Complex Systems **10** (1996) 257–281

Replicability of Neural Computing Experiments

D. Partridge[*]
W. B. Yates
Department of Computer Science,
University of Exeter,
Exeter EX4 4PT, UK

Abstract. The nature of iterative learning on a randomized initial architecture, such as backpropagation training of a multilayer perceptron, is such that precise replication of a reported result is virtually impossible. The outcome is that experimental replication of reported results, a touchstone of "the scientific method," is not an option for researchers in this most popular subfield of neural computing. This paper addresses the issue of replicability of experiments based on backpropagation training of multilayer perceptrons (although many of the results are applicable to any other subfield that is plagued by the same characteristics) and demonstrate its complexity. First, an attempt to produce a complete abstract specification of such a neural computing experiment is made. From this specification an attempt to identify the full range of parameters needed to support maximum replicability is made and it is used to show why absolute replicability is not an option in practice. A statistical framework is proposed to support replicability measurement. This framework is demonstrated with some empirical studies on both replicability with respect to experimental controls, and validity of implementations of the backpropagation algorithm. Finally, the results are used to illustrate the difficulties associated with the issue of experimental replication and the claimed precision of results.

1. Introduction

Experiments based on the iterative training of neural networks (NNs) are known to be sensitive to initial conditions in weight space and some studies have begun to explore the nature and extent of this sensitivity (e.g., [3, 7]). However, the initial conditions for training involve much more than the initial position chosen (usually by means of some "random" procedure) in weight space. Some facets of the initial conditions are routinely reported (e.g., the

[*]Electronic mail address: `derek@dcs.exeter.ac.uk`.

network architecture, the number of inputs, hidden and output units), while others are not (e.g., the precise criterion for learning algorithm termination). Some aspects of the initial conditions are thought to be insignificant (e.g., batch or online update, an “implementation detail” within backpropagation, see [1]) and others are simply unrecognized as characteristics of the initial state (e.g., the composition and size of the training and test sets).

In this paper we investigate the issue of replicability, that is, what needs to be specified in order to make a neural computing experiment replicable, and what determines the degreee to which it is replicable.

We present a framework and mechanisms for quantifying the sensitivity of such experiments to variations in the initial conditions. By “sensitivity” we mean extent of effect on the resultant trained network, that is how much does the parameter under study affect the behavior of the trained net? This view of sensitivity can be contrasted with, for example, the studies in [3, 7] into the effect on the rate of convergence to a solution during training. Our method of specification will concentrate on elucidating the parameters on which our experiments depend and making explicit the correctness criteria that must be satisfied in order to conduct them.

Ultimately, our goal is to clarify the requirements for replicability in such NN experiments, to introduce a statistical framework for improved replicability, and to begin to relate the degree of replicability achieved to the precision that can be justifiably associated with empirical results.

1.1 Background

In [7] it is reported that backpropagation is extremely sensitive to the choice of initial weights. While that study revealed some interesting chaotic behaviors and provided analytic insights into a number of observed phenomena, it does not have much impact from an engineering applications standpoint (as was freely admitted). In our study it is the nature of the solution, the trained net, that is of primary importance. The only attempt that was made to address solution sensitivity in [7] is based on a weak notion of equivalent solutions: “Two networks are considered equivalent if their weights have the same sign.” It is acknowledged in a footnote that “it is extremely difficult to know precisely the equivalence classes of solutions, so we approximated.” We shall also approximate, but less approximately. By using appropriate metrics and measures we present a definition of the approximate correctness we require of our networks, and use this to define a notion of functional equivalence for a population of trained networks that exhibit structural differences. Our notion of equivalence is founded on the similarities and differences in the observable behavior of trained networks, more succinctly in the parlance of connectionism: equivalent networks generalize identically, that is, two networks are equivalent if they fail on precisely the same test patterns. Furthermore, we take the view that an experiment has been replicated if it yields an equivalent network.

Testing, as a strategy for estimating degree of equivalence between two networks, is itself vulnerable, because the outcome must depend, to some degree, on the nature of the test set used. Yet the measured quantity is one supposedly inherent to the networks.

For a well-defined "toy" problem, such as the one we use (see section 2.3), the test set structure can be controlled, defined, and hence reproduced. But for a "real world" problem this luxury cannot be taken for granted. However, in practice all software is primarily validated by testing (proofs of correctness in terms of program infrastructure remains an academic exercise restricted to a few simple well defined examples).

Similarly, validation of NN implementations of real world problems must (for the foreseeable future, at least) be testing based. A crucial determinant of test set validity is that it exhibits the "normal usage profile" of the system under test. Without pretending that it is either easy or straightforward to construct valid test sets from available, but necessarily incomplete data, some reasonable attempt must be made when dealing with NN implementations of real world problems. There is no other option, and it is just such "valid" test sets that we use to assess replicability.

1.2 Approach

Our approach to the replicability issue is from two rather different strategies: first, we explore the potential for replicability based on formal specifications, and second, we measure replicability achieved using several numerical quantities derived from the network testing phase.

This study aims to deliver a precise and detailed description of our NN experiment consisting of formal specifications for the NN, the learning algorithm, and the task. From the specification we abstract a list of experimental parameters designed to fully characterize the initial conditions of our experiments and so provide a proper basis for maximum replicability. We then present the results of an investigation into the effect of varying some of these parameters on the observable behavior of a number of trained NNs, and hence an indication of the replicability actually achievable under various relaxations of controls. We provide a quantification of the effects of these changes using statistical techniques. Finally, a validation scheme for implementations of algorithms such as backpropagation is proposed and demonstrated.

2. A formal definition of an experiment

In this study we shall concern ourselves with two-layer feedforward networks trained to implement a simple function by a backpropagation learning algorithm with momentum. We restrict ourselves to this particular neural computing paradigm because it is the simplest backpropagation variant, and is one of the most widely used, and so constitutes (in a practical sense) an important type of neural computing experiment to be able to replicate.

Most studies that use this type of network and training algorithm refer to the seminal paper in [11] as the "definition" of their experimental setup. It is tacitly assumed that all these experiments use functionally identical versions of backpropagation, that is they differ only in implementation detail. Unfortunately, problems experienced by ourselves and other researchers in the field when trying to reproduce published experiments contradict this assumption.

In this section we present a formal definition of our experiments with the intention of making explicit the experimental parameters and assumptions on which our studies depend.

2.1 The neural network

We concern ourselves with a class of feedforward two-layer NNs based on a definition presented in [5]. They specify a two-layer feedforward NN over the real numbers $\mathcal{R}$ by a triple (r, A, G) consisting of a number $r \in \mathcal{N}$, a family of *affine* functions A, and an activation function G. The family of functions A has the form

$$\begin{aligned} &(\forall a_j \in A) \quad a_j : \mathcal{R}^r \rightarrow \mathcal{R} \\ &(\forall x \in \mathcal{R}^r) \quad a_j(x) = \sum_{i=1}^{r} w_{ji}^1 x_i + b_j^1 \end{aligned}$$

for $j = 1, \ldots, q$, where $x \in \mathcal{R}^r$ are the inputs to the network, $w_{ji}^1 \in \mathcal{R}$ is the weight from the ith input unit to the jth hidden unit in layer 1, and $b_j^1 \in \mathcal{R}$ is the bias of the jth hidden unit. A single hidden layer feedforward NN with one output is specified by the function $f : \mathcal{R}^r \rightarrow \mathcal{R}$ defined by

$$f(x) = \sum_{j=1}^{q} w_j^2 G(a_j(x))$$

where $w_j^2 \in \mathcal{R}$ is the weight from the jth hidden unit to the output unit in layer 2 and q is the number of hidden units. We shall assume that our networks have several output units labeled $k = 1, \ldots, s$ each with the activation functions G (see [5]). In this study G is the *logistic* or sigmoid activation function (see [11], page 329).

Later, when we come to specify our convergence criteria we shall, for the sake of clarity, represent our network as a parameterized function

$$f : W^n \times A^r \rightarrow A^s$$

with the intention that $f(w, a,) \in A^s$ is the output pattern computed by the NN f, parameterized with weights $w = (w^1, b^1, w^2, b^2)$ on input pattern a. In this case $W = \mathcal{R}$ and $A = [0, 1]$, and $n = q(r + s) + q + s$ is the number of weights and biases, r is the number of inputs, and s is the number of outputs.

2.2 The learning algorithm

We train our two-layer feedforward NN $f : W^n \times A^r \to A^s$ using an online backpropagation learning algorithm with momentum. The learning algorithm itself is specified by three equations (see [11], page 327). It is a NN based implementation of the steepest (gradient) descent optimization algorithm, due originally to Fermat (see [11]). The task of the learning algorithm is encoded in a (usually finite) set of patterns P, with elements

$$p = (x_p, y_p) = (x_{p1}, \dots, x_{pr}, y_{p1}, \dots, y_{ps}).$$

The algorithm processes a sequence of patterns from P, and computes a weight update $\Delta w_{ij}(p)$ for each weight w_{ij} and each pattern p.[1] The weight updates are used to modify the weights of the network $w \in W$, until, typically, the error of the network, as measured by the function

$$E = \sum_{p \in P} E_p = \sum_{j=1}^{s} (y_{pj} - o_{pj})^2$$

falls below some given magnitude. Formally, consider a nonempty set of patterns P and let T be the natural numbers. The set $[T \to P]$ is the set of all sequences or *streams* over P with the intention that $a \in [T \to P]$ is seen as a sequence

$$a(0), a(1), a(2), \dots$$

of patterns from P. The learning algorithm computes a weight update $\Delta w_{ij}(a(t))$ for each $t \in T$. These updates are added to the current weights of the network and in this way the learning algorithm produces a sequence of weights.

In symbols we have

$$\begin{aligned} W_{ji}(0, a, w) &= w_{ji} \\ W_{ji}(t+1, a, w) &= W_{ji}(t, a, w) + \Delta w_{ji}(a(t)) \end{aligned}$$

where w_{ji} is the initial weight of the connection and $t \in T$ indexes the pattern being processed.

In our experiments we also employ a *momentum term* α which determines the effect of past weight updates on the current direction of descent (see [11], page 330).

With this in mind we represent our learning algorithm by the stream transformer

$$L : [T \to A]^{r+s} \times W^n \times \mathcal{R}^2 \to [T \to W]^n$$

where $L(a, w, \eta, \alpha)(t)$ is the set of weights returned at time t on pattern stream a, with initial weights w, learning rate η, and momentum α.

[1] We employ the functional notation $\Delta w_{ij}(p)$ instead of the more usual subscript notation $\Delta_p w_{ij}$ to highlight the dependence on the current pattern being processed.

2.3 Task specification

The launch interceptor problem has been used in a number of software engineering experiments concerning correctness and reliability (e.g., [6]). Our task is to implement one of the 15 so-called launch interceptor conditions, LIC1. In [6] LIC1 is specified as a predicate that evaluates to true if:

> "There exists at least one set of two consecutive data points that are a distance greater than the length `LENGTH1` apart
>
> $(0 \leq \texttt{LENGTH1})$."

Here, `LENGTH1` is a parameter of the condition. We shall abstract a functional representation of this precise, though informal description. The intention here is to make explicit the exact structure and semantics of the network task.

Let $S = [0,1] \times [0,1]$ represent the set of all data points, let $a = (x_1, y_1)$ and $b = (x_2, y_2)$ represent *two consecutive data points* and note that the function

$$d(a,b) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

is the euclidean distance on S. Formally, we require a NN implementation of the bounded function

$$f : S \times S \times [0,1] \rightarrow \mathcal{B}$$

defined by

$$f(a, b, \texttt{LENGTH1}) = \begin{cases} 1 & \text{if } d(a,b) > \texttt{LENGTH1}; \\ 0 & \text{otherwise.} \end{cases}$$

In fact, as our task will ultimately be executed on some digital computer, we shall restrict our attention to the finite subset of rationals, with six decimal places, in $S \times S \times [0,1]$.

LIC1 is a simple, well defined problem, although highly resistent to "correct" implementation with NNs. It is not a typical neural computing problem. It was however a conscious choice to use such a problem for the purposes of illustration in this paper. The resultant demonstrations of various aspects of the replicability difficulties are not confused by absence of information on test set composition. It then remains to provide assurance that the observed replicability phenomena are, in general, not artifacts of the chosen problem, or problem type.

2.4 Training and test sets

Many NN experiments reported in the literature fail to adequately characterize the structure of the training and test sets. A consequence of this is

```
seed random number generator
for i = 0 to 1000 do
    $x_1$ = random()
    $y_1$ = random()
    $x_2$ = random()
    $y_2$ = random()
    $LENGTH1$ = random()
    print$\langle x_1, y_1, x_2, y_2, LENGTH1, g(x_1, y_1, x_2, y_2, LENGTH1)\rangle$
```

Figure 1: Training set generator algorithm, where `random()` is the pseudorandom number generator of a Silicon Graphics computer system.

```
for $x_1$ = 0 to 1.0 step 0.1 do
  for $y_1 = 0$ to 1.0 step 0.1 do
    for $x_2 = 0$ to 1.0 step 0.1 do
      for $y_2 = 0$ to 1.0 step 0.1 do
        for $LENGTH1 = 0$ to 1.0 step 0.1 do
          print $\langle x_1, y_1, x_2, y_2, LENGTH1, g(x_1, y_1, x_2, y_2, LENGTH1)\rangle$
```

Figure 2: Test set generator algorithm.

that any reported training and generalization results are difficult to interpret, and virtually impossible to reproduce. Proper characterization of the training and test sets is one essential prerequisite of replicability.

The five training sets (that we shall use later) were constructed by selecting a seed $i = 1, 2, \ldots, 5$ and generating 1000 random triples $\langle a, b, \texttt{LENGTH1}\rangle$ and applying the task specification g (see Figure 1). In principle, we are able to generate the entire set of example patterns (recall that we have restricted our attention to rationals of six decimal places). In practice, however, this is (usually) not the case and a specification of the task function is usually encoded in a (small) training set acquired during the requirements analysis phase of the network design.

Our test set consists of 161051 patterns equally spaced throughout the pattern space, and constructed according to the algorithm shown in Figure 2. The resulting test set (of 161051 or 11^5 patterns) ensures that our networks are tested over the whole input space as opposed to some subset.

2.5 Convergence criteria

Given a set of patterns P that encodes our task specification, we usually require that the error of the learning algorithm E falls below some given magnitude. Such a convergence criterion does not allow us much control over the nature of acceptable network solutions and fails to take into account any *a priori* knowledge of a task. We shall employ a stronger convergence criterion that addresses these deficiencies.

Specifically, let μ be a σ-finite measure defined on a σ-algebra in X (see [4]), and let d be a metric in Y (see [2]). We require that our learning algorithm either terminates at time $t = e$, or for some time $t < e$ returns a set of weights $L(p, w, \eta, \alpha)(t) = w$ such that the NN satisfies

$$[\mu\{x_p | d(g(x_p), f(w, x_p)) \geq \varepsilon\} < \delta].$$

And, in this case, we shall say that the learning algorithm has *converged* to an approximately correct implementation of the target function g, to some accuracy ε and some tolerance δ, under metric μ.

In this study, let $r = 5$, $s = 1$, and $d(x, y) = |x - y|$. Assume that X is finite, and let μ be the measure defined over the power set of X by

$$\forall x_p \mu(\{x_p\}) = \frac{1}{|X|}$$
$$\forall U \mu(U) = \frac{|U|}{|X|}.$$

The measure μ can be interpreted as a uniform probability distribution, that is, every point in X has an equal probability of occurring naturally. Intuitively, in this case, our correctness specification implies that learning terminates when our NN is ε-close to our target function, on all but some fraction δ of training patterns.

2.6 Initial conditions

From these formal specifications we can extract a set of the initial conditions designed to prescribe a replicable neural computing experiment. Table 1 lists the parameters necessary (it also includes specific values that are used later). At this point the objective is to explicitly layout the parameters that must be specified.

Despite our efforts to provide a complete and precise specification of our experiments, we recognize that it falls short of our goal. The various "generators" (initial weights, training and test sets) will inevitably produce slightly different results as the hardware and software platforms, upon which they are implemented, vary. In an ideal computational environment these generators would be presented as abstract specifications that always deliver the same results on any correct implementation of some standard virtual machine. However, the reality is that results are likely to differ from (real) machine to machine, and (in some cases) from compiler to compiler. And, while these

Table 1: Experimental parameters.

Level	Name	Parameter
Network	Input Units	$r = 5$
	Hidden Units	$q = 8, 9, 10, 11, 12$
	Output Units	$s = 1$
	Activation	$G = sigmoid$
Learning Algorithm	Initial Weights Seeds	$i = 1, 2, 3, 4, 5$
	Initial Weights Range	$[-0.5, 0.5]$
	Learning Rate	$\eta = 0.05$
	Momentum	$\alpha = 0.5$
	Update	online or batch
Task	Accuracy	$\varepsilon = 0.5$
	Tolerance	$\delta = 0.0$
	Training Set Seeds	$i = 1, 2, 3, 4, 5$
	Training Pattern Generator	specified in Figure 1
	Number of Training Patterns	$m = 1000$
	Epochs	$e = 20000$
	Test Pattern Generator	specified in Figure 2
	Number of Test Patterns	$l = 161051$

possibly small discrepencies may not be significant in conventional computational experiments, the iterative nature of neural computing is such that they can become compounded to high levels of significance. Hence our earlier assertion that total replicability is impossible in most cases.

3. Generalization diversity

Having isolated a number of experimental parameters and requirements, we now turn to the question of quantifying the differences which may be observed between different NN experiments. We make use of the following two measures.

1. The "generalization diversity" or GD, that occurs within groups of networks (see [9]).

2. The "intergroup diversity" or ρ that occurs between groups of networks (see [8]).

3.1 Intraset diversity

Consider a set A of N NNs each trained to perform a well defined task. Assume that each network in A is trained using differing initial conditions and then evaluated on a set of l test patterns. If the number of patterns that have failed on (any) $n = 1, \ldots, N$ networks in A is denoted k_n, then the probability p_n that exactly $n = 1, \ldots, N$ versions fail on a randomly selected test pattern is given by the formula

$$p_n = \frac{k_n}{l}.$$

Thus, the probability that a *randomly* selected version in A fails on a randomly selected input is given by

$$P(1 \text{ fails in } A) = \sum_{n=1}^{N} \frac{n}{N} p_n.$$

The probability that two versions in A selected at random (*with replacement*) both fail on a randomly selected input is given by

$$P(2 \text{ both fail in } A) = \sum_{n=1}^{N} \frac{n^2}{N^2} p_n.$$

Note that in this case it is possible that the same network is selected twice. In contrast, the probability that two versions in A selected at random (*without replacement*) both fail on a randomly selected input is given by

$$P(2 \text{ different versions both fail in } A) = \sum_{n=2}^{N} \frac{n(n-1)}{N(N-1)} p_n.$$

In this case, we note that it is impossible to select the same network twice. Thus, we define the generalization diversity of a set of networks A to be

$$GD = \frac{P(1 \text{ fails in } A) - P(2 \text{ different versions both fail in } A)}{P(1 \text{ fails in } A)}.$$

The measure GD has a minimum value of 0 when the N networks are all identical (as measured by test set failures), and a maximum value of 1 when all versions are maximally different, that is, every test pattern failure is unique to one of the versions.

3.2 Interset diversity

In order to measure the diversity between two groups of networks we shall employ a measure of correlation ρ (see [8]). Consider two nonempty sets of networks A and B, each with N_A and N_B networks, trained to perform a well defined task. Assume that each network in A and B is trained using differing initial conditions and then evaluated on a common set of l test patterns.

If $k_{n_A n_B}$ is the number of patterns that have failed on n_A versions in A *and* on n_B versions in B, then the probability that exactly n_A nets in group A *and* exactly n_B nets in group B all fail on a randomly selected test pattern is given by the formula

$$p_{n_A n_B} = \frac{k_{n_A n_B}}{l}.$$

From this, the probability that two randomly selected versions (one from each group) both fail on a randomly selected pattern is given by

$$P(1 \text{ fails in } A \;\&\; 1 \text{ fails in } B) = \sum_{n_A}^{N_A} \sum_{n_B}^{N_B} \frac{n_A n_B}{N_A N_B} p_{n_A n_B}.$$

Thus we define ρ as

$$\rho = \frac{P(1 \text{ fails in } A \;\&\; 1 \text{ fails in } B) - P(1 \text{ fails in } A)P(1 \text{ fails in } B)}{\sqrt{(P(2 \text{ both fail in } A) - P(1 \text{ fails in } A)^2) * (P(2 \text{ both fail in } B) - P(1\text{fails in } B)^2)}}.$$

We may interpret different values of ρ as follows.

1. A value of $\rho = 1$ indicates that the results of our groups are identical, that is, the experiments are exact replications of one another. This implies that every network in one group has an exact replication in the other group.

2. A value of $\rho > 0$ indicates that our groups have not replicated exactly, although the differences in the results are (statistically) similar, and the value of ρ indicates the degree of similarity.

3. A value of $\rho = 0$ indicates that the differences in the results of our groups are (statistically) unrelated; there is no evidence of replication between individual networks in the two groups.

4. A value of $\rho \leq 0$ indicates that test pattern failures in one group are unlikely to be replicated in the other group, and again the value of ρ indicates the degree to which this occurs.

5. A value of $\rho = -1$ indicates that within each group all failures are coincident, that is, each network in a group fails on the same patterns, while between the two groups no failures are coincident, that is, no pattern that fails in group A also fails in group B, and vice versa. In this (extreme) case we note that the GD of each group is 0, that is, total lack of replication of failures between the two groups requires total replication within each group.

The correlation coefficient ρ is thus a measure of the extent to which the individual network experiments in one group have been replicated in another group. It permits measurement of replicability over a set of individual networks and therefore can be used to avoid the idiosyncracies of individual nets.

4. An empirical study of replicability

In order to demonstrate the application of our measures, we present a small, but systematic investigation into the effect of changing update strategy used in the backpropagation learning algorithm from *online* to *batch*. Although, in theory, both update strategies lead to implementations of gradient descent, in practice they can lead, as we shall show, to different NN solutions. In the literature, the choice of update strategy is often seen as merely an implementation detail, and so the choice of update strategy is omitted from the report of the experiment entirely.

Specifically, we shall quantify the effect of the choice of update strategy with respect to a population of 30 networks. The population was constructed by training a group of 15 networks using first online, and then batch update. The group consists of three subgroups of five networks each (denoted W, A, and T), and constructed using the parameters shown in Table 1, according to the following prescriptions.

1. Group W: *Initial Weights.* A NN with $r = 5$ input units, $n = 10$ hidden units, and $s = 1$ output unit was trained on a single randomly constructed training set p_3 of 1000 patterns, generated using seed 3, using five different sets of weights w_i, randomly generated from initial seeds $i = 1, 2, 3, 4, 5$. We shall label these networks T3-A10-Wi.

2. Group A: *Hidden Units.* Five NNs with $r = 5$ input units, $n = 8, 9, 10, 11, 12$ hidden units, and $s = 1$ output unit were trained on a single randomly constructed training set p_3 of 1000 patterns generated using seed 3, using one set of weights w_3, randomly generated from initial seed 3. We shall label these networks T3-An-W3.

3. Group T: *Training Set.* Five NNs, each with $r = 5$ input units, $n = 10$ hidden units, and $s = 1$ output unit, were each trained on one of five randomly constructed training sets p_i, of 1000 patterns generated using seeds $i = 1, 2, 3, 4, 5$, using one set of weights w_3, randomly generated from initial seed 3. We shall label these networks Ti-A10-W3.

Where appropriate we distinguish between networks generated with online or batch update by the insertion of a "b" in the label of networks generated using the latter process. The two larger groups, online and batch, are distinguished by denoting the component groups as A, W, or T and Ab, Wb, or Tb, respectively.

4.1 Training

The results of training our population of 30 NNs is shown in Table 2. The results demonstrate a high level of convergence, with only three networks failing to learn after 20000 epochs. In these cases the numbers of patterns remaining unlearned was small.

The first point to note is that one network experiment, that involving network T3-A10-W3, is repeated within each of the six groups. Note further that it appears to train in precisely the same way within the three batch groups, and within the three online groups (same error to six decimal places and same number of epochs to convergence), but that *between these two groups* this network trains very differently (e.g., nearly 50% more epochs to convergence when using batch rather than online weight update).

This result suggests two things. First, we are completely and precisely replicating the basic initial conditions from group to group, because we do seem to obtain complete replicability for individual network experiments (but remember we are using the same hardware, software, random number generator, *etc.*, all of which would not normally be possible for researchers at another site, or at another time). Second, the switch from batch to online update strategy appears to wreck the replicability at the level of individual nets. However, it may be that apparently large training differences do not become manifest as similarly large generalization differences, that is, all six instances of the trained network T3-A10-W3 may be much the same in terms of generalization behavior. To explore this possibility we need to test the trained nets.

4.2 Testing

The results of testing each of these NNs on our test set of 161051 is shown in Table 3. The results demonstrate a high level of generalization (minimum 96.04%, maximum 97.67%) and appear to indicate a low level of diversity, that is, it appears that our networks have all converged to roughly equivalent solutions as witnessed by the similarity in generalization results. This suggests that, in terms of the generalization results, we might have achieved replicability to within less than 2%.

In addition, it can be seen that again, within the batch and online groups, the evidence points to complete replicability which strongly suggests that the test is indeed identical within each group of three. Further examination of the successes and failures of the test patterns on network T3-A10-W3 confirmed that it was precisely the same 5694 patterns that failed within the online group, and the same 6332 patterns that failed within the batch groups.

Apart from this example of complete replicability when precisely the same network is subjected to precisely the same experiment, notice that no other examples of total replication are evident in Table 3, which suggests that varying weight initialization, varying number of hidden units, and varying the training set composition (i.e., different randomly selected patterns) will each individually destroy replicability at the level of individual nets. This is as we would expect, but note also that there are no examples of replicability between the batch and online groups.

Between these two groups the most similar experiment appears to be the T3-A10-W4 network with 5515 test pattern failures in the W group, and 5520 failures within the Wb group (shown in bold in Table 3). This is a difference of only five patterns in a test set of 161051, that is, 0.003%, which is not too significant, perhaps. But what is hidden by this presentation is whether the "overlap" in test failures, that is, the 5515 tests that they both failed on, are the *same* 5515 tests. Clearly, to support the claim that the T3-A10-W4 experiment was effectively replicated between the batch and online strategies, these 5515 failures ought to be exactly (or very nearly exactly) the same 5515 test patterns. But when we examine the test results more

Table 2: The results of training the online and batch groups.

Incorrect	Error	Epochs	Version	Group
0	2.046708	5331	T3-A10-W1	
0	1.414763	10757	T3-A10-W2	
0	1.885597	5029	T3-A10-W3	W
0	2.106218	6953	T3-A10-W4	
0	1.849380	6627	T3-A10-W5	
0	2.289617	9906	T3-A8-W3	
0	2.232139	6343	T3-A9-W3	
0	1.885597	5029	T3-A10-W3	A
0	2.920789	3871	T3-A11-W3	
0	1.441672	6186	T3-A12-W3	
1	1.111853	20000	T1-A10-W3	
0	4.774262	1863	T2-A10-W3	
0	1.885597	5029	T3-A10-W3	T
0	1.846951	2824	T4-A10-W3	
0	4.150957	1430	T5-A10-W3	
0	1.050877	9211	T3-A10-Wb1	
3	5.272037	20000	T3-A10-Wb2	
0	2.076072	7461	T3-A10-Wb3	Wb
0	1.922335	7622	T3-A10-Wb4	
0	2.435892	12153	T3-A10-Wb5	
0	1.525628	7006	T3-Ab8-W3	
0	2.253490	15224	T3-Ab9-W3	
0	2.076072	7461	T3-Ab10-W3	Ab
0	1.384206	8118	T3-Ab11-W3	
0	1.445946	9845	T3-Ab12-W3	
2	2.218627	20000	Tb1-A10-W3	
0	4.151362	2660	Tb2-A10-W3	
0	2.076072	7461	Tb3-A10-W3	Tb
0	1.992639	8255	Tb4-A10-W3	
0	2.005054	3149	Tb5-A10-W3	

closely (as provided in Table 4), we find that both versions only have 3114 test failures in common; the other 2401 seemingly joint failures were, in fact, a different 2401 test failures for each of the two versions. So a more accurate measure of the replicability achieved in this case is that 4807 test patterns failed uniquely on one of the two versions of T3-A10-W4, that is a difference of $4807/161051 \approx 3\%$, a much more potentially significant discrepancy.

If we plot the test failures for T3-A10-W4 and T3-A10-Wb4 (see Figures 3 and 4) the difference in test failures can be clearly seen. Each test failure is plotted as euclidean distance (i.e., $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$) against

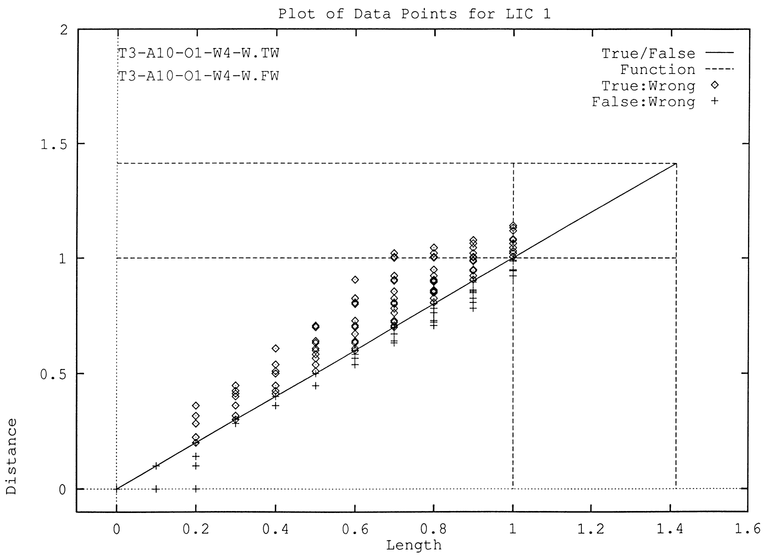

Figure 3: Graphical representation of the failures of T3-A10-W4 during testing.

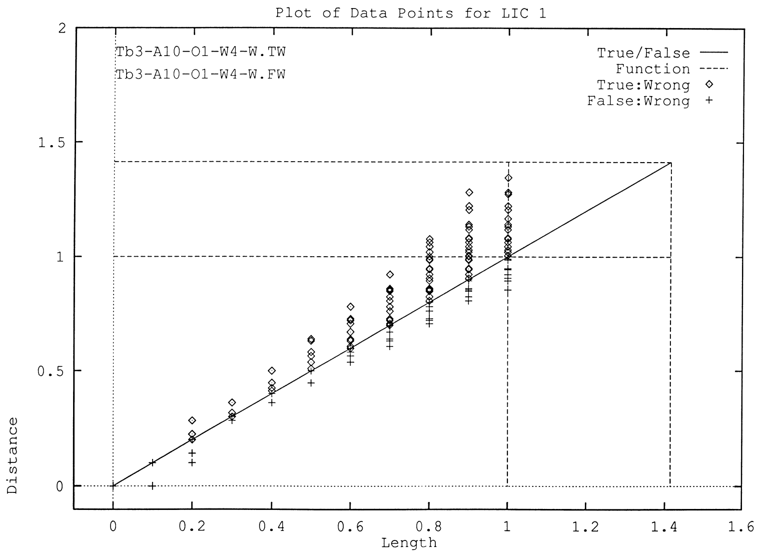

Figure 4: Graphical representation of the failures of T3-A10-Wb4 during testing.

*LENGTH*1. The decision boundary for the task LIC1 is then the line *Length* = *Distance*; points above this line are tests that should evaluate to 1 but actually produced 0, and points on and below the line should have evaluated to 0 but actually gave 1.

A comparison of these two plots clearly reveals the many test pattern failures that were different between the two networks. In addition, it can be seen that the test failure differences are heavily concentrated in the 1 results wrongly computed as 0, that is, on tests above the decision boundary.

From Table 3, the maximum lack of replicability appears to be for networks T3-A11-W3 and T3-Ab11-W3 (shown in bold in Table 3). In group A it exhibits 6376 test pattern failures and in group Ab there are 4006 such failures. In this case there might be as many as 4006 common failures (i.e., the same test pattern failing on both versions), but further inspection reveals that only 2580 test patterns, in fact, failed on both versions. The number of unique failures was 5222, which also gives a difference of $5222/161051 \approx 3\%$.

The tentative conclusion must be that the choice of batch or online update strategy is *not* an implementation detail. It appears to introduce performance differences of around 3% in individual nets. And furthermore, this lack of replicability may be almost completely hidden in overall generalization performance measures. Generalization performance for T3-A10-W4 is 96.58% in group W and 96.57% in group Wb.

4.3 Replicability "on average"

It is possible that, although the choice of the weight update strategy may undermine replicability between individual network experiments, it may be effectively cancelled out by working with groups of networks. This is the next hypothesis examined.

Using the measure of generalization diversity (*GD* specified earlier), which is based on the numbers of unique and nonunique test failures in a set of nets, we can quantify how differently individual networks in a set of networks perform on a given test set. Clearly, if the five networks that constitute the W group, say, were identical, that is, the same training had replicated the same trained network five times despite different random weight initialization, then there would be no performance difference, the *GD* value would be 0. At the other extreme, if the five networks in group W were maximally different (i.e., different weight initialization led to minimum replicability in experiments), then all test failures would be unique to just one of the networks (any common failure indicates similarity of performance), and the *GD* value would be 1.

As an example, Table 5 provides the *GD* value for each group and the average of the generalization performances for the five networks in each group. Again, it is clear that lack of complete replicability is evident even in these "average" or group measures. Between batch and online update groups, the maximum difference between *GD* values is 4% (between W and Wb), and the maximum difference between average generalization performance is 0.43% (between A and Ab). The two most similar groups appear to be T and Tb,

Table 3: The results of testing the online and batch groups.

Group	Version	Failures	Success	Prob(fail)	Prob(success)
W	T3-A10-W1	5390	155661	0.0335	0.9665
	T3-A10-W2	6164	154887	0.0383	0.9617
	T3-A10-W3	5694	155357	0.0354	0.9646
	T3-A10-W4	**5515**	**155536**	**0.0342**	**0.9658**
	T3-A10-W5	6064	154987	0.0377	0.9623
Wb	T3-A10-Wb1	5722	155329	0.0355	0.9645
	T3-A10-Wb2	6374	154677	0.0396	0.9604
	T3-A10-Wb3	6332	154719	0.0393	0.9607
	T3-A10-Wb4	**5520**	**155531**	**0.0343**	**0.9657**
	T3-A10-Wb5	5254	155797	0.0326	0.9674
A	T3-A8-W3	4930	156121	0.0306	0.9694
	T3-A9-W3	4776	156275	0.0297	0.9703
	T3-A10-W3	5694	155357	0.0354	0.9646
	T3-A11-W3	**6376**	**154675**	**0.0396**	**0.9604**
	T3-A12-W3	6083	154968	0.0378	0.9622
Ab	T3-Ab8-W3	4124	156927	0.0256	0.9744
	T3-Ab9-W3	5871	155180	0.0365	0.9635
	T3-Ab10-W3	6332	154719	0.0393	0.9607
	T3-Ab11-W3	**4006**	**157045**	**0.0249**	**0.9751**
	T3-Ab12-W3	4106	156945	0.0255	0.9745
T	T1-A10-W3	5307	155744	0.0330	0.9670
	T2-A10-W3	4734	156317	0.0294	0.9706
	T3-A10-W3	5694	155357	0.0354	0.9646
	T4-A10-W3	4779	156272	0.0297	0.9703
	T5-A10-W3	3804	157247	0.0236	0.9764
Tb	Tb1-A10-W3	5745	155306	0.0357	0.9643
	Tb2-A10-W3	4217	156834	0.0262	0.9738
	Tb3-A10-W3	6332	154719	0.0393	0.9607
	Tb4-A10-W3	3748	157303	0.0233	0.9767
	Tb5-A10-W3	4452	156599	0.0276	0.9724

Table 4: Test pattern failure summary.

Group	Version	Unique Failures	Common Failures
W	T3-A10-W4	2401	3114
Wb	T3-A10-Wb4	2406	3114
A	T3-A11-W3	3796	2580
Ab	T3-Ab11-W3	1426	2580

Table 5: Intragroup generalization diversity and average generalization.

Group	*GD*	Avg. Gen.(%)
A	0.47	96.54
Ab	0.50	96.97
W	0.45	96.42
Wb	0.49	96.37
T	0.61	96.98
Tb	0.60	96.96

Table 6: Inter-group similarity

Groups	ρ
A,Ab	0.83
W,Wb	0.84
T,Tb	0.85

with a *GD* difference of only 1% and generalization difference of only 0.02%. These results might be taken to suggest that across a population of networks variation of the (random) composition of the training set more or less cancels out the lack of replicability between batch and online update for individual nets. However, we must examine the results more carefully before we can even begin to conclude this.

These results also suggest that, within a group, variation among the individual networks is substantially more sensitive to training set composition ($GD \approx 0.6$ in both T and Tb groups) than to either weight initialization or to number of hidden units ($GD \approx 0.5$ for both W and both A groups), and that sensitivity to variation of these latter two parameters is about equal. More extensive studies (e.g., [9]) confirm these tentative observations.

In Table 6 the intergroup diversity values are given. This quantity is a measure of the similarity between the variety of test failures in two groups of nets. If in both groups precisely the same number of networks fail on precisely the same test patterns, then the ρ value will be 1, but if between the two groups every test pattern failure is unique to its own group then the correlation coefficient value will be negative. A failure distribution with a ρ value very close to 1 is illustrated later in Table 9.

As can be seen, although the amount of diversity within each group is similar (see Table 5), the specific differences within each batch update group are quite different from the differences in each online update group. The intergroup ρ values indicate that the between group differences are of a similar magnitude, although this tells us nothing about whether these observed differences are similar in terms of the coincidence of specific test failures. Taking groups T and Tb as examples: they differ in average generalization performance by only 0.02%; they differ in GD by only 1.7%; but the similarity of individual failures *between the two groups* correlate to only 0.85. So, as with replicability between single-network experiments, there is an illusion of similarity in the average measures on groups of nets, but these similar "difference" values are composed from very different constituents. The ρ values suggest that, in terms of group performance (rather than individual networks), the lack of replicability (with respect to specific test failures) is more that 10%. So lack of replicability between individual networks does not "cancel out" within groups. On the contrary, it appears to become magnified.

5. Validating implementations of backpropagation

A crucial element in achieving replicability in neural computing experiments is to ensure that different implementations of the basic algorithms are indeed equivalent, that is, they are correct implementations of the specified algorithms. Although we do not present a proof that the software employed in our practical experiments is a correct implementation of our NN model and learning algorithm, we can show that a particular piece of software is in fact an "acceptable" implementation of a designated algorithm. This statistics-based empirical validation is particularly applicable to NN experiments where random initialization and the iterative numerical nature of the computations being performed make exact replicability practically impossible from one site to another (because of differences in hardware and software). In addition, the fault tolerant nature of NNs themselves make it difficult to spot "obvious" errors.

Essentially, the idea is to compare the outputs of two systems; one being the software we are attempting to validate, the other being either a machine executable version of our specification or an accepted standard implementation. We execute both programs in parallel, on identical test sets and compare the results. In theory, the results of the computation should be identical. In practice however, there may be many discrepancies and in this case we may employ our statistical framework to quantify these discrepancies in order to make a reasoned judgement as to whether our software is a close enough approximation to our standard. In this section we illustrate this idea with a comparison between two versions of backpropagation developed independently.

5.1 Experimental structure

We use two different versions of online backpropagation; bpsig, which was used to generate the results of section 4, and a new version, bpmon. The versions are assumed to be functionally identical but were developed independently. The aim is to test and quantify the validity of this assumption,

Table 7: Training results.

Incorrect	Error	Epochs	Version	Group
0	2.048734	5329	T3-A10-W1	
0	1.418497	10748	T3-A10-W2	
0	1.884790	5033	T3-A10-W3	bpmom
0	2.105070	6960	T3-A10-W4	
0	1.849360	6630	T3-A10-W5	
0	2.046734	5331	T3-A10-W1	
0	1.414893	10774	T3-A10-W2	
0	1.885629	5048	T3-A10-W3	bpsig
0	2.106218	6972	T3-A10-W4	
0	1.849377	6646	T3-A10-W5	

and to demonstrate that even when our NNs and learning algorithms are simple and well documented, complete replicability cannot be guaranteed.

Two groups of five networks each, T3-A10-Wi (where $i = 1, \ldots, 5$) were trained and tested using all the initial conditions specified in Table 1. One group was trained with the bpsig implementation and the other with the bpmom.

5.2 Training

Before training, our 10 networks were loaded with identical weight sets and tested. The networks produced identical results over our test set of 161051 patterns and thus we shall regard them as identical. The results after training (shown in Table 7) demonstrate that our two learning algorithm implementations differ very slightly. An interesting point to note here is the slight difference between the bpsig results presented here and the bpsig results presented in the previous sections (see Table 2 for the W group). Although the experimental setup was identical, in the previous section the weights were generated "in memory" using 64 bits of precision, while in this experiment the weights were loaded from an ASCII file using six decimal places of precision. The difference, while small, is enough to create slight differences in the results, demonstrating the extreme sensitivity of backpropagation to initial conditions.

5.3 Testing

The results of testing our networks are shown in Table 8. They are, as one would expect, very similar. This observation is supported when one notes that the correlation between the two groups is $\rho = 0.9996$. The correlation between the bpsig results reported here and the bpsig results of the previous section is $\rho = 0.9999$.

Table 8: bpmom program failure probabilities.

Group	Version	Failures	Success	Prob(fail)	Prob(success)
bpsig	T3-A10-W1	5390	155661	0.0335	0.9665
	T3-A10-W2	6159	154892	0.0382	0.9618
	T3-A10-W3	5694	155357	0.0354	0.9646
	T3-A10-W4	5515	155536	0.0342	0.9658
	T3-A10-W5	6064	154987	0.0377	0.9623
bpmom	T3-A10-W1	5391	155660	0.0335	0.9665
	T3-A10-W2	6154	154897	0.0382	0.9618
	T3-A10-W3	5698	155353	0.0354	0.9646
	T3-A10-W4	5516	155535	0.0343	0.9657
	T3-A10-W5	6065	154986	0.0377	0.9623

These high correlations indicate that within each group, as a whole, precisely the same numbers of nets failed on precisely the same test patterns. It is just possible that the two groups of very similar looking nets (as presented in Table 8) are in fact quite dissimilar when looking at which individual nets failed on which individual test patterns. Recall the earlier example of the two nets (batch and online trained) that agreed overall to 0.003% and yet exhibited 3% difference in the specific test patterns that failed.

In order to examine the situation in this case of apparently very high replicability between two sets of experiments, we again choose net T3-A10-W4 and examine it more closely. According to the summary data in Table 8 the net T3-A10-W4 when trained with the bpsig backpropagation algorithm failed on 5515 test patterns (precisely the same number as earlier, see the T3-A10-W4 entry of the W group in Table 3). When trained with the bpmom implementation of backpropagation 5516 test failures were recorded.

Closer inspection confirmed that the test failures are in fact identical, save for the one extra failure by the bpmom version. This confirms that the two groups of trained nets are indeed as similar as the correlation coefficient suggests. One set of five experiments has been replicated (to a high degree of accuracy), experiment for experiment, in the other set. From the table of coincident failures (Table 9), this high correlation between the two groups of networks can be clearly seen. It can also be seen that it is not perfect. To produce a ρ value of 1, all failures would have to be on the diagonal from top-left to bottom-right. All but 72 (i.e., 0.0004%) of the test failures are on this diagonal, and each of the 72 that are not on it are in adjacent positions. This means that, at most, only one individual network was involved in the lack of total replication observed. Each entry in this table is the number of test patterns that failed on precisely n_A and n_B networks, where both indices are integers ranging from 0 to 5, set A is the bpsig group and set B is the bpmom group.

Table 9: Coincidence of test pattern failures between two implementations of the backpropagation algorithm.

	bpmom					
	0/5	1/5	2/5	3/5	4/5	5/5
bpsig						
0/5	149169	9				
1/5	12	4304	10			
2/5		10	2412	8		
3/5			6	2038	8	
4/5				5	1838	2
5/5					2	1218

Table 10: Generalization diversity and average failure probabilities.

Group	*GD*	Avg. Gen.
bpsig	0.4474	96.42
bpmom	0.4472	96.42

From our results, it is clear that although our versions are not identical, there is a strong correlation indicating that they are similar enough to justify the view that they are both implementations of backpropagation. The ρ value of 0.9996 indicates that our assumption of equivalence is valid to a degree greater than 0.01%. It would, however, take a proper numerical analysis of these values to transform them into reliable quantifications of replicability, but the direct comparisons we have presented should provide a guide to replicability, that is, the higher the ρ value, the better the replication achieved.

6. Conclusions

This paper has accomplished the following.

1. Presented a formal specification of a simple NN experiment and attempted to establish the problems that make replicability of simple experiments virtually impossible.

2. Proposed and demonstrated a statistical framework that provides a general measure of replicability achieved in the context of a well specified set of initial conditions.

3. Demonstrated the relative effects on replicability of a number of conditions, some well known as important (but not how important), some

generally seen as unimportant (e.g., batch or online weight update), and some not generally recognized as conditions at all (e.g., precision of hardware).

4. Proposed and demonstrated how the statistical framework for replicability assessment can also be used as a way of validating supposed implementations of the backpropagation algorithm (or indeed of any other similarly "awkward" algorithm).

A systematic exploration of the possible interactions between the many parameters is desirable, but will require a number of substantial studies in their own right. The results of one study, such as [9] on the *GD* measure, indicate that the examples presented in this paper are representative of general trends on the LIC1 problem. Subsequent studies across problem types LIC1, LIC4 (a more complex well defined problem), and OCR (a data-defined letter recognition problem) as in [10], indicate that the results generalize to other similar problems as well as to more traditional data-defined real world problems. The results presented are then generally illustrative of the backpropagation style of neural computing, which is not surprising given that it relies on a dynamic "balancing" of many separate accumulations of small increments in real-number space. What is surprising, and revealed by the results, is the extent, pervasiveness, and complexity of the nonreplicability.

Further complicating the problem of replicability is the question of what particular features one is trying to replicate. If, for example, one considers generalization performance and commonality of test pattern failure, the former is often quite replicable while the latter is not, even though the former is a direct consequence of the latter. Recall, for example, the online and batch results with the net T3-A10-W4 (Table 3). The generalization performance was replicated to less than 0.001%, but the actual test set failures varied by 40% (Table 4). A replication in terms of one feature is far from a replication in terms of another.

One rather different approach to replicability would be for researchers to establish publically accessible data sets consisting of the training set, test set, and initial weight set (together with architecture and necessary mappings). Even this would not ensure complete replicability, but it would, however, go some way towards reducing the inevitable diversity of network solutions.

This paper has pointed out that the claimed accuracy of experimental results must be related to the experimental control exercised, both the degree of control and what is controlled. This is, of course, in addition to all the usual requirements of experimental validity. A proper numerical analysis is needed to firmly establish the exact relationship between statistical measures of replicability (e.g., ρ and *GD*) and the accuracy of reported results. But even without such an analysis the replicability measures can be used as a rough guide to the acceptable accuracy of reported results.

The (generally) consistent reproducibility of a coarse-grained measurement, such as generalization performance, contrasts sharply with the lack of replicability at the fine-grained level of specific test pattern failures of individual networks. The situation is reminiscent of many natural complex systems (e.g., the real gases of physics, or many features of biological systems) where macro-predictability is secure despite micro-unpredictability, for example, the pressure exerted by a volume of gas at a given temperature can be accurately predicted, despite the fact that the speed and trajectories of the individual molecules (which collectively produce this pressure) cannot. Classic science tackles this problem by attempting to replicate only the macro-phenomena, although there is also some considerable discussion about how, whether, and in what way the more micro-phenomena in, say, biology can be treated as reproducibly "scientific" phenomena.

Neural computing, which is a man-made artifact rather than a "given" that the scientist must make the best of, begins to look all too organic in practice, even though it is a well-defined abstraction. If this "macro-determinacy but micro-indeterminacy" view is accepted, then one can only reasonably expect replicable experiments in terms of the macro-properties of neural computing.

This paper demonstrates that lack of replicability is a more insidious problem than most neural-computing practitioners realize because of the following.

1. Not only do "mere implementation" differences lead to different results, but even slight changes in the usage of exactly the same implementation can also yield differences.

2. Common measures of replication, such as percent correct, can suggest accurate replication, whereas our more discriminating technique reveals that this is far from true.

In conclusion, we view the controls needed to guarantee replicability of a single-net neural computing experiment to be excessive (and in practice usually impossible). In neural computing, a replicable experiment seems to imply group averages if not statistics in one form or another. In fact, the long-running saga in classic science of whether cold fusion is a reality or an illusion has caused many to question the apparent simplicity and significance of replicability as the foundation stone of scientific method.

References

[1] E. Barnard and J. E. Holm, "A Comparative Study of Optimization Techniques for Backpropagation," *Neurocomputing*, **6** (1994) 19–30.

[2] J. Dugundji, *Topology* (Allyn and Bacon, London, 1966).

[3] S. E. Fahlman, "Faster-learning Variations on Backpropagation: An Empirical Study," in *Proceedings of the 1988 Connectionist Models Summer School*, edited by D. Touretzky, G. E. Hinton, and T. J. Sejnowski (Morgan Kaufmann, Los Altos, CA, 1988).

[4] P. R. Halmos, *Measure Theory* (Van Nostrand, New York, 1950).

[5] A. K. Hornik, A. M. Stinchcombe, and H. White, “Multilayer Feedforward Networks are Universal Approximators,” *Neural Networks*, **2** (1989) 359–366.

[6] J. C. Knight and N. G. Leveson, “An Experimental Evaluation of the Assumption of Independence in Multiversion Programming,” *IEEE Transactions on Software Engineering*, **12** (1986) 96–109.

[7] J. F. Kolen and J. B. Pollack, “Backpropagation is Sensitive to Initial Conditions,” *Complex Systems*, **4** (1990) 269–280.

[8] B. Littlewood and D. R. Miller, “Conceptual Modeling of Coincident Failures in Multiversion Software,” *IEEE Transactions on Software Engineering*, **15** (1989) 1596–1614.

[9] D. Partridge, “Network Generalization Differences Quantified,” *Neural Networks*, **9** (1996) 263–271.

[10] D. Partridge and W. B. Yates, “Engineering Multiversion Neural-net Systems,” *Neural Computation*, **8** (1996) 869–893.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representation by Error Propagation,” in *Parallel Distributed Processing. Explorations in the Microstructure of Cognition. Volume 1: Foundations*, edited by D. E. Rumelhart and J. L. McClelland (MIT Press, Cambridge, MA, 1989).