A Study of Early Stopping, Ensembling, and Patchworking for Cascade Correlation Neural Networks

Mike J. W. Riley, Karl W. Jenkins, and Chris P. Thompson

***Abstract*— The constructive topology of the cascade correlation algorithm makes it a popular choice for many researchers wishing to utilize neural networks. However, for multimodal problems, the mean squared error of the approximation increases significantly as the number of modes increases. The components of this error will comprise both bias and variance and we provide formulae for estimating these values from mean squared errors alone.**

We achieve a near threefold reduction in the overall error by using early stopping and ensembling. Also described is a new subdivision technique that we call *patchworking*. Patchworking, when used in combination with early stopping and ensembling, can achieve an order of magnitude improvement in the error. Also presented is an approach for validating the quality of a neural network's training, without the explicit use of a testing dataset.

***Index Terms*—Bias, Cascade Correlation, early stopping, ensembling, multimodal functions, patchworking, subdivision method, variance.**

I. Introduction

Neural networks are commonly used for regression modelling. However, a perennial problem in the specification is determining the topology of the network. Constructive topology neural networks have gained in popularity because hand crafting this structure is very time consuming. Cascade correlation [1] is a well known member of the constructive neural networks, with hundreds of associated publications each year. Rather than requiring decisions from the user such as: how many hidden layers, how many neurons in each layer, and which activation functions should be used, cascade correlation automatically makes these choices during its supervised learning process.

The first version of cascade correlation was intended to work best as a classifier, but subsequently, its author made some minor changes that improved its performance in regression roles [2]. The new algorithm was named "Cascade II" but is referred to in this paper as "CasCor".

Manuscript received October 8, 2010. This work was supported by an EPSRC Doctoral Training Grant.

Mike J. W. Riley is a Doctoral Research Student in the School of Engineering, Cranfield University, Cranfield, Bedford MK43 0AL, UK. (e-mail: m.riley@cranfield.ac.uk).

Karl W. Jenkins is a senior lecturer in the Department of Engineering Computing and Cybernetics at Cranfield University and specializes in computational engineering. (e-mail: k.w.jenkins@cranfield.ac.uk).

Chris P. Thompson is the Head of the Department of Engineering Computing and Cybernetics at Cranfield University.
(e-mail: chris.thompson@cranfield.ac.uk)

The aim of this paper is to present mechanisms that improve the fit of the CasCor neural network, focusing on multimodal surfaces. Test functions for global optimization were being used by the current authors to create training surfaces, and curiosity grew as to why certain functions caused exceptional mapping problems for CasCor neural networks. Whilst undertaking work to resolve these problems, the most successful method discovered was subdividing the input domain (see [3]).

Ensemble averaging and early stopping are two techniques commonly used to reduce neural network generalization errors [4]. We found clear benefits from employing these techniques for the functions under consideration. Ensembling and early stopping address the *variance* problem of neural networks. We introduce a subdivision method called "Patchworking" that addresses the *bias* problem of CasCor networks, by raising their *information capacity*. This capacity is a measure of a neural network's ability to represent the features within the training set. By using patchworking for domain subdivision the information content in the training sets, and hence the error, is much reduced. The total information capacity of the patchwork has grown – thus we obtain improved generalization on multimodal test functions.

The layout of this paper is as follows: Section II describes the three techniques we use to improve the fit for multimodal functions. Section III gives details of the early stop, training set size, ensembling, and patchworking experiments. Section IV contains the results, and closing remarks are made in Section V.

II. Improving the Fit of the CasCor Neural Network

Three techniques are presented in this paper, all of which are designed to improve the fit of the CasCor neural network to given datasets thereby improving generalization. These three methods are:

1) Early stopping
2) Ensemble averaging
3) Patchworking

Our previous work [3] indicated other areas for future work such as: how the sizes of training datasets influence the result of training, and, what are the effects of the size of the validation set when we use early stopping? The current work answers these questions.

A. Early stopping

One of the disadvantages of CasCor neural networks is their propensity to overfit on the training data, thus losing generalization of the underlying function [4]. Inspecting the

monotone decrease of the training mean squared error (MSE) gives no indication of this. Typically, the error during training is seen to reduce, almost uninterrupted, until one of the stopping criteria is met and the network is pronounced as "trained". If, however, a call-back function is set, the training progress can briefly be interrupted to test the (still evolving) neural network against the validation dataset.

The validation dataset is wholly independent from the training set and it allows us to determine an early stopping point. The MSE graph on this validation data typically takes the approximate form of a hockey stick outline – initially the validation MSE falls as the network fits to the underlying function, but at some point too many neurons are added, there is a loss of generality, and the MSE starts to increase. Early stopping halts the training at or around this minimum point thus minimizing negative impacts from overfitting. In reality, the profile of the validation error is not smooth and some form of heuristic needs to be used to halt the training at an appropriate moment; the heuristic is described in Section III.D.

B. Ensembling

Tetko and Villa [4] described ensemble averaging, or a "committee of machines", as acting to reduce the variance that is common in neural networks. Multiple neural networks are trained on the same dataset, but in use, the arithmetic mean is taken across the output responses of the ensemble members. The testing error of these ensembles is much lower than the average test errors of their constituent parts and, when compared to the basic CasCor neural network, often represents a reduction in the MSE by a factor of two to three - the only penalty being an increase in required training time.

C. Patchworking - a subdivision method

A third technique for improving the fit is "patchworking", a method of subdivision that addresses the bias problem of CasCor neural networks by raising their capacity. This technique is particularly suited to highly multimodal response surfaces and its benefits are shown in Section IV.E.

Determined empirically, we define "highly multimodal" as six or more distinct extrema over a multi-dimensional surface; the fit deteriorates significantly when the extrema exceed nine. Functions such as these are used in this paper to demonstrate CasCor's difficulty in fitting the underlying function (Table I). These poor fits appear as high MSEs on testing sets and are also clearly visible in surface plots. Neither early stopping, nor ensembling, are sufficient to overcome these poor fits as the source of this problem is the inability of a single CasCor neural network to represent the complex features in the dataset.

The ensembled and early-stopped plot of the Schwefel function, Fig. 2, correctly maps the global minimum and global maximum, but is clearly a poor approximation of Schwefel's form (Fig. 1). The Langermann function, Fig. 3 likewise challenges the mapping ability of the CasCor neural network even with ensembling and early stopping (Fig. 4).

Some of the greatest strengths attributed to the CasCor type of neural network are as a result of it growing its own topology during training. An intrinsic feature is that at any point during training, no more than one new neuron will be having its weights optimized. It is widely believed that this distinguishing behaviour results in rapid training times; however, this is challenged by [5], in which Squires et al. conclude that freezing of formerly trained weights can be detrimental to effective learning.

The universal function approximation abilities of the CasCor neural network, mathematically proven in [6], are only applicable if we assume that correct choices have been made when each and every neuron was inserted. By taking a system view of the training process, we argue that correct choices are frequently not made when mapping multimodal functions.

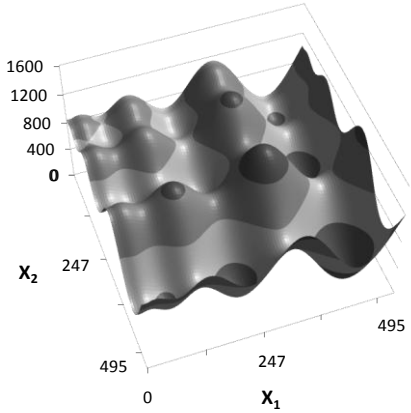

Figure 1 Schwefel function, range x(i) [0,500]

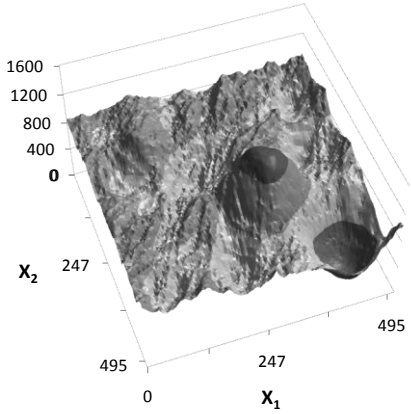

Figure 2 CasCor mapping of Schwefel with ensembling and early stopping

Informally, the training process plays the role of an agent in the system. This agent aims to train and fix in the network one neuron at a time that, in isolation, reduces the MSE on the training set by the largest possible amount. Several time steps later in the training, more neurons have been added and we see, with the benefit of hindsight, that incorrect choices have been made in the early stages of training. What were once apparently optimal additions to the network are ultimately conspiring to deflect the network from a good mapping of the underlying function. The training algorithm dictates that once neurons have been placed in the network, they may not be removed or re-trained (*weight freezing*) and so the problem becomes irreconcilable [5].

The problem is one of *decision theory* – specifically *evidential decision theory*: how can a training process place a neuron in the network which, later in time, will combine with downstream neurons in only a beneficial way?

A more formal description can be found in [7] where they consider the problems caused when training on the simple "double-tanh" function. The problem is seen to be sufficient to preclude, or at least delay, convergence of the CasCor

network. Variants of the CasCor neural network include one that only adds neurons to a single hidden layer (breadth) [2] and one that chooses whether to add depth or breadth to the network [8]. Both have mixed success against the standard CasCor.

In our training experiments with datasets that contain highly multimodal functions (Table I), the training problem becomes clearer when monitoring the validation MSE. As the network is training, the insertion of new neurons should be conferring a greater information capacity to the neural network, and the validation MSE should decrease. Inserting the first two or three hidden neurons does cause a small decrease in the validation MSE, but soon after, this error increases resulting in a very poor generalization of the underlying function.

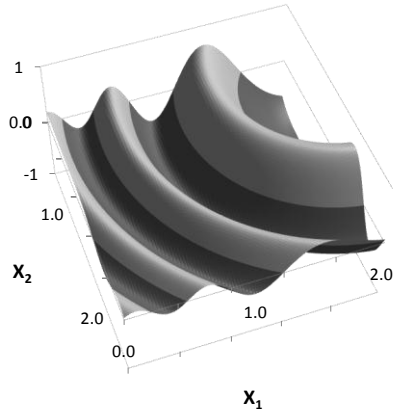

Figure 3 Langermann function, range x(i) [0,2]

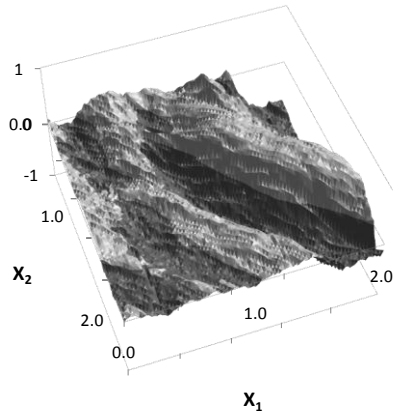

Figure 4 CasCor mapping of Langermann with ensembling and early stopping

The hypothesis behind patchworking is that by subdividing the input domain, the number of extrema that any one neural network must approximate is kept below the multimodal threshold. Hence, CasCor networks with a small number of neurons can approximate the function over each subdivision with a lower MSE. In this way, patchworking overcomes the problems associated with *weight freezing*. Ensembling and early stopping can be used in conjunction with patchworking and are, in fact, logical accompaniments.

III. Experimental set up

The architecture of the CasCor algorithm is well known [1],[2],[9]. The CasCor neural networks under consideration are created from the open source library created by Nissen [10]. The library contains an implementation of the Cascade Correlation II algorithm based on the original Lisp code written by Fahlman in 1996 (unpublished).

Here, the FANN C source code is used with default settings chosen for CasCor training. The target MSE for the training is 10^{-4} when early stopping is not used and an arbitrary setting of 10^{-5} when early stopping is used. In use, the lower target will never be reached, due to early stopping triggering a halt to the training. The current release, 2.1.0-Beta, does not yet provide a neural network copy utility or functions that correctly scale and de-scale datasets, and so these have been added to our implementation.

One traditional test for the quality of regression fits (such as presented in the current work) is to calculate the MSE against a testing set, in which the samples differ from those in the training set. Lower is better, and so we can measure the success of the techniques herein by how much they reduce the MSE. Our testing sets are generated from the algorithm in [11]. The size is chosen as $1000 \times d$ where d is the number of inputs to the neural network (or dimensions). The positioning of so many points is computationally expensive, especially when trying to maintain space filling properties. For this reason only one template was generated for each of the four different dimensions that were tested.

The range of all inputs and outputs is normalized to the interval [0.1,0.9] with the scaling factors saved after processing. These factors are later used to scale down the queries and scale up the neural network response.

Note: All MSE errors presented in this paper are also calculated on scaled data [0.1,0.9], thus making possible fair comparisons between otherwise disparate function output ranges.

A. Sample size

When choosing the size of the training datasets, how many samples should be used? Too few samples will mean that our training set may not accurately represent the underlying pattern. However, in situations where generating training data is very time-expensive, we would like to know the minimum size that can be of practical use in training our neural networks. We would also like to answer the question of how the demand for training data varies with the dimensions of the problem at hand. To determine the answers to these questions we performed CasCor training using 13 test functions (defined in Table I) in two, three, four and five dimensions with training datasets sizes in the range $[16 \times d, 384 \times d]$ (where d is the number of dimensions).

Orthogonal arrays (OAs) were chosen to generate our training datasets. An OA is defined in the form $OA.N.k.s.t$ indicating an orthogonal array with N runs, k factors, s levels, and strength t. This is an array of size N by k, with entries from 0 to $s-1$ with the property that in any of the t columns each of the s^t possibilities occurs equally often [12].

The training set is made up from repeated runs of $OA.16.5.4.2$ [13]. With 16 evaluations being made each time, 6 runs of this OA would be required to generate a training dataset of 96 points. The selection of the factors in each subsequent OA is known as the infill criteria [14]; when subsequent OAs are evaluated, each of its factors is chosen to be numerically furthest from all previously tested factors.

The use of orthogonal arrays in the current work is only an artifact of the intended application of this work in surrogate

Table I Multimodal test functions

Function Name		Range	
Ackley	$= -20 \cdot \exp\left(-\frac{1}{5}\cdot\sqrt{\frac{1}{n}\sum_{j=1}^{n}x_j^2}\right) - \exp\left(\frac{1}{n}\cdot\sum_{j=1}^{n}cos(2\pi x_j)\right) + 20 + \exp(1)$	$-30 \le x_j \le 30$ $j = n\ variables$	(1)
De Jong's 5th	$= \left(0.002 + \sum_{i=1}^{25}(i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6)^{-1}\right)^{-1}$ where $\begin{pmatrix}a_{1i}\\a_{2i}\end{pmatrix} = \begin{pmatrix}-32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32\end{pmatrix}$	$-20 \le x_j \le 20$ $j = 1,2$	(2)
Langermann	$= \sum_{i=1}^{5} c_i \exp\left(-\frac{1}{\pi}\sum_{j=1}^{2}(x_j - a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{2}(x_j - a_{ij})^2\right)$ where $(a_{ij}) = \begin{pmatrix}3 & 5 & 2 & 1 & 7\\ 5 & 2 & 1 & 4 & 9\end{pmatrix}^{\mathbf{T}}\ (c_i) = (1\ \ 2\ \ 5\ \ 2\ \ 3)^{\mathbf{T}}$	$0 \le x_j \le 2$ $j = 1,2$	(3)
Michalewicz	$= -\sum_{j=1}^{n}\sin(x_j)\cdot\left(\sin\left(\frac{j\cdot x_j^2}{\pi}\right)\right)^{20}$	$j = n\ variables$ When $j = 2,\ 0 \le x_j \le \pi$ When $j = 5,\ 1.0 \le x_j \le 1.5$	(4)
Schwefel	$= 418.9829n - \sum_{j=1}^{n}\left(x_j \sin\sqrt{\lvert x_j\rvert}\right)$	$j = n\ variables$ When $j = 2,\ 0 \le x_j \le 500$ When $j = 4,\ 100 \le x_j \le 300$ When $j = 5,\ 100 \le x_j \le 300$	(5)
Shubert	$= \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\cdot\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$	$-8 \le x_j \le -6.2$ $j = 1,2$	(6)
Six Hump Camel Back	$= \left(4 - 2.1x_1^2 + {x_1^4}/{3}\right)\cdot x_1^2 + x_1x_2 + (-4 + 4x_2^2)\cdot x_2^2$	$-1.9 \le x_1 \le 1.9$ $-1 \le x_2 \le 1$	(7)
Hartmann, $H_{3,4}$	$= -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right]$ where i | a_{ij} | c_i | p_{ij} 1 | 3.0 10 30 | 1.0 | 0.6890 0.1170 0.2673 2 | 0.1 10 35 | 1.2 | 0.4699 0.4387 0.7470 3 | 3.0 10 30 | 3.0 | 0.1091 0.8732 0.5547 4 | 0.1 10 35 | 3.2 | 0.0381 0.5743 0.8828	$0 \le x_j \le 1$ $j = 3\ variables$	(8)
Rosenbrock	$= \sum_{j=1}^{n-1}\left[100\left(x_j^2 - x_{j+1}\right)^2 + \left(x_j - 1\right)^2\right]$	$-10 \le x_j \le 10$ $j = n\ variables$	(9)

modelling, and their inclusion is not believed to alter the findings of this paper. In creating training datasets, less complex sampling methods should be sufficient to repeat our results.

B. Early stopping

Several tests were undertaken in order to answer two questions: 1. What is the smallest size of validation set that can be used? 2. Does the use of larger size validation sets have any beneficial effect on improving the fit of the trained networks? The validation sets ranged from a size of 5% of the training set to 100% of the training set. Code from Beachkofski and Grandhi [11] provides the method of distributing the samples in the validation set. This "improved Latin hypercube" sampling was chosen because:

1) Generating validation sets of less than 1000 points is not computationally expensive and can be done at run time,
2) The algorithm in [11] produces points that fill the hypercube uniformly, the statistical properties of which are desirable as described in [14],
3) The technique is fundamentally different from that used to generate the training set - ensuring that most, if not all, of the validation data points are automatically independent from those in the training set.

After the validation error is initialized to 1.0, our heuristic algorithm for early stopping is run each time a new hidden neuron is added to the network, and is given below:

- Test the network against the validation set.
- If this new validation error is less than the old one, update the old validation error with this new value and make a copy of this "best network so far".
- Do not initiate early stopping until at least five hidden neurons exist in the network.
- Trigger early stopping on the earliest of:
  - The error on the validation set becoming less than 5×10^{-5} (suitably low error)
  - The validation error growing to be 50% larger than the smallest experienced validation error (network is diverging)
  - More than 31 hidden neurons existing in the network (likelihood of a diverging network)
- When early stopping occurs, the "best network so far" is recalled from memory to replace the active network. The

training is halted and the network is saved to permanent storage. The saved neural network is that which had the smallest validation error.

C. Dispensing with the testing set

Our early stopping validation set shares the same property of a testing set in that they both contain samples wholly independent from the training dataset. The only difference is that testing sets are usually of a large size. Testing sets are useful in determining how successful a neural network's training has been. However, there may be circumstances where sampling is very time-expensive, for example surrogate modelling. If we want to avoid the cost of generating a large testing set, yet still retain a test for the quality of the fit, is there a size of validation set that can give us a reasonable approximation to the results we would get from a testing set? Experiments were performed that compare the MSE calculated from validation sets of sizes [5%,100%] of the training set against MSE calculations from our much larger testing sets of size $1000 \times d$.

D. Ensembling

When preparing an ensemble, we need to answer the question of how many neural networks to include in that ensemble. Others have chosen an arbitrary number [4],[15] for their ensembles, but we investigated the ensemble size with respect to its influence on reducing the MSE.

Ensembles of CasCor neural networks were trained on our 13 test functions; each test was repeated ten times for the larger ensembles and 30 times for ensembles smaller than ten.

E. Patchworking

The algorithm used to construct the patchwork is shown in the appendix. It allows for a user defined number of subdivisions known as "depth" and can be applied to as many input dimensions as is practical. Note, though, that the number of required networks grows exponentially $2^{(depthxdimensions)}$ and so this method may not be practical if the dimensions number more than nine or ten. The patchworking technique is shown in Fig. 5 and is applied as follows:

1. Train at first without subdividing the domain (patchwork depth=0)
2. Test the MSE after this training.
3. Subdivide the input domain if the test error is undesirably high (depth = depth + 1).
4. Create more training samples if necessary and re-train on these subdivisions (or 'patches').
5. Repeat steps 2-4 until the testing MSE is satisfactorily low.

A relatively simple algorithm can be constructed to query such a patchwork, assuming that we have stored on file the minimum and maximum bounds of each network's domain.

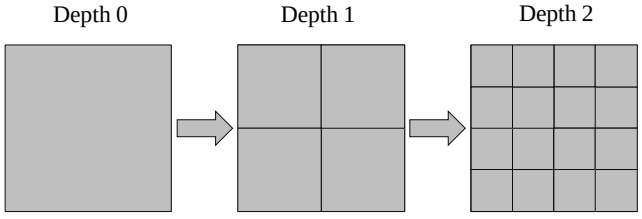

Figure 5 Patchworking subdivisions for a 2D function

IV. Results

A. Sample size

Fig. 6 shows the results of the sample size test. Neural networks were trained on the 13 test functions, covering two to five dimensions. Each test was repeated ten times. After each training, the quality of the fit was evaluated by a testing set of size $1000 \times d$. The resulting MSEs often differed by one or two orders of magnitude, hence a need to normalize the results. In normalizing the results, the set of mean squared errors for each function were scaled such that the size of the training dataset that yielded the worst error was attributed 1.0; the dataset set size with the lowest MSE was attributed a score of 0.0. Therefore, Fig. 6 shows the mean of the normalized results of training across the 13 test functions.

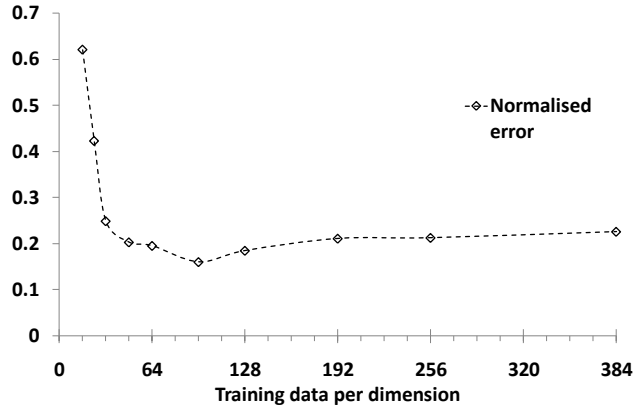

Figure 6 Change in test MSE against training set size

The demand for training data scales linearly with the number of dimensions. Less than 32 samples/dimension leads to poor mappings of the underlying function. Optimal training occurred when the training datasets were of size 96 samples/dimension. For more cost-effective training, 48 to 64 samples/dimension are sufficient to yield low mapping errors.

B. Early stopping

Fig. 7 shows the results of an experiment that aimed to determine how big the validation set should be with respect to the training set. For this experiment, we chose to train our neural networks with 96 samples per dimension. As before, we trained on all 13 test functions and each test was repeated ten times – Fig. **7** showing the mean average of the results.

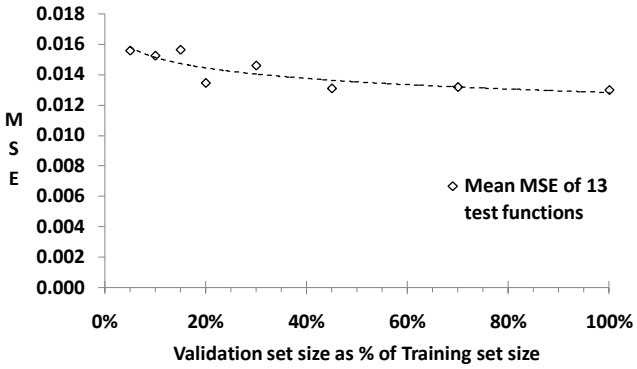

Figure 7 Reductions in the tested MSE with larger validation sets

A trend line has been added to Fig. 7 that shows the error reducing by 25% as we increase the size of the validation set

from 5% to 100% of the training set. The conclusion we make is that validation set sizes as small as 5% (or minimum size of 10 samples) can be relied on to achieve the early stopping behavior that we desire.

In column five of Table II, the results of early stopping are displayed. For all the experiments in this table, the training datasets were created from 48 samples per dimension and the validation sets were set at 20% of the size of the training datasets. The mean reductions in the MSE range from 10% to 57% due to early stopping (ES). In all test cases, early stopping has reduced the common tendency of the CasCor neural network to overfit.

C. Dispensing with a testing set

There was one other early stopping experiment for which we desired an answer. If we are using unseen data for our early stopping set, then could we dispense with a testing set entirely – relying only on the MSE calculated from the validation dataset? If this approach is viable then, in circumstances when creating datasets is time-expensive, we could dispense with the creation of a testing set - relying solely on our validation error as a test for the quality of our fit.

The results in Fig. 8 were generated from the same experiment performed for the results in Fig. 7. However, for each size of validation set, we also compared the MSE calculated from the validation set with the MSE calculated from our much larger testing sets ($1000 \times d$). The results suggest that validation sets of 20% or greater are sufficient to give a close approximation to the results from a much larger testing set. Taking a two dimensional test function as an example; the training set would have numbered $48 \times 2 = 96$ samples, and a 45% validation set would have been of size $96 \times 0.45 = 44$. The total number of samples we would have created $= 140$. With this validation set, Fig. 8 predicts that the MSE calculated from this, size $= 44$, validation set will be within 7% (σ = 5%) of the MSE calculated from a testing set of size $= 2000$ samples.

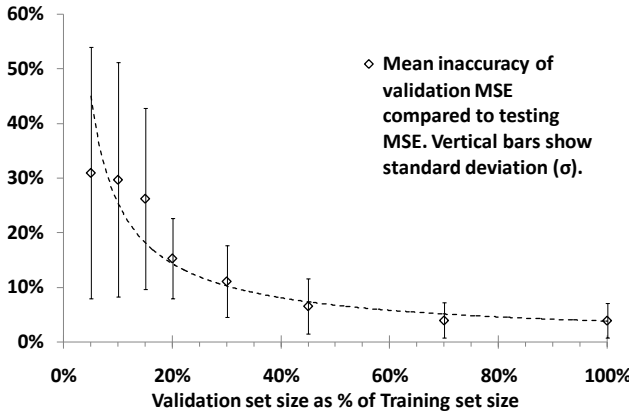

Figure 8 How close the validation MSE is to the testing set MSE

D. Ensembling

For clarity, only three of the thirteen test functions are shown in Fig. 9, however, the form of the line graphs were similar throughout all 13 functions; the MSE reduced rapidly as the ensemble size increased from one to seven. Smaller reductions in the MSE occurred until ensembles with a size greater than 25 were seen to deliver little benefit. We also used early stopping in this experiment and so the MSEs in Fig. 9 reflect the combination of both techniques.

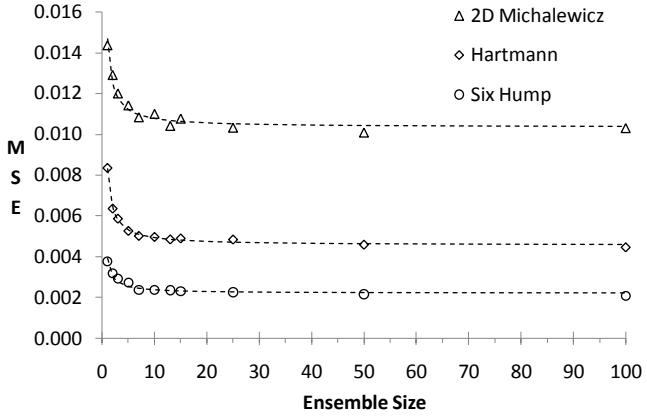

Figure 9 Reduction in mean squared error due to ensembling

The curves in Fig. 9 take the form:

$$MSE_{Ensemble} = \frac{\left(\overline{MSE}_{EnsSize=1}\right) - Bias^2}{EnsSize} + Bias^2 \tag{10}$$

where $\overline{MSE}_{EnsSize=1}$ is the mean MSE of the neural networks that constitute the ensemble. $Bias^2$ is the asymptote to which the curves tend. Effectively, the bias is an MSE boundary that no size of ensemble can reduce because ensembling acts only on the part of the error that is due to variance. Likewise the early stopping, provided by our validation set, acts only to reduce the variance by limiting overfitting.

Equation (10) can be derived from the equations presented in the seminal paper of Geman [16] where he describes the bias/variance dilemma of neural network training. The general form of the error is given in their paper as:

$$Error = Variance + Bias^2 \tag{11}$$

and it can be shown that (10) and (11) are equivalent. Equation (10) provides us a convenient test for the relative contribution of variance and bias to the overall error. Evaluating the MSE is a function commonly built into neural network libraries and so, using MSE evaluations alone, we can estimate the bias (12) and then the variance (13) for any ensemble. If variance is found to dominate, then creating a larger size of ensemble will reduce the MSE and improve the mapping of the underlying function. If we find that the bias is the largest component of our mapping error, we know that the information capacity of our CasCor neural network has been exceeded. Installing more neurons will confer additional capacity and patchworking provides that utility.

$$Bias^2 = \frac{(EnsSize \times MSE_{Ensemble}) - \overline{MSE}_{EnsSize=1}}{(EnsSize - 1)} \tag{12}$$

$$Variance = \frac{\left(\overline{MSE}_{EnsSize=1}\right) - Bias^2}{EnsSize} \tag{13}$$

By way of example, Fig. 10 presents a smaller region of Fig. 9 and, for clarity, only the Michalewicz data is

re-plotted. Say that an ensemble of size 10 has been created. We calculate the MSE of that ensemble and also calculate the mean MSE of the 10 members of that ensemble. Now, by using (12) and (13), we find that our $Bias^2$ = 0.01 and the *Variance* of our ensemble = 0.0004. Ensembling to a size of 15 would reduce our variance to 0.00027, but it is clear that the dominant component of our MSE is the bias. A CasCor ensemble that possesses a high bias indicates a highly multimodal function in the training dataset. When the MSE is undesirably high (and dominated by bias), the application of our patchworking method is advocated.

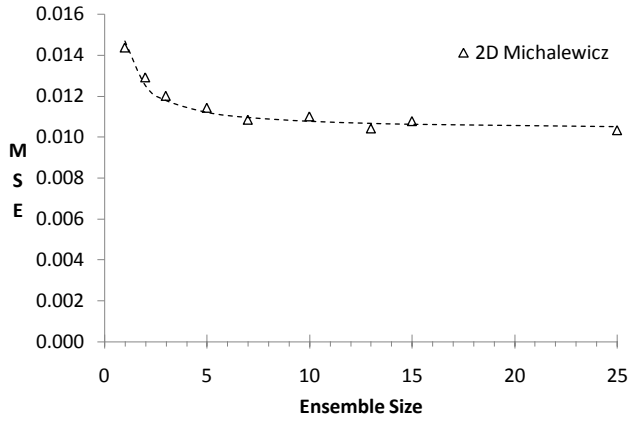

Figure 10 Michalewicz scatter plot

Table II Benefits of ES, Ens, and patchworking

Test function	Dims	Size of train + early-stop sets. Patchwork Off/On	$10^3 MSE$ Cascade Correlation (CasCor)	**CasCor + ES**	**CasCor with Patchworking**	**CasCor with Ens + ES**	**CasCor with Patchworking +Ens + ES**
Ackley	2	116/461	33.79	14.33	6.31	3.10	1.53
			Reduction in error:	57.59%	81.33%	90.82%	95.47%
DeJongs5th	2	116/461	176.33	80.06	33.20	58.10	11.23
			Reduction in error:	54.60%	81.17%	67.05%	93.63%
Langermann	2	116/461	77.33	33.32	3.82	22.43	1.48
			Reduction in error:	56.91%	95.06%	70.99%	98.09%
Michalewicz	2	116/461	22.90	14.38	5.23	10.78	3.27
			Reduction in error:	37.22%	77.16%	52.92%	85.72%
Schwefel	2	116/461	36.73	19.96	3.77	4.39	0.80
			Reduction in error:	45.67%	89.75%	88.06%	97.81%
Shubert	2	116/461	32.08	20.24	3.11	4.59	0.27
			Reduction in error:	36.89%	90.31%	85.69%	99.15%
Six Hump	2	116/461	13.39	6.77	1.46	4.26	0.36
			Reduction in error:	49.42%	89.09%	68.15%	97.34%
Ackley	3	173/1383	14.66	6.36	4.78	5.64	2.37
			Reduction in error:	56.62%	67.38%	61.56%	83.84%
Hartmann	3	173/1383	12.67	11.60	2.44	6.50	2.38
			Reduction in error:	8.40%	80.76%	48.66%	81.18%
Rosenbrock	4	231/3687	18.27	14.41	4.88	8.19	2.99
			Reduction in error:	21.10%	73.27%	55.18%	83.61%
Schwefel	4	231/3687	27.47	20.73	2.84	13.70	2.37
			Reduction in error:	24.51%	89.66%	50.12%	91.36%
Michalewicz	5	288/9216	10.64	9.52	1.74	5.38	1.35
			Reduction in error:	10.53%	83.62%	49.45%	87.35%
Schwefel	5	288/9216	44.77	22.61	3.66	22.07	1.55
			Reduction in error:	49.50%	91.83%	50.71%	96.54%
Average reduction in error				**39.15%**	**83.88%**	**64.57%**	**91.62%**

E. Patchworking

In Table II $\text{Ens}_{\text{size}} = 15$ was used and the basic CasCor results are shown alongside the benefits of early stopping, patchworking, ensembling (Ens) + early stopping (ES), and all three combined. Patchworking is applied to a depth of one. The same computer program was used to generate all the neural networks, the only changes being flags that turn on/off the features shown. Results shown are formed from the arithmetic mean of ten trials.

When compared to a standalone CasCor neural network, the mean effect of patchworking is to reduce the error 6.2 times. Employing ensembling and early stopping on these functions reduces the error by a mean factor of 2.8. However, the real benefit of patchworking is that it can be combined with the techniques of early stopping and ensembling – here delivering a mean reduction in neural network testing error of 11.9 times (91.6%).

F. Visualization of patchworking + Ens + ES results

Fig. 2 showed the ensembled and early stopped mapping of a small part of the Schwefel function. Similarly, Fig. 4 showed the ensembled and early stopped plot of part of the Langermann function. After patchworking to a depth of one, Figs. 11 and 12 show clearly the significant improvement achieved from using the patchworking method.

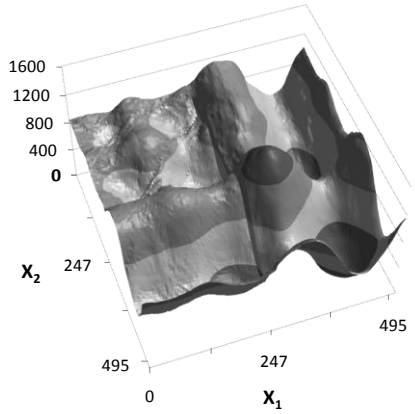

Figure 11 CasCor mapping of Schwefel (Patchworking + Ens + ES)

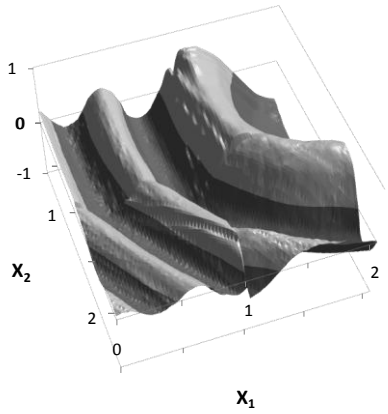

Figure 12 CasCor mapping of Langermann (Patchworking + Ens + ES)

G. Patchworking for larger depths and dimensions

From our experience with the CasCor neural network, no more than nine features can be mapped satisfactorily by one network alone. Taking the full domain of the two dimensional Schwefel function as an example, Fig. 13, we see significantly more than nine stationary points on this surface. Patchworking to a depth of one, Fig. 14, begins to approximate the Schwefel surface but, using the recursive facility of the patchworking algorithm, we can see the significant improvement in Fig. 15 when patchworking has been allowed to continue to a depth of three.

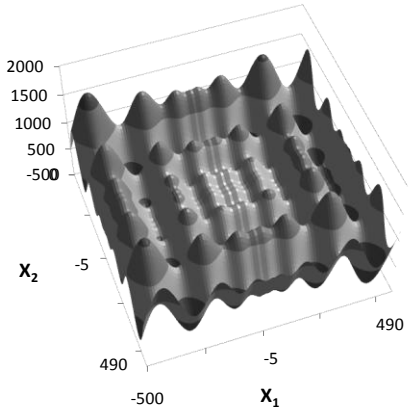

Figure 13 Schwefel function, range x(i) [-500,500]

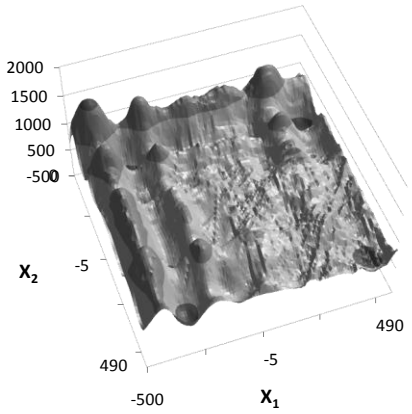

Figure 14 CasCor of Schwefel (Patchworking, depth=1 + Ens +ES)

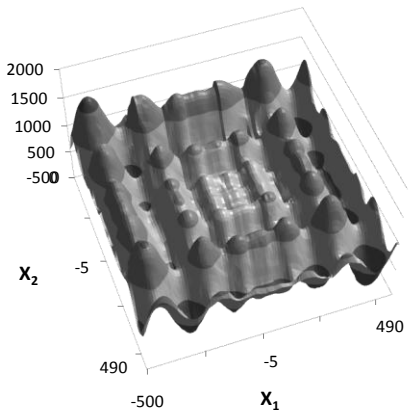

Figure 15 CasCor of Schwefel (Patchworking, depth=3 + Ens +ES)

The required sizes of training datasets per patch remain the same for any given problem, but the number of patches grows exponentially = $2^{(dept\ h \times dimensions\ )}$ therefore, so too will the total training data required. Some fields in which patchworking may be appropriate are those which already have very large datasets e.g. health databases, astronomy data, chemical process data, or any other collection of data samples where the data available is exponentially larger than

the dimensions of that data. The information capacity of patchworked CasCor networks also grows exponentially and so we can provide a calculation for the number of features that can be mapped. In the general case:

$$Maximum\ features\ mappable = 9 \times 2^{(depth x dimensions)}$$

Therefore, given an eight-dimensional problem, patchworking to a depth of one could map as many as 2,304 unique features in a training dataset numbering 98,304 samples.

V. Conclusion

The architecture of the Cascade Correlation neural network means that it is quick and simple to configure for training. However, its *weight freezing* mechanism hinders the mapping of multimodal functions. To address the *bias/variance* problem for this neural network type, three techniques have been presented: *early stopping*, *ensembling*, and *patchworking*.

Early stopping and ensembling have been shown to be valuable tools in reducing the variance component of error. Early stopping sets as small as 5% of the training set have been shown to be effective in reducing the variance error. Our work also suggests that there may be no need for a separate testing set. A validation set of size 45% of the training set can substitute for a testing set 45 times larger, returning an MSE calculation within 7% of the MSE from that testing set $(\sigma = 5\%)$. This offers the possibility of saving a significant amount of time that would otherwise have been spent sampling for a testing set.

Ensembling has been shown to be more effective than early stopping in reducing variance and, in the limit, will reduce the variance to zero. Equations have been presented in this work that will provide approximations for the variance and the bias of an ensemble using mean square error calculations alone.

Our *patchworking* technique has been introduced to reduce the bias component of error by raising exponentially the *information capacity* of the Cascade Correlation neural network. Although patchworking does require exponentially larger training datasets, it overcomes the weight freezing problem of this neural network type and leads to significantly improved fits for multimodal problems - yielding a reduction in error of over ten in some cases.

Appendix

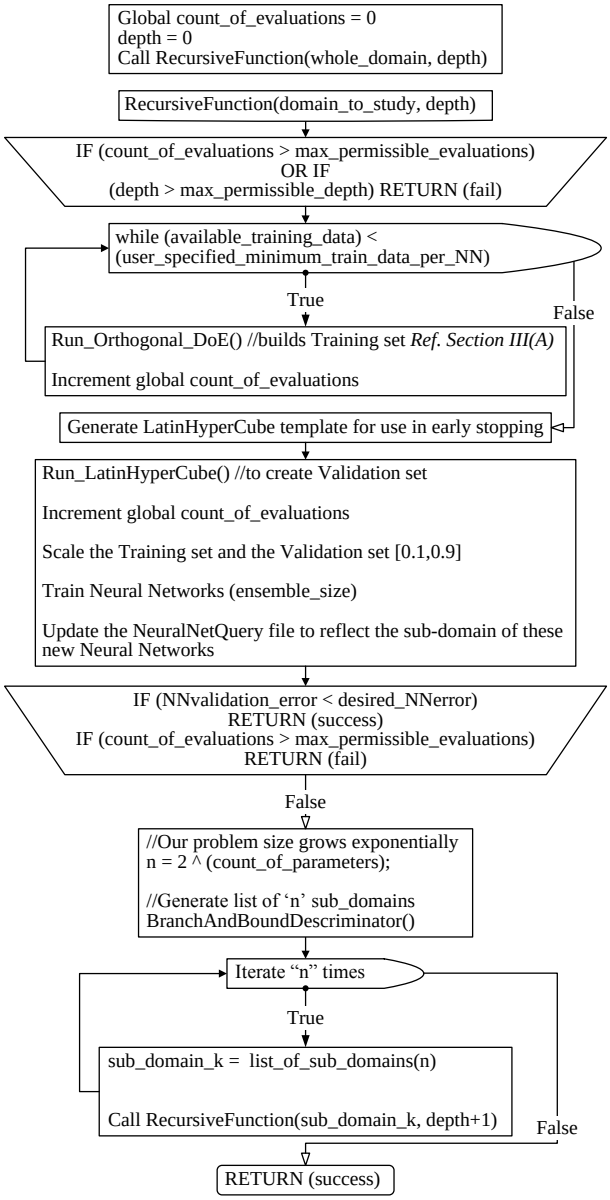

Figure 16 The Patchworking algorithm

Acknowledgment

The authors are grateful to Adeline Schmitz of California State University for a discussion about her ensembling/early stopping experiments with the cascade correlation neural network, and, Steffen Nissen for creating and sharing FANN. Mike Riley thanks Rick Drury for his comments on this paper.

References

[1] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," *National Science Foundation under Contract Number EET-8716324 and Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976 under Contract F33615-87-C-1499.*, 1991.

[2] L. Prechelt, "Investigation of the cascor family of learning algorithms," *Neural Networks*, vol. 10, pp. 885–896, 1996.

[3] M. J. W. Riley, C. P. Thompson, and K. W. Jenkins, "Improving the Performance of Cascade Correlation Neural Networks on

Multimodal Functions," *Lecture Notes in Engineering and Computer Science: Proceedings of The World Congress on Engineering 2010*, WCE 2010, 30 June - 2 July, 2010, London, U.K., pp. 1980-1986.

[4] I. V. Tetko and A. E. P. Villa, "An enhancement of generalization ability in cascade correlation algorithm by avoidance of overfitting/overtraining problem," *Neural Processing Letters*, no. 6, pp. 43–50, 1997.

[5] C. S. Squires, Jr., and J. W. Shavlik, "Experimental analysis of aspects of the cascade-correlation learning architecture," Computer Sciences Department, University of Wisconsin-Madison, Tech. Rep., 1991.

[6] T.Y. Kwok and D.Y. Yeung, "Objective functions for training new hidden units in constructive neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 1131–1148, Sep 1997.

[7] G. P. Drago and S. Ridella, "On the convergence of a growing topology neural algorithm," *Neurocomputing*, vol. 12, no. 2-3, pp. 171–185, 1996.

[8] S. Baluja and S. E. Fahlman, "Reducing network depth in the cascade-correlation learning architecture," Carnegie Mellon University, School of Computer Science, Tech. Rep., 1994.

[9] K. Mehrotra and S. Ranka, *Elements of artificial neural networks*. The MIT Press, 1996.

[10] FANN, *http://leenissen.dk/fann/*. Fast Artificial Neural Network Library. (Retrieved Jan 12th 2010.)

[11] B. Beachkofski and R. Grandhi, "Improved distributed hypercube sampling," in *43rd Structures, Structural Dynamics, and Materials Conference*, 2002.

[12] N. J. A. Sloane, *http://www2.research.att.com/ njas/oadir/*. A Library of Orthogonal Arrays. (Retrieved Feb 8th 2010.)

[13] W. F. Kuhfeld, *http://support.sas.com/techsup/technote/ts723.html*. Orthogonal Arrays Provided as a Service of SAS (Retrieved Feb 8th 2010).

[14] A. Forrester, A. Sobester, and A. Keane, *Engineering Design Via Surrogate Modelling: A Practical Guide*. Wiley Blackwell, 2008.

[15] A. Schmitz and H. Hefazi, "Constructive neural network ensemble for regression tasks in high dimensional spaces," *Sixth International Conference on Machine Learning and Applications*, pp. 266–273, 2007.

[16] S. Geman, "Neural Networks and the Bias/Variance Dilemma," *Neural Computation*, vol. 4, pp. 1-58, 1992.