

# Diversity, Neural Nets and Safety Critical Applications\*

A.J.C. Sharkey, N.E. Sharkey and O.C. Gopinath  
Department of Computer Science,  
University of Sheffield, U.K.

## Abstract

A Neural Net that generalises to previously unseen examples, at a level of about 95% sounds impressive unless it forms part of a real-life application. In such cases, a greater level of reliability would be required. N-version programming is a popular technique for increasing reliability in software programs; the idea being that if programs are independently developed they will fail independently, and that independent N-versions, in combination with a voter, will be more likely to produce a correct output than a single program. It has been argued (Knight and Leveson, 1986) that true independence is unlikely to be achieved; and that the aim should be one of promoting methodological *diversity*, with the aim of finding negatively correlated methodologies (Littlewood and Miller, 1989). Our aim, in this paper, was to apply the concept of diversity to Neural Nets, and to conduct an investigation to examine the relative merits of different potential methods for creating diversity. The data we employed for this purpose was simulated data from a ship's engine, generated for the purpose of fault diagnosis. The methods investigated were (i) varying the initial conditions of nets (ii) varying the training sets (iii) using contrasting measures. Our experiments suggest the third method is the most appropriate, since it resulted in the lowest correlations between the failures of different nets.

## 1 Introduction

The issue of safety-critical applications and the need for reliability has not yet received much attention in the Neural Net literature. Similarly, little attention has been paid to methods of testing, although there is an extensive literature on the subject in software engineering. Instead, Neural Net researchers have usually been content to divide their data into a training set and a test set, and to accept a measure of correct generalisation to the test set as the main indicator of a net's performance; where a performance measure of 95% correct generalisation to previously unseen inputs would be considered impressive. However, as Neural Nets begin to be used in real applications, issues of reliability become more important. It is easy to imagine situations where a generalisation level of 95% would not be adequate. A neural net controlled wing flap adjuster on an aircraft, that was only

---

<sup>o</sup>We would like to thank the EPSRC/DTI Grant No.GR/H85427 IED4/1/9301 for funding this research.

accurate 95% of the time, would be of little use. In the software engineering literature there has been extensive discussion of methods of improving the reliability of software programs. In this paper, we shall consider the software concept of *diversity*, in the context of research on Neural Nets. Our aim will be both to look at the application of a software engineering concept to Neural Nets, and to consider the potential contribution that Neural Nets might make to this area of software engineering.

In software engineering, the concept of diversity originates from investigations of N-version programming. N-version programming is employed with the aim of increasing the fault-tolerance of conventional programs; the traditional approach being the independent development of alternative versions of a piece of software (Knight and Leveson, 1986). These versions can then be executed in parallel, each receiving identical inputs and producing the required output. The outputs can then be collected by a voter, with the majority vote being chosen in the case of disagreement. N-version programming is made use of in real systems. For example, the Airbus Industry A310 aircraft makes use of dual programming in the slat and flap control system (Martin, 1983). Similarly, Taylor (1981) describes the application of dual programming to point switching, signal control and traffic control in the Gothenburg area by Swedish State Railways.

The assumption was that independently developed N different versions would fail indendently, and therefore increase reliability (Dahll and Lahti, 1980). However, it has become clear that this is not necessarily the case. In an important paper in the area, Knight and Leveson (1986) present evidence that, even when working independently, people tend to make the same mistakes when solving a difficult intellectual problem. This means that even when programs are written by different people, when they do fail they do not fail independently; thus if one program fails on a particular input the probability of other programs also failing on that input are increased. Eckhardt and Lee (1985) developed this idea and provided a formal definition of independence of versions, showing that even in the unlikely event of achieving independence, versions would still not show independent behaviour. According to Littlewood and Miller (1989), the important idea is that of *diversity* rather than independence. They introduce the idea of promoting diversity through the employment of different methodologies, and argue that it is possible to do better than achieving independence of failures. That is, to achieve a situation in which there is a low, or even *negative* correlation between the failures of the two methodologies, such that if a particular input fails on several programs from one methodology, that input succeeds in programs from the other methodology. They also argue that greater diversity of design results in a greater chance of freedom from coincident failures. Their argument is reinforced by the empirical results presented by Adams and Taha (1992). Littlewood and Miller (1989) define the degree of methodological diversity in terms of the size and the

sign of the (product moment) correlation between probability of failures for two contrasting methodologies. They make the point that reliability can be increased not only by employing contrasting methodologies, but also by creating several versions within a methodology.

The question to be investigated in this paper is to look at the best ways of creating diversity in a Neural Net implementation of a solution to a particular problem. The problem to be considered in this case is that of Fault Diagnosis in a ship's engine, (Gopinath, 1994), but the results are intended to have a more general relevance (related investigations, using a different source of data, are reported by Sharkey and Partridge, 1992). The methods which have previously been used to decrease the number of coincident failures in standard symbolic programs have included the following; (a) working from different specifications (Ramamoorthy et al, 1981); (b) using different programmers (Knight and Leveson, 1986); and (c) using different types of programming language (procedural versus logic programming), (Adams and Taha, 1992). The differences between Neural Nets and standard programs mean that different manipulanda are more likely candidates for producing diversity between net solutions. The manipulanda to be investigated here are (i) initial conditions; (ii) training sets and (iii) contrasting measures.

In our investigations of diversity in Neural Nets, our aim is not simply to create nets that are diverse in the sense that their failures are negatively correlated; we also need them to be reasonably accurate, and to cover the function. This point can be made clearer if we look at an example of contrasting methodologies which *are* negatively correlated, but which do *not* provide good coverage of the function. In Figure 1, the probability of failing (number of inputs to fail on that number of versions) in one methodology is plotted against the probability of failing in a different methodology. The results on which this plot is based were taken from a different study, not reported here, in which nets were trained on a simple function. The nature of the function is irrelevant for our present purposes; the illustration is included because it provides an example of a negative correlation accompanied by a high incidence of coincident failures. The two methodologies plotted in this study correspond to one set of nets trained on the function (Methodology 1), and a second set not trained on any function (Methodology 2). Although the failures in the two methodologies were negatively correlated ( $\rho = -0.6381$ ), examination of Figure 1 should make it clear that the combined performance of the two nets would be poor, due to the lack of combined successes, and the presence of inputs which fail on all versions of both methodologies.

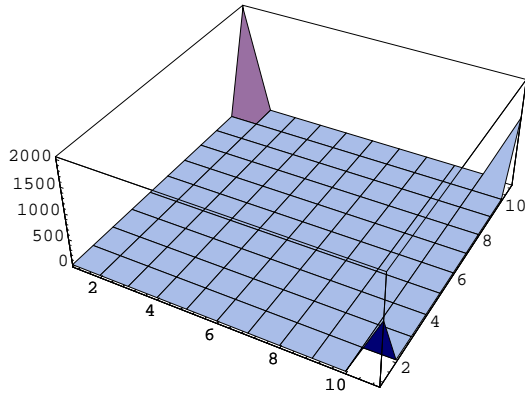


Figure 1: Three dimensional plot of the probability of failure of one methodology against another. The  $x$  axis shows the number of versions on which failures occurred in Methodology 1, and the  $y$  axis shows the number of versions on which failures occurred in Methodology 2. The height of the plot represents the number of failures. This plot represents a high negative correlation (the result of plotting trained nets against untrained nets), but there are still a large number of inputs which fail on 10/10 versions in both methodologies.

By contrast, Figure 2 shows an idealised result, where two methodologies are negatively correlated, but the majority of inputs are correct on both methodologies. In this example, where failures occur in one methodology, they have a zero chance of failing on the other methodology. Thus, if the two methodologies were combined, there is an increased chance of a correct response being produced by one of the two methodologies.

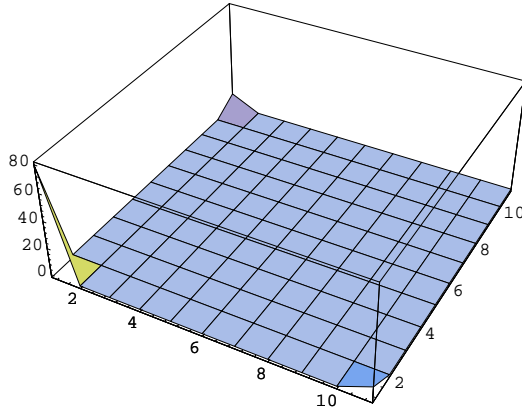


Figure 2: Constructed example of an idealised pattern of negative correlation. There are no inputs that fail on all ten versions of both methodologies; of the 100 inputs, 80 are correct on all versions of both methodologies, and 10 fail on 10/10 versions of one methodology whilst being correct on the other, whilst a different 10 inputs fail on 10/10 versions on the other methodology whilst being correct on the first.

## 2 Data

The data to be investigated were generated through the use of the MERLIN<sup>1</sup> simulator of engine data, which provides a close approximation to real engine data.<sup>2</sup> Two fault types were simulated: (i) Retarded injection of fuel and (ii) Advanced injection of fuel. Retarded fuel injection means that fuel is injected into the cylinder later than normal, causing 'after-burning' whereby complete combustion of the injected fuel does not occur. Unburnt fuel escapes from the cylinder and reduces turbocharger efficiency, which in turn makes less air available for combustion, which causes more unburnt fuel to leave the cylinders. Advanced fuel injection means that the fuel is injected too early, combustion occurs earlier, and high firing pressures result, with consequent stress and power imbalance. In this paper we shall consider two measures which indicate combustion quality. The first is the Pressure in the cylinder; the second, more indirect measure, is the Temperature of the gas inside the cylinder at the corresponding point

<sup>1</sup>Engine simulation software developed by Lloyds Register of Shipping, London, made accessible through the assistance of Dr K. Banisoleiman.

<sup>2</sup>The data were generated from the MERLIN simulator by O.C. Gopinath. This particular method of generating faulty data for Neural Net training for the purpose of fault diagnosis of a ship's engine is his idea, and forms the basis of his MSc thesis (Gopinath, 1994).

that the pressure was measured.

MERLIN was used to produce 588 examples of each of the two faults, together with 588 examples of ideal combustion, making a total of three classes. This data set was divided, on the basis of random sampling, into a test set of 414 examples (138 from each class), and 9 training sets of 150 examples (50 from each class). The initial set of inputs generated by MERLIN consisted of 1440 values for both Pressure and Temperature. These were sampled, whilst retaining the points of interest, such that each Pressure example consisted of 52 Pressure input points, and each Temperature example consisted of 71 input points. The data were normalised such that their values fell between 0 and 1. The three output classifications (Retarded Injection, Advanced Injection and Ideal combustion) were indicated in the data as outputs of 010, 001, and 100 respectively.

### 3 Experiments and Results

A standard feedforward architecture was used and nets were trained, using backpropagation, to classify input data (either Pressure, or Temperature) as either an instance of Ideal combustion, Retarded Injection or Advanced Injection. A total of 81 nets were trained on Pressure inputs; using 9 different initial random seeds for 9 different training sets each containing 150 examples. The 52-2-3 nets were trained to a tolerance of 0.1, using a learning rate of 0.6 and momentum factor of 0.2. A further 81 nets, with a 71-2-3 architecture, were trained in the same manner on corresponding Temperature inputs. In the first two result sections which follow only the Pressure data will be reported. They are not reported here, but the same experiments were also carried out based on the Temperature data, and essentially the same pattern of results was obtained.

We shall employ two methods of illustrating the results we obtained; (i) tables of the correlations between failures in contrasting methodologies and (ii) 3-dimensional plots of the probability of failing on a pair of methodologies. In both cases, the failures are the failures of trained nets tested on the previously unseen test set (the average generalisation for pressure was 98.04%, and for temperature was 98.15%). As mentioned earlier, three different methods for creating diverse solutions were explored; these being the manipulation of (i) Initial conditions; (ii) Training set and (iii) Alternative measures.

*Initial conditions: Pressure data* The effect of variations in Initial conditions on Diversity was investigated by treating each Initial condition as a Methodology, and each training set trained with that Initial condition as a version within that methodology. Using the statistical methods of Littlewood and Miller (1989), it was then possible to calculate the correlation between the failures of pairs of methodologies, testing across several versions. In Table 1, it is possible to see the correlations between nets trained

	1	2	3	4	5	6	7	8	9
1		0.971	0.948	0.974	0.943	0.959	0.963	0.963	0.945
2			0.971	0.969	0.963	0.979	0.971	0.972	0.951
3				0.969	0.951	0.959	0.972	0.975	0.974
4					0.943	0.963	0.983	0.985	0.976
5						0.972	0.950	0.936	0.927
6							0.978	0.960	0.954
7								0.990	0.977
8									0.975
9									

Table 1: Correlations between Random Initial Conditions methodologies

in different Methodologies based on different initial conditions. Each entry in the table corresponds to the correlation between two methodologies (two different random seeds), where each methodology consists of 9 different versions, or training sets. As is apparent from the table, there is little evidence here of any resulting diversity. The failures between the methodologies are all highly correlated. Figure 3 shows a plot of one of these comparisons (between Random Seed 1, and Random seed 2). As should be apparent from the figure, as the probability of failing on one methodology increases, so does the probability of failing on the other.

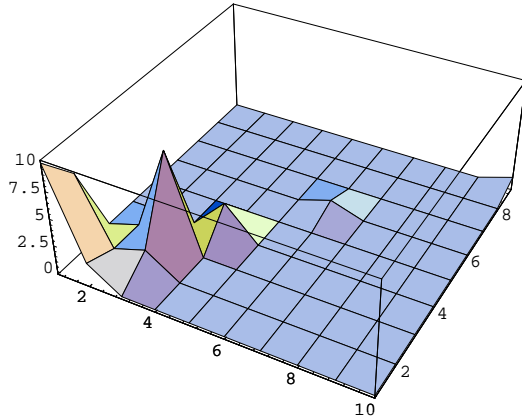


Figure 3: Number of inputs failing on two methodologies, where each are based on a different random seed. (To increase the visibility of the results, the number of failures was truncated at 10 from 384). It can be seen that as probability of failing on one methodology increases so does the probability of failing on the other methodology.

*Training sets: Pressure data* The effect of variations in training sets on Diversity was investigated by treating each training set as a methodology, and each net trained on that set from a different random seed, as a version within that methodology. In Table 2, the correlations between nets trained on different training sets are shown. Each entry in the table corresponds to the correlation between one training set methodology, and another training set methodology. Examination of the table shows that the correlations here are more variable, and often lower than in the case where the Initial Conditions were treated as the methodology. Nonetheless, the level of diversity achieved is still not ideal, as can be seen by the illustration provided in Figure 4. Here it is evident that even in a case where the correlation was relatively low, there is still one input which fails on all versions in both methodologies, and other inputs which fail on several versions in both methodologies.

	1	2	3	4	5	6	7	8	9
1		0.268	0.319	0.675	0.536	0.368	0.467	0.446	0.760
2			0.832	0.477	0.684	0.173	0.705	0.727	0.499
3				0.567	0.563	0.196	0.902	0.486	0.608
4					0.629	0.262	0.574	0.594	0.871
5						0.626	0.551	0.733	0.703
6							0.250	0.453	0.314
7								0.465	0.650
8									0.649
9									

Table 2: Correlations between Training set methodologies

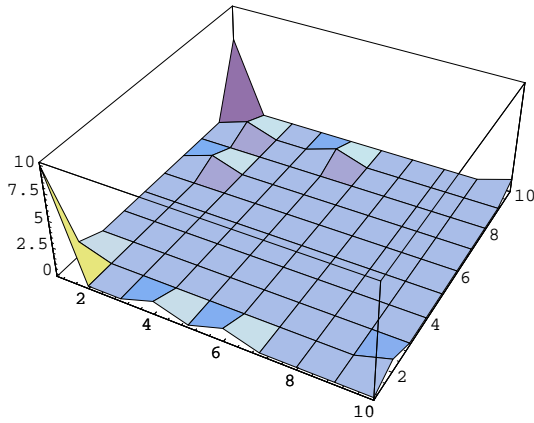


Figure 4: Number of inputs failing on two methodologies, based on two different training sets (Set1 vs Set2). To aid this visualisation, the number of shared successes was truncated at 10, from 399. The correlation between these methodologies was 0.2676, but there is one output which fails on all versions in both methodologies, and some inputs which fail on several versions in both methodologies.

*Alternative Measures* Our final manipulation was to look at the extent to which the failures of nets trained to classify faults based on pressure data, would correlate with the failures of nets trained to classify faults based on temperature data. Here corresponding training sets, based on either Pressure or Temperature data, were treated as methodologies, with different random seed versions being treated as versions within that methodology. In both cases the inputs were based on the same engine cycle, but they represent alternative measures which can be taken from the engine; thus

	1	2	3	4	5	6	7	8	9
1	0.079								
2		0.360							
3			0.571						
4				0.109					
5					0.218				
6						0.596			
7							0.497		
8								0.223	
9									0.055

Table 3: Correlations between Temperature and Pressure methodologies.

each entry represents a comparison between two methodologies where both used a corresponding training set. Examination of Table 3 shows that they are all positive, but vary from a positive correlation of 0.0550 to a correlation of 0.5957. Thus, the correlations achieved are lower than those found as a result of the other two methods. The lowest correlation achieved as a result of varying the initial conditions (Table 1) was 0.9273, and as a result of varying the training sets (Table 2) was 0.1732. When the correlation between the ninth Pressure training set, and the corresponding ninth Temperature training set is examined more closely in Figure 5, it is apparent that the distribution of the failures is close to the ideal, in that there are no inputs which fail on all versions in both methodologies, and that as the probability of failing on one methodology increases, the probability of failing on the other remains close to zero.

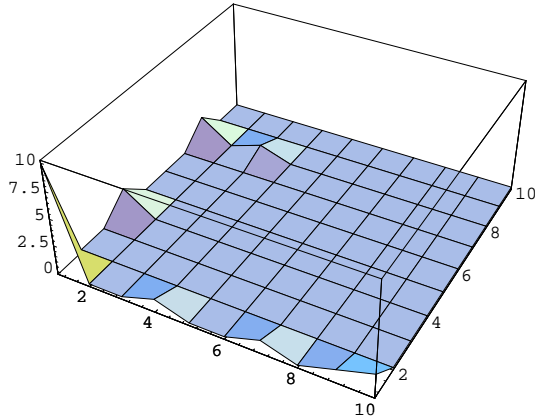


Figure 5: Number of failures per version for Pressure methodology, plotted against number of failures for Temperature methodology. Actual number of shared successes was 404/414, but that number is truncated to 10 for this diagram. The two methodologies show a low level of correlation ( $\rho = 0.0550$ ). As can be seen, as the probability of failing on one methodology increases, the probability of failing on the other methodology stays close to zero.

In summary, the results reported here indicate that of the three methods investigated, it is the use of contrasting measures which provides the greatest diversity of failures. Although the negative correlations discussed by Littlewood and Miller (1989) were not in evidence, some of the comparisons are close to the ideal situation, in which failures in one methodology are backed up by successes in the other methodology.

## 4 Conclusions

The results reported here have implications for both Neural Net research, and for Software Engineering. In terms of Software Engineering they demonstrate the general applicability of the notion of diversity. The possibility has been raised (Partridge and Sharkey, 1994) that Neural Net solutions could themselves be treated as a diverse methodology in combination with standard programming solutions. Our results show that it is possible also to create methodological diversity *within* the Neural Net paradigm, thus offering a potential increase in reliability, and showing the relevance of the notion of diversity to 'network programming'. It remains to be seen whether the notion of promoting diversity through reliance on different sources of data could also be usefully fed back into conventional Software Engineering practice.

In terms of Neural Net research, it is clear that the best means of producing diverse methodologies is to make use of alternative measures. In our experiments, variations in initial conditions whilst holding training sets constant, had little effect on diversity; variations in training sets reduced the positive correlations between failures, but varying the measure on which the training sets were based resulted in the lowest correlations between the failures. The implication here is that, where possible, it would be better to train different nets on the basis of data from different sensors, than to attempt to create diversity by varying training sets based on one sensor, or varying the initial conditions of one training set. Of course, the strategy of developing nets which fail diversely, implies a need for the deployment of voters to arbitrate between nets in the case of disagreement. The topic of voting has also received extensive consideration in the software engineering literature, (eg Avizienis, 1985; Voges, 1988), and we are currently looking at ways in which voters can be modified and combined with diverse neural net solutions to form more reliable neural net systems.

## 5 References

Adams, J.M. and Taha, A. (1992) "An Experiment in Software Redundancy with Diverse Methodologies," *Proc of the Twenty-Fifth Hawaii International Conference on Systems Sciences*.

Avizienis, A. "The N-Version approach to fault tolerant software" *IEEE Transactions on Software Engineering*. Vol 11, 12, pp 1491-1501.

Dahll, G. and Lahti, J. (1980) "An investigation of methods for production and verification of highly reliable software", In L. Lauber (ed) *Safety of Computer Control Systems (Proc. SAFECOMP'79)*, New York: Pergamon.

Eckhardt, D.E. and Lee, L.D. (1985) "A theoretical basis for the analysis of multiversion software subject to coincident errors". *IEEE Trans. Software Eng.*, vol SE-11, no 12, pp 1511-1517.

Gopinath, O.C. (1994) "A neural net solution for diesel engine fault diagnosis", MSc thesis, University of Sheffield.

Knight, J.C. and Leveson, N.G. (1986) An experimental evaluation of independence in multiversion programming. *Trans on Software Eng*, Vol SE-12 no 1.

Littlewood, B. and Miller, D.R. (1989) Conceptual modeling of coincident failures in multiversion software. *IEEE Trans. on Software Engineering*, 15,(12).

Martin, D.J. (1983) Dissimilar software in high integrity applications in flight controls," In *Software for Avionics* (AGARD Conf. Proc. 330), Jan, 1983, pp36-1-36-9.

Partridge, D. and Sharkey, N.E. (1994) Neural computing for software reliability. *Expert Systems*, 11, 3, 167-175.

Ramamoorthy, C.V., Mok, Y.R., Bastani, E.B., Chin, G.H. and Suzuki, K. (1981) "Application of a methodology for the development and validation of reliable process control software" *IEEE Trans. Software Eng.*, vol SE-7, pp 537-555.

Sharkey, A.J.C. and Sharkey, N.E. (in press) Cognitive Modelling: Psychology and Connectionism. In (Ed.) M.A. Arbib *The Handbook of Brain Theory and Neural Networks*, Bradford Books/MIT Press.

Sharkey, N.E. and Partridge, D.P. (1992) The statistical independence of network generalisation: an application in software engineering. In P.G. Lisboa & M.J. Taylor (Eds) *Neural Networks: Techniques and Applications* Chichester, UK: Ellis Horwood.

Sharkey, N.E. and Sharkey, A.J.C. (1994) Understanding Catastrophic Interference in Neural Nets. University of Sheffield, Department of Computer Science Research Report CS-94-4

Taylor, J.R. (1981) "Letters from the editor", *ACM Software Eng Notes*, vol 6, no1, pp 1-2.

Voges, U., (1988) (ed). "Software Diversity in Computerised Control Systems", Springer-Verlag.

White, H. (1992) *Artificial Neural Networks: Approximation and Learning Theory*. Blackwell: Oxford, UK