# Ensemble Techniques for Avoiding Poor Performance in Evolved Neural Networks

John A. Bullinaria

*Abstract*— **The idea of using evolutionary techniques to optimize the performance of neural networks is now widely used, but some approaches have been found to result in the evolution of risky learning strategies that lead to occasional instances of very poor performance. In this paper I shall present a series of simulations that explore the nature of those problems, and examine the extent to which one can use ensemble techniques to alleviate them.**

## I. Introduction

An important application area for neural networks is in autonomous agents – systems that are expected to learn from, or adapt to, their environment without any external assistance. Typically this means that they must learn to generalize as well as possible, as quickly as possible, from as little training data as possible. It is now well established that using simulated evolution is an extremely powerful approach for optimizing such performance [1, 2], but recently it has been observed that some of the most obvious evolutionary approaches can lead to the evolution of rather risky learning strategies that sometimes result in very poor performance [3, 4]. Of course, if we are employing a whole population of essentially disposable autonomous devices, then occasional instances of disastrous performance will not be too problematic. However, if we have a single autonomous system, sent at great expense to another planet (say), then the possibility of an occasional lapse that could lead to its destruction would clearly be unacceptable. The aim of this paper is to look in more detail at the problematic learning mechanisms that can arise for evolved neural networks, and then explore the possibilities for avoiding these problems using ensemble techniques.

In the next section I shall outline the main evolutionary approaches that are appropriate for optimizing autonomous neural network systems. I then describe a series of simulations that explore the relevant issues for a simplified class of classification tasks that cover the crucial properties of many real world situations. The results of those simulations are then presented and analyzed, and the potential problems inherent in the emergent systems become clear. This leads on to a discussion of ensemble techniques, and an exploration of how they can be used to alleviate the problems that the evolved networks suffer. I then look briefly at the idea of avoiding the problems by using evolved time dependent learning rates, and show how the same ensemble techniques are able to produce improved performance in that case too. I end with some discussion and conclusions.

J. A. Bullinaria is with the School of Computer Science, The University of Birmingham, Birmingham, B15 2TT, UK (phone: +44 121 414 2590, e-mail: j.bullinaria@physics.org).

## II. Neural Network Evolution

The aim of neural network evolution here is to generate autonomous systems that can be sent into a new environment and learn to perform (i.e., generalize) as well as they can, as quickly as they can, on whatever task is given to them. They will receive a steady stream of inputs from their environment and be expected to produce appropriate outputs (i.e. actions). We will assume that they obtain feedback after they have produced their output as to whether it was correct or not, and hence be able to learn from their mistakes. Obviously, if we knew the exact task in advance, we could do the training in advance, and would not need an autonomous system. For autonomous systems, we will normally only have some idea of the class of environments/tasks that can be expected to arise, and so the aim is to evolve systems that will have good performance in any situation from that class.

The simulated evolution proceeds by taking a whole population of neural networks, and generating a stream of inputs corresponding to one environment from the class of expected environments, and measuring how well each network learns to perform. We then take the fittest (i.e., best performing) individuals to create the next generation of neural networks, using appropriate forms of cross-over and mutation. Such evolution by 'natural selection' will result in good innate properties (e.g., parameter values) proliferating in the populations, and poor ones being lost. If we change the environment for each new generation, we eventually evolve a population of neural networks that will work well in any environment they find themselves in.

There are several distinct approaches to simulated evolution that one can follow [3]. The simplest are the *generational* approaches in which the individuals compete with each other one generation at a time. If we allow each individual to learn from a fixed number $N$ blocks of training examples and test their generalization performance during the last block, we can use that performance as their fitness and form the next generation from the fittest. We shall call this the basic *G1* generational approach. We can see that this is likely to cause good generalizers to evolve, but there appears to be nothing to encourage the good performance to arise quickly, in less than $N$ blocks of training data. It is also far from obvious how one can choose an appropriate value of $N$ before we have even started the evolutionary process.

A sensible refinement over the *G1* approach is to start off

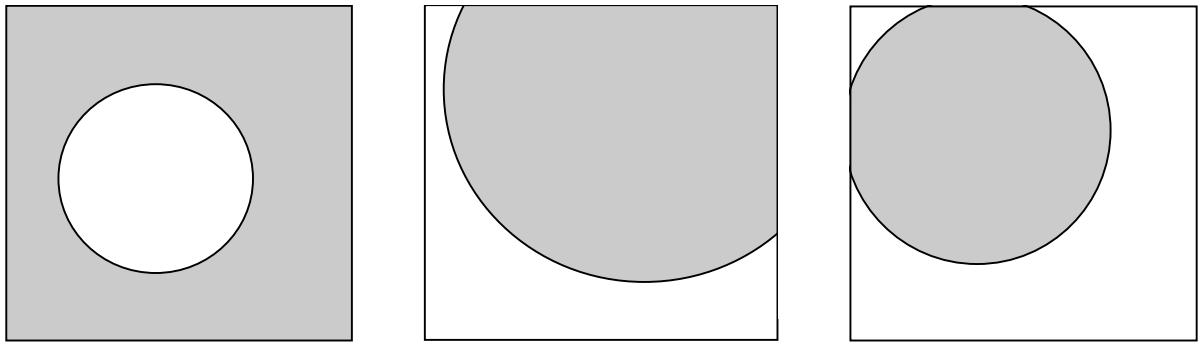

Figure 1. Three typical training data sets – circular classification boundaries in a 2D continuous input space.

with $N$ at some sufficiently large value, and as soon as the evolution has resulted in (say) half the population reaching a clear maximum performance in that time, average the generalization performance after $N$-1 training data blocks too. If we keep measuring the fitness over increasingly large numbers of data blocks at the end of training, in line with the improving performance, we encourage the networks to learn more and more quickly. The first *G1* stage of evolution improves the generalization, and the second improves the speed. We shall call this the *G1+G1* approach.

An alternative approach for improving the learning speed is to take the fitness to be the number of blocks of training data required to reach the maximum level of performance. This is the *G2* approach. The obvious problem with this is that we will generally not be able to know in advance what the maximum level of performance is likely to be. However, we can start the evolution with the *G1* approach to establish that level, and then switch to the *G2* approach. We shall call this the *G1+G2* approach.

It is clear that most biological populations evolve in a rather different manner. They typically include competing individuals of all ages, with relatively few individuals replaced by children at each stage. The need to compete with more experienced individuals inevitably encourages faster learning of good performance. We shall call such *steady state* evolution the *SS* approach.

## III. Simulation Details

To understand the similarities and differences between the neural networks produced by the four evolutionary approaches just described (*G1, G1+G1, G1+G2, SS*), we need to run some simulations, which requires us to narrow down to a specific class of tasks and type of network. Most real-world classification tasks involve learning non-linear classification boundaries in a space of real valued inputs. So let us look at the class of simple classification tasks, based on a two dimensional continuous input space with circular classification boundaries. Some representative examples are shown in Figure 1. This set-up is simple enough to allow extensive simulations, yet covers the crucial features and difficulties of many real world problems. Each network is assigned a random classification boundary which it must learn from a stream of randomly drawn data points. The performance measure will be the generalization ability, i.e. the average number of correct outputs (e.g. within 0.2 of the binary targets) *before* training on them.

Again for simplicity, we shall take our neural networks to be traditional Multi-Layer Perceptrons with one hidden layer, sigmoidal processing units, trained by gradient descent using the cross-entropy error function that is appropriate for classification tasks [2]. As previous studies have shown, one gets better performance by evolving separate learning rates $\eta_L$ and initial weight distributions $[-r_L, +r_L]$ for each of the four distinct network components $L$ (the input to hidden weights $IH$, the hidden unit biases $HB$, the hidden to output weights $HO$, and the output unit biases $OB$), rather than having single parameters for the whole network [2, 3]. These, together with a standard momentum parameter $\alpha$ and a weight decay regularization parameter $\lambda$, give us a total of ten evolvable innate parameters for each network. One could also evolve the number of hidden units, but doing this usually results in the maximum allowed number being used, slowing down the simulations considerably [4], so we kept this fixed at 20 for all networks, which is plenty for the given tasks, yet not so large as to place unnecessary demands on our computational resources.

Each of the evolutionary approaches involve a number of additional parameters that we must set. To ease comparison between the steady state and generational approaches, we define a 'simulated year of experience' to be a block of 1200 training data samples. In the steady state simulations, pair-wise comparisons of fitness after each simulated year select 10% of the least fit individuals to be removed from the population. In addition, a random 20% of individuals aged over 30 simulated years are removed each year, to prevent the populations being dominated by a few very old and very fit individuals. In the generational simulations, the least fit 50% of the population are removed and the most fit 50% are regenerated from their innate parameters at each generation, which for the *G1* stages corresponds to 60 simulated years. A population size of 200 is maintained throughout, with the removed individuals being replaced by children generated from random pairs of the most fit individuals. Each child inherits innate parameters that are chosen randomly from the corresponding range spanned by its two parents, plus a

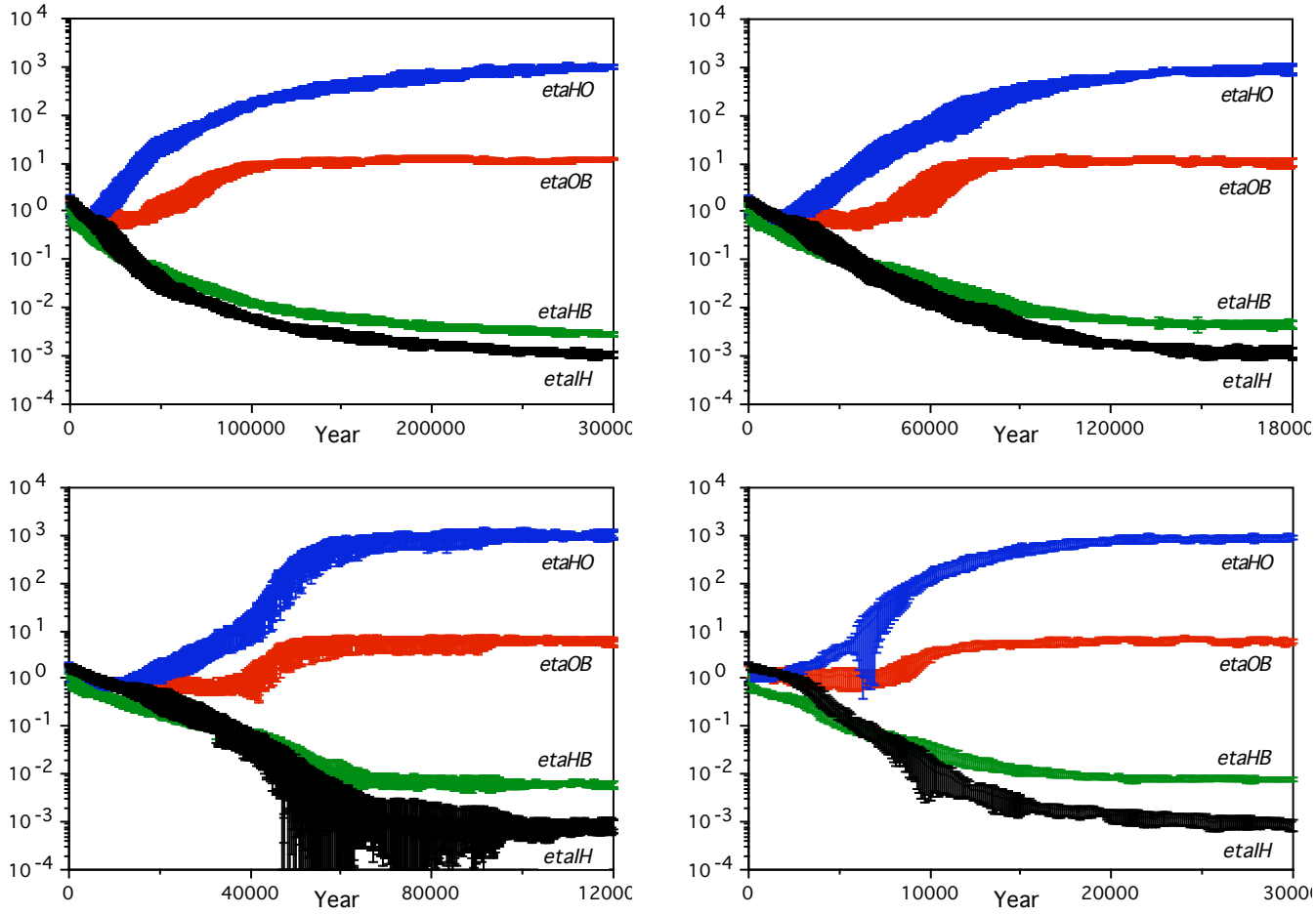


Figure 2. Evolution of the neural network learning rates, with means and standard deviations over 10 runs, for the four evolutionary approaches: *G1* (top left), *G1+G1* (top right), *G1+G2* (bottom left), *SS* (bottom right).

random mutation (from a Gaussian distribution) that gives it a reasonable chance of falling outside that range.

## IV. Simulation Results

If we generate our initial populations randomly with their innate parameters chosen in the range from zero to about twice those used in traditional hand-crafted networks, we stand a good chance of starting with a range of reasonably competent individuals. Our simulated evolution as specified above should then be able to adapt those parameters to optimize the performance. Occasionally, if we fail to maintain the population diversity adequately, the evolution can become stuck in local maxima of fitness, particularly in the *SS* approach. However, for present purposes at least, it is easy to spot the resultant inferior 'species' and simply disregard them [3]. All of the following results will be presented as averages over 10 successful runs of each type using different random numbers.

The evolution of the learning rates for each of the four approaches are shown in Figure 2. We see that a similar pattern emerges in each case, but over rather different evolutionary time scales. The *G1* approach takes something like 300 thousand simulated years to settle down, the *G1+G1* approach takes around 180, the *G1+G2* approach takes around 120, and the *SS* approach requires only around 30. These differences are a clear reflection of the increasing amounts of evolutionary pressure to improve the learning speeds across the four approaches. It is interesting to note that all four approaches exhibit similar large differences between the evolved learning rates for the four network components, and that they take on values that differ by orders of magnitude from those traditionally used. Evolving a single learning rate for all four network components results in a value of around 0.2, which is much more in line with the values traditionally used in this type of network.

We find analogous patterns of evolution for the other innate network parameters. The important property here though, is the resultant performance of the evolved networks. Since the idea is to evolve networks that can perform well whatever training set they are given from the specified class, each evolved individual was tested on 100 different randomly chosen data sets. Figure 3 shows the mean errors and their variances, as a function of age, for each of the four main approaches. For comparison, it also shows a typical population only allowed single parameters

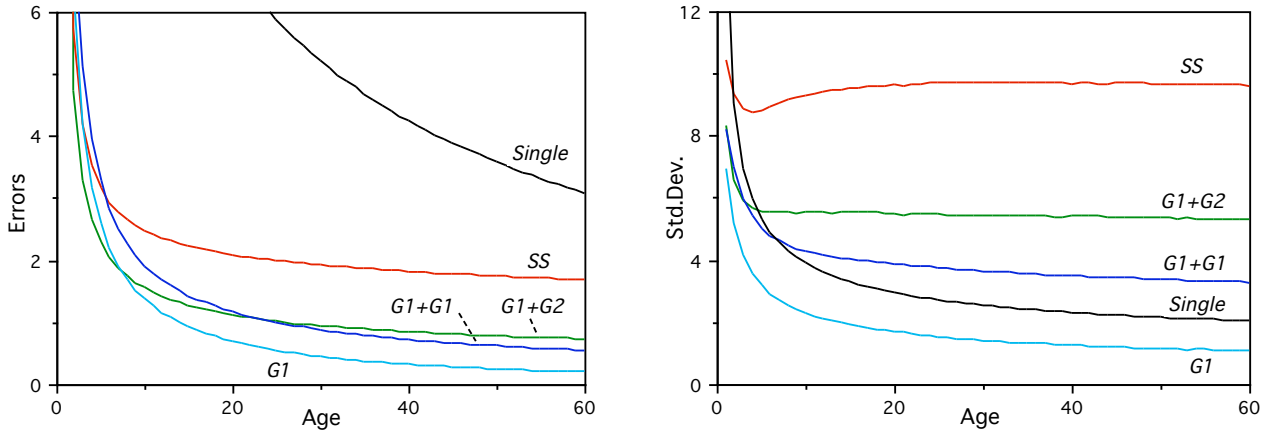


Figure 3. The error rates during learning for the evolved networks: means (left) and variances (right). The four main populations are shown (*SS, G1, G1+G1, G1+G2*), plus the single component *G1* population (*Single*).

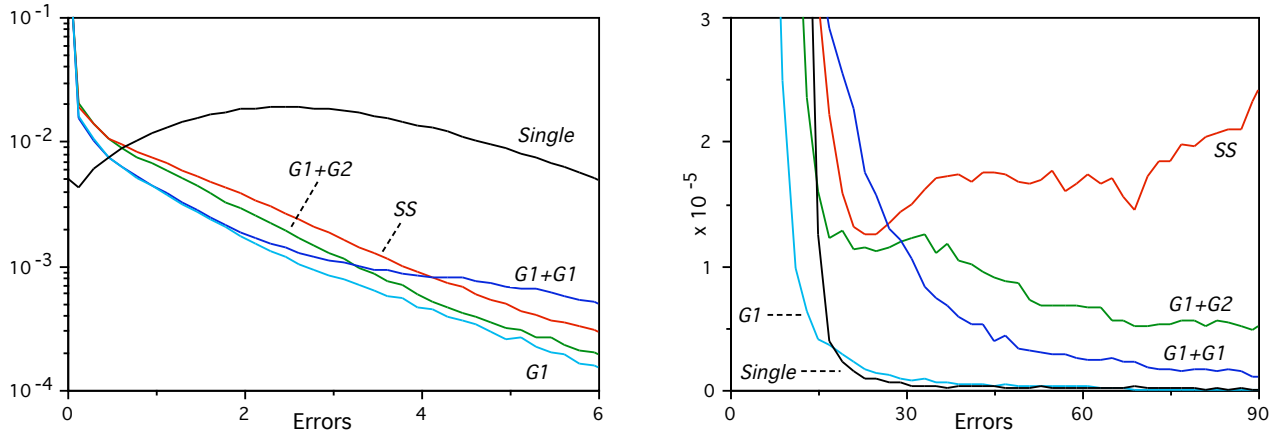


Figure 4. Two views of the average error distributions between ages 50 and 60: peaks (left) and tails (right). The four evolved populations are shown (*SS, G1, G1+G1, G1+G2*), plus the single component *G1* population (*Single*).

across all four network components. The standard errors on the means are a factor of 140 smaller than the standard deviations shown, so the differences between the means across approaches are quite significant, despite the evolved learning rates looking rather similar. In both the errors and the variances, we see again the hierarchy of the approaches in terms of forcing faster learning, and a trade-off between speed and accuracy. It is also clear from the relative sizes of the variances and means that the error distributions must be somewhat skewed, and this warrants further investigation.

Figure 4 shows the error distributions averaged over evolved individuals between the ages 50 and 60, by which time little further learning takes place. We see that there is relatively little difference in the peaks across the four main evolutionary approaches, with each of them showing a massive concentration around zero errors, as one would hope. The tails of the distributions, however, are much more variable, with the same hierarchy that we saw in Figures 2 and 3. The evolutionary approaches that specifically encourage faster learning result in individuals producing many times the number of very large errors than the *G1* and single component cases. In effect, the faster learners are adopting riskier learning strategies that work well overall, but occasionally results in extremely poor performance.

The fact that the riskier strategies emerge is evidence of their benefit from an evolutionary point of view, but to judge whether they are worthwhile for our more practical concerns, it is instructive to look more carefully at their effect on faster learning. The median error rates will not be influenced by the long tails of the error distributions in the same way as the means. These are shown on the left of Figure 5. The usual hierarchy is still apparent, but the differences don't appear to be that great. The differences are clearer, though, if we plot the distribution of ages at which individuals reach their first complete year of perfect performance, as shown on the right of Figure 5. The wide distributions reflect the range of task difficulties, as well as individual differences. As we go through the hierarchy of approaches, the peak shifts from around eight years for *G1* to around four for *SS*, which really could be a worthwhile level of improvement for many applications.

A closer investigation of the pattern of errors reveals that the problems are not due to a small number of consistently very poorly performing individuals, nor due to brief periods

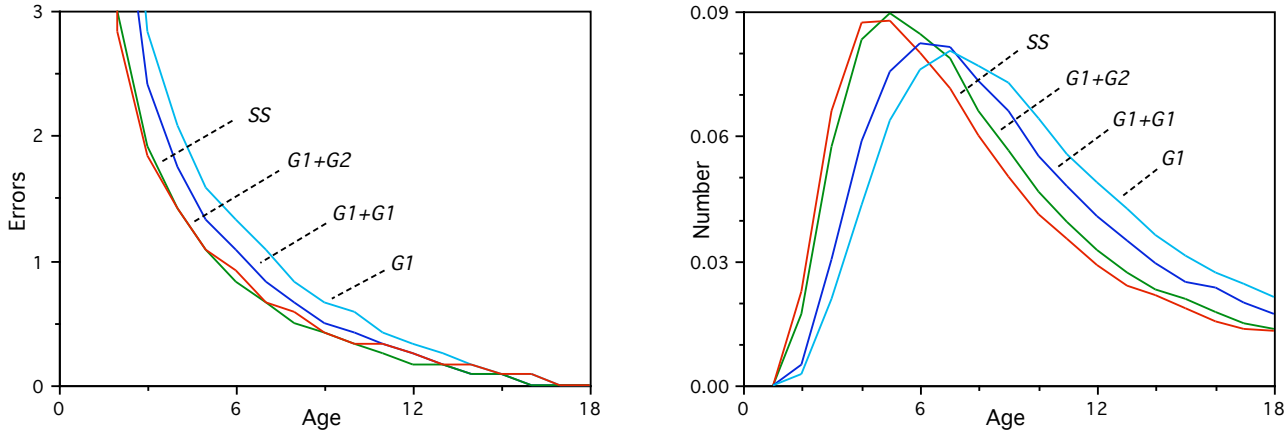


Figure 5. The median error rates during learning for the evolved networks from each approach (left), and the corresponding distributions of ages at which individuals reach their first complete year of perfect performance (right).

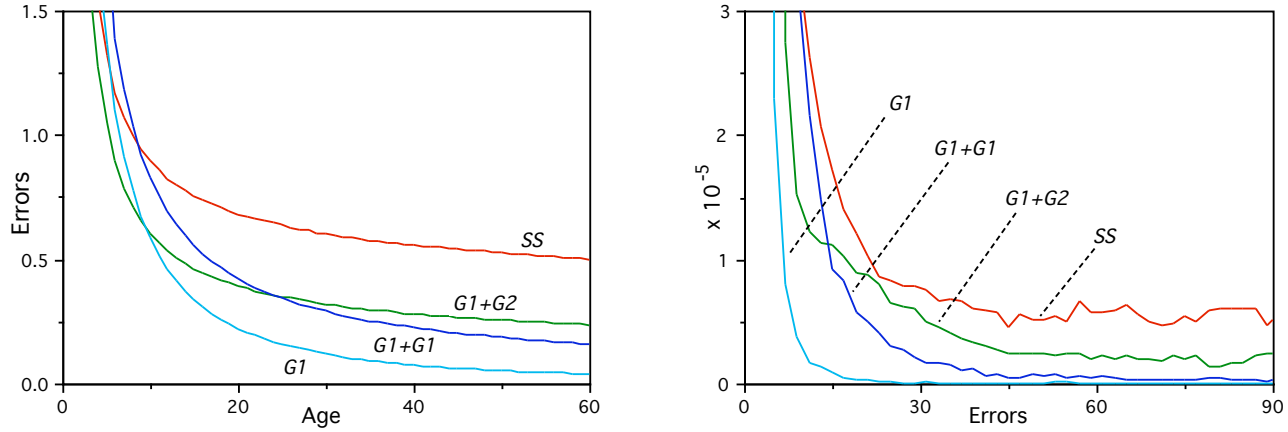


Figure 6. The error rates during learning for the evolved networks when each network uses voting over three runs starting from different random initial weights (left), and the corresponding average error distributions between ages 50 and 60 (right).

of very poor performance during lifetimes of otherwise reasonable performance. They actually arise from different individuals not being able to cope at all on a small number of tasks, and it seems difficult to predict which individuals will have problems on which tasks. The remainder of this paper will look at ways of avoiding this unfortunate situation.

## V. Ensemble Performance

Given that the instances of very poor performance are fairly rare, it is quite possible that a simple committee or ensemble approach could be sufficient to remove them. The basic idea is that if an ensemble of three or more individual neural networks arrive at a combined output by a simple voting procedure, the occasional instances of very poor performance will be out-voted and no longer be a problem (e.g., as discussed in [5, 6]). Such an approach has certainly proved to be successful with other types of evolved neural networks in the past (e.g., [7]). However, if the individual networks all make the same rare mistakes, we will clearly need to adopt a more sophisticated approach.

Three is clearly the minimum number that can form a workable voting committee, so we shall investigate that first. There are two obvious ways to proceed: we can take three independent individual networks from each evolved population, or we can take a single individual and train it three times starting from different sets of random initial weights drawn from its innate distribution. Both of these approaches will (assuming we do not have the ability of parallel processing) increase the total learning time by a factor of three, but we may still end up with better overall performance. Figure 6 shows the mean error rates during learning, and the error distributions between ages 50 and 60, when we have voting over three runs of each individual. Compared with the corresponding results for single runs of each individual, we see reductions in errors by factors of three or four, which is encouraging, but there remain a significant number of very large errors. Figure 7 shows the corresponding results for voting ensembles made up of three independent individuals. This is clearly a much more successful approach for avoiding the large error cases, as one might expect given the reduced likelihood for error correlations across the ensemble [8], so we shall concentrate on this approach from here on.

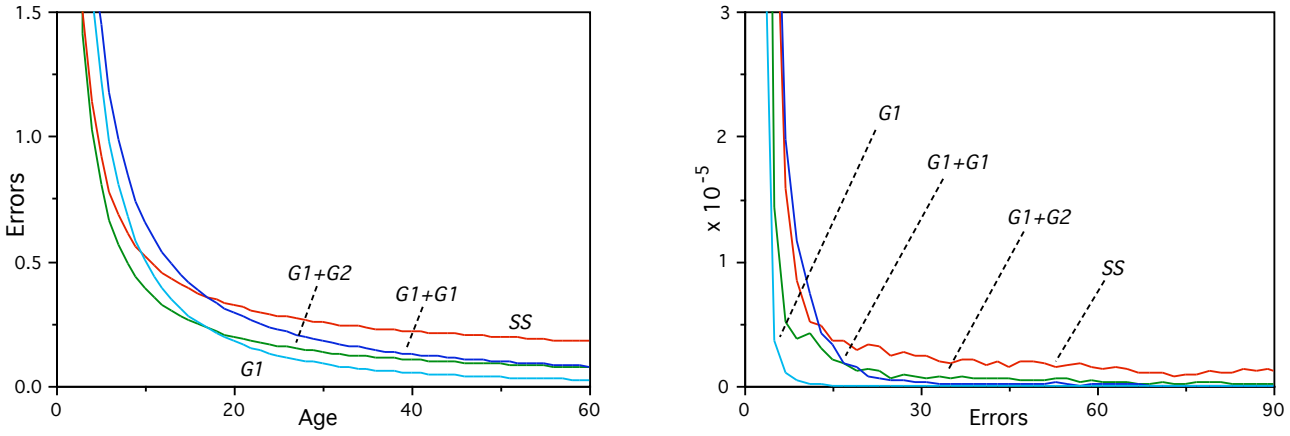


Figure 7. The mean error rates during learning for the evolved networks using voting in groups of three independent individuals (left), and the corresponding average error distributions between ages 50 and 60 (right).

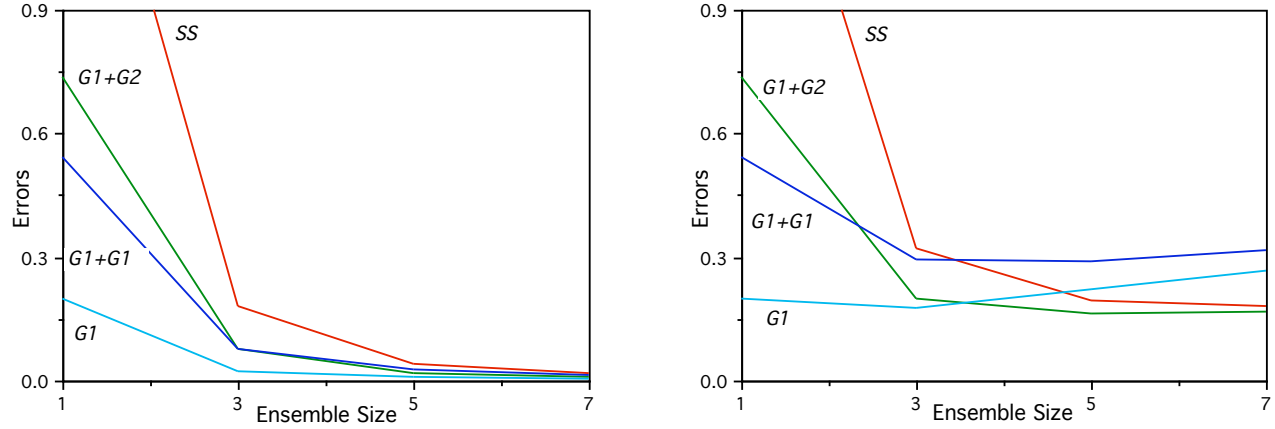


Figure 8. The mean error rates at age 60 for increasing ensemble sizes (left), and the corresponding error rates when the computational effort is taken into account, i.e. counting errors at age 60/*EnsembleSize* (right).

The natural extension to voting in threes is to consider larger voting ensembles. On the left of Figure 8 we see how the mean error rates at age 60 are reduced with increasing ensemble sizes. However, if we take into account the extra computational effort of multiple training runs, i.e. look at the error rates at ages 60/*EnsembleSize,* we get the rather different pattern of results seen on the right of Figure 8. There is now relatively little improvement overall for ensemble sizes greater than three, and the *G1* and *G1+G1* approaches actually get worse.

A useful measure of the rate of occurrence of the problematic very large error cases is the total proportion of errors above 30 arising between the ages of 50 and 60. This is plotted for the various ensemble sizes on the left of Figure 9, and shows clearly the expected benefit of the ensemble approach. In this case, taking the computational effort into account, i.e. dividing each of the specified ages by the *EnsembleSize*, makes very little difference. This is because the large errors persist throughout a small number of training runs, but are absent from a very early age in the majority of runs. It would seem then, that ensembles of five or seven do have significant benefits in this respect, even if the additional computational costs are taken into account.

## VI. Age Dependent Plasticity

Human learning is known to involve, in many cases, age dependent plasticity with associated critical periods for learning [9], and my own earlier studies have already shown how such features can be evolved in artificial neural network systems to give improved performance [10, 4]. I shall now briefly explore how such ideas interact with the ensemble approach discussed above.

There exist a number of powerful approaches to adaptive plasticity (e.g., [11]), but for current purposes a simple age $t$ dependent scale factor multiplying the learning rates will suffice. A general piecewise linear scale factor typically evolves to take a form that is not far from an exponential fall to a baseline value [10]. To simplify matters here, we therefore consider a simple two-parameter scale factor $s(t)$:

$$\eta_L(t) = s(t)\,\eta_L(0) \quad , \quad s(t) = \beta + (1-\beta)\,e^{-t/\tau}$$

in which the time constant $\tau$ and baseline $\beta$ are allowed to

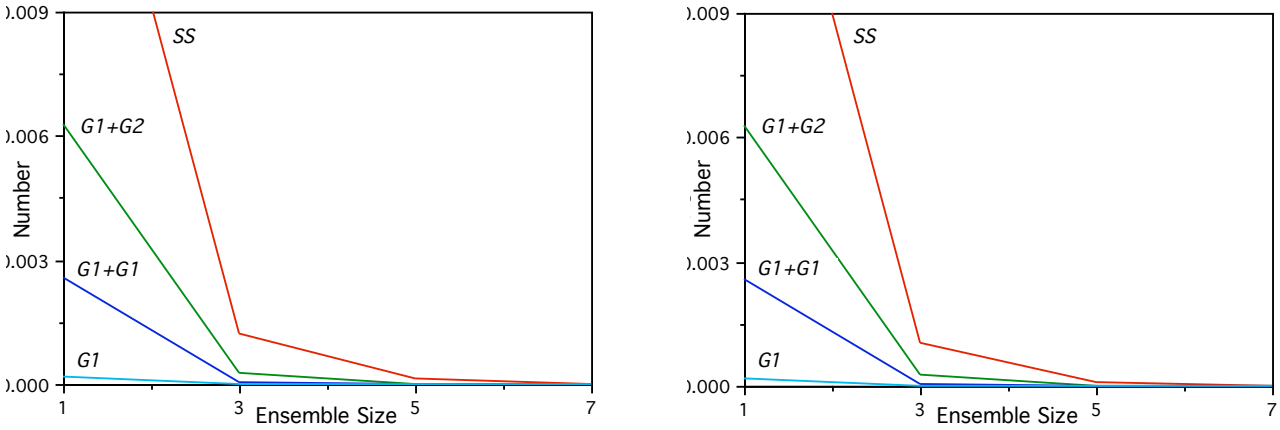


Figure 9. The total proportion of large errors (above 30) between ages 50 and 60 (left), and the corresponding proportions when we the computational effort is taken into account, i.e. using ages/*EnsembleSize* (right).

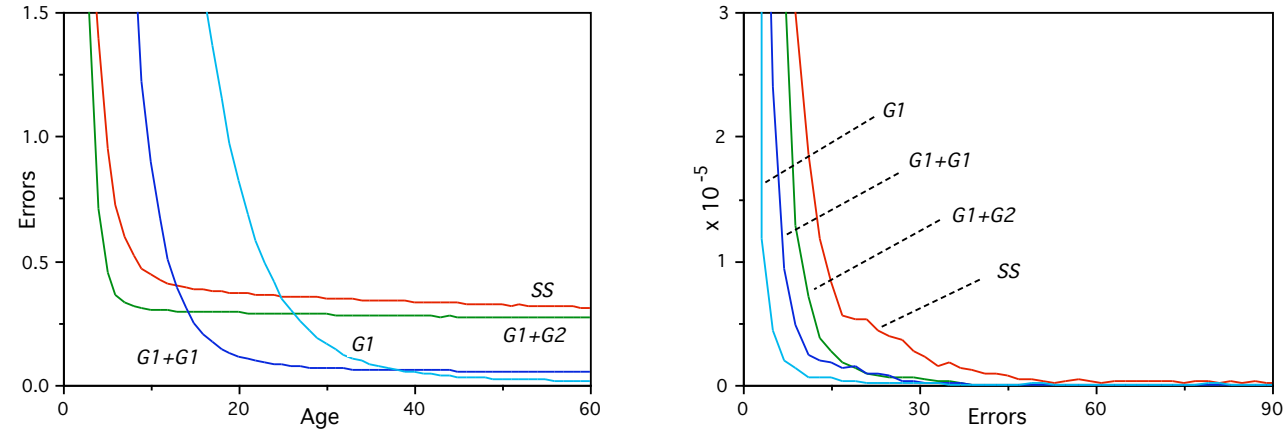


Figure 10. The mean error rates during learning for evolved networks that are allowed age dependent learning rates (left), and the corresponding average error distributions between ages 50 and 60 (right).

evolve as additional innate parameters in the same four evolutionary approaches considered above.

The simulations reveal that the interaction of these two new parameters with the existing parameters is rather complex [4]. In each approach, the evolved learning rates fall exponentially with age by a factor of about 10, but the time constants $\tau$ vary from around 20 years for *G1*, to around 7 for *G1+G1*, and around 2 for *G1+G2* and *SS*. These result in the mean error rates during learning and the old age error distributions shown in Figure 10. All four approaches show clear reductions in the number of large error cases, in some ways better than the reductions obtained by the ensemble approach (seen in Figure 7). However, in terms of the mean errors at each age, the differences are mixed. There are clear improvements in final mean error, for all four approaches, over the single networks of Figure 6, but the *G1* and *G1+G1* approaches achieve this at the expense of a much slower reduction in errors at earlier ages. Comparison with the ensemble results of Figure 7 also shows mixed differences, with the variable plasticity *G1* and *G1+G1* slower to learn initially, but reaching lower errors by age 50, and the opposite pattern for *G1+G2* and *SS*.

The important question to ask now is: can the ensemble approach further improve the variable plasticity results, or have we reached the limit of what is achievable? Figure 11 shows the mean error rates at age 60 for the various ensemble sizes, for comparison with the fixed plasticity results in Figure 8. The slower initial reduction in errors for the variable plasticity *G1* and *G1+G1* cases will clearly be problematic if we take the computational cost into account, but otherwise, the ensembles do give further improvement. This is particularly clear in the graphs of Figure 12 showing the proportion of very large error cases, where we see that variable plasticity and ensembles together can completely eliminate the problem of occasional very large errors.

## VII. Discussion And Conclusions

We have studied the evolution of neural network classifier systems that would form the basis of many autonomous systems that are required to learn good generalization performance as quickly as possible from a continuous stream of training data. Simulations of four natural evolutionary approaches have shown clearly that the more one encourages

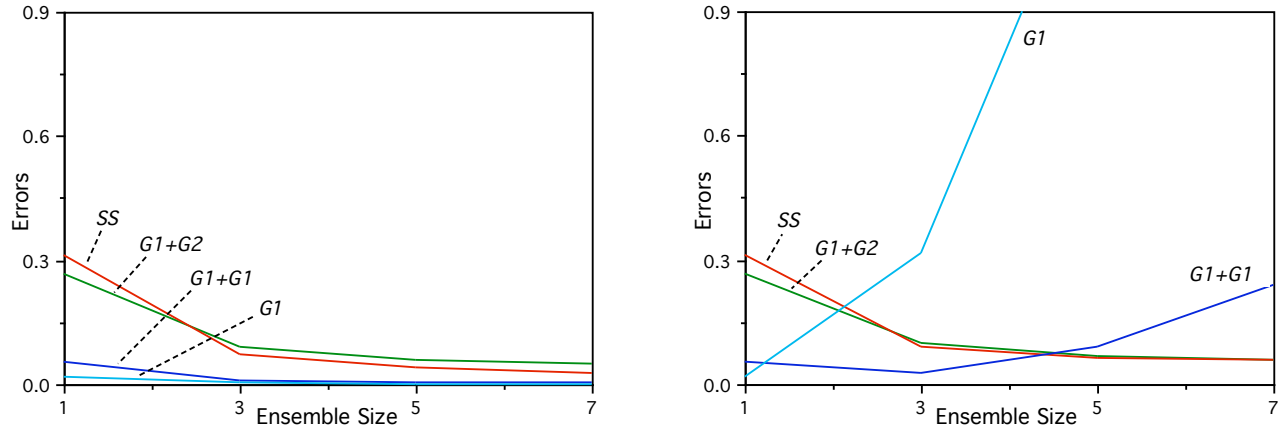


Figure 11. The mean error rates at age 60 for ensembles of the variable plasticity networks (left), and the corresponding error rates when the computational effort is taken into account, i.e. counting errors at age 60/*EnsembleSize* (right).

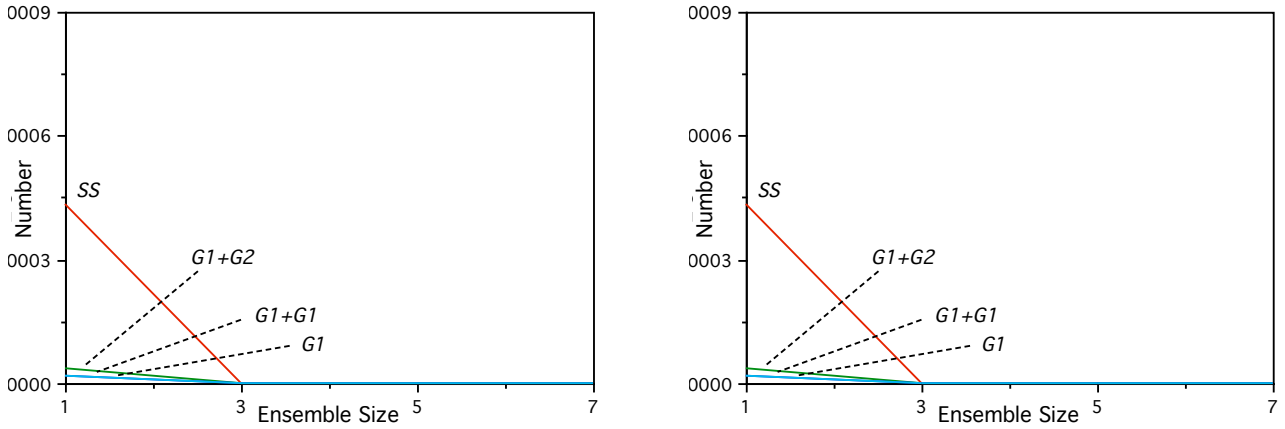


Figure 12. The total proportion of large errors (above 30) between ages 50 and 60 for ensembles of the variable plasticity networks (left), and the corresponding proportions found when using ages/*EnsembleSize* (right).

faster learning, the more likely will be the emergence of risky learning strategies that occasionally result in very poor performance, which would be potentially disastrous for many real world applications.

We then saw how these problems could be significantly reduced by using appropriate ensemble techniques, and also, to a similar degree, by evolving learning rates that decrease during the learning process. By employing both of these techniques together, we found that it is possible to eliminate the problematic high error cases completely. Of no less practical importance are the more detailed results presented here, showing clearly the trade-offs between the various approaches and performance measures. It is these that will allow better informed design choices that will surely depend on which aspects of evolved performance are most crucial for each particular application.

## REFERENCES


[1] X. Yao, Evolving Artificial Neural Networks. *Proceedings of the IEEE*, **87**, pp. 1423-1447, 1999.

[2] J. A. Bullinaria, Evolving Efficient Learning Algorithms for Binary Mappings. *Neural Networks*, **16**, pp. 793-800, 2003.

[3] J. A. Bullinaria, Generational versus Steady-State Evolution for Optimizing Neural Network Learning. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2004)*, pp. 2297-2302. Piscataway, NJ: IEEE, 2004.

[4] J. A. Bullinaria, Evolved Age Dependent Plasticity Improves Neural Network Performance. In: *Proceedings of the Fifth International Conference on Hybrid Intelligent Systems (HIS 2005)*, pp. 79-84. Piscataway, NJ: IEEE, 2005.

[5] L. K. Hansen, and P. Salamon, Neural Network Ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, pp. 993-1000, 1990.

[6] R. Battiti, and A. M. Colla, Democracy in Neural Networks: Voting Schemes for Classification. *Neural Networks*, **7**, pp. 691-709, 1994.

[7] X. Yao, and Y. Liu, Making Use of Population Information in Evolutionary Artificial Neural Networks. *IEEE Transactions on Systems, Man, and Cybernetics, B*, **28**, pp. 417-425, 1998.

[8] K. Turner, and J. Ghosh, Error Correlation and Error Reduction in Ensemble Classifiers. *Connection Science*, **8**, pp. 385-403, 1996.

[9] D. B. Bailey, J. T. Bruer, F. Symons, and J. W. Lichtman, (Eds), *Critical Thinking About Critical Periods*. Baltimore: Brookes, 2000.

[10] J. A. Bullinaria, From Biological Models to the Evolution of Robot Control Systems. *Philosophical Transactions of the Royal Society of London* A, **361**, pp. 2145-2164, 2003.

[11] R. A. Jacobs, Increased Rates Of Convergence Through Learning Rate Adaptation. *Neural Networks*, **1**, pp. 295-307, 1988.