# Evolving Neural Networks for Improved Incremental Learning Performance

**John A. Bullinaria**

School of Computer Science, University of Birmingham
Edgbaston, Birmingham, B15 2TT, UK

*j.a.bullinaria@cs.bham.ac.uk*

**Abstract**: It is well known that incremental learning is often difficult for traditional artificial neural networks, due to newly learned information interfering with what was previously learned, and this proves problematic for cognitive models. This paper present a series of computational experiments that explore the extent to which evolutionary techniques can be used to generate simple neural network incremental learners that exhibit improved performance over existing "hand-crafted" networks. The nature of the learning difficulties are inevitably application dependent, so a sequence of representative cases are considered. These range from pure memory networks, that must learn new items without forgetting old items, to generalization networks, that must learn from new items to improve their classification performance. It is shown for all cases that the evolutionary optimized traditional neural networks perform considerably better than equivalent neural networks with standard parameter values, casting doubt on the need for more complex specialist incremental learning systems. The implications for brain modelling are discussed.

**Keywords:** Neural networks, Evolution, Incremental learning, Catastrophic forgetting, Memory, Generalization.

## 1. Introduction

Good *incremental learning* is the ability to learn effectively from new information as it becomes available, without needing access to previous information (Polikar, Udpa, Udpa & Honavar, 2001). This is of practical importance for humans and many real world applications where learning really does need to be an ongoing process (Giraud-Carrier, 2000). The human brain is certainly good at usefully absorbing new information into what it already knows. Traditional artificial neural networks (Bishop, 1995), on the other hand, usually work best if they are trained on all the available data during a single learning session, and it is often difficult to improve them by updating with new information that becomes available later. This poor incremental learning ability is obviously problematic, both for building brain models, and for real world applications. The problem is that they do not handle the *stability-plasticity dilemma* (Grossberg, 1987) very well – the learning of new data tends to interfere with the previously learned information.

While it is quite normal for humans to gradually forget what they have previously learned, particularly during the learning of new information, in traditional artificial neural networks the forgetting tends to be more catastrophic, and this proves to be a serious limitation of associated cognitive models (McCloskey & Cohen, 1989; Ratcliff, 1990; French, 1999, 2003). Human brains have presumably evolved by natural selection to minimize this problem (Sherry & Schacter, 1987). This paper aims to take inspiration from that, and explore the extent to which simulated evolution can be used to improve the performance of artificial neural networks too.

The neural networks should be able to use any new training data to improve their performance, without requiring access to the previous data. This may involve accommodating whole new classes of information that are introduced with the new data, and remembering old classes that are not represented in the new data. There are two broad application areas to consider: memory tasks and generalization tasks. Memory tasks have been much studied in the cognitive modelling literature, and involve the learning of new data patterns without forgetting those learned previously. Generalization tasks have been studied more in the context of practical Artificial Intelligence (AI) applications, with the aim of further improving existing generalization abilities as new data arrives. Actually, both types of task are relevant to both cognitive modelling and practical AI applications, and both will be studied in this paper.

The remainder of this paper begins by reviewing the idea of incremental learning in general, and what this means for generalization and memory tasks. Then the principal previous approaches for improving incremental learning performance and avoiding catastrophic forgetting in memory tasks

are outlined. The idea of using simulated evolution to improve neural network performance is then considered, and the particular approach adopted for this study is described. The main part of the paper consists of empirical studies of evolved neural network learners for representative memory and generalization tasks, comparisons against existing systems, and analyses of how the evolved networks achieve their improved performance. The paper ends with some conclusions and a discussion of how the results tie in with known brain structures and their evolution.

## 2. Incremental Learning

Chalup (2002) has provided a wide ranging review of the various types of incremental learning in both biological and machine learning systems, and Lange & Grieser (2002) have presented a more theoretical analysis of incremental learning. For present purposes, a successful incremental learning system will be defined, following Polikar et al. (2001), as one that involves the training data becoming available in batches and satisfies the following useful properties:

1. No access is required to any of the previous training data from which the current state was learned.
2. Learning from new training data does not cause large scale forgetting of previously acquired information.
3. Additional information can be acquired from the new data, and hence the performance can be improved.
4. There is no problem in accommodating any new data classes that are introduced in the new training data.

Human brains have these properties, yet achieving them in artificial neural networks seems to pose serious difficulties.

When a neural network that has been trained on one set of data is subsequently trained on a new set, the performance on the original set is affected. This is a direct consequence of the stability-plasticity trade-off, and is known to be a major problem for artificial neural network models. The difficulty arises because the crucial feature that gives connectionist networks their generalization and graceful degradation abilities, the single set of shared weights that forms a distributed representation of the old information, necessarily gets modified as the new information is learned. The aim, of course, is to find a way of modifying the weights in a manner that is consistent with the four incremental learning properties listed above.

For memory tasks, attempts at incremental learning frequently result in catastrophic forgetting, whereby the new training patterns seriously disrupt the patterns that were previously learned (French, 1999, 2003). This is because the new training patterns generally have little overlap with the old patterns, and consequently it is difficult to satisfy incremental learning properties 1 and 2.

For generalization tasks (both classification and regression), the interference will generally be much less catastrophic, particularly if the new training patterns only represent minor changes to, or more information about, the existing classification boundaries or regression curves, but one still needs to ensure that the new data improves the overall generalization performance rather than making it worse. In particular, it is important to ensure that the additional training does not cause over-fitting, and it is not obvious how best to incorporate standard regularization techniques such as stopping early and weight decay (Bishop, 1995) into incremental regimes. Here, incremental learning properties 1 and 2 tend to be less of a problem, but satisfying the other two properties is not so straightforward.

## 3. Previous Approaches

Numerous processes have already been developed with the aim of allowing neural networks to learn new information without disrupting the existing information. Perhaps the most obvious procedure is *interleaved learning* (Ratcliff, 1990; McClelland, McNaughton & O'Reilly, 1995), whereby the entire original training data set is mixed together with the new patterns, and the network is retrained on the new expanded set. This can involve continued training of the existing network, or discarding the previously learned weights and starting the training from new random initial values. Either way, the key incremental learning properties are satisfied, except that access to past data is required, which clearly violates property 1. This approach clearly needs some refinement as a model of human learning, and also tends to be impractical for real world applications in that it requires permanent storage of all the old data for retraining. Moreover, the nature of the learning task often varies over time anyway, rendering the old data unhelpful.

The interleaving approach may also prove computationally expensive due to the longer training times of the increasingly large training sets. One can attempt to reduce the size of the full training set by only using carefully chosen subsets of the new patterns. Engelbrecht & Britis (2001), for example, devised a system whereby the candidate set of new training patterns is divided into clusters, and only the most informative pattern from each cluster is added to the training set. This approach, however, still has the problem of needing access to all the old data, and as the number of clusters

increases, so does the computational complexity.

An extension of interleaving, that avoids the need to use the entire original training set, involves employing *rehearsal* (Ratcliff, 1990; Robins, 1995), whereby only a subset of the original data is used for further training. By choosing such subsets appropriately, variations over time and recency effects can also be incorporated. Better still, *pseudo-rehearsal* (Robins, 1995, 2004; Robins & McCallum, 1999) has the advantages of rehearsal without requiring access to the original data. Here, after training the neural network on the original data, it is used to generate *pseudo-patterns* by producing output vectors for randomly generated input vectors. These pseudo-patterns approximate the earlier learning of the network, to a degree that depends on their distribution throughout the input space, and can be used for training along with the new data, hopefully preserving what was learned before. This approach has been shown to reduce substantially the interference between sequential training items while allowing new information to be learned, but there remain the problems of storing the pseudo-patterns and knowing how many of them to produce and how to distribute them in the input space. Frean & Robins (1999) carried out a formal analysis of the pseudo-rehearsal approach for linear networks and showed that it is "guaranteed to succeed in high dimensions under fairly general conditions", but found that it can fail in low dimensions.

Since the underlying cause of catastrophic forgetting is interference in the shared weights, many approaches have attempted to reduce that interference. Typically they restrict the way in which the hidden unit activations are distributed in order to modify the connection usage. French (1992) explored activation sharpening algorithms to reduce the overlaps between hidden unit activations for different input patterns. McRae & Hetherington (1993) showed how pre-training the neural networks could affect the pattern of hidden unit activations in a such a way that catastrophic interference was reduced. The HARM (Hebbian Autoassociative Recognition Memory) model of Sharkey & Sharkey (1995) effectively implements a neural network lookup table in which learning is divided into two stages, first eliminating the overlap between input patterns at the hidden layer using an orthogonalizer, and then using Hebbian learning to map the hidden representations to appropriate outputs. Other approaches have involved allowing two sets of weighted connections between the nodes, such as the additive dual-weights of Hinton & Plaut (1987) with fast weights to learn new patterns temporarily and slow weights for long-term storage. Robins (1997, 2004) has discussed how a second set of weights can be used as an alternative to explicit storage of pseudo-patterns in the pseudo-rehearsal approach.

More complex modular/localist neural network approaches have also been proposed. A range

of ART (Adaptive Resonance Theory) networks (Grossberg, 1987; Carpenter & Grossberg, 1988, 2003) use pattern matching to enable the rapid learning of new information while preserving previously learned patterns. Similarly, the CALM (Categorization and Learning Module) approach of Murre (1992) and the ALCOVE (Attention Covering Map) model of Kruschke (1992) have also attempted to avoid catastrophic forgetting, in rather different ways, by recognizing new patterns for storing on uncommitted nodes.

Further approaches have been based more directly on the brain structures that humans are believed to employ for these tasks (McClelland, McNaughton & O'Reilly, 1995; O'Reilly & Rudy, 2000). The idea is that humans do not suffer from catastrophic forgetting because their brains have evolved two distinct areas to deal with the problem (Sherry & Schacter, 1987): a hippocampal system that allows rapid learning of new information, and a neocortical system that slowly consolidates the new information with the old for long-term storage, presumably using some form of interleaved learning. Some very successful dual-model architectures consisting of two distinct networks, one for early processing and another for long-term storage of previously learned information, together with an information transfer mechanism using pseudo-patterns, have been developed to simulate this separation (French, 1997; Ans & Rousset, 1997, 2000; French, Ans & Rousset, 2001; Ans et al., 2002). Robins (1996) and Robins & McCallum (1999) have also considered the relation of such consolidation processes to those that are thought to take place in the sleeping brain.

Other researchers, more concerned with generalization problems, have developed somewhat different approaches to eliminate the need for accessing the old training data. One particularly successful approach has been the *Learn++* algorithm of Polikar et al. (2001) which employs an ensemble of weak classifiers that generate multiple hypotheses by sampling the training data using carefully customized distributions. Empirical results for a range of benchmark classification problems have shown how this algorithm satisfies all four incremental learning properties listed above (Polikar, Byorick, Krause, Marino & Moreton, 2002; Muhlbaier, Topalis & Polikar, 2009). Unfortunately, it involves numerous important parameters that need their values set by hand, and it is not at all clear how the algorithm might relate to the processes that take place in brains. Further approaches have been developed to cope with incremental learning, such as the Neocognitron of Fukushima (2004), ILUGA (Incremental Learning Using Genetic Algorithm) of Hulley & Marwala (2007) and the NCL (Negative Correlation Learning) approaches of Lin, Tang & Yao (2008) and Minku, Inoue & Yao (2009), but with neural architectures and algorithms of considerably increased

complexity. Recently, simpler approaches have also had some success, but at the expense of requiring access to past data (e.g., Albesano, Gemello, Laface, Mana & Scanzio, 2006).

All the previous approaches noted above have been based on extensions of traditional neural networks, with the designers themselves deciding on the architectures, learning algorithms, and various associated parameter values. The proposed approach here, to be presented in the remainder of this paper, is to return to more traditional non-modular feed-forward neural networks with simple gradient descent learning algorithms and employ evolutionary techniques to optimize them. The aim is to determine whether the usual problems of incremental learning can be sufficiently minimized in these simple neural networks that more complex architectures are not required. That will provide better artificial systems to deal with these tasks, and better understanding of the computational limitations of such networks. In turn, that will elucidate the driving pressures behind human brain evolution, offer insights into how brains might have evolved to be good incremental learners, and provide a starting point for relating such emergent simulated networks to known brain structures. That will result in a more solid foundation for future brain modelling studies.

## 4. Evolving Neural Networks

The general idea of applying the basic principles of evolution by natural selection to optimize the performance of neural networks is now widely used (e.g., Yao, 1999; Bullinaria, 2003, 2007a,b; Cantû-Paz & Kamath, 2005). A population of individual neural networks is maintained, each with a genotype representing an appropriate set of innate parameters. Then for each generation throughout the evolutionary process, the "fittest" individuals, i.e. those exhibiting the best performance on their given task, are selected as parents. Suitable crossover and mutation operators are then applied to those parents to generate offspring for populating the next generation. This process is repeated, hopefully creating increasingly fit populations. Such an approach can be used to select optimal network architectures, learning algorithms, transfer functions, connection weights, and any other network parameters.

An important feature of evolving neural networks in this way is that any aspect of the neural network can be encoded and subjected to the evolutionary process, and it is possible for many parameters that interact in complex manners to be optimized simultaneously. This means that the crucial, and usually extremely difficult, task of setting good neural network parameter values can be left completely to the evolutionary process, rather than having to be done by hand by the network designer. The performances obtained from evolved neural networks are regularly reported to be

significantly better than traditional hand built neural networks (Yao, 1999; Bullinaria 2003, 2007a,b).

In this paper, the aim is to evolve the various neural network topology and learning parameters to produce systems that are good incremental learners with minimal catastrophic forgetting. The connection weights themselves are determined, as in humans, by the lifetime learning algorithm, rather than being specified as innate parameters by the evolutionary process (Elman et al., 1996). To satisfy incremental learning property 1, that access to past data is not required, the evolved incremental learning performance is measured on data distinct from that used by the evolutionary optimization process, in the same way that humans have evolved to perform well on data that is distinct from, but of the same type as, that experienced by previous generations. The underlying network architecture and learning algorithm will be fixed to be standard Multi-Layer Perceptrons with one hidden layer and sigmoidal processing units, trained by gradient descent weight updating (back-propagation) with the Cross Entropy error measure (Bishop, 1995; Bullinaria 2003).

The simulated evolution involves populations of individual neural networks, each learning their weights starting from random initial values drawn from their own innately specified distributions. The process starts from an initial population with random innate parameters. Then, for every generation, each network goes through the chosen incremental learning process and has its fitness (i.e. learning performance) determined. The fittest half of the population are copied into the next generation, and also randomly select a mating partner to produce one child, thus restoring the population size. The children inherit their innate characteristics (i.e. parameter values) randomly from the corresponding ranges spanned by both parents, with random Gaussian mutations added to allow final values outside the parental ranges (Bullinaria, 2003, 2007a,b). For each new generation, all the networks, both copies and new children, learn starting from new random initial weights. This process is repeated for a sufficient number of generations that no further improvements are evident and a network, or group of networks, that best satisfy the performance criteria is obtained.

The application of simulated evolution is not totally straightforward, however. Obtaining optimal networks relies on maintaining diversity in the populations, and the evolutionary process can easily become trapped in local optima, particularly when computational resource limitations restrict the size of the population. Using appropriate initial populations is important, as is identifying good representations, crossovers and mutations, but often one must simply run the simulations many times and discard any that have clearly not achieved their full potential (Bullinaria, 2007a).

The main difficulty is in deciding which of the innate parameters in the genotype are worth evolving, and which are better left fixed. This paper aims to explore a fairly complete range of

possibilities, and determine which factors contribute to improved incremental learners, and which are no help at all. Since it is likely that different architectures and parameter values will emerge from the evolutionary process depending on the details of the incremental learning problem under study, the two extreme cases are initially considered separately: first pure memory tasks, and then pure generalization tasks. That then leads to a better understanding of what is happening when mixed cases are considered. All the simulation results presented are based on populations of 100 individuals, and involved at least ten independent evolutionary runs using different random numbers.

## 5. Memory Tasks

The empirical part of this study begins by looking at catastrophic forgetting in pure memory tasks. The immediate problem is that humans and other animals typically learn items one at a time after a single presentation, and that appears incompatible with the neural network learning of batches of items over many epochs of training that this study aims to investigate. To proceed, one has to assume that the training patterns are maintained somehow, either individually or in batches, long enough for sufficient numbers of epochs of gradient descent training to take place. Although many of the details remain uncertain, the idea that the hippocampus provides such an initial storage medium in the human brain has considerable support (e.g., McClelland, McNaughton & O'Reilly, 1995; Nadel & Moscovitch, 1997), so this paper will concern itself only with the later gradient descent learning phase. How realistic sensory inputs get encoded into suitable internal representations for that learning to take place efficiently is something that also remains a topic of ongoing research, so again this paper will leave that to be studied elsewhere (e.g., O'Reilly & McClelland, 1994; Stern et al., 1996; Deadwyler & Hampson, 1999; Eichenbaum, 2001, 2004; Davachi, 2006; Rolls & Kesner, 2006). However, once it is clear what emerges from the simulated evolution of simple gradient descent based networks, that will then need to be placed into the context of known memory systems and how they may have evolved (e.g., Sherry & Schacter, 1987; O'Reilly & Rudy, 2000). What is needed here is a convenient memory association task that is simple enough for large numbers of simulations to run reasonably quickly, yet complex enough to be representative of realistic problems, and flexible enough for a range of batch sizes and task difficulties to be explored.

For consistency and ease of comparison with earlier work, a variation of the task used by Hinton & Plaut (1987) was chosen, namely random associations of 12 bit random binary patterns with 6 bits "on". Sparser representations might have been more ecologically valid (e.g., Weliky et

al., 2003; Olshausen & Field, 2004), but that would render the task easier, and a task is needed for this study that will not run into ceiling effects before the full range of potential performance improvement mechanisms have been exhausted. The aim here is to have brain like solutions emerge from the evolutionary simulations, not to build them in from the beginning. To ensure that networks evolve which can cope with the general task, rather than particular sets of training patterns, new random data sets of this specification were generated for each generation. For every generation, each network was trained on the same initial set of patterns until it had learned all those patterns (i.e. had all its output activations within a particular *learning tolerance* of their target values) or until a maximum number of epochs of training had been reached. They were then trained on a new set of patterns in the same manner. When the new patterns had been learned to the same tolerance as the original patterns, each network was tested again to see how well the original patterns were remembered. This was measured as the percentage of output units over all the original patterns that were still "correct" (i.e. within a particular *testing tolerance*). The fittest individuals were naturally those with the highest percentage remembered.

## 5.1 Baseline Performance

Before embarking on the evolutionary simulations, some baseline performance levels were established using traditional neural network learning parameters. This allowed the identification of tasks of appropriate complexity, and provided starting points for the evolutionary study. Enough of the potential 12 bit random binary associations needed to be used to make the learning task reasonably hard and statistically representative, yet not too many that it rendered the proposed systematic investigation computationally infeasible. Each network was therefore trained on 20 such patterns until the error on each output bit was less than the learning tolerance of 0.1. It was then trained in the same way on a different set of $N$ such patterns, and the number of output bits remembered correctly over the original 20 patterns was determined using an increased testing tolerance of 0.2. The left graph in Figure 1 shows the mean percentages remembered over 2500 individuals with different random training data sets, trained using a traditional back-propagation learning rate of 0.1 and random initial weights uniformly distributed in the standard range [–1, +1], for four different values of $N$. There is a fairly large variance (of the order of 5%) in all the results due to some data sets being "easier" than others, but the large number of individuals leads to the small standard errors shown on the means. Naturally, the fewer new patterns learned, the better the original ones are remembered. The best remembering performance in each case is obtained using

very large numbers of hidden units.

That having more hidden units results in less interference has been observed before (e.g., Yamaguchi, 2004), but one must not be too quick to draw this conclusion. A well known difficulty with analyzing neural network performance is that the various parameters can interact in unexpected ways, and this can lead to misleading results. For example, simply increasing the testing tolerance to 0.5, which allows the maximum level of interference in the output activations without "correct" outputs being more wrong than right, leads to the changed pattern of results seen in the graph on the right of Figure 1. Here, increasing the number of hidden units provides improvement initially, but then performance worsens, before rising and leveling off, sometimes at a level above the first peak, and sometimes below. Exactly how this depends on the other training parameters is not clear. The key idea behind this paper is that evolutionary computation techniques can be used to optimize simultaneously all the relevant parameters, including the number of hidden units, to achieve the best possible remembering performances. The crucial questions are whether the resulting performance levels of evolved simple neural networks can reach those of the more complex hand-crafted neural networks that have been designed to deal with the same problem, and how they relate to the evolution of known brain structures.

## 5.2 Evolving the Learning Parameters

As might be predicted from the left graph of Figure 1, a difficulty that the proposed evolutionary approach quickly runs into is that the evolved number of hidden units $N_{Hid}$ just keeps on increasing. On a serial computer, that brings the simulations to a standstill, the crucial learning parameters never settle down to optimal values, and a fully systematic investigation is rendered computationally infeasible. The various "design problems" relating to changes in real brain sizes have been discussed by Kaas (2000), and Striedter (2005) provides a good overview of the evolutionary pressures that determine the sizes of brains and brain regions. The crucial issue is that real brains have various processing overheads (such as growth costs, energy/oxygen consumption, heat dissipation, wiring volumes, and such like) which constrain their sizes. Ultimately, the models should include a brain size related cost in the fitness function that will prevent the number of hidden units growing unrealistically, but that depends on being able to reliably assign such costs according to the relative complexities of the tasks and hidden layers in the models, informed by the corresponding factors in real brains, and that is currently proving impossible. A simpler approach is to assume that there will be some maximum number of hidden units $N_{Max}$ corresponding to each

given task, and simply include that as a hard maximum in the model. Since it is difficult to know what that limit should really be, for the following simulations it was first set at the computationally feasible number of 50, which is close to the first performance peaks in the right graph of Figure 1, and then at 10000, which is in the region where performance stabilizes in both graphs of Figure 1. This allows the evolutionary process to concentrate on optimizing the other parameters, while providing an indication of how the results depend on the number of hidden units.

The first idea to be explored was the suggestion of French (1992) that "sharpening" the hidden unit activations might reduce forgetting by developing semi-distributed representations in the hidden layer which lead to less interference between new and old training patterns. At each epoch of training, the input to hidden weights are modified in such a way that the $N_H$ highest activation hidden units become closer to one, and the $N_L$ lowest activations become closer to zero, by a "sharpening factor" of $\alpha$ times the difference. Two variations were simulated. First, the sharpening parameters ($N_H$, $N_L$, $\alpha$) were evolved with the constraint $N_H + N_L = N_{Hid}$ so that all the hidden unit activations were modified. In this case, the sharpening factor $\alpha$ invariably evolved to zero, leaving a standard network. Second, $N_H$ and $N_L$ were allowed to evolve freely. In this case, they both evolved quickly towards zero, again leaving a standard network. The conclusion is that neither form of node sharpening is able to effectively reduce catastrophic forgetting, at least for this class of training data with traditional learning parameters. As a check, the node sharpening parameters were left free to evolve in all the subsequent simulations, in case some added evolved flexibility made them profitable, but in all the simulations the node sharpening was quickly "turned off".

After this initial negative result, attention turned to evolving the more traditional learning parameters in the hope that deviations from the standard values could improve matters. A preliminary study (Seipone & Bullinaria, 2005a) showed that evolving values for each parameter alone provided improved performance, but the parameters interacted in such a way that the best performances were achieved by evolving many of them together. For example, for 50 hidden units, 4 new training patterns and testing tolerance of 0.2, the remembering performance went up from the 68% baseline to 78% by evolving just the learning rates, 75% by evolving just the initial weight distributions, and 87% if they were evolved together. Consequently, for this study, all the traditional neural network parameters were evolved together:

1. The number of hidden units $N_{Hid}$ subject to some specified fixed maximum $N_{Max}$.
2. The connectivity levels between layers ($c_{IH}$, $c_{HO}$), specified as the proportion of possible

connections that are used by the network, with the actual connections chosen randomly.

3. The gradient descent learning rates $\eta_L$, for which earlier studies have established that allowing different values for each of the four network components $L$ (input to hidden weights *IH*, hidden unit biases *HB*, hidden to output weights *HO*, and output unit biases *OB*) can result in massively improved performance over having a single value for all of them as in traditional hand crafted networks (Bullinaria, 2003).

4. The random initial weight distribution for each network component $L$. There are several options for specifying these, such as means and standard deviations of Gaussian distributions $G(\mu_L, \sigma_L)$, but here the lower and upper limits of uniform distributions $[-l_L, u_L]$ were used.

5. A *Sigmoid Prime Offset* $o_{SPO}$ that is added to the sigmoid derivative factor in the hidden layer weight updates to prevent them going to zero when the sigmoids saturate (Bullinaria, 2003).

6. A weight decay regularization parameter $\lambda$ that can act to prevent over-fitting of the training data (Bishop, 1985).

7. The node sharpening parameters described above ($N_H$, $N_L$, $\alpha$).

8. The output error tolerances that determine when a particular output activation is deemed "correct", with different values for the learning and testing phases ($t_l$, $t_t$).

Successful evolution of all these details simultaneously is difficult, and, even with careful choices of initial populations and mutation rates, around 15% of runs fail to evolve well, for example because they become trapped in a local optima of fitness. This is a well known problem when computational feasibility severely restricts the population sizes, and the easiest solution is simply to repeat the runs many times and discard those runs that fail (Bullinaria, 2007a).

The initial populations were created with random parameter values drawn uniformly from traditional ranges (learning rates, initial weight distribution parameters, and connectivity proportions from [0, 1], weight decay parameters from [0, 0.001], sigmoid prime offsets from [0, 0.2], tolerances from [0, 0.5], and numbers of hidden units from $[0, N_{Max}]$). The precise starting parameter ranges were found to have little effect on the final results, but poor values often led to an extremely slow start to the evolutionary process.

Figure 2 shows how the key parameter values then co-evolved for a maximum of 50 hidden units and 4 new training patterns, with the corresponding performance levels. There is remarkable consistency across ten independent evolutionary runs, with the instances of large parameter

variability reflecting their small effect on fitness, rather than runs becoming trapped in different local optima. The number of hidden units rises quickly to near the maximum allowed, as does the number of hidden to output connections. The input to hidden unit connectivity settles to only about 0.38, presumably because such sparsity limits the interference between new and old patterns. The hidden to output learning rates evolve to high values, over 100, while the other learning rates take on more traditional small values, under 0.1. The evolved initial weight distributions are rather different to the traditional hand-crafted ranges of [-1,+1], with the input to hidden distribution close to [-10, 0], the hidden bias distribution close to [0,+2], the hidden to output distribution relatively unconstrained around [-10,+10], and the output bias distribution smaller but still relatively unconstrained around [-0.3,+0.3]. The sigmoid prime offset parameter and weight decay regularization parameters both evolve to very low values, that are effectively zero, as do the node sharpening parameters $N_H$ and $N_L$. The training tolerance is fairly unconstrained around 0.15, but the testing tolerance consistently settles just below 0.5, which is as close as the mutations can get it to the maximum value of 0.5 that allows any output on the right side of 0.5 to be deemed correct. This makes sense, since it renders the remembering as easy as possible without affecting the learning. (This parameter becomes more interesting in situations where the fitness function favours a "don't know" response over an incorrect output, but such cases will not be presented in this paper.) It is hard to see how a human designer could arrive at such a pattern of parameter values. The mean remembering performance is now 88.9%, with standard deviation across data sets of 1.2%, but much lower standard error on the mean. This is clearly a massive improvement over any of the hand-crafted network results for 4 new patterns seen in Figure 1.

### 5.3 Evolving Dual-Weight Architectures

The next question is whether moving to the next level of network complexity, namely to dual-weight architectures, can lead to further improvements. Levy & Bairaktaris (1995) have reviewed the key possibilities, which range from simple feed-forward networks with two additive weights between each pair of nodes, to complex dual-network systems. In line with the aim of seeing what can be achieved by evolving standard feed-forward networks, this study will look at the simplest possible approach. This, as originally proposed by Hinton & Plaut (1987), has a pair of additive weights on each connection, consisting of one standard weight plus an additional "fast weight" that has a learning rate larger by some scale factor, and a larger weight decay rate that prevents it from having long term memory. Both the normal and fast weights are driven by the same error signals, but the

fast weights change more quickly, so initially they contribute most to the overall performance, and on their own would perform much better than the normal weights. But their fast decay causes the error signal to persist and the normal weights slowly adjust to accommodate that until there is no error signal left to cause any weight changes, and the fast weights eventually decay to small values, leaving the normal weights to do all the work on their own. The only persisting contribution of the fast weights is to lead the final normal weights to perform better than if they had not been involved. However, getting such networks to work well relies on them having appropriate fast-weight learning and decay parameter values, and these prove extremely difficult to set "by hand", but this is a problem that the evolutionary approach should be able to deal with easily.

Obviously, with the fast weights decaying over time, it makes a difference whether the networks are tested immediately at each stage of training, or after the fast weights have had time to decay to negligible values. If, as in the preliminary study of Seipone & Bullinaria (2005b), they are tested immediately, evolving the new scale factor and decay rate, along with all the other details described above, constitutes an even more difficult search space than before, with around 40% of runs failing to reach the optimum level. Nevertheless, the successful runs provide dual-weight networks with an improved remembering performance of 95.0%. Not surprisingly, including the extra set of fast weights leads to rather different optimal values emerging for many of the evolved parameters compared with those of Figure 2. Significantly, the evolved training tolerance takes on a much lower and less variable value than before (around 0.01), with an associated large (approximately seven-fold) increase in the number of epochs of training required.

It seems more natural, however, to consider the long term stable state with the fast weights merely facilitating the training process of the standard weights, and having decayed to zero for testing purposes. Evolving everything in this case leads to much more consistency across different evolutionary runs, with all runs achieving the same optimum level as shown in Figure 3. The dual weights now provide a further level of performance improvement up to 97.9%. The involvement of fast weights here still results in different parameter values emerging compared with those of Figure 2, including a much higher input to hidden layer connectivity (around 0.86). Throughout the incremental learning process, any new patterns are learned quickly by the fast weights, and the information is then slowly integrated into the standard weights. The training tolerance is relatively high again (around 0.35), but testing with the fast weights fully decayed requires completed consolidation before the training stops, again resulting in a large (approximately seven-fold) increase in the number of epochs of training needed over the single weight networks. Of course, the

advantage of slow consolidation is not a new idea to memory modelling (e.g., McClelland, McNaughton & O'Reilly, 1995; O'Reilly & Rudy, 2000), and it is interesting to see it emerging automatically here from the evolutionary process.

### 5.4 Varying the Memory Task and Network Complexity

At this stage it is important to consider the issue of network and memory task complexity, and return to the matter of restricting the number of hidden units.

The simulations so far have tested how many of 20 original patterns are remembered after training on 4 new patterns. There is a clear need to explore the extent to which the number of new patterns, and hence the complexity of the task, affects the results. It is also important to test whether general purpose neural networks emerge, that work well for all task complexities, or whether they become over-tuned to particular levels of complexity. The obvious way to proceed is by evolving more networks as before, but to deal with different numbers of new patterns. For reliable testing, however, the simple population means presented in Figures 3 and 4 are not good enough. A significant population diversity is deliberately maintained by mutations and cross-overs to facilitate the evolutionary process, and that results in many sub-optimal individuals in the population. To avoid these artificially poor individuals, the testing was restricted to the best 10% of the final population on the last random data set used for evolution, and averages taken over large enough numbers of new random test data sets that the potential errors on the means were negligible.

Figure 4 shows such performance measures for 50 hidden unit networks evolved to deal with either 2, 4, 10 or 20 new patterns, and tested with 0, 2, 4, 10 and 20 new patterns. For both the standard (left graph) and dual-weight (right graph) cases, the best performance for each number of new patterns is achieved by the networks evolved using that number of new patterns (i.e. line *X* is always top for *X* new patterns), and in each case the performance is better with fewer new patterns and worse with more (i.e. all lines fall with the number of new patterns). One can also attempt to generate general purpose networks, that perform well on any number of new patterns, by evolving the networks using a new randomly chosen number of new patterns from the range [2, 20] for each generation. These result in the thick performance lines shown in Figure 4, exhibiting the kind of performance compromises one would expect.

Referring back to Figure 1 shows that, with the number of hidden units fixed at 50, evolving all the other network parameters leads to improvements far superior to the improvement achievable by simply increasing the number of hidden units by a factor of 200. It remains to be determined

whether allowing more hidden units now can improve the evolved performance even further, or if a performance ceiling has been reached. Figure 5 shows the performance results, analogous to Figure 4, when the maximum number of hidden units is raised to 10000. Evolution quickly increases the number of hidden units up to that maximum, and ends up with new appropriate values for all the other evolved parameters, resulting in further performance improvements. For both the standard (left graph) and dual-weight (right graph) cases, the extra hidden units do allow considerable further improvement in remembering performance.

Figure 6 compares directly the various performance results for the 50 and 10000 hidden unit cases. In each case are shown the remembering performance for traditional networks, evolved networks with and without fast-weights, and tuned to the number of new patterns or general purpose. It is clear that evolving the traditional network learning parameters, as described above, leads to considerable improvement in remembering performance, and further large improvements are achievable with evolved dual-weight networks.

### 5.5 Further Enhancements: Batch Learning and Network Ensembles

To complete the study of pure memory tasks, there are two further potential enhancement techniques that have proved sufficiently successful elsewhere to be worth considering here.

Evolutionary optimized learning parameters and dual weights allow a smoother interleaving of new patterns among previously learned patterns using standard online (i.e. one pattern at a time) gradient descent learning. Using batch learning rather than online learning might be expected to have a similar effect, but this is known to generally require smaller learning rates and more epochs of training (Bishop, 1995). This possibility was tested by evolving the 50 hidden unit networks again using batch learning and the same maximum number of epochs of training as before. The resulting remembering performances for general purpose networks are shown in the left graph of Figure 7. For both the standard and dual-weight cases there is improved performance for small numbers of new patterns, but worse performance for larger numbers of new patterns. It seems that this approach may be useful in certain circumstances, but it is not always beneficial.

Remembering performance was also improved by increasing the number of hidden units and letting the evolution adjust the learning parameters appropriately. It is known for other tasks that more sophisticated extensions of the neural structure can also lead to improved performance, such as combining the outputs of several independent neural modules to produce an overall output, or using error correcting codes to represent the outputs, so that small numbers of errors can be

eliminated. It is not feasible to consider all such possibilities here, but one simple example will be sufficient to demonstrate that further significant performance enhancements for the memory task are easily achievable in this way.

Each individual in an evolved population represents a single neural network. If one takes a number (or ensemble) of these networks to be modules of a larger network, and combines their binary outputs using a simple voting procedure to give an overall binary output, a small number of errors by each module are likely to be out-voted leading to better overall performance (Hansen & Salamon, 1990; Battiti & Colla, 1994; Yao & Liu, 1998; Bullinaria, 2007a). Of course, this depends on the errors in different modules being uncorrelated, and empirical tests are required to see how useful this approach might be in practice for the memory task. The right graph of Figure 7 shows the performance of ensembles of three individual general purpose networks compared with the single individual results obtained previously. There are considerable improvements for all numbers of new patterns in both the standard and dual-weight cases. The left graph of Figure 8 shows that larger ensembles give even more improvements, though correlations in the errors do limit how far this can be taken. The right graph of Figure 8 shows that even more substantial improvements can be achieved in this way for the 10000 hidden unit case too. The remaining errors are now so small that more difficult memory tasks are required to test further the limits of this approach.

### 5.6 Understanding the Enhanced Performance

The question remains as to what it is about the evolved networks that makes them perform so much better than traditional networks. There are several noteworthy features. First, for the standard (no fast-weight) networks, the distributions for the initial input to hidden weights are tightly constrained, indicating the importance of getting good random hidden representations prior to learning. Then the learning rates for the input to hidden weights are very low compared to those for the output layer weights. This corresponds to having relatively stable hidden unit representations that are minimally affected by the learning processes, and thus suffer little disruption when new training patterns are learned. The idea of having fixed hidden representations in neural networks has been shown to be successful in the context of Extreme Learning Machines (Huang, Zhu & Siew, 2006), so it is not surprising to find that relatively fixed hidden representations evolve here. A related factor is that the input to hidden unit connectivity proportion is also low (around 0.38), so changing the input pattern only has limited impact on the hidden representation. This low connectivity and the predominately negative (inhibitory) input to hidden layer weights lead to sparse hidden layer representations that

are already known to minimize interference in memory tasks (e.g., O'Reilly & Rudy, 2000; Olshausen & Field, 2004). These advantages are further enhanced by using as many hidden units as possible, since that results in the need for smaller individual weight changes when a new input-output mapping is learned that are less likely to interfere with existing mappings.

The evolved dual-weight networks allow a smoother interleaving of new patterns with older patterns, minimizing interference and leading to even better performance than the evolved standard networks. Interestingly, compared to the standard networks, the input to hidden layer weights here have nearer full connectivity (around 0.86) and a more symmetric distribution for their initial values, so that prior to training the hidden layer representations are not sparse at all, though sparser representations do emerge from the training process. Individually, each of the evolved factors reduce catastrophic forgetting, and together they result in massively improved performance. Although the best results are achieved by evolving networks to deal with specific numbers of new patterns, particularly when dual weights are employed, general purpose networks can also be evolved with very little degradation in performance.

## 6. Generalization Tasks

Having established that evolving standard neural networks can provide massive improvements for memory tasks, the issue of generalization performance is now considered. For concreteness, the study will be restricted to classification problems, but the applicability to regression problems too should be apparent. In particular, for ease of comparison with earlier research, the main incremental learning data sets studied by Polikar et al. (2001) will be used, namely their artificial "circular regions" data set, and the optical digits database from the UCI machine learning repository (Asuncion & Newman, 2007). The evolutionary approach will be described in detail for the optical digits task, and then results will be presented more briefly for the circular regions task, as a test of how widely applicable the evolved networks are.

The optical digits database contains hand-written samples of the digits 0 to 9 digitised on to an 8×8 grid to create 64 input attributes for each sample. The full training set consists of 3823 such patterns, and a separate test set contains a further 1797 patterns, with both sets fairly evenly spread over the ten classes. To study incremental learning, the training data set is divided randomly into six distinct batches of 200 patterns (each with 20 patterns from each digit class) to be used for six stages of incremental training, plus a further distinct sub-set of 1423 patterns to be used as a validation set during the evolution. That leaves another six batches of 200 unseen training patterns

to be used only after the whole evolutionary process has been completed, which ensures that the evolved networks only see new data and fully satisfy incremental learning property 1 that no access to past data is needed. The aim, of course, is to maximise the generalization performance after training. The validation set is used to estimate that to provide a measure of fitness to drive the evolutionary process. The test set is not used at all until the whole evolutionary process is completed, at which point it is used to evaluate the incremental learning performance of the evolved networks on the unseen training patterns.

The incremental learning takes place over six training sessions $T_i$, during each of which only one batch $B_i$ of 200 patterns is used to train the network to some stopping condition. The networks' classification outputs are taken to be the class corresponding to the highest activated output unit for each input pattern. At the end of each session, the network is re-tested on all the training sets used in the previous sessions to see how much interference has taken place, and also on the validation set to provide a measure of the generalization ability. As more of the training batches are used, the generalization ability is expected to increase, demonstrating good incremental learning capability, but at the same time the performance on the previous batches should not be seriously reduced.

## 6.1 Baseline Performance

Before embarking on the evolutionary simulations, baseline performance levels were established using traditional neural network learning parameters. The same type of network was employed as for the memory task, with the nature of the training data fixing the number of input units to be 64, and the number of output units to be 10, one for each class. One thousand such networks with 100 hidden units were initialized with random weights drawn uniformly from the standard range [-1, 1], and trained for 5000 epochs per data batch, with all the learning rates fixed at 0.02 (which is just below the maximum value that allows stable training for this network) and no weight decay or sigmoid prime offsets. The average performances of these standard networks are shown in Table 1, as percentages with the standard errors in brackets. The columns show the network's classification performance at the end of each of the six stages of training $T_i$, on the current data-batch $B_i$, all previous data-batches $B_{j<i}$, and the test set. A general fall off in performance is observed on each training data batch as the later batches are learned. The generalization (i.e. test set) performance does increase with the first two batches, but then starts falling again as the later batches are learned, presumably because of over-fitting of the training data. This is a clear demonstration of poor incremental learning ability. Of course, different training parameter values may improve or worsen

these baseline results. The aim here is to use evolutionary techniques to find the best possible parameter values, and to compare those optimized standard networks with more complex models.

Polikar et al. (2001) developed their *Learn++* system, and Lin, Tang & Yao (2008) their *SNCL* algorithm, specifically to obtain better incremental learning. The results they achieved with the same training data are shown in Table 2. Although their initial training performances start lower, they remain more steady as further training data sets are used, and there is a steady increase in generalization performance as more data is made available. Their final generalization performance is 92.7% for *Learn++* and 93.5% for *SNCL*, compared with only 86.4% for the baseline standard neural networks. The question to be explored next is whether it is possible to do any better here by evolving neural networks to be good at incremental learning, in the same way that they were evolved to perform better on memory tasks.

## 6.2 Evolving the Learning Parameters

The same evolutionary neural network approach as described above was used, based on standard one hidden layer feed-forward neural networks trained by gradient descent with the cross-entropy cost function. Each of the six training sessions continued until the innately specified stopping criterion was satisfied, or until a maximum number of training epochs was reached. That maximum number of epochs was set large enough that it only prevented successful learning during the first few generations while the learning abilities were still very poor. The individuals that had the lowest error on the validation set after training on all six batches were taken to be the fittest, and used to produce the next generation using crossover and mutation as before.

The learning parameters to be evolved were essentially the same as for the memory task: the number of hidden units $N_{Hid}$, connectivity proportions $c_{IH}$ and $c_{HO}$ between layers, four learning rates $\eta_L$ and initial weight distributions $[-l_L, u_L]$, a sigmoid prime offset $o_{SPO}$, a weight decay regularization parameter $\lambda$, and the training tolerance $t$. The network's classification output was simply the output unit with the highest activation, so no testing tolerance was required. However, unlike in the memory task, where all the training patterns needed to be learned before stopping the training, it could be advantageous to stop the training for generalization tasks before all the training patterns have been learned (to within the tolerance $t$), to prevent over-fitting of noisy data. Therefore, a second training tolerance parameter $s$ was evolved to specify the fraction of training patterns that could be left unlearned (i.e. not within the training tolerance $t$) when the training was stopped. As before, the number of hidden units $N_{Hid}$ tended to evolve to near the maximum number permitted,

inflicting a considerable strain on the computational resources, so this was constrained to be no more than 100, which is many times that needed to learn the given training data, and allows a fair comparison with the baseline performance.

Also as before, all the populations consisted of 100 individuals and the initial populations were created with random innate parameter values drawn from traditional ranges. Then, for each generation, each individual network has new random initial weights drawn from their own innately specified ranges, and learns according to their other innately specified parameters. There were two natural approaches for using the training data during the evolutionary process. One could, for each generation, randomly select (from the set of 2623 patterns not reserved for the final testing phase) new training and validation sets as specified above, or the same sets could be used for each generation. Having different training data for each generation proved to result in better general purpose learners, so that approach was adopted.

To determine the best levels of performance that could reasonably be expected from the evolved incremental learners, other networks were first evolved to generalize as well as possible from non-incremental versions of the same training data. Two separate cases were considered, both using exactly the same evolutionary regime as for the incremental learners, except that there was only one training data batch instead of six. The first case used one standard batch of 200 patterns, and the second used 1200 patterns, equivalent to having all six standard training data batches at once. The best 10% of the evolved individuals on the final validation sets were re-initialized and trained on the relevant unseen training data batch and evaluated on the unseen testing data. The 200 training pattern networks achieved an average test set performance of 91.55 ± 0.03%, while the 1200 training pattern networks achieved 96.21 ± 0.02%.

The emergent incremental learning networks from the full evolutionary simulations were fairly consistent across runs starting from different random initial configurations. The population average results with means and variances across ten runs are shown in Figure 9. The top-left graph shows that the connectivity here rises to near the maximum possible. The next three graphs show how the learning rates and initial weight distributions evolve. The precise values are not very informative, but note again the large variation in learning rates that emerge for the different components, that would be very difficult to get right "by hand". The bottom-left graph shows how the training output tolerance *t* and stopping early parameter *s* evolve to appropriate values. Finally, the bottom-right graph shows how the generalization performance improves little after the first 500 generations, despite some of the other parameters continuing to drift. The persistent variance in performance

reflects the random nature of the training data sets and initial weight distributions. The variance for each parameter reflects how crucial that parameter is to the fitness. Not shown (due to lack of space) are the number of hidden units that quickly rises to near the maximum allowed, the sigmoid prime offset that quickly falls to negligible values (showing that it is of no help here), and the weight decay parameter that settles down to around 0.0000005.

Table 3 shows the performance averages on the unseen data over 100 runs of each of the best 10% of the evolved networks from each of the 10 evolutionary simulations. As expected, all aspects show an improvement over the baseline network results of Table 1. More importantly, improvement is also seen over the *Learn++* and *SNCL* results of Table 2. The performance levels on the earlier training batches still fall slightly as later batches are processed, but those performance levels remain well above those for *Learn++* and *SNCL*. The generalization (test set) performance is also better at each stage, with a gradual improvement as more data batches are used, indicating good incremental learning. The final test set performance of 94.66% appears a modest improvement over the 92.7% of *Learn++*, but it more than halves the gap between that incremental learning performance and the 96.21% obtainable by training on all the data at once. Moreover, the performance on just the first batch of training data of 91.47% is now very close to the 91.55% obtained by networks evolved specifically to perform well on a single batch of training data. By comparison, *Learn++* only achieves 82% after the first data-batch, which is likely to be problematic for many practical applications where good performance generally needs to be established as soon as possible. *SNCL* is also better than *Learn++*, but still not as good as the evolved networks.

For memory tasks it was found that introducing a second set of "fast weights" leads to significant performance improvements due to the ability to incorporate appropriate weight changes for new data patterns into the existing weights with minimal disruption. Introducing the same system of dual weights into the generalization task networks led to the modest improvements in performance seen in Table 4, though the difference in final generalization performance is highly statistically significant (t-test $p < 10^{-8}$). These small potential improvements are sufficient to drive the fast weight parameters to quite consistent values across the ten independent runs (scale = 18±2, decay = 0.0012±0.0002), again indicating the importance of this factor.

Within this framework, there still remains scope for further improvement. The evolution tends to slow the learning to make full use of the maximum number of epochs allowed, and also makes full use of the maximum number of hidden units allowed. Both these factors form part of the evolved regularization process, and further performance improvements may be possible simply by

increasing those maximum values. However, the improvements achievable by doing this prove to be rather limited in relation to the enormous increase in the associated computational costs. Indeed, the computational cost of the general evolutionary process proposed in this paper is also high, but even small improvements are often extremely valuable, so it will generally remain a complex problem dependent decision whether the potential improvements are worth the extra effort.

A less computationally intensive road to potential improvement follows the same ensemble approach that proved successful for the memory tasks. Again, one can take a number (or ensemble) of individual neural networks from an evolved population to be modules of a larger network, combining the modules' binary outputs by a simple voting procedure to give a single overall binary output. In this way, small numbers of uncorrelated errors by individual modules are likely to be out-voted, leading to better overall performance. Table 5 summarizes the generalization (test set) results for the various cases. For the standard evolved network case, the individual overall generalization performance of 94.66 ± 0.02 (from Table 3) increases to 94.78 ± 0.02 for ensembles of three, and 94.88 ± 0.02 for ensembles of nine. These improvements are rather small, but they are both statistically significant (paired t-test $p < 10^{-3}$). For the dual-weight case, the individual overall generalization performance of 95.09 ± 0.03 (from Table 4) increases to 95.18 ± 0.04 for ensembles of three, and 95.24 ± 0.05 for ensembles of nine. These improvements are even smaller than the no fast-weights case, but both are still statistically significant (paired t-test $p < 10^{-3}$). It seems likely that the evolved networks are already operating too consistently and too close to the performance ceiling for the ensembles to provide much improvement here.

### 6.3 Accommodating New Classes

The above simulations have established how well the evolved neural networks satisfy the first three properties of good incremental learning described in Section 2, but property four remains to be explored, namely the ability to accommodate any new classes that may be introduced in the new data. The same optical digits training data was used to test this, but instead of six batches containing equal proportions of all 10 classes, the data was reorganised into four data-batches with each new batch introducing some new classes or removing some previously seen classes. To enable fair comparison, the distribution of classes, as shown in Table 6, was the same as that studied by Polikar et al. (2002). The batches were selected randomly for each run from the full set of training data, leaving 1270 items equally distributed across the ten classes as a validation set, and the same test set of 1797 items as used before. The incremental learning results achieved by traditional non-evolved

neural networks and *Learn++* are shown in Tables 7 and 8.

The evolutionary simulations proceeded in the same way as before. Table 9 shows the performance achieved by evolved networks without fast weights. There is clear improvement over the non-evolved networks of Table 7 and a slight improvement over the *Learn++* results of Table 8. Dealing with new classes is obviously difficult, and the final generalization performance is much lower than the uniform data case of Table 3, even though there are more examples overall of each class in the training data. The situation now has features analogous to the memory tasks considered earlier, in that there are different classes of patterns in the different training batches, and networks need to avoid catastrophic forgetting of previous classes that are not represented in the current batch. The smooth interleaving achievable by incorporating a second set of fast weights might be expected to provide more benefit here than it did for the uniform case of Table 4. This is confirmed in Table 10 which shows the performance of evolved networks with dual weights, with a large statistically significant improvement in the final generalization performance (t-test $p < 10^{-8}$). Further statistically significant improvements are achievable using the ensemble approach, as seen in Table 11, though these are small compared to those achieved by the incorporation of fast-weights.

### 6.4 Applicability to Other Data Sets

To check the generality of the above findings, and to explore the issue of accommodating new classes more carefully, the simulations were repeated with a rather different form of generalization task, namely the "concentric circles" data-set studied by Polikar et al. (2001). This has a circular two dimensional input space of radius 5, with concentric circular decision boundaries of radius 1, 2, 3 and 4 giving five classes to be learned from random samples in the input space. The incremental learning takes place over six training batches, with batches $B_1$ and $B_2$ having 50 instances from each of classes 1, 2 and 3; batches $B_3$ and $B_4$ having 50 instances from each of classes 1, 2, 3 and 4; and batches $B_5$ and $B_6$ having 50 instances from each of all five classes. The validation and test sets both contain 100 previously unseen instances from each class.

The learning performance of traditional networks is shown in Table 12, and the corresponding *Learn++* results of Polikar et al. (2001) are shown in Table 13. Evolving the neural networks without fast weights leads to improved incremental learning performance over both of these, as seen in Table 14. The final generalization performance of 94.22 ± 0.06% is still some way below the 97.20 ± 0.02% that is achieved by neural networks evolved to learn the whole set of training data at once. Since the incremental learning here involves the accommodation of new data classes, one

might expect further improvements to arise from incorporating fast weights. However, evolving dual weights here actually resulted in low fast weight scale parameters and high decay rates that rendered the fast weights ineffective in all ten runs, and the resulting incremental learning performances were not significantly different to the no fast-weights results of Table 14.

One fundamental difference between this case and the varying classes optical digits data-set is that all the classes remain present to the end of training here, so careful interleaving of information by fast weights may not be required because training data exists throughout to preserve the early class information. This hypothesis can be tested by using a new "varying classes" variant of the concentric circles data-set with batches $B_1$ and $B_2$ having 50 instances from each of classes 1, 2 and 3; batches $B_3$ and $B_4$ having 50 instances from classes 2, 3 and 4; and batches $B_5$ and $B_6$ having 50 instances from classes 3, 4 and 5. The validation and test sets each contain 100 previously unseen instances from each class as before. Now one can expect fast weights to help preserve classes 1 and 2 during the later stages of training. Table 15 gives the baseline traditional neural network results, and Table 16 shows that evolving networks without fast weights vastly improves the performance, but not to the levels achieved when the earlier classes remained in the training data till the end (as shown in Table 14). Table 17 shows that employing dual weights here does help preserve the class information learned previously and consequently also significantly improves the generalization (t-test $p < 10^{-7}$), but still not to the level seen in Table 14. Also shown in Table 17 are the significant levels of improvement achieved in this case by ensemble based modular networks.

## 6.5 Understanding the Enhanced Performance

It is well known that choosing good neural network parameter values is extremely problem dependent, and that general "rules of thumb" are notoriously unreliable (e.g., Bishop, 1995). It is certainly true that the evolved parameter values vary considerably over the problems studied in this paper, and often differ widely from those traditionally used in hand-crafted networks. While this implies that it will not be possible to identify evolved parameter values that will work well in general, there are several noteworthy patterns that do emerge. As with the memory tasks, the evolved networks for generalization tasks tend to have relatively stable hidden representations, with considerably lower learning rates for the input to hidden weights than for the hidden to output weights, but the details vary widely. Again, using as many hidden units as possible, while employing other forms of regularization, consistently emerges as a good evolved strategy. It has been known theoretically for some time that the size of the weights is more important for good

generalization than the size of the network (e.g., Bartlett, 1998), and evolution is clearly leading to more appropriate (small) weight sizes than traditional learning parameter values. Unlike for the memory tasks, where relatively low input to hidden unit connectively and/or predominately negative input to hidden layer weights led to sparse hidden unit representations that minimize interference, for all the generalization tasks studied close to full connectivity and fully distributed representations emerged that are known to facilitate generalization (e.g., Bishop, 1995; O'Reilly & Rudy, 2000).

The effectiveness of allowing dual weights was found to depend on the problem being studied. For pure generalization tasks which involve the same classes throughout the incremental learning, or only change the classes by introducing new classes, dual weights do not help very much. However, when some classes are dropped from the training data during incremental learning, an element of memory is required, and using dual weights helps preserve the earlier class information and allow continued good generalization for items from those classes. This is directly analogous to the smooth interleaving of new information and avoidance of catastrophic forgetting in the pure memory tasks. The error correcting abilities of ensemble based modular networks also prove most effective in these harder cases that are prone to catastrophic forgetting.

## 7. Conclusions and Discussion

This paper began by reviewing the problems (such as catastrophic forgetting) that traditional artificial neural networks have in dealing with incremental learning tasks in which the training data arrives in batches, and the principal earlier approaches for overcoming those problems. It then went on to explore, through a series of computational experiments, whether more sophisticated neural architectures and learning algorithms were really needed to solve those problems, because simple ideas from evolution by natural selection could be used to optimize standard feed-forward networks with gradient descent learning to the extent that they performed equally well. Those simulations demonstrated that such emergent networks could indeed perform very well, and now they need to be put into the context of models of known brain structures and their evolution.

It was first shown that simply evolving the parameters (such as the numbers of hidden units, degrees of connectivity, initial weight distributions, learning rates, and regularization parameters) for standard neural networks could provide massive incremental learning performance improvements over equivalent networks trained with traditional parameter values. A simple extension of those standard networks to dual-weight architectures (with one standard set of weights, plus a second set that learn and decay much faster) was already known to potentially lead to further learning

improvements, but setting the associated parameters to effective values "by hand" proves extremely difficult (Hinton & Plaut, 1987). However, including such dual weights in the evolutionary approach was found to render them useable and highly effective for avoiding catastrophic forgetting. Finally, in all cases, further improvements were achieved by simple modular architectures based on ensembles of the evolved standard networks.

It is clear (even from the small number of data-sets studied in this paper) that the neural learning and architecture parameter values emerging from evolutionary optimization are rather problem dependent, so any idea of evolving a single set of parameter values that is optimal for all incremental learning tasks is never going to work, but it does seem feasible to use past data-sets to evolve networks that will work reasonably well on a range of similar data sets in the future, in much the same way that humans have evolved to cope well on the range of tasks they are likely to encounter. It is encouraging in this respect that the networks evolved to work on a range of memory tasks in Section 5.4 performed only slightly worse on each particular task than those evolved specifically to work on that task. More generally, the emergence of distinct optimal structures for different problem types is consistent with existing explanations of why multiple memory systems have evolved in humans and other animals (e.g., Sherry & Schacter, 1987).

The incremental learning tasks studied were chosen to span the full spectrum from the pure memory tasks of Section 5, for which dual weights bestow massive improvements by allowing smooth interleaving of new data items into the weights, to pure generalization tasks, such as the concentric circles classification task of Section 6.4, for which dual weights were no help at all. In between were the "varying classes" generalization tasks of Section 6, which are prone to suffering catastrophic forgetting like the memory tasks, and are also improved by allowing dual weights. Overall, the study confirmed the ability of the evolutionary approach to achieve performance levels far better than traditional neural networks and simple hand-crafted incremental learning systems such as *Learn++* (Polikar, et al., 2001; Polikar et al., 2002), but it remains a topic of future work to explore exactly how much further this approach can be taken, and whether simple evolved networks with dual weights can be made to perform as well as more complex hand-crafted neural systems (such as those of Grossberg, 1987; Ans, et al., 2002).

Extrapolating reliably from the simple models studied in this paper up to the kinds of tasks that humans regularly perform well and brain-like numbers of hidden units is non-trivial, of course, but it does seem that the problem of incremental learning in neural networks might not be quite as problematic as previously thought (e.g., French, 1999). The issue of how simplified neural network

simulations relate to real brain processes is always difficult for cognitive scientists, particularly since the physical structures in brains and their learning algorithms are usually far removed from the simplified models. One might argue that, if extremely simple evolved neural networks can perform incremental learning well, then it is reasonable to expect that more realistic evolved models will be able to do so too. However, if complex artificial neural network architectures and algorithms are required, it may be more difficult to relate them to real brains and more realistic models of them. This was one of the main motivations for exploring what the simplest possible neural networks were actually capable of achieving if optimized appropriately. It is tempting to apply Occam's razor here, but that can be misleading in this context. One has to consider the question of brain evolvability, and how particular brain structures and processes have evolved. It is clear that the kind of brain that has emerged for humans has depended heavily on the details of our evolutionary history and factors related to the need for the structures to be grown at each stage using biological material (e.g., Jacob, 1977; Alon, 2003), so it is dangerous to assume that the most simple or computationally efficient models are the ones most likely to corresponds to how they operate. It seems reasonable to assume that a simple sequence of evolutionary steps is more likely to have actually occurred than a complex sequence, and simple solutions are more likely to have emerged from them than complex solutions, but often it will be simpler for evolution to re-use existing structures than to create new ones, and that may lead to unnecessarily complex solutions in real brains (e.g., Anderson, 2010).

Many of the relevant issues concerning brain evolution have already been discussed elsewhere (e.g., Ohno, 1970; Jacob, 1977; Sherry & Schacter, 1987; Calabretta, Nolfi, Parisi & Wagner, 1998; Alon, 2003; Striedter, 2005; Anderson, 2010). Given the known biological basis and complexity of the constituent neural components, it seems likely that the associated basic neural learning parameter values could be adjusted relatively easily by biological evolution, with the kinds of associated performance improvements found in the simulations in this paper. Similarly, the improvements arising from simple ensemble style modular architectures can be expected to emerge easily through simple duplication type mutations. It also seems likely that Hinton & Plaut (1987) style dual-weight systems could evolve easily via a relatively simple "duplicate and adjust" mechanism, though that still requires a mechanism whereby new training patterns are stored while the training process slowly shifts the information from the fast weights into the standard weights. Indeed, the question of how the training patterns are preserved to enable gradient descent learning still exists even if only the standard weights are present. Fortunately, the literature already contains plausible suggestions about how that could operate. Sherry & Schacter (1987) argued why distinct memory

systems should evolve for episodic remembering and generalizable skill learning, and there is increasing evidence of their existence and interactions (e.g., Poldrack & Packard, 2003). The details remain debatable, but the general idea is that information is acquired rapidly by the Hippocampus and then gets consolidated relatively slowly into the neocortex (Alvarez & Squire, 1994; Nadel, & Moscovitch, 1997; Frankland & Bontempi, 2005), with much of the transfer taking place during sleep (Robins, 1996; Stickgold & Walker, 2007; Peyrache et al., 2009; Diekelmann & Born, 2010). One likely mechanism, with growing supporting evidence, has a largely rapid Hebbian learning based Hippocampus and a largely slow gradient descent based neocortex (McClelland, McNaughton & O'Reilly, 1995; O'Reilly & Rudy, 2000). In that context, the models and simulation presented in this paper show how the consolidation phase can come about in an efficient and successful manner, with performance levels far superior to traditional non-evolved networks.

What adaptations and exaptations might lead such set-ups to evolve in biological systems remain unclear, but given that complex structures like eyes have evolved multiple times (Lamb, Collin & Pugh, 2007), it seems likely that evolvability will not be a major issue here, as long as fitness enhancements for a sequence of plausible evolutionary steps can be identified. As demonstrated by the simulations of this paper, a simple separation of fast and slow weights leads automatically to transfer of information between them and significant performance advantages, and different optimal structures and parameter values emerge readily for different tasks (such as pure memory, pure generalization, or something in between), so there would appear to be relatively easy routes for brain evolution from simple associations to the kind of brain structures found in the human brain today. Whether the second set of fast-weights found to be so effective in the simple networks studied in this paper would still emerge alongside the main weights if access to an existing separate dual network of fast-weights could be employed for that purpose instead (or vice versa) is one of those questions that depend critically on neural re-use issues (Anderson, 2010), and will only be determined by more detailed simulations of more complex networks.

Another important issue here is the fact that brain evolution has depended on numerous interacting factors (e.g., Striedter, 2005), and if these are not accommodated as constraints into the cognitive models, it will remain unclear how reliable those models are. For example, using a similar evolutionary computation approach to this paper has shown that even fundamental concepts like the emergence of modularity depends crucially on basic neurobiological constraints that are usually absent in cognitive models, such as the volume of "wiring" required to connect up different neural architectures in the brain (Bullinaria, 2007b). Ultimately it is hoped that direct empirical evidence

will be sufficient to constrain all brain models, but in the absence of that, it may be that evolution can provide additional clues as to which of competing models of equal competence are most likely to correspond to real brains (e.g., Manns & Eichenbaum, 2006), though there are certainly known difficulties in applying evolutionary ideas to cognitive abilities (e.g., Bolhuis & Wynne, 2009). Of course, understanding how brains have evolved is also of interest in its own right (e.g., Klein et al., 2009; Nairne & Pandeirada, 2010), and that may also help in understanding the continuing presence of common harmful heritable mental dysfunction (e.g., Keller & Miller, 2006). The simulated evolution approach presented in this paper, and extensions of it to more realistic modeling frameworks, should provide a solid basis for exploring such issues.

Perhaps the most obvious way to take the approach of this paper further, then, is to enable the evolutionary simulations to represent more complex and more brain-like structures. As noted above, this has already been done to explore the emergence of modular structures (Bullinaria, 2007b), so incorporating the possibility of evolving dual-network architectures (such as those of French, 1997; Ans & Rousset, 1997, 2000; Ans, et al., 2002) and related architectures and parameters should be feasible. Ultimately, the aim will be to start with models of the simplest of nervous systems and simulate their evolution into structures that resemble the human brain, and see if and how the dual-weight models explored in this paper fit into that evolutionary sequence. It will be instructive to see whether the existing hand-crafted models emerge from such simulated evolution, or if even better versions of those models can be obtained, and whether such simulations can cast any light on the remaining controversy over the operation of learning in the hippocampus and neocortex. The only limiting factor now is the increasing computational resources that are required by the improvements in biological/ecological plausibility and growing model complexities.

## Acknowledgements

Tebogo Seipone is thanked for her collaboration on the early stages of this work. Preliminary versions of several simulations described in this paper were previously published in conference proceedings (Seipone & Bullinaria, 2005a,b,c; Bullinaria, 2009). Three reviewers are also thanked for their helpful suggestions for improving the paper.

## References


Albesano, D., Gemello, R., Laface, P., Mana, F. & Scanzio, S. (2006). Adaptation of artificial neural networks: Avoiding catastrophic forgetting. *Proceedings of the 2006 International Joint Conference on Neural Networks (IJCNN 2006),* 1554-1561. Piscataway, NJ: IEEE.

Alon, U. (2003). Biological networks: The tinkerer as an engineer. *Science*, **301**, 1866-1867.

Alvarez, P. & Squire, L.R. (1994). Memory consolidation and the medial temporal lobe: A simple network model. *Proceedings of the National Academy of Sciences*, **91**, 7041-7045.

Anderson, M.L. (2010). Neural re-use as a fundamental organizational principle of the brain. *Behavioral and Brain Sciences*, **33**, to appear.

Ans, B. & Rousset, S. (1997). Avoiding catastrophic forgetting by coupling two reverberating neural networks. *CR Academie Science Paris, Life Sciences*, **320**, 989-997.

Ans, B. & Rousset, S. (2000). Neural networks with a self-refreshing memory: Knowledge transfer in sequential learning tasks without catastrophic interference. *Connection Science*, **12**, 1-19.

Ans, B., Rousset, S., French, R.M. & Musca, S. (2002). Preventing catastrophic interference in multiple-sequence learning using coupled reverberating Elman networks. *Proceedings of the Twenty-fourth Annual Conference of the Cognitive Science Society*, 71-76. Mahwah, NJ: LEA.

Asuncion, A. & Newman, D.J. (2007). *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Bartlett, P.L. (1998). The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. *IEEE Transactions on Information Theory*, **44**, 525-536.

Battiti, R. & Colla, A.M. (1994). Democracy in neural networks: Voting schemes for classification. *Neural Networks*, **7**, 691-709.

Bishop, C.M. (1995). *Neural Networks for Pattern Recognition*. Oxford, UK: Oxford University Press.

Bolhuis, J.J. & Wynne, C.D.L. (2009). Can evolution explain how minds work? *Nature*, **458**, 832-833.

Bullinaria, J.A. (2003). Evolving efficient learning algorithms for binary mappings. *Neural Networks*, **16**, 793-800.

Bullinaria, J.A. (2007a). Using evolution to improve neural network learning: Pitfalls and solutions. *Neural Computing & Applications*, **16**, 209-226.

Bullinaria, J.A. (2007b). Understanding the emergence of modularity in neural systems. *Cognitive Science*, **31**, 673-695.

Bullinaria, J.A. (2009). Evolved dual weight neural architectures to facilitate incremental learning. *Proceedings of the International Joint Conference on Computational Intelligence (IJCCI 2009*), 427-434. Portugal: INSTICC.

Calabretta, R., Nolfi, S., Parisi, D. & Wagner, G.P. (1998). A case study of the evolution of modularity: Towards a bridge between evolutionary biology, artificial life, neuro- and cognitive science. *Proceedings of the Sixth International Conference on Artificial Life*, 275-284. Cambridge, MA: MIT Press.

Cantû-Paz, E. & Kamath, C. (2005). An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, **35**, 915-927.

Carpenter, G.A. & Grossberg, S. (1988). The ART of adaptive pattern recognition by a self-organizing neural network. *Computer*, **21**, 77-88.

Carpenter, G.A. & Grossberg, S. (2003). Adaptive resonance theory. In M.A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*, 87-90. Cambridge, MA: MIT Press.

Chalup, S.K. (2002). Incremental learning in biological and machine learning systems. *International Journal of Neural Systems*, **12**, 447-465.

Davachi, L. (2006). Item, context and relational episodic encoding in humans. *Current Opinion in Neurobiology*, **16**, 693-700.

Deadwyler, S.A. & Hampson, R.E. (1999). Anatomic model of hippocampal encoding of spatial information. *Hippocampus*, **9**, 397-412.

Diekelmann, S. & Born, J. (2010). The memory function of sleep. *Nature Reviews Neuroscience,* **11**, 114-126.

Eichenbaum, H. (2001). The hippocampus and declarative memory: Cognitive mechanisms and neural codes. *Behavioural Brain Research,* **127**,199-207.

Eichenbaum, H. (2004). Hippocampus: Cognitive processes and neural representations that underlie declarative memory. *Neuron*, **44**, 109-120.

Elman, J.L., Bates, E.A., Johnson, M.H., Karmiloff-Smith, A., Parisi, D. & Plunkett, K. (1996). *Rethinking Innateness: A Connectionist Perspective on Development*. Cambridge, MA: MIT Press.

Engelbrecht, A.P. & Brits, R. (2001). A clustering approach to incremental learning for feedforward

neural networks. *Proceedings of the International Joint Conference in Neural Networks (IJCNN 2001)*, **3**, 2019-2024.

Frankland, P.W. & Bontempi, B. (2005). The organization of recent and remote memories. *Nature Reviews Neuroscience,* **6**, 119-130.

Frean, M. & Robins, A. (1999). Catastrophic forgetting in simple neural networks: An analysis of the pseudorehearsal solution. *Network: Computation in Neural Systems*, **10**, 227-236.

French, R.M. (1992). Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, **4**, 365-377.

French, R.M. (1997). Pseudo-recurrent connectionist networks: An approach to the "sensitivity-stability" dilemma. *Connection Science*, **9**, 353-379.

French, R.M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, **4**, 128-135.

French, R.M. (2003). Catastrophic interference in connectionist networks. In L. Nadel (Ed.), *Encyclopedia of Cognitive Science,* **1**, 431-435.

French, R.M., Ans, B. & Rousset, S. (2001). Pseudopatterns and dual network memory models: Advantages and shortcomings. In R.M French & J. Sougne (Eds), *Connectionist Models of Learning, Development and Evolution*, 13-22. London, UK: Springer.

Fukushima, K. (2004). Neocognitron capable of incremental learning. *Neural Networks*, **17**, 37-46.

Giraud-Carrier, C. (2000). A note on the utility of incremental learning. *AI Communications*, **13**, 215-223.

Grossberg, S. (1987). Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, **11**, 23-63.

Hansen, L.K. & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 993-1000.

Hinton, G.E. & Plaut, D.C. (1987). Using fast weights to deblur old memories. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, 177-186. Hillsdale, NJ: LEA.

Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*, **70**, 489-501.

Hulley, G. & Marwala, T. (2007). Genetic algorithm based incremental learning for optimal weight and classifier selection. *Proceedings of the 2007 International Symposium on Computational Models for Life Sciences (CMLS 2007),* 258-267. Melville NY: American Institute of Physics.

Jacob, F. (1977). Evolution and tinkering. *Science*, **196**, 1161-1166.

Kaas, J.H. (2000). Why is brain size so important: Design problems and solutions as neo-cortex gets bigger or smaller. *Brain and Mind*, **1**, 7-23.

Keller, M.C. & Miller, G. (2006). Resolving the paradox of common, harmful, heritable mental disorders: Which evolutionary genetic models work best? *Behavioral and Brain Sciences*, **29**, 385-452.

Klein, S.B., Cosmides, L., Gangi, C.E., Jackson, B., Tooby, J. & Costabile, K.A. (2009). Evolution and episodic memory: An analysis and demonstration of a social function of episodic recollection. *Social Cognition*, **27**, 283-319.

Kruschke, J.K. (1992). ALCOVE: An exemplar-based connectionist model of category learning. *Psychological Review,* **99**, 22-44.

Lamb, T.D., Collin, S.P. & Pugh, E.N. (2007). Evolution of the vertebrate eye: Opsins, photo-receptors, retina and eye cup. *Nature Reviews: Neuroscience*, **8**, 960-976.

Lange, S. & Grieser, G. (2002). On the power of incremental learning. *Theoretical Computer Science*, **288**, 277-307.

Levy, J.P. & Bairaktaris, D. (1995). Connectionist dual-weight architectures. *Language and Cognitive Processes,* **10**, 265-283.

Lin, M., Tang, K. & Yao, X. (2008). Selective negative correlation learning algorithm for incremental learning. *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN 2008),* 2525-2530. Piscataway, NJ: IEEE.

Manns, J.R. & Eichenbaum, H. (2006). Evolution of declarative memory. *Hippocampus*, **16**, 795-808.

McClelland, J.L., McNaughton, B.L. & O'Reilly, R.C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review,* **102**, 419-457.

McCloskey, M. & Cohen, N.J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation*, **24**, 109-165.

McRae, K. & Hetherington, P.A. (1993). Catastrophic interference is eliminated in pretrained networks. *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, 723-728. Hillsdale, NJ: LEA.

Minku, F.L., Inoue, H. & Yao, X. (2009). Negative correlation in incremental learning. *Natural Computing*, **8**, 289-320.

Muhlbaier, M.D., Topalis, A. & Polikar, R. (2009). Learn++.NC: Combining ensemble of

classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. *IEEE Transactions on Neural Networks*, **20**, 152-168.

Murre, J.M.J. (1992). *Learning and Categorization in Modular Neural Networks*. Hillsdale, NJ: LEA.

Nadel, L. & Moscovitch, M. (1997). Memory consolidation, retrograde amnesia and the hippocampal complex. *Current Opinion in Neurobiology*, **7**, 217-227.

Nairne, J.S. & Pandeirada, J.N.S. (2010). Adaptive memory: Ancestral priorities and the mnemonic value of survival processing. *Cognitive Psychology*, **61**, 1-22.

Ohno, S. (1970). *Evolution by Gene Duplication*. New York, NY: Springer-Verlag.

Olshausen, B.A. & Field, D.J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, **14**, 481-487.

O'Reilly, R.C. & McClelland, J.L. (1994). Hippocampal conjunctive encoding, storage, and recall: Avoiding a trade-off. *Hippocampus*, **4**, 661-682.

O'Reilly, R.C. & Rudy. J.W. (2000). Computational principles of learning in the neocortex and hippocampus. *Hippocampus*, **10**, 389-397.

Peyrache, A., Khamassi, M., Benchenane, K., Wiener, S.I. & Battaglia, F.P. (2009). Replay of rule-learning related neural patterns in the prefrontal cortex during sleep. *Nature Neuroscience,* **12**, 919-926.

Poldrack, R.A. & Packard, M.G. (2003). Competition among multiple memory systems: Converging evidence from animal and human brain studies. *Neuropsychologia*, **41**, 245-251.

Polikar, R., Byorick, J., Krause, S., Marino, A. & Moreton, M. (2002). Learn++: A classifier independent incremental learning algorithm for supervised neural networks. *Proceedings of the 2002 International Joint Conference on Neural Networks (IJCNN 2002),* 1742-1747. Piscataway, NJ: IEEE.

Polikar, R., Udpa, L., Udpa, S.S. & Honavar, V. (2001). Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews,* **31**, 497-508.

Ratcliff, R. (1990). Connectionist models of recognition and memory: Constraints imposed by learning and forgetting functions. *Psychological Review,* **97**, 205-308.

Robins, A. (1995). Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science,* **7**, 123-146.

Robins, A. (1996). Consolidation in neural networks and in the sleeping brain. *Connection Science*,

**8**, 259-275.

Robins, A. (1997). Maintaining stability during new learning in neural networks. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 3013-3018. Piscataway, NJ: IEEE.

Robins, A. (2004). Sequential learning in neural networks: A review and a discussion of pseudorehearsal based methods. *Intelligent Data Analysis,* **8**, 301-322.

Robins, A. & McCallum, S. (1999). The consolidation of learning during sleep: Comparing the pseudorehearsal and unlearning accounts. *Neural Networks*, **12**, 1191-1206.

Rolls, E.T. & Kesner, R.P. (2006). A computational theory of hippocampal function, and empirical tests of the theory. *Progress in Neurobiology*, **79**, 1-48.

Seipone, T. & Bullinaria, J.A. (2005a). Evolving neural networks that suffer minimal catastrophic forgetting. In A. Cangelosi, G. Bugmann & R. Borisyuk (Eds*), Modeling Language, Cognition and Action,* 385-390. Singapore: World Scientific.

Seipone, T. & Bullinaria, J.A. (2005b). The evolution of minimal catastrophic forgetting in neural systems. *Proceedings of the Twenty-Seventh Annual Conference of the Cognitive Science Society,* 1991-1996. Mahwah, NJ: LEA.

Seipone, T. & Bullinaria, J.A. (2005c). Evolving improved incremental learning schemes for neural network systems. *Proceedings of the 2005 IEEE Congress on Evolutionary Computing (CEC 2005),* 273-280. Piscataway, NJ: IEEE.

Sharkey, N.E. & Sharkey, A.J.C. (1995). An analysis of catastrophic interference. *Connection Science*, **7**, 301-329.

Sherry, D.F. & Schacter, D.L. (1987). The evolution of multiple memory systems. *Psychological Review*, **94**, 439-454.

Stern, C.E., Corkin, S., Gonzalez, R.G., Guimaraes, A.R., Baker, J.R., Jennings, P.J., Carr, C.A., Sugiura, R.M., Vedantham, V. & Rosen, B.R. (1996). The hippocampal formation participates in novel picture encoding: Evidence from functional magnetic resonance imaging. *Neurobiology*, **93**, 8660-8665.

Stickgold, R. & Walker, M.P. (2007). Sleep-dependent memory consolidation and reconsolidation. *Sleep Medicine*, **8**, 331-343.

Striedter, G.F. (2005). *Principles of Brain Evolution*. Sunderland, MA: Sinauer Associates.

Weliky, M., Fiser, J., Hunt, R.H. & Wagner, D.N. (2003). Coding of natural scenes in primary visual cortex. *Neuron*, **37**, 703-718.

Yamaguchi, M. (2004). Reassessment of catastrophic interference. *NeuroReport*, **15**, 2423-2425.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE,* **87**, 1423-1447.

Yao, X. & Liu, Y. (1998). Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics,* **28**, 417-425.

## Tables

| ***Baseline*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 100.00 (0.00) | 95.51 (0.05) | 93.88 (0.06) | 92.85 (0.06) | 91.59 (0.09) | 89.98 (0.11) |
| $B_2$ | -- | 100.00 (0.00) | 94.66 (0.06) | 93.08 (0.07) | 91.78 (0.08) | 90.01 (0.11) |
| $B_3$ | -- | -- | 100.00 (0.00) | 93.87 (0.07) | 91.87 (0.08) | 90.08 (0.12) |
| $B_4$ | -- | -- | -- | 100.00 (0.00) | 92.69 (0.09) | 90.44 (0.12) |
| $B_5$ | -- | -- | -- | -- | 99.94 (0.04) | 91.08 (0.12) |
| $B_6$ | -- | -- | -- | -- | -- | 99.59 (0.08) |
| ***Test*** | 88.09 (0.04) | 89.56 (0.03) | 89.43 (0.04) | 88.97 (0.04) | 87.95 (0.07) | 86.39 (0.10) |

Table 1: Incremental learning performance on the optical digits data-set for standard neural networks trained using traditional back-propagation parameters. The percentages of correct classification on the training data batches $B_{j\leq i}$ and generalization test set are given after each stage of training $T_i$. Averages over 1000 runs are shown, with standard errors in brackets.

| ***Learn++*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 94 | 94 | 94 | 93 | 93 | 93 |
| $B_2$ | -- | 93.5 | 94 | 94 | 94 | 93 |
| $B_3$ | -- | -- | 95 | 94 | 94 | 94 |
| $B_4$ | -- | -- | -- | 93.5 | 94 | 94 |
| $B_5$ | -- | -- | -- | -- | 95 | 95 |
| $B_6$ | -- | -- | -- | -- | -- | 95 |
| ***Test*** | 82.0 | 84.7 | 89.7 | 91.7 | 92.2 | 92.7 |

| ***SNCL*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 98.35 | 94.70 | 95.30 | 95.15 | 94.20 | 94.75 |
| $B_2$ | -- | 99.45 | 97.65 | 95.97 | 96.70 | 92.35 |
| $B_3$ | -- | -- | 99.25 | 98.87 | 98.22 | 98.25 |
| $B_4$ | -- | -- | -- | 99.58 | 98.53 | 99.05 |
| $B_5$ | -- | -- | -- | -- | 99.08 | 96.10 |
| $B_6$ | -- | -- | -- | -- | -- | 99.47 |
| ***Test*** | 89.02 | 91.54 | 93.39 | 94.00 | 94.00 | 93.49 |

Table 2: Incremental learning performances on the optical digits data-set achieved by the *Learn++* algorithm of Polikar et al. (2001) and the *SNCL* algorithm of Lin, Tang & Yao (2008).

| ***NFW*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 100.00 (0.00) | 98.92 (0.02) | 98.49 (0.03) | 98.34 (0.03) | 98.27 (0.02) | 98.27 (0.03) |
| $B_2$ | -- | 100.00 (0.00) | 99.01 (0.02) | 98.49 (0.03) | 98.31 (0.03) | 98.21 (0.03) |
| $B_3$ | -- | -- | 100.00 (0.00) | 99.06 (0.03) | 98.57 (0.03) | 98.34 (0.03) |
| $B_4$ | -- | -- | -- | 100.00 (0.00) | 99.15 (0.03) | 98.61 (0.02) |
| $B_5$ | -- | -- | -- | -- | 100.00 (0.00) | 99.15 (0.02) |
| $B_6$ | -- | -- | -- | -- | -- | 100.00 (0.00) |
| ***Test*** | 91.47 (0.03) | 92.97 (0.02) | 93.64 (0.01) | 94.10 (0.02) | 94.43 (0.02) | 94.66 (0.02) |

Table 3: Incremental learning performance on the optical digits data-set for evolved neural networks without fast weights.

| ***FW*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 99.99 (0.00) | 99.15 (0.04) | 98.92 (0.05) | 98.86 (0.04) | 98.83 (0.04) | 98.84 (0.04) |
| $B_2$ | -- | 99.99 (0.00) | 99.32 (0.03) | 99.01 (0.02) | 98.89 (0.03) | 98.83 (0.02) |
| $B_3$ | -- | -- | 99.93 (0.01) | 99.31 (0.02) | 99.02 (0.02) | 98.91 (0.02) |
| $B_4$ | -- | -- | -- | 99.83 (0.02) | 99.29 (0.01) | 99.01 (0.01) |
| $B_5$ | -- | -- | -- | -- | 99.72 (0.03) | 99.24 (0.02) |
| $B_6$ | -- | -- | -- | -- | -- | 99.61 (0.04) |
| ***Test*** | 91.61 (0.08) | 93.45 (0.03) | 94.19 (0.02) | 94.60 (0.02) | 94.88 (0.03) | 95.09 (0.03) |

Table 4: Incremental learning performance on the optical digits data-set for evolved neural networks with fast weights.

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| ***NFW 1*** | 91.47 (0.03) | 92.97 (0.02) | 93.64 (0.01) | 94.10 (0.02) | 94.43 (0.02) | 94.66 (0.02) |
| ***NFW 3*** | 91.54 (0.05) | 93.06 (0.03) | 93.75 (0.03) | 94.17 (0.03) | 94.52 (0.02) | 94.78 (0.02) |
| ***NFW 9*** | 91.52 (0.03) | 93.12 (0.04) | 93.83 (0.02) | 94.26 (0.03) | 94.62 (0.02) | 94.88 (0.02) |
| ***FW 1*** | 91.61 (0.08) | 93.45 (0.03) | 94.19 (0.02) | 94.60 (0.02) | 94.88 (0.03) | 95.09 (0.03) |
| ***FW 3*** | 91.87 (0.04) | 93.62 (0.03) | 94.32 (0.03) | 94.72 (0.03) | 94.99 (0.04) | 95.18 (0.04) |
| ***FW 9*** | 91.98 (0.03) | 93.71 (0.03) | 94.39 (0.05) | 94.78 (0.05) | 95.04 (0.05) | 95.24 (0.05) |

Table 5: Incremental learning test set performances on the optical digits data-set for voting ensembles of 1, 3 and 9 evolved neural networks, with no fast weights (NFW) and with fast weights (FW).

| *Class* | $B_1$ | $B_2$ | $B_3$ | $B_4$ |
|---|---|---|---|---|
| **0** | 100 | 50 | 50 | 25 |
| **1** | 0 | 150 | 50 | 0 |
| **2** | 100 | 50 | 50 | 25 |
| **3** | 0 | 150 | 50 | 25 |
| **4** | 100 | 50 | 50 | 0 |
| **5** | 0 | 150 | 50 | 25 |
| **6** | 100 | 50 | 0 | 100 |
| **7** | 0 | 0 | 150 | 50 |
| **8** | 100 | 0 | 0 | 150 |
| **9** | 0 | 50 | 100 | 50 |

Table 6: The distribution of optical digits classes across the four varying classes training batches, as previously used by Polikar et al. (2002).

| ***Baseline*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|
| $B_1$ | 100.00 (0.00) | 77.16 (0.02) | 68.93 (0.14) | 85.16 (0.12) |
| $B_2$ | -- | 100.00 (0.00) | 90.19 (0.06) | 67.94 (0.06) |
| $B_3$ | -- | -- | 100.00 (0.00) | 80.87 (0.07) |
| $B_4$ | -- | -- | -- | 100.00 (0.00) |
| ***Test*** | 48.47 (0.01) | 75.49 (0.02) | 79.87 (0.07) | 77.07 (0.07) |

Table 7: Incremental learning performance on the varying classes optical digits data-set for neural networks trained using traditional back-propagation parameters.

| ***Learn++*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|
| $B_1$ | 96.6 | 89.8 | 86.0 | 94.8 |
| $B_2$ | -- | 87.1 | 89.4 | 87.9 |
| $B_3$ | -- | -- | 92.0 | 92.2 |
| $B_4$ | -- | -- | -- | 87.3 |
| ***Test*** | 46.6 | 68.9 | 82.0 | 87.0 |

Table 8: The *Learn++* incremental learning performance of Polikar et al. (2002) for the varying classes optical digits data-set.

| ***NFW*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|
| $B_1$ | 99.92 (0.05) | 79.11 (0.10) | 75.81 (0.94) | 95.89 (0.09) |
| $B_2$ | -- | 99.81 (0.13) | 96.18 (0.41) | 82.02 (0.11) |
| $B_3$ | -- | -- | 99.93 (0.04) | 91.00 (0.28) |
| $B_4$ | -- | -- | -- | 99.87 (0.08) |
| ***Test*** | 48.55 (0.05) | 76.46 (0.10) | 85.00 (0.38) | 87.06 (0.23) |

Table 9: Incremental learning performance on the varying classes optical digits data-set for evolved neural networks without fast weights.

| ***FW*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|
| $B_1$ | 94.91 (1.59) | 88.45 (1.01) | 80.81 (0.29) | 97.39 (0.07) |
| $B_2$ | -- | 98.84 (0.18) | 99.38 (0.07) | 97.34 (0.16) |
| $B_3$ | -- | -- | 97.34 (0.20) | 98.24 (0.04) |
| $B_4$ | -- | -- | -- | 94.85 (0.19) |
| ***Test*** | 46.48 (0.73) | 80.20 (0.24) | 87.23 (0.08) | 93.83 (0.02) |

Table 10: Incremental learning performance on the varying classes optical digits data-set for evolved neural networks with fast weights.

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|---|
| ***NFW 1*** | 48.55 (0.05) | 76.46 (0.10) | 85.00 (0.38) | 87.06 (0.23) |
| ***NFW 3*** | 48.64 (0.04) | 76.57 (0.06) | 85.28 (0.41) | 88.06 (0.14) |
| ***NFW 9*** | 48.69 (0.04) | 76.74 (0.05) | 85.54 (0.44) | 88.41 (0.12) |
| ***FW 1*** | 46.48 (0.73) | 80.20 (0.24) | 87.23 (0.08) | 93.83 (0.02) |
| ***FW 3*** | 46.98 (0.46) | 80.39 (0.27) | 87.09 (0.07) | 94.21 (0.04) |
| ***FW 9*** | 47.04 (0.45) | 80.65 (0.30) | 87.19 (0.08) | 94.46 (0.06) |

Table 11: Incremental learning test set performances on the varying classes optical digits data-set for voting ensembles of 1, 3 and 9 evolved neural networks, with no fast weights (NFW) and with fast weights (FW).

| ***Baseline*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 99.99 (0.00) | 94.82 (0.06) | 90.55 (0.09) | 91.85 (0.09) | 90.17 (0.09) | 91.38 (0.08) |
| $B_2$ | -- | 100.00 (0.00) | 90.45 (0.09) | 91.85 (0.09) | 90.23 (0.09) | 91.20 (0.08) |
| $B_3$ | -- | -- | 99.96 (0.00) | 92.24 (0.07) | 86.38 (0.09) | 87.64 (0.08) |
| $B_4$ | -- | -- | -- | 99.98 (0.00) | 86.74 (0.09) | 87.81 (0.09) |
| $B_5$ | -- | -- | -- | -- | 99.94 (0.00) | 88.59 (0.08) |
| $B_6$ | -- | -- | -- | -- | -- | 99.96 (0.00) |
| ***Test*** | 56.36 (0.04) | 56.63 (0.03) | 71.55 (0.06) | 72.93 (0.05) | 85.11 (0.07) | 87.26 (0.06) |

Table 12: Incremental learning performance on the concentric circles data-set for neural networks trained using traditional back-propagation parameters.

| ***Learn++*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 98.7 | 96.7 | 91.4 | 91.4 | 95.3 | 95.3 |
| $B_2$ | -- | 96.1 | 87.1 | 85.8 | 92.2 | 91.6 |
| $B_3$ | -- | -- | 98.3 | 98.3 | 72.0 | 90.8 |
| $B_4$ | -- | -- | -- | 93.6 | 77.0 | 88.4 |
| $B_5$ | -- | -- | -- | -- | 88.0 | 95.2 |
| $B_6$ | -- | -- | -- | -- | -- | 96.4 |
| ***Test*** | 55.6 | 56.8 | 73.2 | 74.4 | 85.8 | 89.6 |

Table 13: The *Learn++* incremental learning performances of Polikar et al. (2001) for the concentric circles data-set.

| ***NFW*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 98.06 (0.11) | 96.27 (0.08) | 94.43 (0.07) | 95.24 (0.05) | 94.44 (0.07) | 95.06 (0.04) |
| $B_2$ | -- | 98.77 (0.08) | 94.50 (0.07) | 95.30 (0.06) | 94.46 (0.07) | 95.15 (0.10) |
| $B_3$ | -- | -- | 98.76 (0.08) | 95.78 (0.05) | 93.45 (0.06) | 94.40 (0.06) |
| $B_4$ | -- | -- | -- | 98.92 (0.05) | 93.38 (0.03) | 94.30 (0.05) |
| $B_5$ | -- | -- | -- | -- | 98.84 (0.06) | 94.99 (0.05) |
| $B_6$ | -- | -- | -- | -- | -- | 99.05 (0.06) |
| ***Test*** | 56.87 (0.06) | 57.53 (0.04) | 75.44 (0.03) | 76.16 (0.04) | 93.16 (0.05) | 94.22 (0.06) |

Table 14: Incremental learning performance on the concentric circles data-set for evolved neural networks without fast weights.

| ***Baseline*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 99.99 (0.00) | 94.84 (0.07) | 73.96 (0.16) | 72.55 (0.15) | 55.03 (0.20) | 50.90 (0.20) |
| $B_2$ | -- | 99.99 (0.00) | 73.97 (0.16) | 72.97 (0.15) | 54.58 (0.20) | 50.70 (0.21) |
| $B_3$ | -- | -- | 99.96 (0.00) | 91.17 (0.09) | 64.07 (0.16) | 61.71 (0.15) |
| $B_4$ | -- | -- | -- | 99.96 (0.00) | 64.65 (0.17) | 62.30 (0.14) |
| $B_5$ | -- | -- | -- | -- | 99.90 (0.010) | 85.14 (0.11) |
| $B_6$ | -- | -- | -- | -- | -- | 99.92 (0.01) |
| ***Test*** | 56.24 (0.04) | 56.71 (0.04) | 61.74 (0.09) | 61.78 (0.10) | 63.32 (0.12) | 62.99 (0.12) |

Table 15: Incremental learning performance on the varying classes concentric circles data-set for neural networks trained using traditional back-propagation parameters.

| ***NFW*** | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $B_1$ | 98.37 (0.84) | 95.68 (0.31) | 89.17 (0.97) | 88.62 (0.72) | 90.18 (0.42) | 88.61 (0.20) |
| $B_2$ | -- | 98.97 (0.46) | 89.26 (1.04) | 88.60 (0.73) | 90.09 (0.45) | 88.59 (0.18) |
| $B_3$ | -- | -- | 98.22 (0.81) | 94.31 (0.35) | 87.55 (1.03) | 86.89 (0.35) |
| $B_4$ | -- | -- | -- | 99.11 (0.40) | 88.16 (1.17) | 87.31 (0.28) |
| $B_5$ | -- | -- | -- | -- | 94.90 (1.94) | 91.97 (0.75) |
| $B_6$ | -- | -- | -- | -- | -- | 98.99 (0.38) |
| ***Test*** | 55.99 (0.54) | 56.94 (0.22) | 70.93 (0.74) | 71.23 (0.46) | 85.96 (1.33) | 88.12 (0.31) |

Table 16: Incremental learning performance on the varying classes concentric circles data-set for evolved neural networks without fast weights.

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| ***NFW 1*** | 55.99 (0.54) | 56.94 (0.22) | 70.93 (0.74) | 71.23 (0.46) | 85.96 (1.33) | 88.12 (0.31) |
| ***NFW 3*** | 56.64 (0.25) | 57.23 (0.18) | 71.75 (0.76) | 71.85 (0.46) | 88.21 (0.98) | 89.74 (0.15) |
| ***NFW 9*** | 56.92 (0.28) | 57.52 (0.15) | 72.42 (0.70) | 72.28 (0.48) | 89.82 (0.81) | 90.95 (0.21) |
| ***FW 1*** | 55.07 (0.81) | 57.12 (0.06) | 72.97 (0.21) | 74.09 (0.32) | 89.78 (0.25) | 91.79 (0.24) |
| ***FW 3*** | 55.85 (0.72) | 57.57 (0.06) | 73.83 (0.22) | 74.82 (0.29) | 91.07 (0.13) | 92.97 (0.18) |
| ***FW 9*** | 56.16 (0.81) | 57.75 (0.10) | 74.26 (0.26) | 75.04 (0.29) | 91.87 (0.22) | 93.58 (0.16) |

Table 17: Incremental learning test set performances on the varying classes concentric circles data-set for voting ensembles of 1, 3 and 9 evolved neural networks, with no fast weights (NFW) and with fast weights (FW).

## Figures

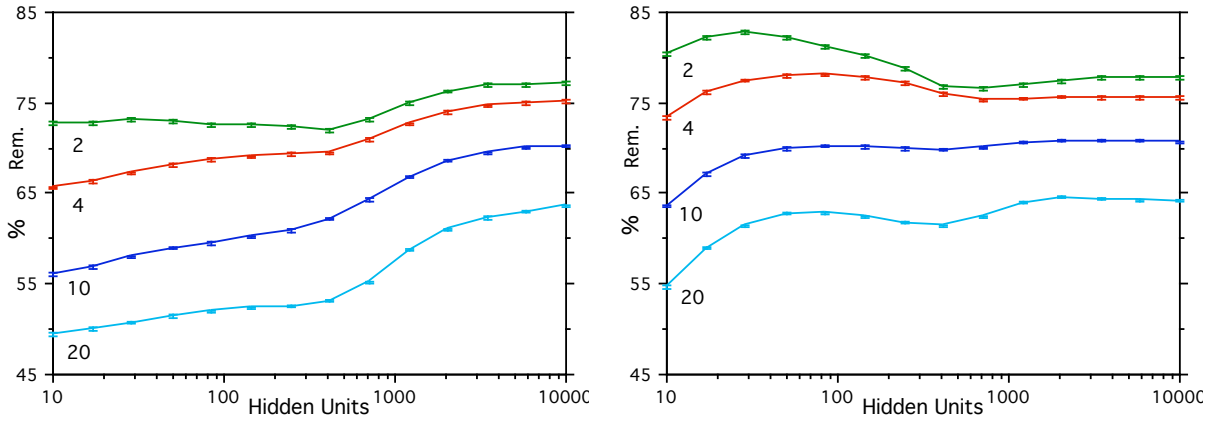


Figure 1: Baseline remembering performance on 20 old patterns after learning new patterns, as a function of the number of hidden units, using standard back-propagation networks with traditional learning parameters and testing tolerance 0.2 (left) and 0.5 (right). The line labels indicate the number of new patterns learned.

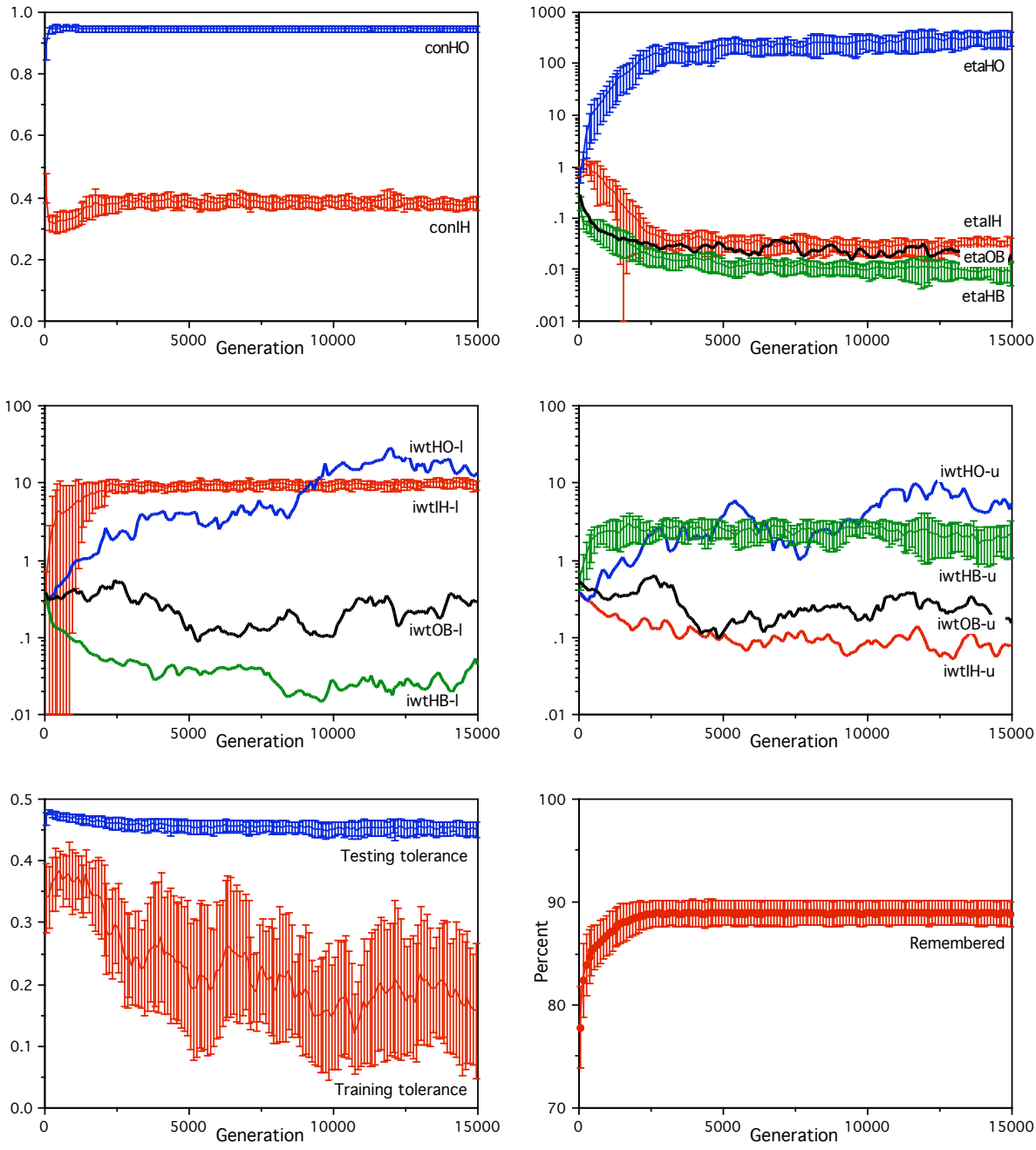


Figure 2: Evolution of basic neural networks with a maximum of 50 hidden units that aim to remember 20 items after learning four new items. The connectivity proportions (top-left), learning rates (top-right), lower and upper ranges of the random initial weight distributions (middle), training and testing tolerances (bottom-left), and the resultant remembering performance (bottom-right). Where no error bars are shown, there is enormous variation that would obscure the rest of the graph, indicating that the precise values of those parameters have little effect on performance.

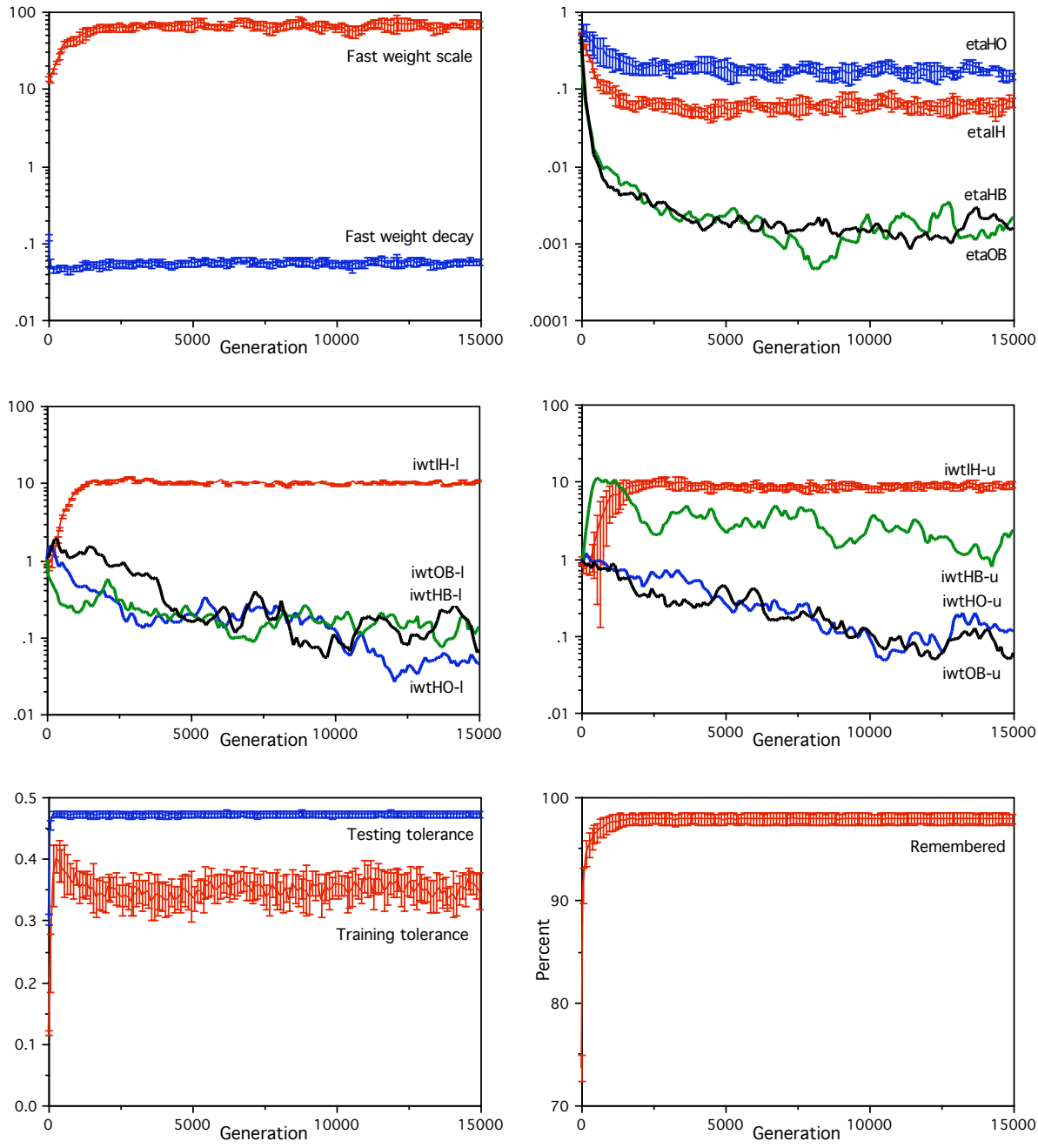


Figure 3: Evolution of neural networks with dual weights and a maximum of 50 hidden units that aim to remember 20 items after learning four new items. The fast weight scale and decay parameters (top-left), learning rates (top-right), lower and upper ranges of the random initial weight distributions (middle), training and testing tolerances (bottom-left), and the resultant remembering performance (bottom-right).

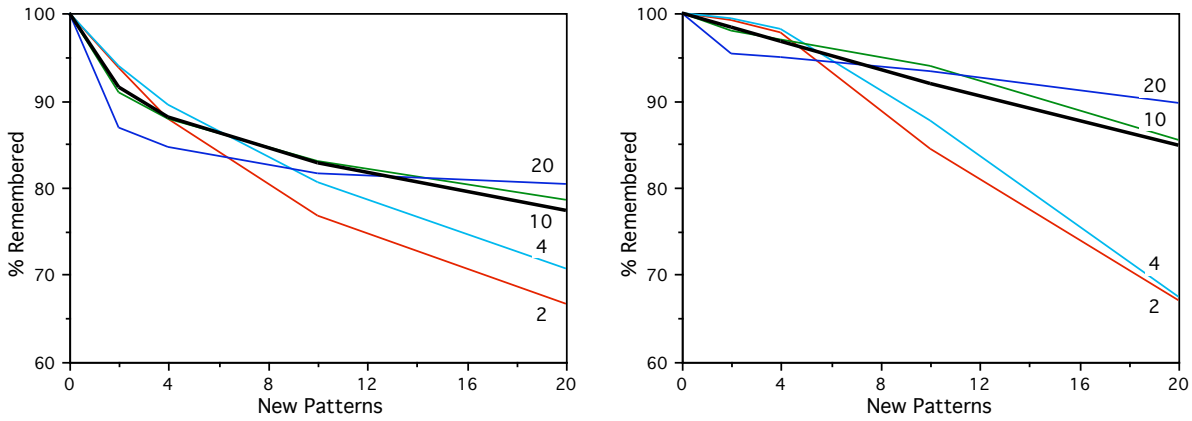


Figure 4: Comparison of remembering performances for evolved 50 hidden unit networks without fast-weights (left) and with fast-weights (right), after various numbers of new patterns have been learned. The thin lines result from evolution using the number of new patterns indicated by the labels. The thick lines result from evolution using a uniform distribution of new patterns.

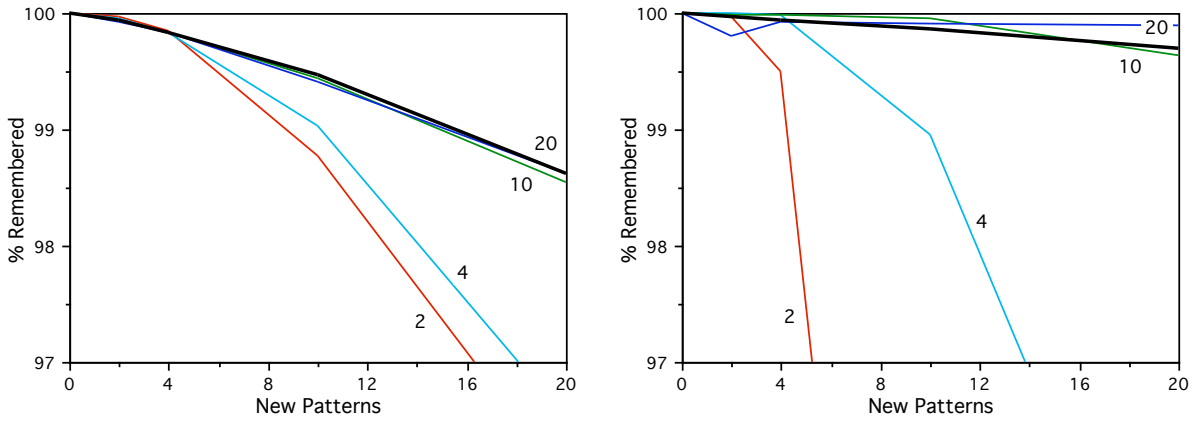


Figure 5: Comparison of remembering performances for evolved 10000 hidden unit networks without fast-weights (left) and with fast-weights (right), after various numbers of new patterns have been learned. The thin lines result from evolution using the number of new patterns indicated by the labels. The thick lines result from evolution using a uniform distribution of new patterns. Note the massive change of performance scale from Figure 4.

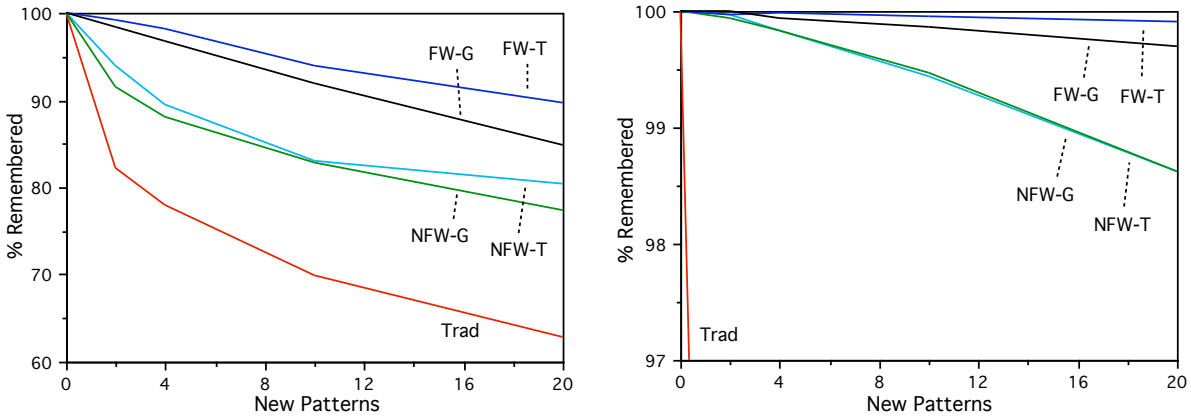


Figure 6: Remembering performance for networks with 50 hidden units (left) and 10000 hidden units (right). Traditional networks (Trad) are compared to evolved networks with fast-weights (FW) and no fast-weights (NFW), either tuned to particular numbers of new patterns (T) or evolved as general purpose networks (G). Note the massive difference in performance scale between the two graphs.

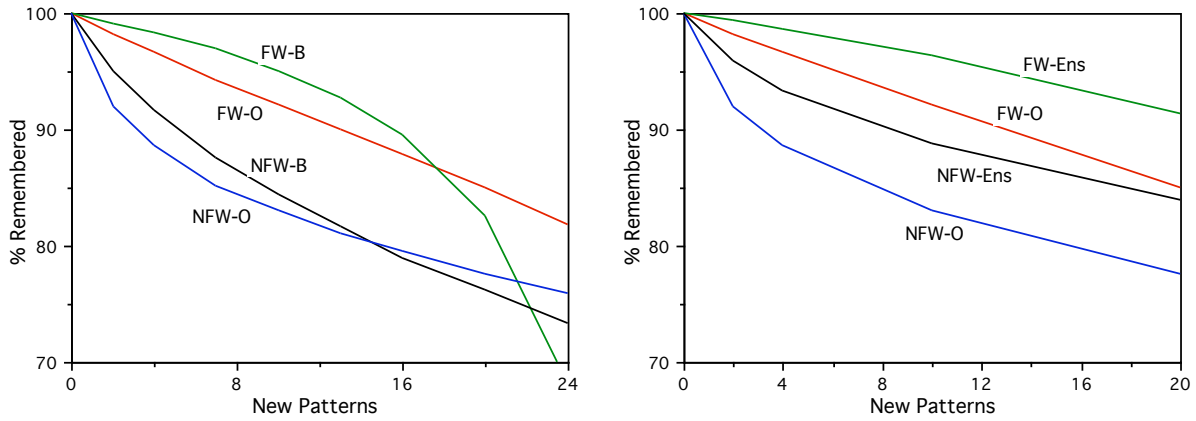


Figure 7: The effect of batch learning and ensemble averaging on networks with 50 hidden units with fast-weights (FW) and no fast-weights (NFW). Comparison of Batch training (B) and Online training (O) in the left graph, and comparison of Ensembles of size 3 (Ens) and individual online networks (O) in the right graph.

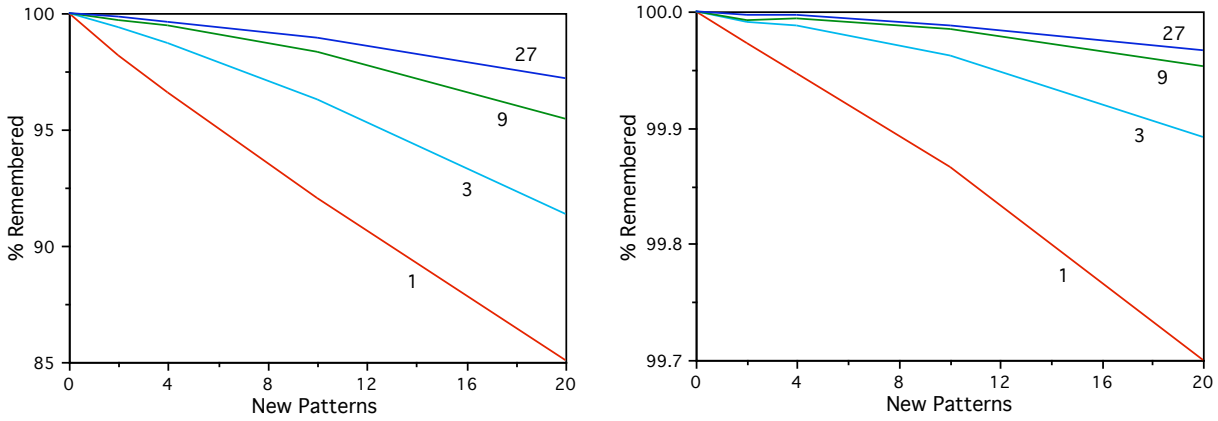


Figure 8: Remembering performance of ensembles of different sizes indicated by the line labels, for fast-weight networks with 50 hidden units (left) and 10000 hidden units (right). Note the massive difference in performance scale between the two graphs.

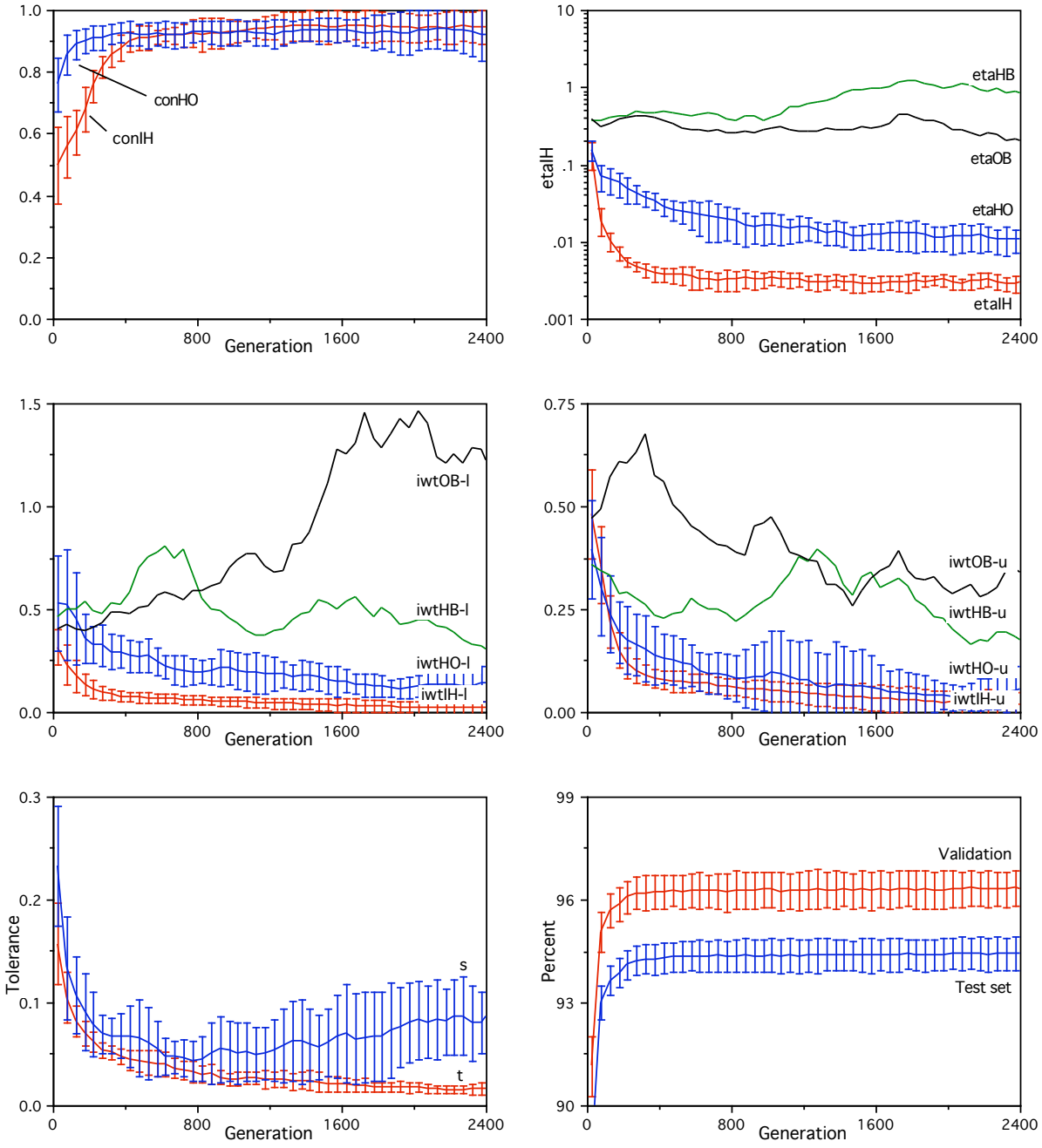


Figure 9: Evolution of networks for the optical digits generalization task with a maximum of 100 hidden units. The connectivity proportions (top-left), learning rates (top-right), lower and upper ranges of the random initial weight distributions (middle), training tolerance $t$ and stopping tolerance $s$ (bottom-left), and the resultant generalization performance measures (bottom-right). Where no error bars are shown, there is enormous variation indicating that the precise values of those parameters have little effect on performance.