	93	60	22	58
Over all	92	69	26	59

Table 2: % Recognition of an MLP net on NZ Vowels

4.The Evolving Classifying Function (ECF)

The ECF [5,6] is an ECOS classification model that partitions a given data set into a number of classes and finds their class centres in the N-dimensional input space by "placing" a rule node. Each rule node is associated with a class and with an influence (receptive) field representing a part of the N-dimensional space around the rule node. Usually, such an influence field in the N-dimensional space is a hyper-sphere. There are two distinct phases of the ECF operation. During the first, learning phase, data vectors are fed into the system one by one with their known classes. The following steps describe the learning sequence:

Step1. Input a vector from the data set and calculate the distances between the vector and all rule nodes already created.

Step 2: If all distances are greater than a maximum radius parameter Rmax, a new rule node is created. The position of the new rule node is the same as the current vector in the input data space and its radius is set to the min-radius parameter; return to step 1.

Else If there is a rule node with a distance to the current input vector less than or equal to its radius and its class is the same as the class of the new vector, nothing will be changed; return to step 1.

Else If there is a rule node with a distance to the input vector less than or equal to its radius and its class is different from those of the input vector, its influence field should be reduced. The radius of the new field is set to the larger value from the distance minus the min-radius, and the min-radius.

Step 3: If there is a rule node with a distance to the input vector less than or equal to the Rmax, and its class is the same as the vector's, enlarge the influence field by taking the distance as the new radius if only such enlarged field does not cover any other rule node which has a different class,

Else create a new rule node the same way as in step 2, and return to step 1.

The recall (classification phase of new input vectors) is performed in the following way:

1. If the new input vector lies within the field of one or more rule nodes associated with one class, the vector belongs to this class.
2. If the input vector lies within the fields of two or more rule nodes associated with different classes, the vector will belong to the class corresponding the closest rule node.
3. If the input vector does not lie within any field, then there are two cases:
   (a) one-of-n mode: the vector will belong to the class corresponding the closest rule node.
   (b) m-of-n mode: take m highest activated by the new vector rule nodes, and calculate the average distances from the vector to the nodes with the same class; the vector will belong to the class corresponding to the smallest average distance.

The ECF has several parameters that need to be optimised according to the data set used. An algorithm for optimisation of these parameters is presented in [6].

5.The Zero Instruction Set Computer (ZISC)

ZISC is a parallel processing neural network that provides fast computing power for technology requiring pattern recognition and information classification. Each neuron is an independent processing element, with integrated memory and learning capability. The ZISC can be considered as an expert system that can recognize and classify objects or situations and take instantaneous decisions based upon accumulated knowledge. It learns from samples of data. The ZISC is a fully integrated, digital implementation of the Radial Basis Function (RBF) neural network model [8]. The ZISC learning process is as follows:

If the input vector does not fall in any of the influence fields of the prototypes already stored in the network, THEN: a new neuron is committed. Its influence field is set to the minimum value between the Maximum Influence Field global parameter and the distance to the closest prototype of the neurons committed in the actual context.

Else If the input vector falls in the influence field of a prototype already stored in the network and their category matches, THEN: no change;

Else If the input vector falls in the influence field of a prototype already stored in the network but their category does not match, THEN: one or more influence fields are reduced so the adjacent neurons with different categories become tangent. **End.**

The reduction of the influence field, however, cannot go beyond a minimum defined by the Minimum Influence Field. As a result, the RBF space mapping may have prototypes with portions of their influence field overlapping one another. This may later cause some uncertain responses from the network. If the reduction of the influence field is decreased to the Minimum Influence Field global parameter, the neuron is labelled as "degenerated".

The classification process in ZISC consists of evaluating whether, or not an N-dimension input vector lies within the influence field of any prototype stored in the network and/or its sub-networks. This is done by computing the distance between the input vector and all stored prototypes, and comparing this distance with the prototype actual influence field. This operation can be iterated to survey all sub-networks, that is, classify the vector for all applicable contexts:

If the input vector does not lie within any influence fields, THEN: it is not recognized.

Else If the input vector lies within the influence field of one or more prototypes associated with one category, THEN: it is recognized and declared as belonging to that category.

Else If the input vector lies within the influence field of two or more prototypes associated with different categories, THEN: it is declared as unidentified, that is recognized but not formally identified.

End.

6. Deterministic Adaptive RAM Networks (DARN)

The third model used here, is a type of Weightless Neural Network and is derived from the work of Aleksander [4], see Figure 1.

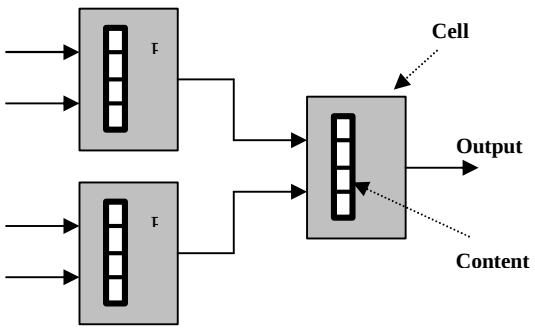

Figure 1: A Weightless Neural Network

The network shown is a very small one, which we chose merely to help with the following description. In a practical system, there may be many cells in each of several layers with 4, or even 8 address/input lines per cell, rather than the 2 shown here. The network appears pyramid shaped since the number of cells decrease as we move towards the output layer. If we require several output classes, a pyramid must be created for each class. However, all the pyramids share a common input vector. This input vector is binary and, when applied to a cell, behaves like an address bus. The contents of an addressed location may be interpreted as holding *zero*, *one*, or *undefined*. Initially, all locations are undefined. In [3], the Aleksander model was modified, with a resulting improvement in the performance. The concept of an *addressable-set* was introduced. When the input/address vector selects an undefined location (U), the cell output propagates that undefined value forward to the next layer. For example, suppose the output from the first layer is the vector 1U. The output cell interprets this as having enabled both addressed locations 10 and 11, forming an addressable set, or {10, 11}. Their network model then has some rules governing what to do about this. The idea of having an addressable set together with some simple rules was developed further in [1]. Even then, however, the generalization properties of the WNN were still not very useful. We now describe the operation of the Deterministic Adaptive RAM Network (DARN), with respect to those earlier models. In the DARN, each addressed location is now a register instead of a single bit. Initially, all of the registers in the network are set to 0. The 0 value is interpreted as an undefined value. During training, as we shall see, a register's value may become positive, or negative.

Training - The input/address vector is presented to the network. The desired output of every cell in the pyramid is the same as the desired output of the entire pyramid, effectively the same as described in [3]. We begin at the input layer. If the desired output is one, the addressed location is incremented; if the desired output is zero, the location is decremented. The next step then involves calculating address vectors for the following layer moving towards the output. The address vector for the output cell, or the next layer in a larger network, is constructed by invoking the recall operation, (described below) for each cell in the current layer. Note that there may be undefined bits in those vectors. When an undefined bit is found (during training), it is replaced by the desired output. The address vector is then applied to the output cell, or the next layer in a larger network and the selected location is modified as already described for the first layer.

Recall - During the recall phase, the content of each addressed location, starting from the input layer, is interpreted as undefined, zero, or one depending upon whether the counter value is 0, negative, or positive respectively. Now, during recall, we should see that the output of a cell might produce an undefined bit. Here we adopt the approach used in [3], of propagating the

addressable-set to the next layer. This may result in several locations being selected at the same time. When such an event occurs (during recall), the following rules are adopted - they are effectively the same as described in [3]:

If at least one zero and no ones appears in the addressable set, the output is zero.

Else If at least one one and no zeros appears in the addressable set, the output is one.

Else the output is undefined Recall is propagated forward through the pyramid until the output is reached. **End.**

The network only requires one pass through the data.

7. Results and Discussions

The following tables illustrate the performance of each of the models when tested with the speech data. The main difference from the MLP, in the way these experiments were performed, is that the new adaptation training was continued from the existing network state having already been trained with the original training set.

The ECF network performed well (Table 5). It has good generalisation capability after a short adaptive training. It trained rapidly, requiring only four passes through the data and was readily adapted to the new training data.

Both ZISC (Table 3) and DARN (Table 4) performed well in other areas [2] and their ease of realisation in hardware makes them the preferred choice in many embedded systems applications. They are most useful though, when generalisation requirements are not too high. One feature that was observed in the behaviour of the ZISC and DARN networks was that although they learn rapidly, they require a larger number of training samples in a given phoneme category, before they recognise phonemes in that category. For example, the poor result for ZISC and DARN for category 4 appears to be related to the 132 training samples in that class, compared to, say, the 395 samples in category 8. Investigations are continuing in this area.

Table3: % Recognition of the ZISC on NZ Vowels

Phonemes (table 1)	Training set	Test set A	Test set B	Adaptation set
1	97	16	0	1
2	84	49	0	17
3	75	20	0	0
4	91	52	0	10
5	44	13	0	6
6	91	29	4	3
7	5	0	0	0
8	84	62	0	33
9	93	50	0	0
10	99	94	93	98
Over all	79	40	14	21

Table4: % Recognition of the DARN on NZ Vowels

Phonemes (table 1)	Training set	Test set A	Test set B	Adaptation set
1	10	10	6	0
2	18	24	15	11
3	3	1	0	2
4	0	0	0	0
5	7	18	7	4
6	15	3	0	44
7	72	57	29	38
8	55	55	20	54
9	27	15	13	18
10	38	25	2	21
Over all	34	28	11	27

References

[1] R. G. Bowmaker and G. Coghill, "Improved Recognition Capabilities for Goal Seeking Neuron", Electronics Letters, 1992, vol. 28, pp. 220-221.

[2] G. Coghill and W. K. Lai, "The Use of Weightless Neural Networks for The Extraction of Feature Information from Web Documents for Taxonomy Construction", Robotics, Vision, Information and Signal Processing (ROVISP), 2003.

[3] E. Filho, M. Fairhurst and D. Bisset, "Adaptive Pattern Recognition using Goal Seeking Neurons", Pattern Recognition Letters, 1991, 12, pp. 131-138.

[4] W. Kan and I. Aleksander, "A Probabilistic Logic Neuron Network for Associative Learning", June 1987, vol. 2, pp. 541-548.

[5] N. Kasabov, "Evolving Connectionist Systems", Springer Verlag, 2002.

[6] N. Kasabov and Q. Song, "GA-Parameter Optimisation of Evolving Connectionist Systems for Classification and a Case Study form Bioinformatics", ICONIP'02 Singapore, 2002.

[7] S. Sinclair and C.Watson, "The Development of the Otago Speech Database", Proc. of ANNES '95, IEEE Computer Society Press, Los Alamitos, CA, 1995.

[8] ZISC-Manual, "Zero Instruction Set Computing (ZISC) ", 2000, http://www.silirec.com/

Table 5: % Recognition of the ECF on NZ Vowels

Phonemes (table 1)	Training set	Test set A	Test set B	Adaptation set
1	95	62	16	67
2	94	75	15	70
3	97	22	7	14
4	94	86	58	56
5	93	44	23	45
6	96	80	54	69
7	97	84	14	46
8	97	81	61	72
9	96	77	39	57
10	100	74	26	64
Over all	96	74	32	58