Investigating the Fault Tolerance of Neural Networks

Article *in* Neural Computation · August 2005

DOI: 10.1162/0899766053723096 · Source: PubMed

Rory Mulvaney
University of Maryland, Baltimore County
7 PUBLICATIONS **118** CITATIONS
SEE PROFILE

Dhananjay Phatak
University of Maryland, Baltimore County
77 PUBLICATIONS **2,206** CITATIONS
SEE PROFILE

Investigating the Fault Tolerance of Neural Networks

Elko B. Tchernev
etchernev@acm.org
Rory G. Mulvaney
rory1@umbc.edu
Dhananjay S. Phatak
phatak@umbc.edu
Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD 21250, U.S.A.

Particular levels of partial fault tolerance (PFT) in feedforward artificial neural networks of a given size can be obtained by redundancy (replicating a smaller normally trained network), by design (training specifically to increase PFT), and by a combination of the two (replicating a smaller PFT-trained network). This letter investigates the method of achieving the highest PFT per network size (total number of units and connections) for classification problems. It concludes that for nontoy problems, there exists a normally trained network of optimal size that produces the smallest fully fault-tolerant network when replicated. In addition, it shows that for particular network sizes, the best level of PFT is achieved by training a network of that size for fault tolerance. The results and discussion demonstrate how the outcome depends on the levels of saturation of the network nodes when classifying data points. With simple training tasks, where the complexity of the problem and the size of the network are well within the ability of the training method, the hidden-layer nodes operate close to their saturation points, and classification is clean. Under such circumstances, replicating the smallest normally trained correct network yields the highest PFT for any given network size. For hard training tasks (difficult classification problems or network sizes close to the minimum), normal training obtains networks that do not operate close to their saturation points, and outputs are not as close to their targets. In this case, training a larger network for fault tolerance yields better PFT than replicating a smaller, normally trained network. However, since fault-tolerant training on its own produces networks that operate closer to their linear areas than normal training, replicating normally trained networks ultimately leads to better PFT than replicating fault-tolerant networks of the same initial size.

1 Introduction

Fault tolerance of feedforward artificial neural networks (ANNs) has been investigated by many researchers (a sampling can be found in Sequin & Clay, 1990, Petsche & Dickinson, 1990; Segee & Carter, 1991, 1994; Neti, Schneider, & Young, 1992; Phatak & Koren, 1992, 1995; Protzel, Palumbo, & Arras, 1993, Murray & Edwards, 1994; Bishop, 1995; Phatak, 1999). Our prior work (Phatak, 1994; Phatak and Koren, 1992, 1995) related fault tolerance to the amount of redundancy required to achieve it. It was demonstrated that for the standard or typical artificial neural network (ANN) architectures wherein the units (or neurons) perform a weighted sum of inputs from other units, followed by a monotonic nonlinear squashing to generate their outputs, less than TMR (triple modular redundancy) is not sufficient to achieve complete fault tolerance. A simple method of replicating a seed network was shown to be one possible way of enhancing fault tolerance of ANNs. This method requires a large amount of redundancy even if the size of the seed network (which gets replicated) is kept minimal. A better alternative might be to use enhanced training algorithms that target fault tolerance. This was analyzed in Phatak (1994) and revealed that using permuted samples or cross validation during training does not lead to a discernible partial fault tolerance (PFT) enhancement. On the other hand, providing initial redundancy and modifying the training algorithm to utilize the extra parameters does improve the PFT to some extent. However, a brute force method of replications proposed in Phatak and Koren (1992, 1995) seems to achieve a higher PFT for the same level of redundancy (as compared with the fault-tolerant gradient descent training), which was a counter-intuitive result. A later work (Tchernev, Mulvaney, & Phatak, 2004) analyzed the exact replication factors needed to obtain perfect fault tolerance for the $n - k - n$ encoder/decoder problem. It was shown that the traditional $n - \log_2 n - n$ architecture, with larger initial size but operating closer to saturation, achieves fault tolerance at a significantly smaller final size than the minimal $n - 2 - n$ architecture. In this letter, we experimentally investigate the relationships between network size, initial PFT, and replicated PFT, and look for optimal-sized seed networks that yield the highest PFT.

2 Experiments

The problems being investigated are the two-class six-area problem, the two spirals problem, and the vowel problem. The two-class six-area problem (see Figure 1) is a specially constructed classification problem with a known minimal size of its solving network: a single hidden layer of three nodes. The two spirals problem (training and testing data sets together) is shown in Figure 4, and the vowel problem is taken from the UCI Machine Learning repository (Blake & Merz, 1998). These are problems ranging from very easy (the two-class six-area problem) through moderate (the two spirals problem) to hard (the vowel problem).

3 Method

For all problems, a strictly layered structure is assumed (all units in a layer connect to only the units in the next adjacent layer, and no layer-skipping connections to other units are present). Networks with various numbers of hidden units are trained with normal gradient descent as well as a modified version that specifically targets enhanced fault tolerance. Each network is then replicated from 1 to 20 times, and the resulting PFT is evaluated for each of the replication sizes under the single fault assumption (i.e., only a single link or unit can fail).

Training for fault tolerance is tantamount to solving a constrained optimization problem, as discussed in Phatak (1994) and Phatak and Tchernev (2002). Instead of trying to solve it outright, as in Neti et al. (1992), the penalty function method approach is used. It is described in Luenberger (2003), and it incorporates the constraints into the objective function. The new objective function becomes $E = \sum_p \sum_o (t_{op} - y_{op})^2 + \alpha \sum_h \sum_f \sum_p \sum_o (t_{op} - y_{opfh})^2$ where t_{op}, y_{op} = target and actual outputs of output unit o on pattern p, y_{opfh} = actual output of output unit o on pattern p upon fault f of hidden unit h, and α = weight of the extra terms relative to normal terms.

Note that now there are two sets of terms in this objective function E. The first summation includes the normal error terms. The second summation accrues the errors that arise due to faults and is the penalty function. When E is minimized, the error between target outputs and the network outputs on faults is also minimized. Thus, the addition of these terms forces the search algorithm to look for a solution with better fault tolerance. These terms in the objective function that encourage fault tolerance are henceforth referred to as *extra terms*. The (nonnegative) parameter α is the weight of the penalty function (or the penalty) relative to the normal error terms, and typically, $\alpha < 1$.

As elaborated in Phatak and Koren (1992, 1995), replication involves adding as many copies of the hidden nodes and all their links as the replication factor is and adjusting the output nodes' biases, while preserving the input and output node counts.

For the links, three types of faults are considered: stuck at +MAX, at −MAX, and at 0, where MAX is the largest weight magnitude in the trained network. For each hidden unit, two types of faults are considered: its output stuck at +Output_Max and at −Output_Max. For a unit, considering a stuck-at-0 fault on the output is not necessary: if the other two faults can be tolerated, then a stuck-at-0 fault on the output of that unit can also be tolerated.

By definition, PFT is the fraction of outputs that remain correct when all single faults are considered. To generate this value, every weight and unit in the tested network is set stuck at each fault type. For each fault, all input-output patterns are applied, and the number of outputs that remain correct is accumulated. This total is divided by the total number of cases, which equals [#outputs $\times$ #io_patterns $\times$ (3 $\times$ #weights + 2 $\times$ #hidden_units)], where #X denotes number of X. The fraction (PFT) is calculated for different numbers of replications and investigated as a function of the total number of hidden units in the resulting network. Note that the magnitude of observed PFT change (and difference between pairs of PFTs) can be quite small—for example, the vowel problem, with 10 inputs, 11 outputs, and 528 training patterns, one nonreplicated network with a single 32 hidden-units-layer (672 weights) can have a PFT step of 1 in 11,894,784 cases, which is about $8.4*10^{-8}$. This can decrease with bigger seed networks, replication, and when averaging over more than one network.

For each training method and size of the seed network, sample networks are generated and trained, and for each, the PFT is evaluated for an increasing number of replications. Only the results from the networks that have 100% classification accuracy on the training data set are taken, averaged, and discussed in this article. (The reason is that by definition, PFT is a correctness measure, and only initially correct networks reach 100% PFT on replication.) The focus of the investigation is on the relationship between PFT and final size of the network in nodes, since network size is both a measure and a constraint in terms of practical use. Replication, on the other hand, is computationally free compared to the cost of training a network.

4 Results

4.1 The Two-Class Six-Area Problem. The chosen network architecture has two inputs, one output, and one hidden layer with a variable number of nodes. The labels on the graphs in Figure 1 correspond to the number of nodes in the hidden layer.

Since this is an easy problem for backpropagation training, the smallest network size to be successfully trained with both normal and fault-tolerant training is the theoretically minimal one of three hidden nodes. The decision lines of these theoretical solution nodes can be seen on Figure 1 together with the data points.

4.1.1 Initial PFT. Table 1 shows the initial (prereplication) averaged PFTs and 95% confidence intervals of the networks that were successfully trained (out of totally 50 runs per size). For hidden node counts of 3, 4, 6, 8, 16, and 24, the corresponding number of successfully trained networks was 35, 42, 47, 49, 49, and 46 for normal backpropagation, and 28, 34, 41, 48, 46, and 46 for fault-tolerant training. The table shows the fraction of correct outputs across all faults, as they change with network size. The two data sets correspond

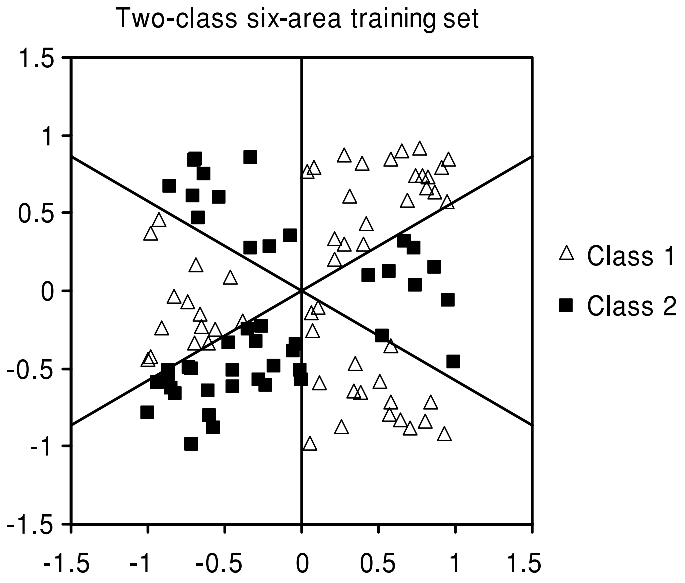


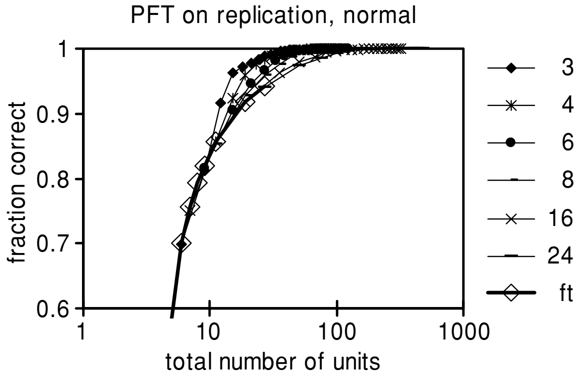
Figure 1: Training data set for the two-class six-area problem.

Table 1: Average PFT and Confidence Intervals of the Correct Runs of the Two-Class Six Area Problem, Normal and Fault-Tolerant Training, by Hidden Node Count.

Hidden	3	4	6	8	16	24
bp PFT	0.698	0.751	0.816	0.853	0.918	0.940
95%	0.000	0.001	0.003	0.003	0.002	0.001
ft PFT	0.701	0.756	0.820	0.856	0.918	0.942
95%	0.000	0.001	0.002	0.002	0.002	0.001

to the normal backpropagation training (bp) and its fault-tolerant variant (ft).

The smallest network to solve the problem, the minimal three-hidden-node network, has a total of six nodes and corresponds to the first (left-most) data column of the table. As expected, PFT increases with network size, the networks become increasingly redundant, and fault-tolerant training produces better PFT than normal backpropagation. The curve of diminishing returns is clearly seen as bigger networks demonstrate less PFT gain. The only network size where the normal and the fault-tolerant training produce statistically identical PFT is at 16 hidden nodes; the two-tailed t-test distance for that pair is 69%.



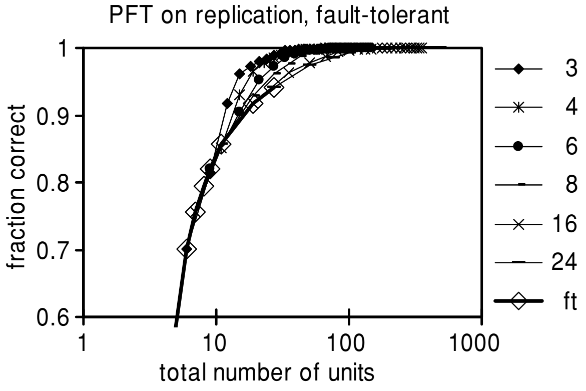
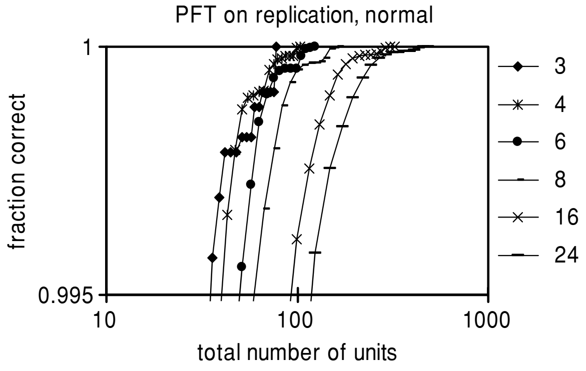


Figure 2: PFT of normally trained and fault-tolerant trained networks with different numbers of hidden-layer units, when using replication, plotted by resulting network size, two-class six-area problem.

4.1.2 Replication PFT. Plots of the entire PFT range, up to 100% PFT, of the successfully trained networks, when using replication, are shown in Figure 2 for plain backpropagation and for fault-tolerant backpropagation. The legend keys correspond to the number of hidden units in the seed networks that are replicated. The 95% confidence intervals would be too small to see on the plots at the data scale so are not included.

The initial (nonreplicated) PFT for fault-tolerant backpropagation is included on both graphs, labeled ft, and demonstrates that replicating a smaller network dominates fault-tolerant training for the same network size, regardless of the replicated net's training method.

The graphs where PFT approaches 100% on replication (perfect fault tolerance) are shown in Figure 3. The resolution on the vertical axis is increased significantly to allow a clear view. The averaged point of reaching perfect



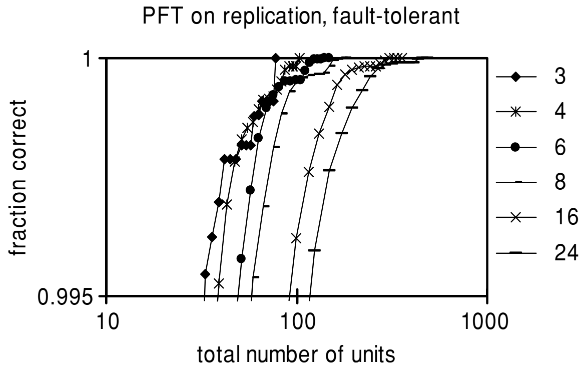


Figure 3: Normal and fault-tolerant trained networks with different number of hidden-layer units achieve perfect PFT when using replication, plotted by resulting network size, two-class six-area problem.

PFT is the result of just one network achieving 100% PFT after all others; even so, the variances are small, and the 95% confidence error bars, while visible at this scale, would only hinder the identification of the data points. Overall, for a given final size, the smallest working network achieves the best PFT on replication.

These graphs show that for certain network sizes, the best PFT is not exhibited by the smallest replicated network; nevertheless, it is the smallest networks that eventually reach 100% PFT. The difference between the PFT of fault-tolerant training and that of normal backpropagation is small and decreases with the number of replications. Eventually, the smallest correct trained networks for both types of training achieve 100% PFT at the same final size.

4.2 The Two Spirals Problem. The training and testing data sets are shown in Figure 4. The chosen network architecture has two inputs, one

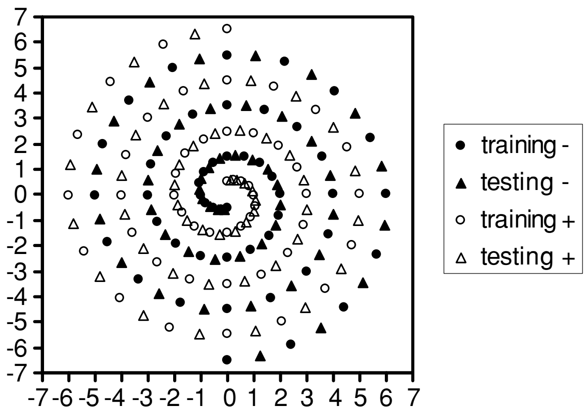


Figure 4: Data sets of the two spirals problem.

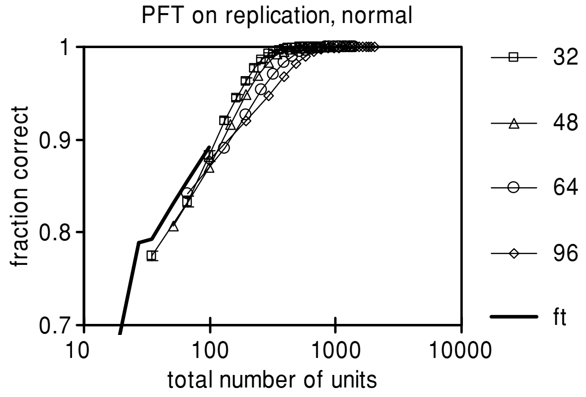
Table 2: Average PFT and Confidence Intervals of the Correct runs of the Two Spirals Problem, Normal and Fault-Tolerant Training, by Hidden Node Count.

Hidden	24	32	48	64	96
bp PFT	—	0.775	0.807	0.841	0.877
95%	—	0.005	0.004	0.005	0.004
ft PFT	0.789	0.793	0.832	0.855	0.892
95%	0.000	0.016	0.005	0.006	0.007

output, and one hidden layer with a variable number of nodes. The labels on the graphs correspond to the number of nodes in the hidden layer. The minimal size of a network of such architecture to solve the problem is not known; the smallest size to actually solve the problem had 32 hidden nodes for normal and 24 hidden nodes for fault-tolerant training.

4.2.1 Initial PFT. Table 2 shows the initial (prereplication) averaged PFTs and 95% confidence intervals of the networks that were successfully trained (out of totally 50 runs per size). For hidden node counts of 24, 32, 48, 64, and 96, the corresponding number of successfully trained networks was 0, 12, 12, 13, and 8 for normal backpropagation and 1, 3, 12, 5, and 3 for fault-tolerant training.

The table shows the fraction of correct outputs across all faults as they change with network size. The two data sets correspond to the normal backpropagation training (bp) and its fault-tolerant variant (ft). The first (leftmost) data column in the table corresponds to the smallest network to solve the problem, which has 24 hidden and 27 total nodes, for fault-tolerant training. PFT increases with network size, and fault-tolerant training achieves a



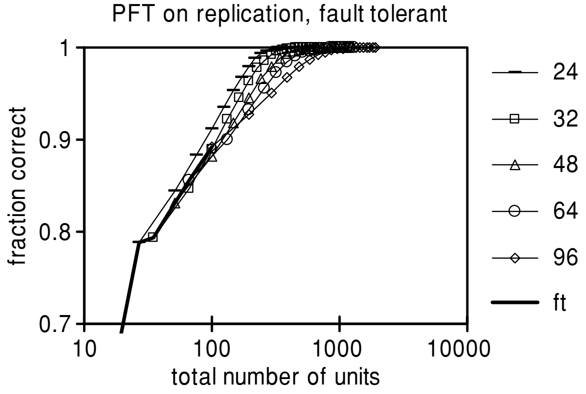
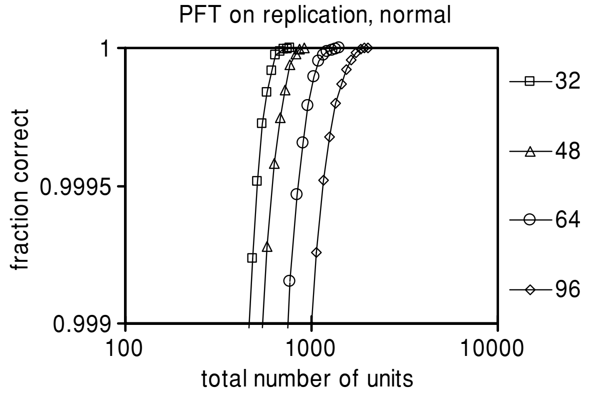


Figure 5: PFT of normally trained and fault-tolerant-trained networks with different number of hidden-layer units, when using replication, plotted by resulting network size, two spirals problem.

higher PFT than normal backpropagation; the difference between the training methods is statistically significant.

4.2.2 Replication PFT. Plots of the entire PFT range, up to 100% PFT of the successfully trained networks, when using replication, are shown in Figure 5 for plain backpropagation and for fault-tolerant training. The legend keys correspond to the number of hidden units in the seed networks that are replicated. The 95% confidence error bars, while visible at this scale, would only hinder the identification of the data points and cause visual clutter so are not included.

The PFT of the nonreplicated fault-tolerant trained networks is included on both graphs, labeled ft, and demonstrates that replicating a smaller, normally trained network is dominated by fault-tolerant training for the same



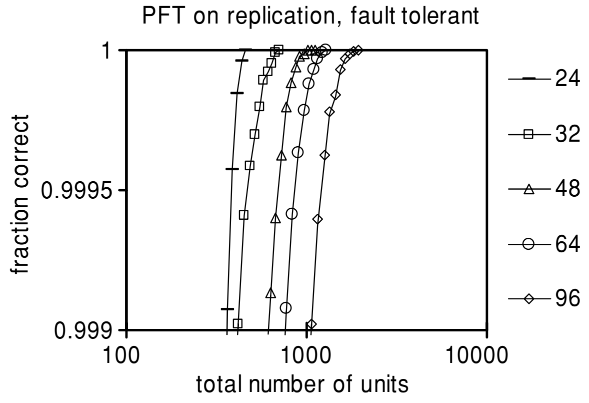


Figure 6: Normally trained and fault-tolerant-trained networks with different numbers of hidden-layer units achieve perfect PFT when using replication, plotted by resulting network size, two spirals problem.

network size. Replicating a smaller fault-tolerant-trained network is similarly dominated by the bigger fault-tolerant-trained ones, except for the case of 24 hidden nodes. Regardless, after several replications, the trend reverses, and for a given final size, the smaller seed network achieves the better PFT. This is illustrated by plotting the areas close to perfect PFT on Figure 6, where it can be seen how the replicated smaller networks achieve 100% PFT at smaller final sizes.

The difference in PFT between networks obtained by replicating corresponding sizes of fault-tolerant trained and normal seed networks, which is positive initially, reduces to insignificance with replication but passes through a statistically significant area where replicated large fault-tolerant-trained networks have worse PFT than replicated normally trained networks of the same size.

Table 3: Average PFT and Confidence Intervals of the Correct Runs of the Vowel Problem, Coded Outputs, Normal and Fault-Tolerant Training, by Hidden Node Count.

Hidden	32	48	64	96
bp PFT	0.966	0.983	0.992	0.997
95%	0.001	0.002	0.001	0.001
ft PFT	0.967	0.990	0.994	—
95%	0.017	0.003	0.003	—

4.3 The Vowel Problem. The vowel problem has 10 inputs and must classify its input patterns into 11 classes. The outputs of a neural network that solves it can encode the solutions either in exclusive form (with 11 output units, only 1 active at a time) or in coded form (with 4 output units, the class coded in binary). Runs were performed with one and with two hidden layers, but the complexity of the network with one hidden layer was such that no correct solutions were found for the fault-tolerant training case. The results presented here are for the two-layer exclusive case (10 input units, 2 hidden layers of variable equal number of units, 11 output units) and for the 2-layer coded case (10 input units, 2 hidden layers of variable equal number of units, 4 output units). The labels on the graphs correspond to the total number of nodes in the hidden layers; each layer contains half as many nodes. The minimal size of networks of such architecture to solve the problem is not known; the smallest sizes to actually solve the problem had 32 hidden nodes (two layers of 16 nodes) for each of the four training method–output combinations.

The vowel problem was testing the more computationally intensive fault-tolerant training with extra terms. Not only was no solution found for any size network with one hidden layer, but the maximum size that this method was able to successfully train was 64 hidden nodes (two layers of 32 nodes each). Normal training (which operates in a smaller state space) fared a little better and was able to obtain solutions in the hardest case (one hidden-layer network), and to train successfully a 96-hidden-units network (two layers of 48 nodes each) in the easiest case (two hidden layers with coded outputs).

4.3.1 Initial PFT. The initial (prereplication) PFTs of the networks that were successfully trained (out of totally 20 runs per size) are shown in Tables 3 and 4 for the coded and the separate outputs architectures, respectively. For the coded case, for hidden node counts of 32, 48, 64, and 96, the corresponding number of successfully trained networks was 2, 5, 10, and 4 for normal backpropagation, and 2, 7, 5, and 0 for fault-tolerant training.

Table 4: Average PFT and Confidence Intervals of the Correct Runs of the Vowel Problem, Exclusive Outputs, Normal and Fault-Tolerant Training, by Hidden Node Count.

Hidden	32	48	64
bp PFT	0.981	0.991	0.996
95%	0.002	0.001	0.000
ft PFT	0.985	0.995	0.997
95%	0.000	0.001	0.000

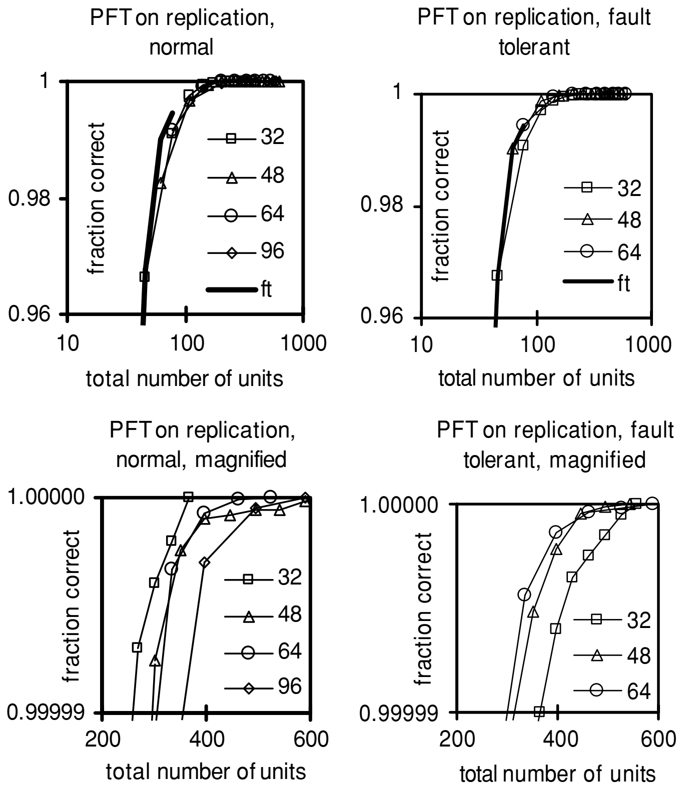


Figure 7: PFT of normally trained and fault-tolerant-trained networks with different numbers of hidden-layer units achieve perfect PFT when using replication, plotted by resulting network size, vowel problem, coded outputs.

For the exclusive case, for hidden node counts of 32, 48, and 64, the corresponding number of successfully trained networks was 4, 3, and 1 for normal backpropagation and 1, 2, and 1 for fault-tolerant training.

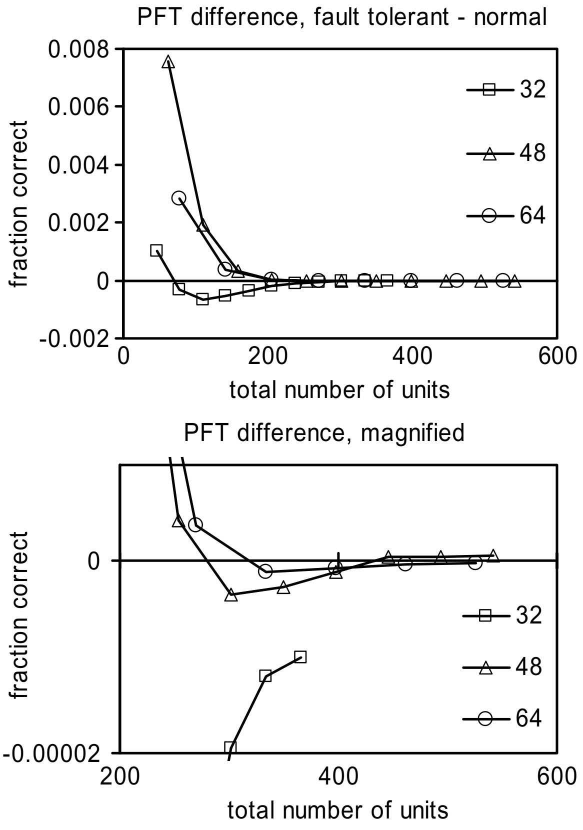


Figure 8: Difference between the PFT of fault-tolerant and normally trained networks with different number of hidden-layer units when using replication, plotted by resulting network size, vowel problem, coded outputs.

The tables show the averaged fraction of correct outputs across all faults, as a function of the hidden nodes count, with 95% confidence intervals. The smallest networks to solve the problem have 32 hidden nodes in two layers of 16 nodes each and correspond to the first (left-most) data columns of the tables. PFT increases with network size, and fault-tolerant training achieves better PFT than normal backpropagation.

4.3.2 Replication PFT. Plots of the entire PFT range of the successfully trained coded output networks, when using replication, are shown in Figure 7 for both normal and fault-tolerant training. The legend keys correspond to the number of hidden units in the seed networks that are replicated. The 95% confidence error bars are not included in these plots, as they would be invisible at the data scale.

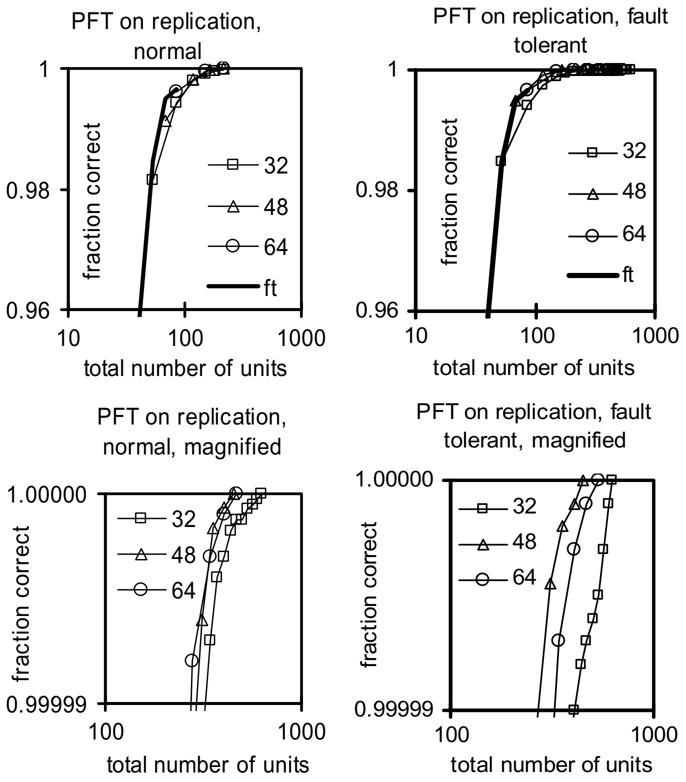


Figure 9: PFT of normally trained and fault-tolerant-trained networks with different numbers of hidden-layer units achieve perfect PFT when using replication. Plotted by resulting network size, vowel problem, exclusive outputs.

The initial PFT for fault-tolerant training is included on two of the graphs, labeled ft, and demonstrates that replicating a smaller, normally trained network is dominated by fault-tolerant training for the same network size.

The data where PFT approaches 100% are shown in the magnified graphs. It can be seen that after enough replications, the initial levels of PFT have shifted. The smallest network to reach 100% PFT for normal training is the replicated 32 hidden units network; the smallest one to solve the problem. For fault-tolerant training, however, the smallest network to reach perfect fault tolerance is not the replicated result of the smallest initial solution.

Actually, the initial relationship between the fault tolerances is preserved in the replications almost to the 100% mark. Until the last few replications, it is the biggest initial network that has the best fault tolerance on replication and is overtaken on the finish line by the two smaller networks.

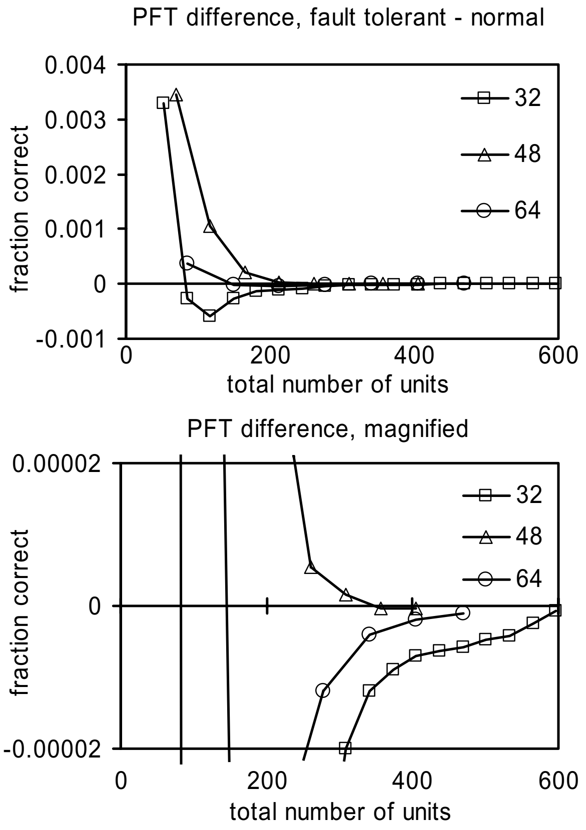


Figure 10: Difference between the PFT of fault-tolerant and normally trained networks with different numbers of hidden-layer units when using replication. Plotted by resulting network size, vowel problem, exclusive outputs.

The difference between the PFT of the fault-tolerant and of the normally trained networks is shown in Figure 8 as an entire data range graph and as a strongly magnified region close to zero. It can be seen that initial better PFT notwithstanding, the replicated normally trained networks exhibit better fault tolerance than the extra-terms-trained one after several replications, and that relationship is retained. In two of the three cases, the normally trained network achieves 100% PFT earlier than the fault-tolerant one.

The PFTs of the replicated successfully trained exclusive output networks are shown in Figure 9 for both normal and fault-tolerant training.

The initial PFT for fault-tolerant training is included on both graphs, labeled ft, and demonstrates that replicating a smaller, normally trained network is dominated by fault-tolerant training for the same network size.

The data where PFT approaches 100% are shown on the magnified graphs. It can be seen that after enough replications, the initial levels of PFT have shifted. The smallest network to reach 100% PFT for normal training is the replicated 48 hidden units one—neither the smallest nor the largest to solve the problem. For fault-tolerant training, it is again the middle-sized network that is the first to reach perfect fault tolerance. The initial relationship between the fault tolerances is preserved in the replications almost to the 100% mark only for the normally trained case. Until the last few replications, it is the biggest initial network that has the best fault tolerance on replication and is overtaken on the finish line by the middle-sized network.

This is not the case for fault-tolerant training, where the superiority of the middle-sized network is established after the first few replications. In none of the cases does the smallest correct network challenge the PFT of the larger ones; it has the worst PFT across all final sizes.

The difference between the PFT of the extra-terms and the normally trained networks is shown in Figure 10 as an entire data range graph and as a strongly magnified region close to zero.

It can be seen that initial better PFT notwithstanding, the replicated normal trained networks exhibit better fault tolerance than the extra-terms-trained networks after several replications, and that relationship is retained. However, in only one of the three cases does the normally trained network achieve 100% PFT earlier than the fault-tolerant one. In the other two cases, the difference diminishes so that they achieve 100% PFT at the same final size.

5 Summary

The data indicate the following trends. As the size of the seed network (which is later replicated) is increased, the initial PFT (without any replications) improves. The improvement exhibits diminishing returns and tapers off as the size of the initial network is further increased, regardless of whether the problem being solved is easy or hard.

Replicating correct seed networks (with 100% classification rates on the training dataset) leads to the formation of networks with 100% PFT with respect to the training data set. Whether the smallest resultant network of this kind is obtained from the smallest seed network or not depends on the difficulty of the problem and the training method. The easier the problem and the training method, the more likely it is that the smallest correct network will achieve 100% final PFT at the smallest final size.

Fault-tolerant training alone improves the PFT of any given seed network. For easy problems, replication PFT dominates fault-tolerant training for the same final size, while the converse is true for hard problems.

On replication, fault-tolerant seed networks produce networks with better PFT than networks obtained by replicating normal seed networks, but only for the first several replications. Eventually, the normal networks take over and in general reach 100% PFT at a smaller final size.

6 Discussion and Conclusion

It is important to realize that the fault tolerance of a neural network depends on the area of operation of its units when classifying data points. Consider, for simplicity, that the last hidden layer before some output performs a mapping of the input space into a hypercube of dimensionality k, where k is the number of hidden-layer units that connect to the output. For classification problems, the output unit has the function of a hyperplane of dimensionality $k-1$, which splits the volume of the hypercube in two classification subspaces. In order for the network to be fault tolerant, each fault-induced change in the position of a data point in that hypercube must leave it on the same side of the hyperplane as it originally was. Replicating the hidden nodes of a network has the effect of reducing the influence of any one node or connection, so that a fault will reduce the amount of displacement in the hyperspace that each data point undergoes. Clearly, the closer to the vertices of the hypercube the points lie (the more saturated the outputs of the hidden units that produce it), the fewer replications are needed to completely eliminate the effect of a fault and achieve 100% fault tolerance. In the extreme case of threshold units and unit weights, triple redundancy would be sufficient to guarantee that a point does not leave its hypercube vertex on both weight and node failures. This effect is the reason for the drastically fewer replications (and total nodes) needed to achieve fault tolerance for the $n - k - n$ problem, as derived in Tchernev et al. (2004).

It is reasonable to posit that backpropagation training on easy problems (like the two-class six-area problem) will tend to find solutions that classify cleanly and drive the hidden and output units close to their saturation levels. Achieving perfect fault tolerance would require approximately the same number of replications from any starting size, which would favor the smallest seed network to reach the smallest solution overall. On the other hand, backpropagation on harder problems might find minimal solutions that barely classify and drive most of their units in their linear area, requiring more replications to become fault tolerant. This interaction of the two factors determines a point of diminishing returns, where the smaller initial size cannot compensate for the increased number of replications that are required. The conclusion with respect to normal backpropagation, then, would be that there exists an optimal seed network size that, when replicated, achieves the smallest perfectly tolerant net. For easy problems, this coincides with the minimal solution network. Training is able to find solutions that operate in a sufficiently saturated state; the large number of required replications is offset by the small size, so that replicating this network is the almost optimal way to go. Larger networks, even if operating in a more saturated state initially and requiring fewer replications, lose out because of their final size. For harder problems, the smallest networks to solve them have nonsaturated networks close to the classification thresholdes; they require a larger number of replications, so that the smallest fault-tolerant final size is produced by some nonminimal seed network.

Fault-tolerant training with extra terms, as implemented, does not lead to the same solutions as normal backpropagation. The extra terms constrain the saturation of the nodes in the network by leading to solutions where each connection has a smaller influence overall. This explains why, even though the initial fault tolerance of these networks is better than that of the normally trained ones, they eventually lose out on replication and do not achieve better fault tolerance for the same final size. In simple problems, where the solving networks are small or minimal, there is not enough redundancy to allow a search for fault tolerance, and that search, performed anyway, prevents the network from reaching a sufficiently saturated operating state. Therefore, not only is a fault-tolerant-trained network worse in PFT than a replicated normal one obtained from a smaller seed, but replication does not increase its fault tolerance as fast as replicating a more saturated normally trained one would. For hard problems, where the solving networks are larger than the (unknown) minimal ones, sufficient extra capacity exists to allow an efficient search for fault tolerance, so that the resulting network is more fault tolerant than a normal one replicated to the same size. However, because of its inherently nonsaturated operation, replicating the fault-tolerant-trained network cannot produce the same PFT gain, and the eventually smallest fault-tolerant network is the result of replicating a normally trained one.

In conclusion, it can be said that for sufficiently hard problems, if 100% PFT is not required, the smallest network to achieve any particular level of PFT would be a fault-tolerance-trained one. This has serious limitations, however, since the method is extremely computationally intensive to be of realistic use for hard real-world classification problems. It is clear that replicating a seed network does not produce the smallest overall network for a given PFT level (unless starting from extremely saturated or threshold units), but there is no realistic alternative. Whether any other method for network construction and/or search (for example, genetic algorithms or genetic programming) can achieve 100% PFT for a realistic problem without using replication is an open question that is subject to further investigation.

Acknowledgments

This work was supported by NSF grants ECS-9875705 and ECS-0196362.

References

Bishop, C. M. (1995). Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, *7*, 108–116.

Blake, C. L., & Merz, C. J. (1998). *UCI repository of machine learning databases*. Irvine: University of California.

Luenberger, D. G. (2003). *Linear and nonlinear programming*. Norwell, MA: Kluwer Academic Publishers.

Murray, A. F., & Edwards, P. J. (1994). Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training. *IEEE Transactions on Neural Networks*, *5*, 792–802.

Neti, C., Schneider, M. H., & Young, E. D. (1992). Maximally fault tolerant neural networks. *IEEE Transactions on Neural Networks*, *3*, 14–23.

Petsche, T., & Dickinson, B. W. (1990). Trellis codes, receptive fields, and fault tolerant, self-repairing neural networks. *IEEE Transactions on Neural Networks*, *1*, 154–166.

Phatak, D. S. (1994). *Fault tolerance of feed-forward artificial neural nets and synthesis of robust nets.* Doctoral dissertation, University of Massachusetts.

Phatak, D. S. (1999). Relationship between fault tolerance, generalization and the Vapnik-Chervonenkis (VC) dimension of feedforward ANNs. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 705–709). Washington, DC.

Phatak, D. S., & Koren, I. (1992). Fault tolerance of feedforward neural nets for classification tasks. In *Proceedings of the International Joint Conference on Neural Networks* (Vol. 2, pp. 386–391). Baltimore, MD.

Phatak, D. S., & Koren, I. (1995). Complete and partial fault tolerance of feedforward neural nets. *IEEE Transactions on Neural Networks*, *6*, 446–456.

Phatak, D. S., & Tchernev, E. (2002). Synthesis of fault tolerant neural networks. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1475–1480). Honolulu, HI.

Protzel, P. W., Palumbo, D. L., & Arras, M. K. (1993). Performance and fault-tolerance of neural networks for optimization. *IEEE Transactions on Neural Networks*, *4*, 600–614.

Segee, B. E., & Carter, M. J. (1991). Fault tolerance of pruned multilayer networks. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 447–452). Seattle, WA.

Segee, B. E., & Carter, M. J. (1994). Comparative fault tolerance of parallel distributed processing networks. *IEEE Transactions on Computers*, *43*, 1323–1329.

Sequin, C. H., & Clay, R. D. (1990). Fault tolerance in artificial neural networks. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 703-708). San Diego, CA.

Tchernev, E., Mulvaney, R., & Phatak, D. S. (2004). *Perfect fault tolerance of the n-k-n network* (Tech. Rep. No. TR-CS-04-14). Baltimore, MD: University of Maryland Baltimore County.

Received May 7, 2004; accepted January 5, 2005.

This article has been cited by:

1. Feng Su, Peijiang Yuan, Yangzhen Wang, Chen Zhang. 2016. The superior fault tolerance of artificial neural network training with a fault/noise injection-based genetic algorithm. *Protein & Cell* **7**:10, 735-748. [CrossRef]
2. Yuwei Cui, Subutar Ahmad, Jeff Hawkins. Continuous Online Sequence Learning with an Unsupervised Neural Network Model. *Neural Computation*, ahead of print1-31. [Abstract] [PDF] [PDF Plus]
3. Yi Xiao, Rui-Bin Feng, Chi-Sing Leung, John Sum. 2016. Objective Function and Learning Algorithm for the General Node Fault Situation. *IEEE Transactions on Neural Networks and Learning Systems* **27**:4, 863-874. [CrossRef]
4. Yi Xiao, Ruibin Feng, Chi Sing Leung, Pui Fai Sum. 2015. Online Training for Open Faulty RBF Networks. *Neural Processing Letters* **42**:2, 397-416. [CrossRef]
5. Lawrence McAfee, Kunle OlukotunEMEURO: A framework for generating multi-purpose accelerators via deep learning 125-135. [CrossRef]
6. Ruibin Feng, Yi Xiao, Chi Sing Leung, Peter W. M. Tsang, John Sum. 2014. An Improved Fault-Tolerant Objective Function and Learning Algorithm for Training the Radial Basis Function Neural Network. *Cognitive Computation* **6**:3, 293-303. [CrossRef]
7. Farhana Kausar, Mohammed Ameen UllaA perspective of fault tolerance for XOR in neural network 1-4. [CrossRef]
8. Amit Prakash Singh, Sartaj Singh Sodhi, Pravin Chandra, Chandra Shekhar RaiMetrics for Weight Stuck-at-Zero Fault in Sigmoidal FFANNs 61-66. [CrossRef]
9. Vahid Malekian, Rasoul Amirfattahi, Saeid Sadri, Mojgan Mokhtari, Alireza Aghaie, Mahdie Rezaeian. 2013. Computer aided measurement of sub-epithelial collagen band in colon biopsies for collagenous colitis diagnosis. *Micron* **45**, 59-67. [CrossRef]
10. J. P. Sum, Chi-Sing Leung, K. I-J Ho. 2012. On-Line Node Fault Injection Training Algorithm for MLP Networks: Objective Function and Convergence Analysis. *IEEE Transactions on Neural Networks and Learning Systems* **23**:2, 211-222. [CrossRef]
11. Robert A. Nawrocki, Richard M. VoylesArtificial neural network performance degradation under network damage: Stuck-at faults 442-449. [CrossRef]
12. Shue Kwan Mak, Pui-Fai Sum, Chi-Sing Leung. 2011. Regularizers for fault tolerant multilayer feedforward networks. *Neurocomputing* **74**:11, 2028-2040. [CrossRef]
13. Joni Pajarinen, Jaakko Peltonen, Mikko A. Uusitalo. 2011. Fault tolerant machine learning for nanoscale cognitive radio. *Neurocomputing* **74**:5, 753-764. [CrossRef]
14. Amit Prakash Singh, C. S. Rai, Pravin ChandraMetrics for measurement of additive noise to weight in sigmoidal FFANNs 2197-2201. [CrossRef]

15. Chi Sing Leung, Hong-Jiang Wang, J Sum. 2010. On the Selection of Weight Decay Parameter for Faulty Networks. *IEEE Transactions on Neural Networks* **21**:8, 1232-1244. [CrossRef]

16. Kevin I.-J Ho, Chi-Sing Leung, John Sum. 2010. Convergence and Objective Functions of Some Fault/Noise-Injection-Based Online Learning Algorithms for RBF Networks. *IEEE Transactions on Neural Networks* **21**:6, 938-947. [CrossRef]

17. Amit Prakash Singh, Pravin Chandra, Chandra Sekhar RaiFault Models for Neural Hardware 7-12. [CrossRef]

18. John Sum, Andrew Chi-Sing Leung. 2008. Prediction error of a fault tolerant neural network. *Neurocomputing* **72**:1-3, 653-658. [CrossRef]

19. Ralf Eickhoff, Ulrich Rückert. 2007. Robustness of radial basis functions. *Neurocomputing* **70**:16-18, 2758-2767. [CrossRef]