Protein Cell 2016, 7(10):735–748
DOI 10.1007/s13238-016-0302-5

Protein & Cell

RESEARCH ARTICLE

The superior fault tolerance of artificial neural network training with a fault/noise injection-based genetic algorithm

Feng Su[1,2,3], Peijiang Yuan[1], Yangzhen Wang[2,3], Chen Zhang[2,3]✉

[1] Robotics Institute, Beihang University, Beijing 100191, China
[2] State Key Laboratory of Membrane Biology, School of Life Sciences, Beijing 100871, China
[3] PKU-IDG/McGovern Institute for Brain Research, Peking University, Beijing 100871, China
✉ Correspondence: ch.zhang@pku.edu.cn (C. Zhang)
Received June 20, 2016 Accepted July 12, 2016

ABSTRACT

Artificial neural networks (ANNs) are powerful computational tools that are designed to replicate the human brain and adopted to solve a variety of problems in many different fields. Fault tolerance (FT), an important property of ANNs, ensures their reliability when significant portions of a network are lost. In this paper, a fault/noise injection-based (FIB) genetic algorithm (GA) is proposed to construct fault-tolerant ANNs. The FT performance of an FIB-GA was compared with that of a common genetic algorithm, the back-propagation algorithm, and the modification of weights algorithm. The FIB-GA showed a slower fitting speed when solving the exclusive OR (XOR) problem and the overlapping classification problem, but it significantly reduced the errors in cases of single or multiple faults in ANN weights or nodes. Further analysis revealed that the fit weights showed no correlation with the fitting errors in the ANNs constructed with the FIB-GA, suggesting a relatively even distribution of the various fitting parameters. In contrast, the output weights in the training of ANNs implemented with the use the other three algorithms demonstrated a positive correlation with the errors. Our findings therefore indicate that a combination of the fault/noise injection-based method and a GA is capable of introducing FT to ANNs and imply that the distributed ANNs demonstrate superior FT performance.

KEYWORDS artificial neural networks, fault tolerance, genetic algorithm

INTRODUCTION

The brain is composed of biological neural networks (BNNs) that contain billions of interconnecting neurons with the ability to perform computations. Artificial neural networks (ANNs), mathematical models that mimic BNNs, are typically built as structured node groups with activation functions and connection weights that are adjusted based on the applied learning rules (Hampson, 1991, 1994; Basheer and Hajmeer, 2000; Krogh, 2008). Because of their powerful computational and learning abilities, ANNs are being used increasingly in various fields, including computation, engineering, machine learning, clinical medicine, and cognitive science (Presnell and Cohen, 1993; Baxt, 1995; Dybowski and Gant, 1995; Forsstrom and Dalton, 1995; Kamimura et al., 1996; Almeida, 2002; Lisboa, 2002; Rajan and Tolley, 2005; Lisboa and Taktak, 2006; Patel and Goyal, 2007; Hu et al., 2013; Street et al., 2013; Azimi et al., 2015).

Fault tolerance (FT), an important feature of BNNs, ensures the fidelity and reality of a system's input-output relationship. The FT of BNNs is thought to rely on extensive parallel interconnections, distributed information storage and processing, and self-learning and self-organizing characteristics. For instance, Alzheimer's patients lose a significant number of neurons (sometimes equaling half the normal brain mass) but still maintain certain brain functions (Fayed et al., 2012; Li et al., 2012; Weiner et al., 2015; Pini et al.,

Feng Su, Peijiang Yuan and Yangzhen Wang have contributed equally to this work.

Electronic supplementary material The online version of this article (doi:10.1007/s13238-016-0302-5) contains supplementary material, which is available to authorized users.

2016). Moreover, structural measurements of various areas of the brain have revealed that brain volume has no direct correlation with cognitive decline in patients (Braskie and Thompson, 2014). Fault tolerance is also an important consideration in the construction of ANNs, especially in highly variable or "fail-safe" systems (Protzel et al., 1993; Phatak and Koren, 1995b). A fault-tolerant ANN is a special ANN system designed to work normally, or at least to a certain degree of normalcy, even if some of its components are unexpectedly damaged. Recently, FT performance has become more important, partly due to the fact that the soft errors caused by transient faults are an unavoidable concern in very large-scale integration (VLSI) technology, whose dimension is approaching the nanoscale (Mahdiani et al., 2012).

To construct a fault-tolerant ANN, neurons (nodes) are replicated in the hidden layer (Emmerson and Damper, 1993; Medler and Dawson, 1994; Phatak and Koren, 1995a; Tchernev et al., 2005). In this way, FT is introduced to the ANNs at the expense of increased complexity. For instance, ANNs with thousands of artificial neurons and up to a million interconnections in the hidden layer are required to solve complex problems, such as mapping landslide susceptibility (Arnone et al., 2014), modeling pectus excavatum corrective prostheses (Rodrigues et al., 2014), and reconstructing traffic networks (Jiang et al., 2014). However, this increase in network complexity makes the hardware implementation of ANNs relatively difficult and inefficient. Other methods that have been proposed to build fault-tolerant ANNs include an adjustment of the distributing weight values (Cavalieri and Mirabella, 1999a) using an empirical equation to deduce the mean prediction error (Sum and Leung, 2008) and adopting two objective functions (i.e., one that deals with open-weight fault and another that deals with open node fault (Mak et al., 2011)) during the training process to improve network FT. However, to our knowledge, no studies have investigated in detail whether a genetic algorithm might enhance the FT of ANNs.

A genetic algorithm (GA) is a heuristic algorithm used to search for a non-random optimal solution to a problem by mimicking the evolutionary process of natural selection (Holland, 1975; Goldberg, 1989). The GA process is iterative and includes initialization, selection, and genetic operation. The genetic operation usually consists of inheritance, mutation, and crossover. In each iteration, which is also called a generation, a fitness function is used to evaluate the fitness of individuals to find the best solution. Thus, a GA requires only a solvable function, which makes it suitable for complex and non-linear problems. Genetic algorithms have been applied to solve a variety of problems, especially when the basic functions are not discontinuous or non-differentiable (Forrest, 1993; Maddox, 1995; Willett, 1995; Pedersen and Moult, 1996; Meurice et al., 1998; Weber, 1998; Liu and Wang, 2001; Rothlauf et al., 2002; Jamshidi, 2003; Leardi, 2007; Wu, 2007; Gerlee et al., 2011; Manning et al., 2013; Pena-Malavera et al., 2014).

Thus, this study proposes an approach that combines an FIB learning algorithm with a GA to build fault-tolerant ANNs and to demonstrate this method's superior FT performance in comparison with that of a general GA (GE-GA) and two classic existing algorithms, the back-propagation (BP) algorithm and the modification of weight (MW) methods, in solving an exclusive OR (XOR) problem or an overlapping classification problem.

RESULTS

Training ANNs to solve an XOR problem with a GA

An XOR problem was used to train the ANN with either a GE-GA or an FIB-GA. Figure 1A illustrates the architecture of the ANN, and Fig. S1 illustrates the artificial neuron model. Two classic algorithms were included in the comparisons: the back-propagation (BP) and the modification of weights (MW). The BP algorithm is a traditional learning method based on a gradient descent, and the MW algorithm modifies the weight during the learning phase if the absolute value of the weight exceeds a certain threshold. For each experiment, the training proceeded until the terminating condition (i.e., error less than 0.001 or number of iterations reaching 1,000) was satisfied. Figure 1B illustrates changes in the error (or minimum error in training with a GE-GA or FIB-GA) in one iteration versus the number of iterations. Errors with all four fitting methods declined with increasing iterations. The BP and MW methods reached the terminating condition much faster (BP: 3.0 ± 0.0, MW: 3.8 ± 0.8; GA: 610.6 ± 274.8; FT: 905.6 ± 148.4 iterations) compared with the other two methods. Furthermore, the $\mathrm{Error}_{\mathrm{BP}}$, $\mathrm{Error}_{\mathrm{MW}}$ and $\mathrm{Error}_{\mathrm{GE-GA}}$ decayed approximately in an exponential manner (τ = 5.6196 ± 0.8967, fitting function: $f(x) = a \cdot e^{-\tau x}$, 4.1171 ± 0.6494, and 0.0186 ± 0.0050 $\mathrm{iteration}^{-1}$, respectively); however, the $\mathrm{Error}_{\mathrm{FIB-GA}}$ decayed much more slowly and in an irregular manner (Fig. 1B), suggesting low efficiency in the parameter optimization process. As there are four elements in the output vector ($c = [c_1\ c_2\ c_3\ c_4]$) and the calculated error in one iteration comprises the average from all four individual elements, we also examined, in each training period, the error of each element that is given by

$$\mathrm{Error\_c_i} = c_i^{\mathrm{calculated}} - c_i^{\mathrm{actual}} \quad (i = 1, 2, 3, 4).$$

Figure 1C illustrates the fluctuations in the error of each element during 20 independent trainings. The average fluctuations during the training of ANNs with BP, MW, GE-GA, and FIB-GA were BP: 0.0001 ± 0.0000, MW: 0.0001 ± 0.0000, GE-GA: 0.0007 ± 0.0009, and FIB-GA: 0.0024 ± 0.0025, respectively. Statistical analysis revealed that the FIB-GA method showed the biggest fluctuation when compared with the other three methods (Table S1). Taken together, all four methods demonstrated the capability of training the ANN successfully, although at different speeds.

Typically, in an FT ANN, the impact of each node is distributed as evenly as possible so as to avoid *dominant*

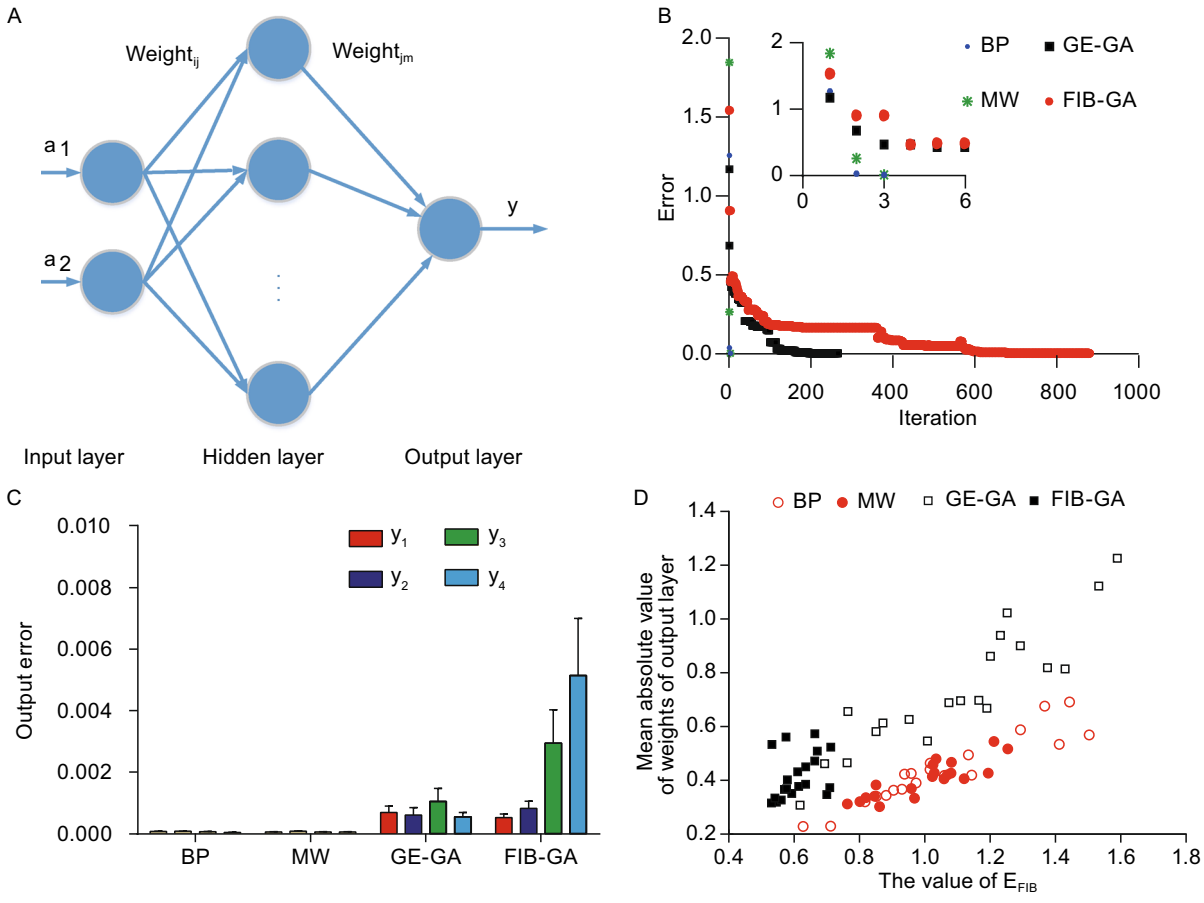


Figure 1. The use of BP, MW, GE-GA, and FIB-GA in training ANNs to solve an XOR problem. (A) The topology of the artificial neural networks. (B) The plot of the fitting errors versus the number of iterations. The inset shows the changes of the error within the initial six iterations. (C) The summary graph comparing the error of each element in the output vectors between BP ANN, MW ANN, GE-GA ANN, and FIB-GA ANN. (D) The plot of the weights for the output neuron versus the fitting errors in ANN training with the BP, MW, GE-GA, and FIB-GA.

Table 1. Correlation between each category of parameters and average errors

ANN	BP ANN				MW ANN			
Parameter	$\exists^{\mathcal{H}}_{\mathcal{MA}}$	$\lfloor^{\mathcal{H}}_{\mathcal{MA}}$	$\exists^{\mathcal{O}}_{\mathcal{MA}}$	$\lfloor^{\mathcal{O}}$	$\exists^{\mathcal{H}}_{\mathcal{MA}}$	$\lfloor^{\mathcal{H}}_{\mathcal{MA}}$	$\exists^{\mathcal{O}}_{\mathcal{MA}}$	$\lfloor^{\mathcal{O}}$
r	−0.659	0.158	0.936	−0.152	0.051	0.018	0.857	0.064
p	0.002	0.507	0	0.521	0.831	0.940	0	0.788
ANN	GE-GA ANN				FIB-GA ANN			
Parameter	$\exists^{\mathcal{H}}_{\mathcal{MA}}$	$\lfloor^{\mathcal{H}}_{\mathcal{MA}}$	$\exists^{\mathcal{O}}_{\mathcal{MA}}$	$\lfloor^{\mathcal{O}}$	$\exists^{\mathcal{H}}_{\mathcal{MA}}$	$\lfloor^{\mathcal{H}}_{\mathcal{MA}}$	$\exists^{\mathcal{O}}_{\mathcal{MA}}$	$\lfloor^{\mathcal{O}}$
r	−0.789	−0.070	0.901	0.291	−0.260	0.524	0.339	−0.258
p	0	0.768	0	0.214	0.268	0.018	0.144	0.272

nodes. Thus, we evaluated the correlation index between different ANN parameters and the fitting errors. All 25 parameters were grouped into four categories: 12 weights (weight$_{ij}$) and six biases (bias$_j$) for neurons in the hidden layer and six weights (weight$_{jm}$) and one bias (bias$_m$) for the output neuron. Table 1 summarizes the correlation efficiency and significance between each category of the parameters and errors. The weights for the neurons in the hidden-layer

ANN training using the BP and the GE-GA strongly correlated negatively with the fitting error; however, those in the other two ANNs did not. The bias for the output neuron has no significant correlation with the fitting errors in all four algorithms. All the output neuron weights in the ANN training with the GE-GA, BP, and MW strongly correlated with the errors (Fig. 1D); however, those in the FIB-GA ANN training did not. These results showed that in the ANN training with the FIB-GA, no parameter set correlated with the fitting error, a finding that implies there is no dominant parameter in ANNs trained via the use of an FIB-GA.

The FT performance of ANNs in solving an XOR problem with a single fault

Fault tolerance is the property that allows an ANN or BNN to operate properly in the event one or more components are lost. We began by comparing the errors among the ANNs generated by the BP, MW, GE-GA, and FIB-GA methods in which one randomly selected network parameter was changed to 0 (void). The plot of the errors versus the faulty parameters in 20 independent experiments clearly shows that the ANNs constructed using the FIB-GA contains the least number of errors (Fig. 2A). The averaged errors from 20 independent experiments are as follows: BP: 0.2623 ± 0.0614, MW: 0.2507 ± 0.0355, GE-GA: 0.2746 ± 0.0698, and FIB-GA: 0.1527 ± 0.0150 (statistical test in Table S2). Assuming the error equals or exceeds 0.4 as a fault output, the error rates show a similar trend (Fig. 2B): BP: 24.80 ± 9.68%, MW: 21.60 ± 7.94%, GE-GA: 23.80 ± 10.66%, and FIB-GA: 8.60 ± 5.24% (statistical test in Table S2). Next, the FT performances of the four ANNs were compared when one neuron (rather than one parameter) in the hidden layer completely lost its responsiveness. The performances of all four ANNs were reduced when compared with the fully functional ANNs, while the FIB-GA ANNs showed the fewest errors (Fig. 2C): BP: 0.3900 ± 0.1041, MW: 0.3645 ± 0.0567, GE-GA: 0.5167 ± 0.1413, and FIB-GA: 0.2936 ± 0.0410; (statistical test in Table S3) and the lowest error rates, with a 0.4 threshold (Fig. 2D): BP: 45.00 ± 19.57%, MW: 40.83 ± 13.76%, GE-GA: 54.17 ± 20.86%, and FIB-GA: 27.50 ± 13.55% (statistical test in Table S3).

As the output matrix is composed of four elements in the XOR problem, the errors of the individual elements were compared. The distributions of Error – c_i among the four ANNs were plotted while voiding one parameter or one neuron in the hidden layer (Fig. S2). Among the four algorithms, ANN training with the FIB-GA consistently showed the least number of errors (Fig. S2A): BP: 0.2623 ± 0.0614, MW: 0.2507 ± 0.0355, GE-GA: 0.2746 ± 0.0698, and FIB-GA: 0.1527 ± 0.0150 (statistical test in Table S4) and the lowest error rate (Fig. S2B): BP: 25.40 ± 7.94%, MW: 25.90 ± 5.39%, GE-GA: 25.75 ± 7.35%, and FIB-GA: 14.85 ± 3.30% (statistical test in Table S4). When one neuron in the hidden layer was voided randomly, the average error and the error rate with the 0.4 threshold showed a similar trend (Fig. S2C): average error: BP: 0.3900 ± 0.1041, MW: 0.3645 ± 0.0567, GE-GA: 0.5167 ± 0.1413, and FIB-GA: 0.2936 ± 0.0410 (statistical test in Table S5). Figure S2D shows the error rate with a 0.4 threshold: BP: 43.75 ± 17.34%, MW: 42.50 ± 12.72%, GE-GA: 48.13 ± 16.36%, and FIB-GA: 31.25 ± 6.55% (statistical test in Table S5). Together, these results clearly show that the FIB-GA ANN has superior FT when one parameter or one neuron in the network is lost.

The FT performance of ANNs in solving an XOR problem with multiple faults

In both ANN and BNN, errors typically happen at multiple sites but are not restricted to one element. Thus, the performances of ANNs were compared to solve an XOR problem in which two to four parameters are disabled simultaneously. As each ANN has 25 parameters, there are 300 (C_{25}^2), 2,300 (C_{25}^3), and 12,650 (C_{25}^4) combinations when two, three, and four parameters are all set to 0, respectively. Figure 3A illustrates the distribution of error; the summarized data clearly show that the FIB-GA-trained ANN still performed best under multiple-fault conditions (Table 2; statistical test in Tables S6–7). Next, we examined the errors occurring when two to six neurons in the hidden layer are voided. Under these circumstances, the ANN trained using the GE-GA showed the largest number of errors, while the ANN trained using the FIB-GA demonstrated the best performance (Fig. 3B). The error rates with a 0.4 threshold displayed the same order in the fitting performance (Fig. 3C). Not surprisingly, the performance of the ANNs trained using the FIB-GA was significantly better than the other three ANNs; however, the performance of the FIB-GA-trained ANNs weakened as the number of voided neurons increased (Table 3 and Fig. S3; statistical test in Tables S8–9). Since the performance of ANNs highly relies on the number of nodes in the hidden layer(Xu and Xu, 2013; Sasakawa et al., 2014). we next investigated whether the number of hidden neurons could affect the FT performance of the four ANNs in solving the XOR problem. In the ANNs with three or nine neurons in the hidden layer, the FIB-GA-trained ANN continued to demonstrate an FT performance that was superior to that of the BP, MW, and GE-GA ANNs (three neurons: Table 4; statistical test in Tables S10–11; nine neurons: Table 5; statistical test in Tables S12–13).

Together, these results demonstrate that the FIB-GA ANN has superior FT in solving XOR problems when multiple parameters or neurons in the network are lost.

The FT performance of ANNs in solving an overlapping classification problem

Next, we examined the FT performance of the four ANNs in solving an overlapping classification problem (Fig. 4A) that is more complicated than an XOR problem. Solving

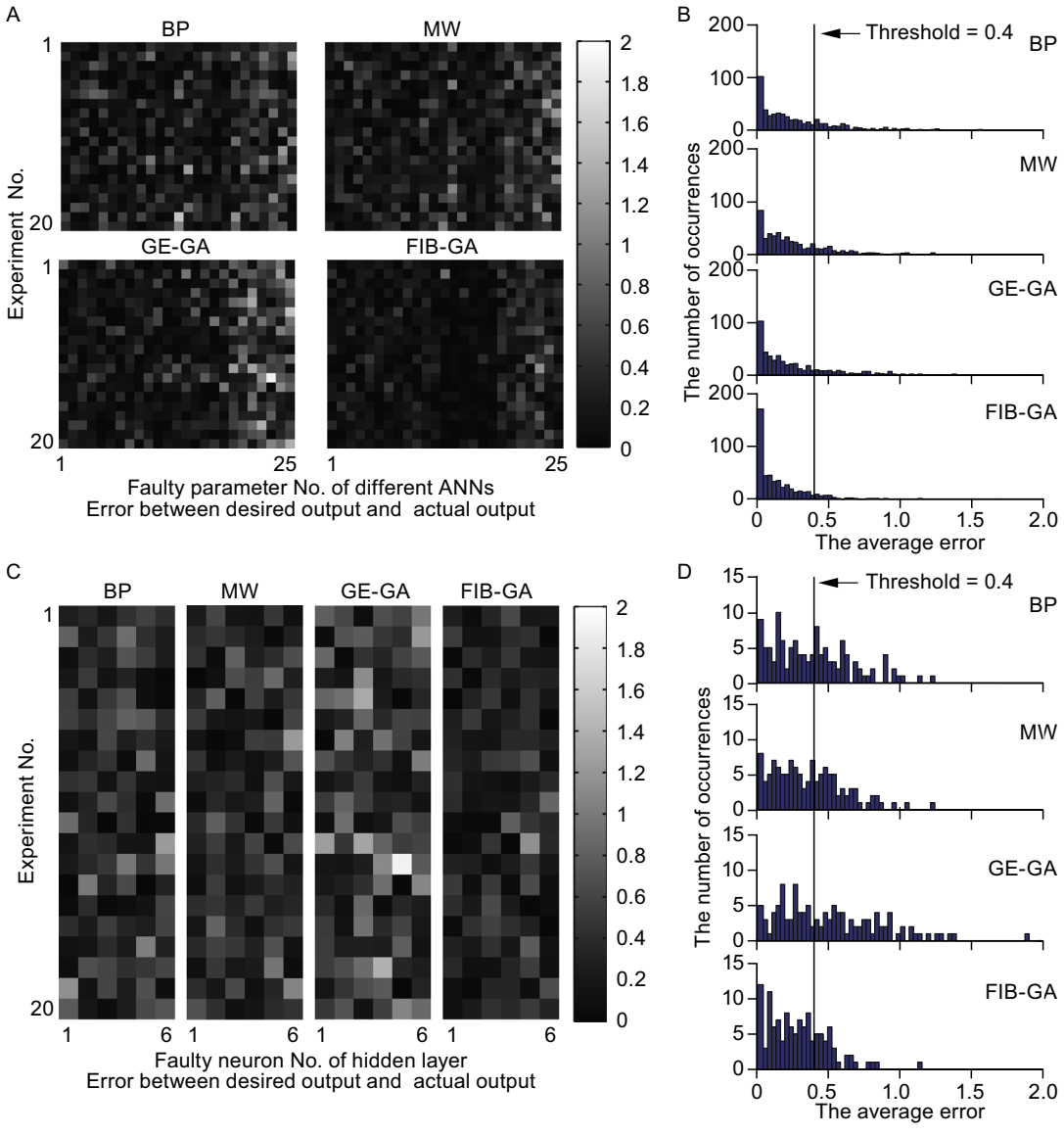
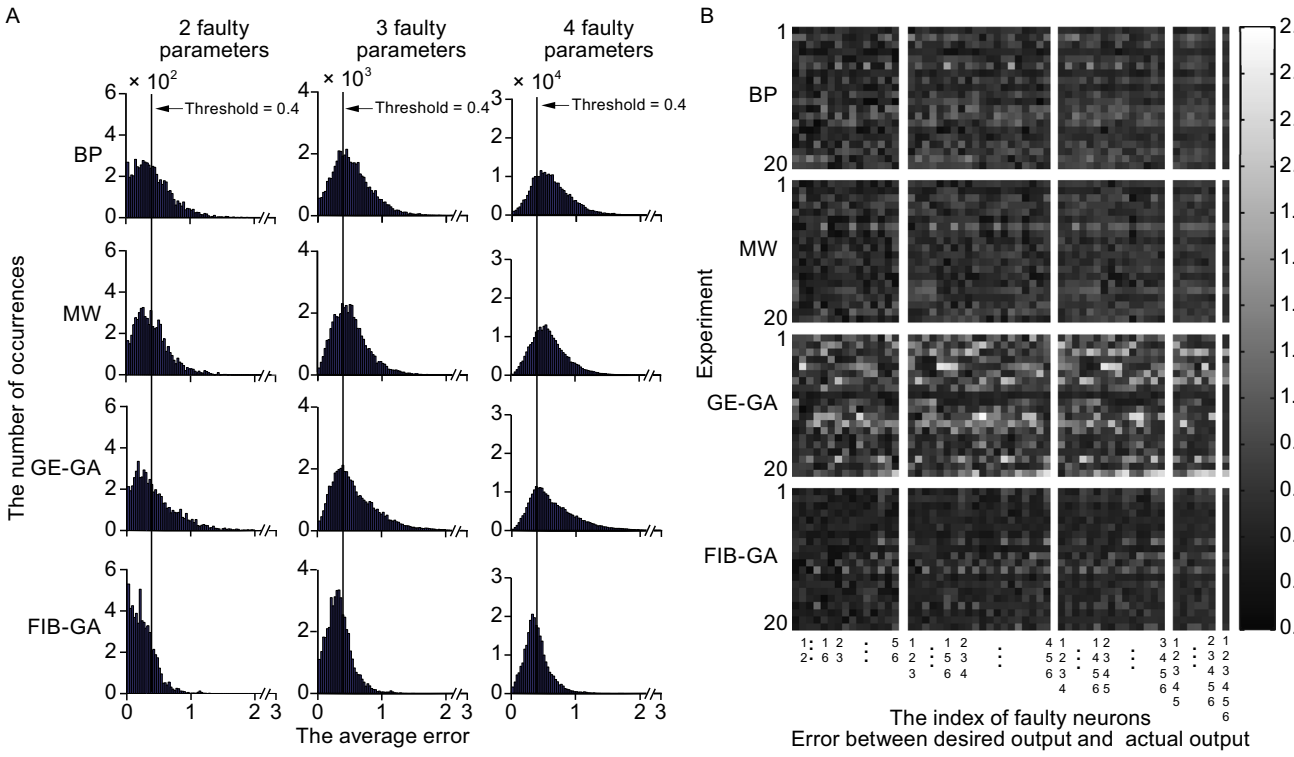
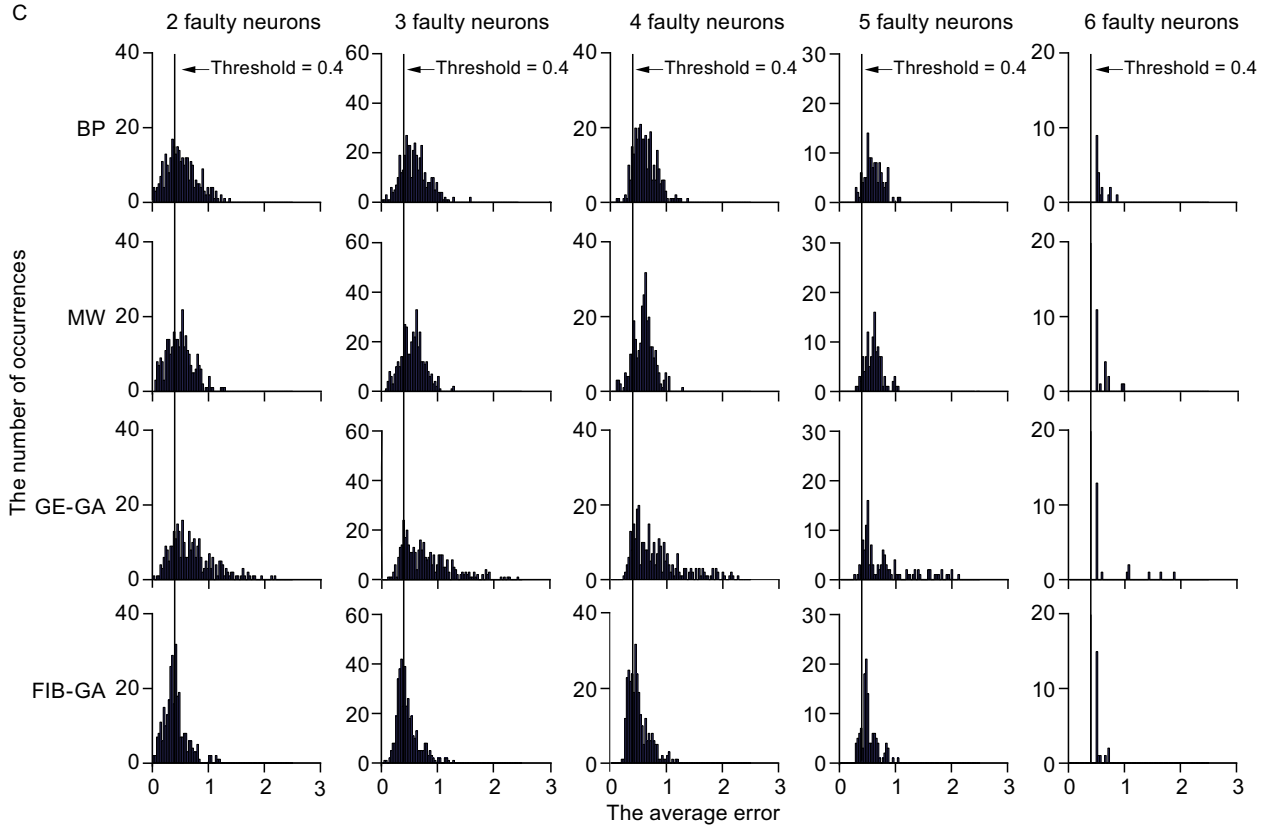


Figure 2. The FT performance of ANNs in solving an XOR problem with a single faulty parameter or neuron in the hidden layer. (A and B) The plot of errors versus the faulty parameter (A) and a histogram of error occurrence (B) in 20 independent experiments using ANNs trained via BP, MW, GE-GA, and FIB-GA methods. (C and D) The plot of errors versus the faulty neurons in the hidden layer (C) and a histogram of error occurrence (D) in 20 independent experiments using ANNs trained via BP, MW, GE-GA, and FIB-GA methods. In panels C and D, a fault output represents a fitting with the error equal to or exceeding a threshold of 0.4 (red lines).

overlapping classification problems using ANNs has been investigated extensively in the pattern recognition and machine-learning areas (Lovell and Bradley, 1996; Tang et al., 2010; Xiong et al., 2010). We adopted an ANN with the same structures used in these previous studies (see Fig. 1A). The terminating condition was set at 1,000 iterations, since none of the four ANNs could satisfy the condition that the fitting error must be equal to or less than 0.001 within 1,000 iterations, partly due to the complexity of the problem. Figure 4B illustrates the changes in the correct rates versus the number of iterations. The correct rates of BP and MW ANNs increased significantly faster compared with those of GE-GA and FIB-GA (fitting function: $f(x) = a \cdot e^{-\tau x}$, square class RCR: BP: 0.4662 ± 0.0006, τ = 0.2717, R^2 = 0.9880, MW: 0.4672 ± 0.0010, τ = 0.2529, R^2 = 0.8855, GE-GA: 0.4605 ± 0.0166, τ = 0.0040, R^2 = 0.9068, FIB-GA: 0.4578 ± 0.0083, τ = 0.0030, R^2 = 0.8502, and circle class RCR: BP: 0.4542 ± 0.0006, τ = 0.2210, R^2 = 0.9230, MW: 0.4541 ± 0.0115, τ = 0.2581, R^2 = 0.8996, GE-GA: 0.4547 ± 0.0115, τ = 0.0084, R^2 = 0.9014, FIB-GA: 0.4605 ± 0.0074, τ = 0.0111, R^2 = 0.7846 (statistical test in





◀ **Figure 3. The FT performance of ANNs in solving an XOR problem with multiple faulty parameters or neurons in the hidden layer.** (A) The histogram of error occurrence in 20 independent experiments using ANNs trained via BP, MW, GE-GA, and FIB-GA methods. (B and C) The plot of errors versus faulty neurons (B) and a histogram of error occurrence (C) in 20 independent experiments using ANNs trained via BP, MW, GE-GA, and FIB-GA methods.

Table S14). The number of iterations that occurred when RCR reached 0.45 are shown for each method as follows: BP 23.2 ± 14.3, MW 15.8 ± 8.3, GE-GA 613.7 ± 341.8, and FIB-GA 583.1 ± 287.7. We then examined the FT performance of these four ANNs in solving an overlapping classification problem and found that the FIB-GA ANN showed significantly fewer errors when one parameter or one neuron was voided. When any of the parameters were voided one at a time in 20 independent experiments, the square class RCRs for FIB-GA ANN was the highest compared with the other three ANNs, while the other three ANNs were not significantly different (square class RCRs: BP: 0.3407 ± 0.0397, MW: 0.3536 ± 0.0317, GE-GA: 0.2863 ± 0.0501, and FIB-GA: 0.4116 ± 0.0247; circle class RCRs: BP: 0.2978 ± 0.0600, MW: 0.3075 ± 0.0301, GE-GA: 0.2151 ± 0.0601, and FIB-GA: 0.4045 ± 0.0258 (statistical test in Table S15) (Fig. 4C). When one neuron in the hidden layer was voided randomly, FIB-GA ANN still significantly outperformed the other three ANNs (square class RCRs: BP: 0.2392 ± 0.1049, MW: 0.2646 ± 0.0743, GE-GA: 0.1334 ± 0.1160, and FIB-GA: 0.4224 ± 0.0586; circle class RCRs: BP: 0.2396 ± 0.1523, MW: 0.2574 ± 0.0911, GE-GA: 0.0073 ± 0.1242, and FIB-GA: 0.4164 ± 0.0640 (statistical test in Table S16) (Fig. 4D). We next randomly voided two to three parameters or neurons in these ANNs and compared their FT performance. As illustrated in Figure 5, voiding two to three parameters or neurons reduced the performance of all four ANNs. The FIB-GA showed the fewest errors and lowest error rates under almost all the fault conditions tested (Tables S17–18). Thus, our data clearly demonstrate that, compared to the ANNs trained using the BP, MW, and GE-GA methods, the FT ability of the ANN trained using the FIB-GA, at a relatively low training speed, is superior.

DISCUSSION

This study compared the FT performances of ANNs constructed with four different algorithms: BP, MW, GE-GA, and FIB-GA. The FIB-GA was constructed via the use of FIB learning algorithms, which has been proven to be a common and efficient method of training fault-tolerant neural networks that includes the addition of noise to the input, weights, or nodes (Leung and Sum, 2008; Ho et al., 2010). Our results clearly show that the FIB learning algorithm is an efficient

Table 2. Average errors and error rates of multiple faulty parameters with six hidden neurons

Error		BP	MW	GE-GA	FIB-GA
Average error	2	0.4295 ± 0.0925	0.4092 ± 0.0538	0.4534 ± 0.1092	0.2603 ± 0.0271
	3	0.5419 ± 0.1086	0.5158 ± 0.0649	0.5752 ± 0.1334	0.3392 ± 0.0373
	4	0.6199 ± 0.1167	0.5903 ± 0.0716	0.6609 ± 0.1490	0.3983 ± 0.0453
Error rate	2	47.03% ± 13.11%	45.03% ± 9.92%	45.45% ± 15.25%	20.05% ± 7.50%
	3	63.52% ± 12.68%	62.87% ± 9.52%	62.07% ± 15.54%	31.93% ± 9.06%
	4	74.61% ± 10.75%	74.31% ± 8.18%	73.43% ± 13.86%	43.29% ± 9.75%

Table 3. Average errors and error rates of multiple faulty neurons with six hidden neurons

Error		BP	MW	GE-GA	FIB-GA
Average error	2	0.5229 ± 0.1127	0.4891 ± 0.0794	0.7041 ± 0.1804	0.4077 ± 0.0657
	3	0.5960 ± 0.1231	0.5593 ± 0.0878	0.7891 ± 0.2036	0.4665 ± 0.0837
	4	0.6227 ± 0.1210	0.5974 ± 0.0969	0.8235 ± 0.2398	0.5009 ± 0.0909
	5	0.6115 ± 0.1174	0.6133 ± 0.1176	0.8096 ± 0.3082	0.5267 ± 0.0905
	6	0.5765 ± 0.1094	0.6037 ± 0.1489	0.7618 ± 0.4398	0.5382 ± 0.0740
Error rate	2	62.67% ± 12.31%	62.00% ± 13.87%	77.67% ± 18.00%	47.33% ± 15.58%
	3	80.00% ± 13.76%	78.00% ± 12.61%	81.75% ± 16.96%	55.25% ± 16.26%
	4	88.33% ± 14.00%	86.67% ± 11.03%	89.00% ± 13.90%	65.00% ± 14.49%
	5	90.83% ± 16.64%	94.17% ± 11.18%	95.00% ± 9.52%	81.67% ± 13.13%
	6	100.00% ± 0.00%	100.00% ± 0.00%	100.00% ± 0.00%	100.00% ± 0.00%

Table 4. Error rates of multiple faulty parameters and neurons with three hidden neurons

Error		BP	MW	GE-GA	FIB-GA
Parameters	1	33.85% ± 14.21%	41.15% ± 16.03%	38.85% ± 13.08%	8.46% ± 6.06%
	2	63.91% ± 12.94%	71.15% ± 13.27%	70.51% ± 13.50%	30.77% ± 7.51%
	3	82.52% ± 8.51%	87.13% ± 7.39%	86.00% ± 9.24%	56.59% ± 4.78%
Neurons	1	71.67% ± 16.31%	75.00% ± 18.34%	86.67% ± 19.94%	18.33% ± 17.01%
	2	95.00% ± 16.31%	95.00% ± 16.31%	88.33% ± 16.31%	46.67% ± 42.44%
	3	100.00% ± 0.00%	100.00% ± 0.00%	100.00% ± 0.00%	100.00% ± 0.00%

Table 5. Error rates of multiple faulty parameters and neurons with nine hidden neurons

Error		BP	MW	GE-GA	FIB-GA
Parameters	1	22.70% ± 9.62%	23.11% ± 8.01%	20.95% ± 11.29%	1.22% ± 1.38%
	2	46.67% ± 12.70%	47.70% ± 12.56%	38.20% ± 15.48%	4.44% ± 1.91%
	3	63.45% ± 13.11%	64.35% ± 12.40%	52.43% ± 16.25%	9.57% ± 2.85%
Neurons	1	41.67% ± 15.24%	45.00% ± 14.18%	48.89% ± 20.52%	1.67% ± 4.07%
	2	60.14% ± 16.05%	65.69% ± 14.20%	66.39% ± 17.40%	10.83% ± 6.62%
	3	71.43% ± 12.67%	76.31% ± 11.55%	75.42% ± 14.32%	20.89% ± 7.74%

method for improving the FT performance of ANNs. The data of this study show that FIB-GA results in a significant improvement in errors between the actual and desired inputs when one or multiple neurons are voided. It is worth noting that, when solving an XOR problem or an overlapping classification problem with the continuous and differentiable basis function, the GE-GA does not offer an advantage over the BP and MW algorithms. This might suggest that FT is not an intrinsic property of GA ANNs. In contrast, compared to the two GAs, the BP and MW algorithms showed much faster training speeds, which implies that efficiency competes with fault tolerance in ANNs. An option for increasing the FT performances of ANNs is to avoid weights or neurons with significant effects on errors. Our analysis showed that the weights between the hidden layer and the output layer in the ANNs trained with the GE-GA, BP, or MW, but not those trained with the FIB-GA, are correlated with the output errors, a finding which clearly supports the notion that robustness is greater in a distributed ANN. This is also consistent with previous attempts to improve the partial FT of ANNs by distributing the absolute value of weights uniformly (Cavalieri and Mirabella, 1999a, b). In addition, Macia and Sole reported that degeneracy, rather than redundancy, is necessary for reliable designs of NAND (NOT AND, a binary operation) gate-forming systems(Macia and Sole, 2009). Considering the fact that the BNNs are also distributed systems yielding a high FT performance, distributed storage and processing seem to be key properties in both ANNs and BNNs. Together, our results propose that a fault/noise injection-based genetic algorithm would serve as an efficient approach for improving the FT in ANNs.

METHODS

The architecture of the ANN

In this study, a three-layer ANN was constructed: an input layer of two neurons, a hidden layer comprised of different numbers of neurons, and an output layer of one neuron. Figure 1A shows the architecture. Each neuron receives multiple weighted inputs and sends one weighted output to the connected neurons. Simply stated, neurons are interconnected through a unidirectional manner (input → hidden → output direction), and there is no connection within a layer (Fig. 1A). The input of the neuron *j* in the hidden layer is given by

$$x_j = \sum_{i=1}^{n} (\mathrm{Input}_{ij} \times \mathrm{weight}_{ij}) + b_j$$

where Input_{ij}, weight_{ij}, and b_j denote the input, weight, and input bias of the postsynaptic neuron *j*, which is connected to the presynaptic neuron *i*, and n denotes the number of the presynaptic neurons connecting to the neuron *j*.

The output of the neuron *j* is given by the following tansig function:

$$y_j = f(x_j) = \frac{1 - e^{-2x_j}}{1 + e^{-2x_j}}$$

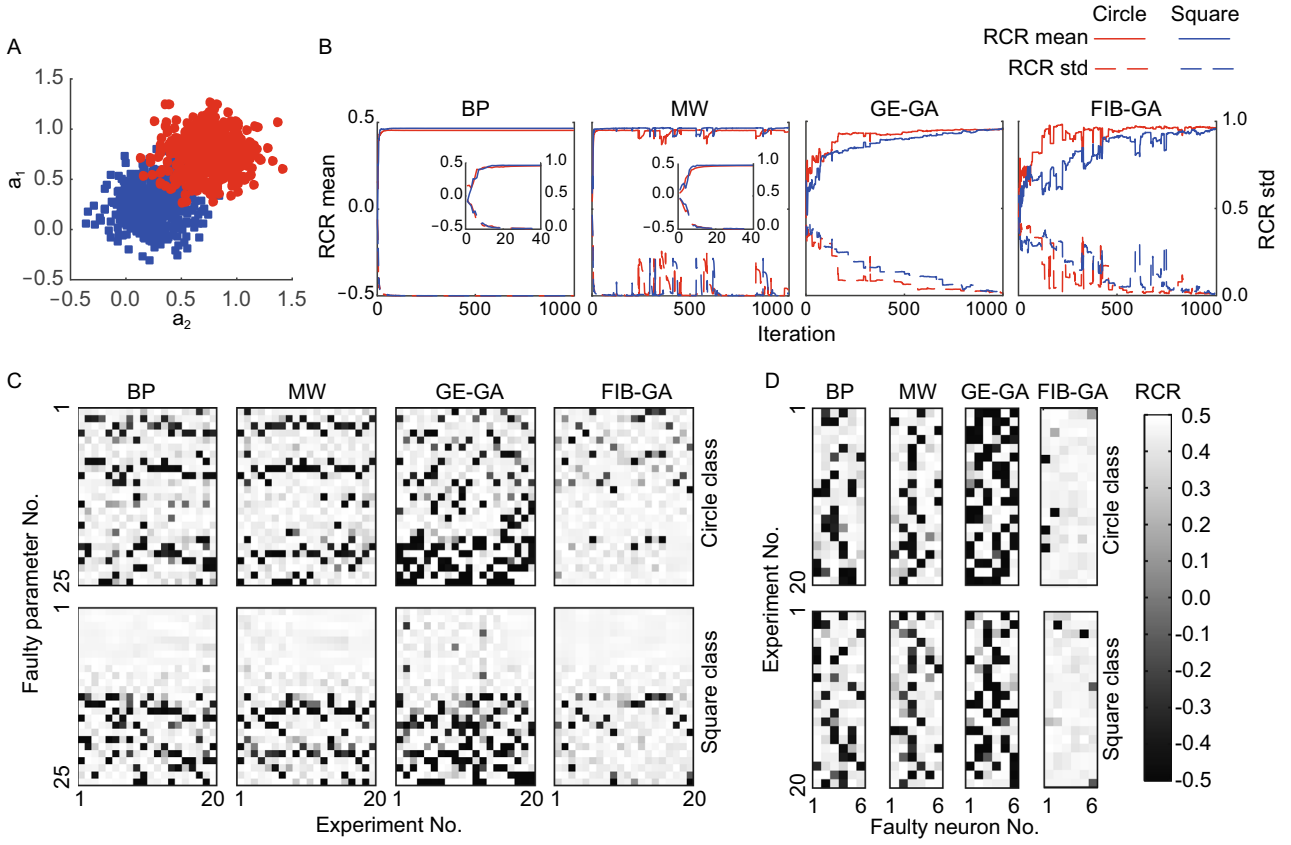


Figure 4. The FT performance of different ANNs with six hidden-layer neurons in solving an overlapping classification problem. (A) Two classes of Gaussian noise sources. (B) The plot of the relative correct rate versus the number of iterations. The inset shows the changes of the error within the initial 40 iterations. (C and D) The plot of errors versus the faulty parameters (C) and the plot of errors versus the faulty neurons in the hidden layer (D) in 20 independent experiments when using ANNs trained via BP, MW, GE-GA, and FIB-GA methods.

Calculation of the ANN to solve the XOR problem

A classic XOR problem was selected for use in training and examining the performance of the ANN. According to the architecture of the ANN used in this study (Fig. 1A), the training data, a_1 and a_2, are defined as follows:

$$a_1 = [1 \quad 0 \quad 1 \quad 0]$$

$$a_2 = [1 \quad 1 \quad 0 \quad 0]$$

Thus, the actual output of the XOR problem with these two inputs is given by

$$y = [y_1 \quad y_2 \quad y_3 \quad y_4] = a_1 \oplus a_2 = (\neg a_1 \wedge a_2) \vee (a_1 \wedge \neg a_2) = [0 \quad 1 \quad 1 \quad 0].$$

For one solution set, the output of the ANN is given by

$$c = [c_1 \; c_2 \; c_3 \; c_4]$$

where $c_p = \sum_{j=1}^{6}(f(x_j) \times \text{weight}_{jm}) + b_m$ and x_j denote the input to the neuron j in the hidden layer from the two presynaptic neurons (a_1 and a_2), which is given by $x_j = \sum_{i=1}^{2}(a_i(p) \times \text{weight}_{ij}) + b_j$.and Weight_{jm} and b_m denote the weight and bias of the neuron m in the output layer, respectively.

Thus, the error for one solution set is given by

$$\text{Error} = \frac{1}{4}\sum_{p=1}^{4} |c_p - y_p|$$

Training of the ANN using a GE-GA

A GE-GA was adopted for training the ANN. The basic idea of a GA is to mimic the process of natural selection and to find the best solution to a problem after several generations. In this study, the upper limit of iteration was set at 1,000, and 20 individuals (i.e., sets of solutions) were used in each generation (i.e., training cycle). The best individual is defined as the one set of solutions having minimum errors. In the first generation, each individual was assigned randomly. In the subsequent generations, the 20 individuals consisted of three parts: two elite individuals (N_{elite}), which are the two individuals carried forward from the previous generation and

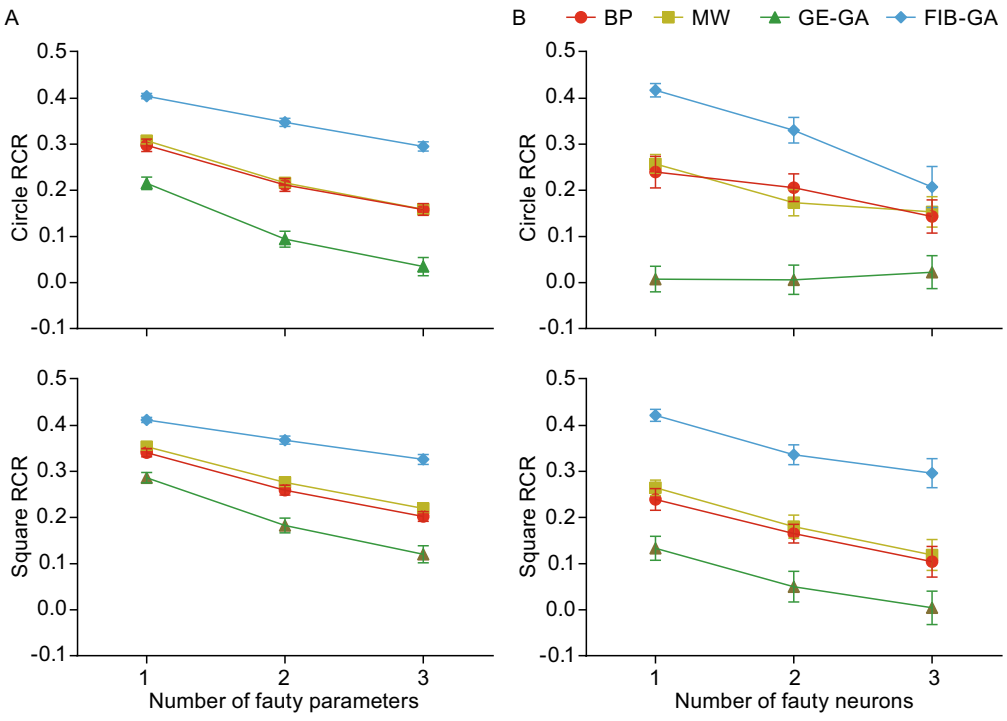


Figure 5. Relative correct rates of different ANNs with six hidden-layer neurons in solving an overlapping classification problem. (A) The plot of circle RCR (top) and square RCR (bottom) versus the number of faulty parameters. (B) The plot of circle RCR (top) and square RCR (bottom) versus the number of faulty neurons.

having the fewest errors, 14 crossover individuals ($N_{crossover}$), which are generated by combining two selected parents, and four mutation individuals ($N_{mutation}$). The selection criteria for parents is based on the scaled position ($Scaled_i$) of each individual within its generation. $Rank_i$ is redefined as the position of $Individual_i$ when sorting all the individuals in one generation by Error in ascending order. Thus, for individual i, the probability to be selected as a parent is calculated as follows:

$$P_i = Scaled_i \Big/ \sum_{j=1}^{20} \frac{1}{\sqrt{j}} = Scaled_i / 7.5953$$

where $Scaled_i = 1/\sqrt{Rank_i}$, ($Rank_i \neq Rank_j$ if $semicoloni \neq j$).

A line segment was then drawn that consisted of lines whose lengths were proportional to the P_i of each individual. A step size was given by $1/N_{parent}$, where $N_{parent} = 2 \cdot N_{crossover} + N_{mutation} = 2 \times 14 + 4 = 32$ and an initial position is denoted as $Initial_{position}$, where $0 < Initial_{position} < 1/N_{parent}$. A cursor is then placed at $Initial_{position}$ and is moved along in steps of $1/N_{parent}$. For each step, the position on which the cursor lands is selected as a parent. Thus, this algorithm generates 32 parents in one generation (Fig. S4). The crossover process generates a child by crossing two parents ($Parent_1 = [parP1_1, \cdots\cdots, parP1_{25}]$ and $Parent_2 = [parP2_1, \cdots\cdots, parP2_{25}]$) with a randomly generated binary vector $Coef = [Coe_1, \cdots\cdots, Coe_{25}]$, where Coe_i is assigned to 0 or 1 based on rounding a value that is randomly selected in the open interval (0,1). The parameter vector of the $Child = [parC_1, \cdots\cdots, parC_{25}]$ generated by the crossover is given by

$$parC_i = parP1_i \times Coe_i + parP2_i \times |Coe_i - 1|.$$

The mutation process generates a child from one parent ($Parent_1 = [parP1_1, \cdots\cdots, parP1_{25}]$) with a vector $Coef = [Coe_1, \cdots\cdots, Coe_{25}]$, where Coe_i follows a Gaussian distribution centered at 0 (Fig. S4C). The standard deviation of the Gaussian distribution in the first generation is 1, and it shrinks to 0 linearly when reaching the last generation. The parameter vector of the $Child = [parC_1, \cdots\cdots, parC_{25}]$ generated by the mutation is given by

$$parC_i = parP1_i + Coe_i.$$

The goal of training the ANN with a GE-GA is to search for the individual with the minimal $Error_{GE-GA}$ which is given by

$$Error_{GE-GA} = \frac{1}{4} \sum_{p=1}^{4} |c_p - y_p| \quad (1)$$

Training of ANN using with a FIB-GA

In addition to the GE-GA, a FIB-GA was another approach used to train the ANN. In the FIB-GA, faults on the ANN parameters were considered during the training process. Thus, the error for one set of solutions is given by

$$\text{Error}_{\text{FIB-GA}} = \frac{1}{25} \times \sum_{p=1}^{25} \text{Error}_i \quad (2)$$

where Error_i is the error when the i^{th} of the 25 parameters is forced to 0, assuming the corresponding parameter becomes faulty.

Overlapping classification problem

Two classes of Gaussian noise sources were considered (shown in Fig. 4A). The first class is shown as a blue square, with a mean at (a_1, $semicolona_2$) coordinates of (0.25, 0.25). The second class is shown as a red circle, with a mean at (a_1, $semicolona_2$) coordinates of (0.75, 0.75). Both classes have a standard deviation of 0.2. The coordinates (a_1, $semicolona_2$) were used as input in the ANN training, and the output is 0.5 and −0.5 for the circle class and the square class, respectively. Each class had 500 scatters in total, and all the data were shuffled before the training was initiated. An actual output value larger than 0 for a point in the circle class and an actual output value less than 0 for a point in the square class were regarded correct.

For each class with N ($0 \leq N \leq 500$) points classified correctly, the relative correct rate (RCR) is defined as follows:

$$\text{RCR} = \frac{N}{500} - 0.5.$$

ACKNOWLEDGEMENTS

This work was supported by grants from the National Basic Research Program (973 Program) (Nos. 2014CB942804, 2014BAI03B01, and 2012YQ0302604), Beijing Institute of Collaborative Innovation (15I-15-BJ), and the Seeding Grant for Medicine and Life Sciences of Peking University (2014-MB-11).

ABBREVIATIONS

ANNs, artificial neural networks; BP, back-propagation; FIB, fault/noise injection-based; FT, fault tolerance; GA, genetic algorithm; MW, modification of weights; VLSI, very large-scale integration; XOR, exclusive OR.

COMPLIANCE WITH ETHICS GUIDELINES

Feng Su, Peijiang Yuan, Yangzhen Wang and Chen Zhang declare that they have no conflict of interest. This article does not contain any studies with human or animal subjects performed by the any of the authors.

AUTHOR CONTRIBUTIONS

F.S., P.Y., Y.W., and C.Z. carried out the experiments; F.S., P.Y., Y.W., and C.Z. contributed to the planning of the work; F.S., P.Y., Y.W., and C.Z. wrote the paper.

OPEN ACCESS

REFERENCES

Almeida JS (2002) Predictive non-linear modeling of complex data by artificial neural networks. Curr Opin Biotechnol 13:72–76

Arnone E, Francipane A, Noto LV, Scarbaci A, La Loggia G (2014) Strategies investigation in using artificial neural network for landslide susceptibility mapping: application to a Sicilian catchment. J Hydroinf 16:502–515

Azimi P, Mohammadi HR, Benzel EC, Shahzadi S, Azhari S, Montazeri A (2015) Artificial neural networks in neurosurgery. J Neurol Neurosurg Psychiatry 86:251–256

Basheer IA, Hajmeer M (2000) Artificial neural networks: fundamentals, computing, design, and application. J Microbiol Methods 43:3–31

Baxt WG (1995) Application of artificial neural networks to clinical medicine. Lancet 346:1135–1138

Braskie MN, Thompson PM (2014) A focus on structural brain imaging in the Alzheimer's disease neuroimaging initiative. Biol Psychiatry 75:527–533

Cavalieri S, Mirabella O (1999a) A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks. Neural Netw 12:91–106

Cavalieri S, Mirabella O (1999b) A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks. Neural Netw 12:91–106

Dybowski R, Gant V (1995) Artificial neural networks in pathology and medical laboratories. Lancet 346:1203–1207

Emmerson MD, Damper RI (1993) Determining and improving the fault-tolerance of multilayer perceptrons in a pattern-recognition application. IEEE Trans Neural Netw 4:788–793

Fayed N, Modrego PJ, Salinas GR, Gazulla J (2012) Magnetic resonance imaging based clinical research in Alzheimer's disease. J Alzheimers Dis 31:S5–18

Forrest S (1993) Genetic algorithms: principles of natural selection applied to computation. Science 261:872–878

Forsstrom JJ, Dalton KJ (1995) Artificial neural networks for decision support in clinical medicine. Ann Med 27:509–517

Gerlee P, Basanta D, Anderson AR (2011) Evolving homeostatic tissue using genetic algorithms. Prog Biophys Mol Biol 106:414–425

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Pub. Co, Reading

Hampson S (1991) Generalization and specialization in artificial neural networks. Prog Neurobiol 37:383–431
Hampson S (1994) Problem solving in artificial neural networks. Prog Neurobiol 42:229–281
Ho KI, Leung CS, Sum J (2010) Convergence and objective functions of some fault/noise-injection-based online learning algorithms for RBF networks. IEEE Trans Neural Netw 21:938–947
Holland JH (1975) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor
Hu X, Cammann H, Meyer HA, Miller K, Jung K, Stephan C (2013) Artificial neural networks and prostate cancer–tools for diagnosis and management. Nat Rev Urol 10:174–182
Jamshidi M (2003) Tools for intelligent control: fuzzy controllers, neural networks and genetic algorithms. Philos Trans R Soc Lond A 361:1781–1808
Jiang DD, Zhao ZY, Xu ZZ, Yao CP, Xu HW (2014) How to reconstruct end-to-end traffic based on time-frequency analysis and artificial neural network. Aeu-Int J Electron Commun 68:915–925
Kamimura R, Konstantinov K, Stephanopoulos G (1996) Knowledge-based systems, artificial neural networks and pattern recognition: applications to biotechnological processes. Curr Opin Biotechnol 7:231–234
Krogh A (2008) What are artificial neural networks? Nat Biotechnol 26:195–197
Leardi R (2007) Genetic algorithms in chemistry. J Chromatogr A 1158:226–233
Leung CS, Sum JP (2008) A fault-tolerant regularizer for RBF networks. IEEE Trans Neural Netw 19:493–507
Li J, Pan P, Huang R, Shang H (2012) A meta-analysis of voxel-based morphometry studies of white matter volume alterations in Alzheimer's disease. Neurosci Biobehav Rev 36:757–763
Lisboa PJ (2002) A review of evidence of health benefit from artificial neural networks in medical intervention. Neural Netw 15:11–39
Lisboa PJ, Taktak AF (2006) The use of artificial neural networks in decision support in cancer: a systematic review. Neural Netw 19:408–415
Liu F, Wang J (2001) Genetic algorithms and its application to spectral analysis. Guang Pu Xue Yu Guang Pu Fen Xi 21:331–335
Lovell BC, Bradley AP (1996) The multiscale classifier. IEEE Trans Pattern Anal Mach Intell 18:124–137
Macia J, Sole RV (2009) Distributed robustness in cellular networks: insights from synthetic evolved circuits. J R Soc Interface 6:393–400
Maddox J (1995) Genetics helping molecular-dynamics. Nature 376:209
Mahdiani HR, Fakhraie SM, Lucas C (2012) Relaxed fault-tolerant hardware implementation of neural networks in the presence of multiple transient errors. IEEE Trans Neural Netw Learn Syst 23:1215–1228
Mak SK, Sum PF, Leung CS (2011) Regularizers for fault tolerant multilayer feedforward networks. Neurocomputing 74:2028–2040
Manning T, Sleator RD, Walsh P (2013) Naturally selecting solutions: the use of genetic algorithms in bioinformatics. Bioengineered 4:266–278
Medler DA, Dawson MR (1994) Training redundant artificial neural networks: imposing biology on technology. Psychol Res 57:54–62
Meurice N, Leherte L, Vercauteren DP (1998) Comparison of benzodiazepine-like compounds using topological analysis and genetic algorithms. SAR QSAR Environ Res 8:195–232
Patel JL, Goyal RK (2007) Applications of artificial neural networks in medical science. Curr Clin Pharmacol 2:217–226
Pedersen JT, Moult J (1996) Genetic algorithms for protein structure prediction. Curr Opin Struct Biol 6:227–231
Pena-Malavera A, Bruno C, Fernandez E, Balzarini M (2014) Comparison of algorithms to infer genetic population structure from unlinked molecular markers. Stat Appl Genet Mol Biol 13:391–402
Phatak DS, Koren I (1995a) Complete and partial fault-tolerance of feedforward neural nets. IEEE Trans Neural Netw 6:446–456
Phatak DS, Koren I (1995b) Complete and partial fault tolerance of feedforward neural nets. IEEE Trans Neural Netw 6:446–456
Pini L, Pievani M, Bocchetta M, Altomare D, Bosco P, Cavedo E, Galluzzi S, Marizzoni M, Frisoni GB (2016) Brain atrophy in Alzheimer's disease and aging. Ageing Res Rev 28:30002
Presnell SR, Cohen FE (1993) Artificial neural networks for pattern recognition in biochemical sequences. Annu Rev Biophys Biomol Struct 22:283–298
Protzel PW, Palumbo DL, Arras MK (1993) Performance and fault-tolerance of neural networks for optimization. IEEE Trans Neural Netw 4:600–614
Rajan P, Tolley DA (2005) Artificial neural networks in urolithiasis. Curr Opin Urol 15:133–137
Rodrigues PL, Rodrigues NF, Pinho ACM, Fonseca JC, Correia-Pinto J, Vilaca JL (2014) Automatic modeling of pectus excavatum corrective prosthesis using artificial neural networks. Med Eng Phys 36:1338–1345
Rothlauf F, Goldberg DE, Heinzl A (2002) Network random keys: a tree representation scheme for genetic and evolutionary algorithms. Evol Comput 10:75–97
Sasakawa T, Sawamoto J, Tsuji H (2014) Neural network to control output of hidden node according to input patterns. Am J Intell Syst 4:196–203
Street ME, Buscema M, Smerieri A, Montanini L, Grossi E (2013) Artificial neural networks, and evolutionary algorithms as a systems biology approach to a data-base on fetal growth restriction. Prog Biophys Mol Biol 113:433–438
Sum J, Leung ACS (2008) Prediction error of a fault tolerant neural network. Neurocomputing 72:653–658
Tang W, Mao KZ, Mak LO, Ng GW (2010) Classification for overlapping classes using optimized overlapping region detection and soft decision. Paper presented at: information fusion
Tchernev EB, Mulvaney RG, Phatak DS (2005) Investigating the fault tolerance of neural networks. Neural Comput 17:1646–1664
Weber L (1998) Applications of genetic algorithms in molecular diversity. Curr Opin Chem Biol 2:381–385
Weiner MW, Veitch DP, Aisen PS, Beckett LA, Cairns NJ, Cedarbaum J, Green RC, Harvey D, Jack CR, Jagust W et al (2015) 2014 update of the Alzheimer's disease neuroimaging initiative: a review of papers published since its inception. Alzheimers Dement 11:001

Willett P (1995) Genetic algorithms in molecular recognition and design. Trends Biotechnol 13:516–521

Wu AH (2007) Use of genetic and nongenetic factors in warfarin dosing algorithms. Pharmacogenomics 8:851–861

Xiong H, Wu J, Liu L (2010) Classification with class overlapping: a systematic study. ICEBI-10

Xu C, Xu C (2013) Optimization analysis of dynamic sample number and hidden layer node number based on BP neural network. Springer, Berlin

