Article:
Sharkey, A.J.C. and Sharkey, N.E. (1997) Combining diverse neural nets. The Knowledge Engineering Review, 12 (3). pp. 231-247. ISSN 0269-8889

https://doi.org/10.1017/S0269888997003123

The Knowledge Engineering Review, Vol. 12:3, 1997, 231–247

Combining diverse neural nets

AMANDA J. C. SHARKEY and NOEL E. SHARKEY

Department of Computer Science, University of Sheffield, UK

Abstract

An appropriate use of neural computing techniques is to apply them to problems such as condition monitoring, fault diagnosis, control and sensing, where conventional solutions can be hard to obtain. However, when neural computing techniques are used, it is important that they are employed so as to maximise their performance, and improve their reliability. Their performance is typically assessed in terms of their ability to generalise to a previously unseen test set, although unless the training set is very carefully chosen, 100% accuracy is rarely achieved. Improved performance can result when sets of neural nets are combined in ensembles and ensembles can be viewed as an example of the reliability through redundancy approach that is recommended for conventional software and hardware in safety-critical or safety-related applications. Although there has been recent interest in the use of neural net ensembles, such techniques have yet to be applied to the tasks of condition monitoring and fault diagnosis. In this paper, we focus on the benefits of techniques which promote *diversity* amongst the members of an ensemble, such that there is a minimum number of coincident failures. The concept of ensemble diversity is considered in some detail, and a hierarchy of four levels of diversity is presented. This hierarchy is then used in the description of the application of ensemble-based techniques to the case study of fault diagnosis of a diesel engine.

1 Introduction

There are a number of reasons for supposing that neural computing techniques might be usefully employed on safety-critical, or safety-related projects. For certain tasks, neural computing can provide an attractive alternative to conventional software engineering systems, or mathematical programming solutions. Neural nets are often easy to implement, efficient, and once trained, fast to execute (especially in parallel hardware). Multi-layer feedforward networks have been shown to be universal approximators (White *et al.*, 1992). That is, they can approximate any function given a sufficiently representative sample of the input-output pairs of the function (and sufficient complexity).

Neural computing techniques are not a panacea; their effective use depends on careful selection of the tasks and problems to which they are applied. When a task can readily be accomplished by means of conventional code, a neural net is unlikely to result in better performance. Consider, for instance, the simple example of a function for deciding whether the length of a line described by a set of coordinates is greater than a reference length. This may be accurately determined using a simple program but requires the generation, selection and combination of many nets to achieve anything approaching comparable performance (Partridge and Yates, 1996).

The type of tasks for which neural nets provide an attractive option are to be found when the data is noisy, where explicit knowledge of the task is not available, when execution speed is required and when the task could change in time. Such situations are often found in areas such as control, diagnosis and sensing, since physical data are usually noisy and incomplete, and a precise program

specification in these areas can be difficult to obtain. It is possible to envisage a useful, if subsidiary, role for neural nets in the safety-critical industry, in which they are employed for continuous on-line monitoring of individual components such as fans, engine combustion quality, or cooling systems. And in fact there has been an increasing amount of work using neural nets for tasks such as fault diagnosis (e.g., Boek, 1991; Duyar and Merrill, 1992), condition monitoring (e.g., MacIntyre *et al.*, 1993), and more recently, fault detection (e.g., Marko *et al.*, 1996; Worden, 1997).

If neural nets are to employed in safety-critical, or safety-related areas, it would make sense to adopt some of the standard procedures for reducing the number of errors, or at least mitigating their effect. One of the standard procedures applied to software relies on the concept of reliability through redundancy. This concept is one that can be usefully applied to neural computing. Reliability through redundancy, and diversity, are two approaches which are referenced in existing safety standards, as discussed by Croll *et al.* (1995). Thus the international IEC65A standard, which is specifically aimed at 'Software for Computers in the Application of Industrial Safety-Related Systems', identifies a number of techniques to deal with faults, one of which is that of N-version programming. Similarly, the Health and Safety Executive (UK) guidelines for designing Safety Related Computer Control Systems advocate the adoption of both diversity and redundancy based techniques to guard against failures.

The basic idea of N-version programming, is to produce N versions of a program such that the versions fail independently (Littlewood and Miller, 1989). These can then be combined by means of a majority vote to produce a more reliable system. This idea of diversity of failure can be used to improve the performance of neural nets. Diverse nets can be combined to produce a more reliable output.

The idea of combining nets to improve performance is not new; (e.g., Breiman, 1992, 1994; Drucker *et al.*, 1994; Hansen and Salamon, 1990; Perrone and Cooper, 1993; Sharkey *et al.*, 1996; Tumer and Ghosh, 1996; Wolpert, 1992). See Sharkey (1996) for a review of this work. However, although these researchers consider the combining of nets for the purposes of improved performance, they do not address the idea from the perspective of software engineering, and N-version programming. Sharkey and Partridge (1994) also present an account of combining neural nets in terms of software engineering and diversity, but they do not discuss the relationship to other neural net research on ensemble combination. Similarly, although neural nets have been applied to fault diagnosis and condition monitoring problems, previous work in these areas has not, to our knowledge, made use of the technique of combining diverse nets in order to improve performance. This paper discusses the benefits of putting together these three areas: (a) combining nets, (b) reliability through redundancy, and (c) condition monitoring and fault diagnosis.

In what follows, we shall examine in more detail the way in which the concepts of diversity, and of reliability through redundancy can be applied to neural nets, and in particular to the problem of fault diagnosis. We shall begin with the notion of *diversity*; and then turn to a consideration of the methods which can be employed to create a set of redundant nets which fail diversely. We shall illustrate the utility of some of these methods by describing some recent research in our laboratory in which combinations, or *ensembles* of nets have been used to improve the performance of a fault diagnosis system for a diesel engine.

2 Diversity

The notion of network combination for the formation of more reliable ensembles can be traced back to Nilsson (1965), and the notion of combining evidence is evident in a variety of fields (e.g., econometrics and forecast combining; Granger, 1989). The advantage of combining nets, is that of avoiding the loss of information that might result if the best performing net of a set of several were selected and the rest discarded. This idea of exploiting, rather than losing, the information contained in imperfect estimators, is central to the notion of combining neural nets for improved performance. Much of the work in this area has focussed on establishing appropriate methods by which sets of nets can be combined. Methods of combining, or merging the outputs of multiple nets include:

ensemble averaging (Lincoln and Skrzypek, 1990; Breiman, 1994; Hansen and Salamon, 1990; Perrone and Cooper, 1993), weighted averaging (Hashem and Schmeiser, 1993), majority voting, (Sharkey *et al.*, 1996), stacked generalisation (Wolpert, 1992) merging regression predictors (Breiman, 1992), and combining beliefs in the Dempster–Shafer sense (Rogova, 1994; Xu et al, 1992).

A complementary alternative to concentrating on the means by which the members of an ensemble are to be combined, is to look at the composition of the nets included in an ensemble, and to attempt to come up with a set of nets which can be effectively combined. It is clear that, to state the obvious, there are no advantages to combining nets which exhibit identical generalisation. In much of the work on combining nets, sets of nets have been generated through the blind application of a particular method (e.g., varying the initial weight settings, the topology of the net, or the content of the training set). The contrasting approach relies on active attempts to generate sets of nets that can be effectively combined in an ensemble. What is needed for effective combination is a set of nets, each of which generalises well and makes only a small amount of errors. However, where errors are made, it is important that they are not shared by all the nets in the set or ensemble. This idea can be expressed in terms of *diversity*.

The term "diversity" has origins in the software engineering literature (e.g. Littlewood and Miller 1989). The aim is to increase the reliability of conventionally programmed solutions by combining programs which failed independently, or whose failures were uncorrelated. The point is to combine programs which when they do fail, fail on different inputs. The same idea can be applied to ensembles of neural nets so that failures on one net can be compensated by successes on others. When applied to neural computing, the concept of diversity is inextricably linked to that of generalisation. Therefore in our examination of the relevance of the concept of diversity to neural computing, we shall focus on the notion of generalisation.

2.1 Generalisation and diversity

Generalisation is the term used to refer to the ability of neural nets to produce a correct response when tested on inputs it has not been trained on. An account of generalisation in neural nets, or more particularly, Multi-Layer Perceptrons (MLPs), relies on the distinction between training and testing. An example of a multi-layer net consisting of a layer of input units, a layer of hidden units, and a layer of output units is shown in Figure 1. There are weighted connections between the input units, and the hidden units, and between the hidden units and the output units where the weight associated with the connection indicates its strength. Such nets can be trained to map a set of inputs onto a set of outputs, where the inputs and outputs are either binary or continuously valued representation vectors. This mapping may be arbitrary, between randomly associated pairs, or it may be functional, i.e., between sets of ordered pairs.

When nets are trained using the backpropogation learning rule (Rumelhart *et al.*, 1986), the mapping between an input and an output occurs in two time steps. At time t_1 the input state vector, $\mathbf{v}$ is translated into hidden unit space as a vector, $\mathbf{h}$, by an update function $f(\mathbf{v}) = \mathbf{h}$. Standardly, the update function for the j^{th} hidden unit is $h_j = 1/1 + e^{-x}$, where x is the weighted sum of the inputs to the unit. At time step t_2, the hidden unit states are propagated to the outputs by the same update function $f(\mathbf{h}) = \mathbf{o}$.

During *training*, the actual output is compared to the required output, or target, and the error is backpropogated through the network such that the weighted connections between all the units are adjusted in the right direction. These steps may be repeated until the correct output is produced (within a certain error tolerance) in response to each input. During *testing*, the weights are no longer adjusted and the performance of the net can be tested by presenting new inputs, and observing the output.

When a net is tested on inputs, an output will be produced that reflects the way in which the weights have been set up as a consequence of training. Where the novel input is sufficiently similar to the inputs on which the net was trained, the output produced by the net can be correct (although it

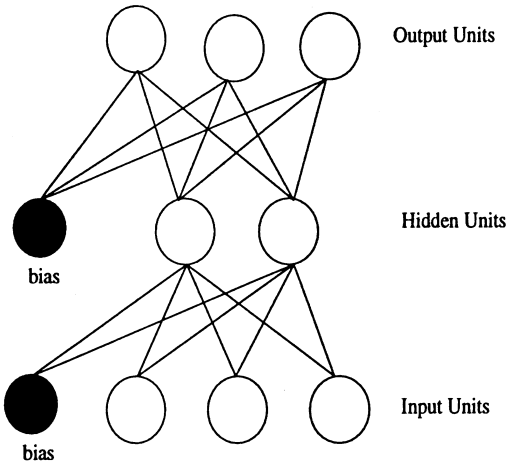

Figure 1 A feedforward net with two weight layers and three sets of units

should be noted that neural nets are good at interpolation, rather than extrapolation). This ability to generalise can be seen as both an advantage and a disadvantage. It is an advantage in that it makes it possible to achieve good performance on a problem for which the total set of ordered pairs that defines the function is not available for training (either because it is not known, or because it is too large, and possibly unbounded). However, it can also be seen as a disadvantage since the generalisation performance of nets is rarely perfect. It is for this reason, that the concepts of reliability through redundancy, and of diversity can be so usefully applied to neural computing.

The idea of generalisation is further illustrated in Figure 2 which consists of Venn diagrams for ordered pairs corresponding to a target function and for alternative functions that are also compatible with the training set.[1] In these diagrams, the target or desired function is shown as a closed circle, where S represents a training sample and T represents a test set. A net trained on a training sample of the ordered pairs which constitute the target function can be tested on another sample of the ordered pairs, T, as shown in Figure 2(a).

The reason why neural nets are imperfect estimators (i.e., why they often correctly generalise to less than 100% of the test set) is that, as shown in Figure 2(b), the sample on which a net is trained may be compatible with an alternative function such as that represented by the set of ordered pairs labelled P' (see Denker *et al.*, 1987; Sharkey and Sharkey, 1995c, for similar accounts). Another way of putting this is to say that the "guess" made about the target function on the basis of the training sample is not quite correct, and results in the induction of an alternative function.

Since the generalisation performance of a net can only be determined with respect to test sets it is important that efforts are made to develop test sets that are representative of the target function, and which do not contain any of the input-output pairs used for training (or the level of correct generalisation will be artificially inflated). The usual way in which a test set is constructed is by taking a random sample of the set of possible input-output pairs that constitute the target function, having first excluded those input-output pairs used for training.

In the same way that generalisation performance is determined with reference to a test set, diversity is also defined with respect to a particular test set. A pair of nets can be said to be diverse with respect to a test set if they make different generalisation errors on that test set, and not to be diverse when they show the same pattern of errors.[2] Thus, in Figure 2(c) the nets that induced the

[1] A common misunderstanding of these Venn diagrams is to think of them as representing an ordered input space. They actually represent an unordered set of ordered pairs, i.e., the set of of ordered pairs that constitutes the target function. Thus the position of the S or T in Figure 2(a) does not indicate anything about their distribution in input space.

[2] Here we assume that a particular output is in error if it is not within a specified tolerance level of the target output.

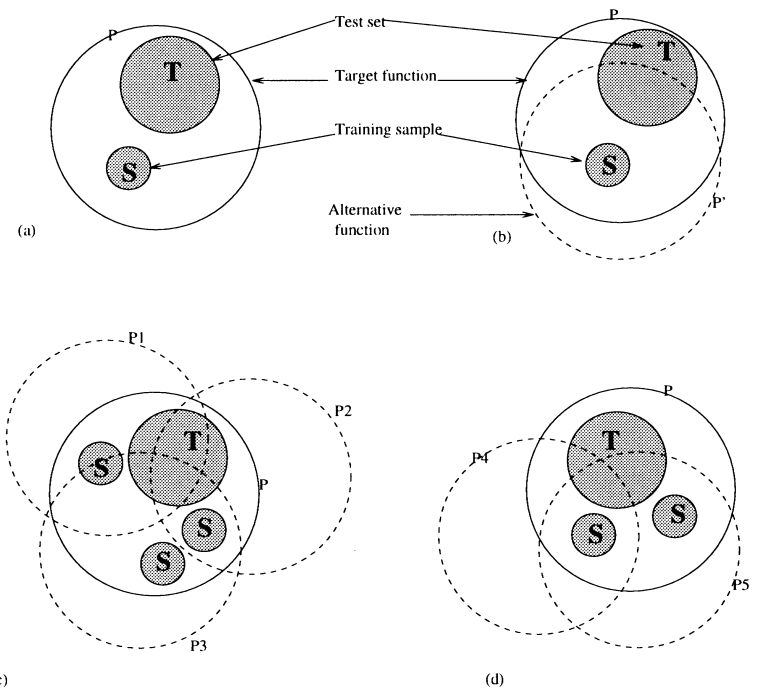

Figure 2 Training and testing neural nets. In these Venn diagrams closed circles are used to represent the set of ordered pairs (input/output) for the target function. The sets of ordered pairs corresponding to alternative functions (P', $P1$, $P2$, $P3$, $P4$, $P5$) are also shown as dashed circles. In (a), the set of ordered pairs defining the target function is shown, together with ordered pairs that make up the training sample, S, and the test sample, T. In (b) the dashed circle represents a set of ordered pairs corresponding to a function other than the target function, one that is also compatible with the training set S. In (c) and (d) more than one training set is shown. Again, the closed circle represents the set of ordered pairs for the derived function, and the dashed circles represent sets of ordered pairs corresponding to alternative functions which are compatible with (ie contain) a training set S.

alternative functions $P1$, $P2$ and $P3$ can be described as diverse. What matters here is the distribution of failures. Two nets might have taken different numbers of cycles to train, and might consist of different sets of weights, but if they show the same patterns of generalisation and both fail on the same inputs in the test set, they represent the same solution, and can be said not to exhibit any diversity.

Sets of nets can be combined to exploit the diversity they exhibit. For example, if the individual members of a set of nets have each induced a function that differs from the target function, such as leads to the situation represented by the intersecting dashed circles in Figure 2(c) and 2(d), then there is something to be gained from their combination. In Figure 2(c), it can be seen that between them, the dashed circles corresponding to the sets of ordered pairs, $P1$, $P2$ and $P3$ cover the test sample (labelled T). Therefore, an alternative to choosing the net which shows the best performance on the test set is to use a combination of the nets which have induced alternative functions. By combining the nets appropriately it might be possible to construct an ensemble which exhibits 100% generalisation on the test set. This would represent an improvement over the selection of any of the individual nets since none of them will produce a correct response for all the elements in the test set. In Figure 2(d) there is an area of the test set which is not included in either $P4$ or $P5$. Even here, it is apparent that better generalisation on the test set will be achieved if the nets that have induced the alternative functions corresponding to the sets of ordered pairs $P4$ and $P5$ are combined, than if only $P4$ were used.

2.2 Levels of diversity

We have found it useful to go a step further than characterising sets or pairs of nets as diverse or not,

and to talk about the *level* of diversity they exhibit[3]. It is possible to identify four levels of diversity which a set of artificial neural networks (ANNs) can exhibit with respect to a test set, ranging from the ideals of Level 1 and Level 2 Diversity to the minimum diversity of Level 4. An advantage of this hierarchy is that the level of diversity exhibited by a set of nets also gives a good indication of the best way of combining them.

Level 1 Diversity: there are no coincident failures, and the function is covered. By no coincident failures, we mean that there were no inputs that resulted in failure for more than one net. By function coverage, we mean that for every input in the test set there is always a net that produces the correct output. When the outputs of the ensemble are combined by means of a majority vote, the correct output will always be produced, since only one net will ever fail on each tested input. Level 1 Diversity is an ideal to aim for in ensemble performance on a test set. This level requires $n > 2$ (where n is the number of nets in the ensemble).

Level 2 Diversity: there are some coincident failures, but the majority is always correct, and the function is covered. Level 2 Diversity does not meet all of the criteria of Level 1, since it does allow some coincident failures. However in ensembles that exhibit Level 2 Diversity, the majority is always correct. Thus, although a particular input pattern might result in an error on more than one net, the number of correct outputs for that input pattern from ensemble members will always be greater than the number of errors. Therefore 100% performance on the test set will still be achieved. This level requires $n > 4$. ($n > 4$ is required to make it possible for the correct response to be in the majority even when two nets fail). It is possible that a Level 2 system is *upwardly mobile* in that removing some of the ANNs could eliminate the coincident failures and lead to a Level 1.

Level 3 Diversity: a simple majority vote will not always result in the correct answer, but the nets in the ensemble do cover the function such that the correct output for each input pattern in the test set is always produced by at least one net (see Figure 2(c)). This means that it might be possible to weight the outputs of the nets in such a way that either the correct answer was always obtained, or at least that the correct output was obtained often enough that generalisation was improved. Some of the more complex methods of combining ensembles might be appropriate (e.g., weighted averaging (Perrone and Cooper, 1993) or stacked generalisation (Wolpert, 1992)). It is possible that a set of ANNs that exhibits Level 3 Diversity may contain subsets of ANNs with either Level 1 or Level 2 Diversity and thus, like Level 2, Level 3 may be upwardly mobile.

Level 4 Diversity: the function is not entirely covered by the ANNs. Level 4 Diversity can never be reliable since there are failures that are shared by all of the ANNs (see Figure 2 (d)). However, although Level 4 can never be upwardly mobile, it can still be used to improve generalisation, provided that it fulfils the criteria for minimal diversity that there should be at least two ANNs in an ensemble such that there is at least one correct input generalisation in each that is not shared by the other. This level is equivalent to the minimal diversity, or "useful diversity" identified by Partridge and Griffith (1995).

To summarize; the level of diversity achieved by an ensemble can be only estimated through an examination of the coincident failures on a test set. If there are no coincident failures, the ensemble exhibits an estimated Level 1 Diversity with respect to that test set; if there are some coincident failures, but the majority is always correct, the ensemble exhibits an estimated Level 2 Diversity; and if there are coincident failures, and the majority is not always correct, the ensemble exhibits either an estimated Level 3 or Level 4 diversity depending on whether or not there are some inputs which fail on all the members of the ensemble.

The concept of diversity employed in the definitions of these levels is related to the ideas expressed by Krogh and Vedelsby (1995) in their discussion of *ensemble ambiguity* (which in turn is

[3] Although previously (Sharkey and Sharkey, 1995b, 1997; Sharkey, 1996) we have used the term *type*, instead of *level*.

related to the concept of variance, Geman et al., 1992), and by Wolpert (1992) when he suggests that what is required is that the nets should be *mutually orthogonal*. Levels of diversity however differ from these approaches because they take account of the overall accuracy of the ensemble output. That is, rather than simply requiring that the nets should disagree over some inputs, the level of diversity indicates the form that this disagreement takes and the extent to which it results in coincident failures.

3 Creating diverse ensembles

In the preceding section we examined the notion of diversity, and provided an account of the levels of diversity which can be exhibited by an ensemble. In this section, we shall consider ways of creating ensembles of neural nets that exhibit high levels of diversity.

The neural computing methods which can be employed for ensemble creation differ from those used for the same purpose in software engineering, where efforts to create diverse programs have made use of different teams of programmers (Knight and Leveson, 1986), different types of programming language (Adams and Taha 1992) and different specifications (Ramamoorthy *et al.*, 1981). In fact, in software engineering there have been relatively few reported uses of the concept of reliability through redundancy because of the difficulties (and cost) of generating program versions which fail on different inputs. For example, consider the cost involved in employing separate teams of programmers to develop alternative solutions. And even when separate teams of programmers are employed, there is evidence (Knight and Leveson, 1986) that they still tend to make similar errors due to the occurrence of common misunderstandings of certain aspects of tasks. By contrast, in neural computing, there are several aspects of neural net training which can be altered at little cost, and which, arguably, are less susceptible to shared comprehension failures. The candidate manipulanda, for the *extensional programming* of neural nets (c.f. Sharkey and Sharkey, 1995a) include the set of initial weights from which a net is trained, the architecture of the net, the input and output representations, and the content of the training sets. The question then is which of these manipulanda are more likely to lead to ensembles with high levels of diversity?

First, to achieve some diversity, it is important that the component nets in an ensemble show different patterns of generalisation. Nets trained on different training sets are more likely to show different patterns of generalisation than nets trained on the same training set from the starting point of different initial conditions, or with different numbers of hidden units, or using a different algorithm. This follows from the observation that it is the data on which a net is trained that determines the function it extracts (i.e., the guess that it makes about the target function). Thus two nets, each trained on different data sets, might be expected to induce, or guess, different alternative functions on the basis of those data sets. On the other hand, nets trained on the same data, but from the starting point of different sets of initial weights could induce the same function, namely that implied by the training data.

In support of the reasoning that different patterns of generalisation are more likely to result from changes in the training set, we can point to the paucity of evidence that nets trained on the same training set from different initial conditions, or with different numbers of hidden units, generalise differently. It has been argued (Kolen and Pollack, 1990) that backpropogation is sensitive to initial conditions, and the susceptibility of backpropagation to local minima is well known. However, this sensitivity/susceptibility is reflected in whether or not it is able to learn the training data (i.e., converges), and how many cycles it takes to do so. It seems that there is usually only one function that is induced on the basis of a set of data. Thus different nets that have been successfully trained on the same training data are likely to induce the same function. This analysis is supported by the empirical results reported by Sharkey *et al.* (1995) who found that nets trained from different initial conditions did differ in the number of training cycles they took to converge, and in whether or not they were able to learn that training data. However, Sharkey *et al.* (1995), also found that nets that had successfully learnt the training data showed the same patterns of generalisation. In other words, the nets had induced the same (alternative) function, or made the same guess about the target

function. Similarly, we can expect that alterations in the the number of hidden units would affect whether or not a net was able to learn the training data, but that nets that had learnt the training data would make the same guesses about the target function.

There is then reason to suppose that different patterns of generalisation are more likely to be obtained from nets that are trained on different training sets than through the manipulation of other neural net parameters. There are a number of alternative ways in which different training sets could be created. Techniques designed to produce different training sets include cross validation and bootstrapping (Raviv and Intrator, 1996; Krogh and Vedelsby, 1995); non-linear transformations (Sharkey *et al.*, 1996); injection of noise during training (Raviv and Intrator, 1996); data from different sensors (Sharkey *et al.*, 1996); the boosting algorithm (Drucker *et al.*, 1994); and the use of different methods of preprocessing. Cross-validation and bootstrapping both involve taking overlapping subsamples of a data set. Non-linear transformations, and injection of noise during training involve changing the inputs in a training set such that a new function is computed. Data from different sensors refers to the situation where the same classification can be made on the basis of more than one kind of input. For instance, in a study of engine faults, if a diagnosis could be made on the basis of either a measure of in-cylinder pressure, or in-cylinder temperature, then these two measures would be an example of data from two sensors (see section 3, case study). In the boosting algorithm, successive nets are trained on input-output pairs that have been filtered by previous nets.

The choice between such methods of ensemble creation will inevitably be determined, to an extent, by the availability of the data. For example, the technique termed Data from Different Sensors (DDS) will only be applicable where such data exists. Similarly, the boosting algorithm, which requires large amounts of data, will only be appropriate where data is in plentiful supply. However, as well as the availability of the data, the main consideration is to choose a method which will result in ensembles that exhibit good diversity (e.g., Level 1 or 2).

As was discussed above, a requirement for diversity is that the component nets in an ensemble should show different patterns of generalisation. Although there is reason to suppose that all the methods which involve varying the data might result in different patterns of generalisation, some methods may be more effective than others. Those methods which promote differences in the kind of patterns which are included in a training set are likely to be particular efficacious in this regard. Thus, the filtering involved in the boosting algorithm forces the training sets to be different. Similarly data taken from different sensors is likely to be different in kind. Nonlinear transformations, and different methods of preprocessing, also systematically change the data in the training set and are likely to result in different patterns of generalisation. Because the former methods promote differences in the content of the training set they are likely to be more effective than methods such as cross-validation and bootstrap which rely on taking different samples of the same master data set. Sampling can result in the same patterns of generalisation (it is quite possible for the same function to be extracted from different data sets), and there is no extra process here to promote differences between the samples.

A further reason for supposing that sampling methods such as cross-validation or bootstrap may be less effective is that they may result in individual nets which each generalise less well. High levels of diversity are more likely to be achieved if as well as showing different patterns of diversity, the individual nets in an ensemble all generalise well. The more failures each net exhibits, the more likely that the errors, when made, will overlap with those made by other nets. For instance, if each net only makes errors on 0.5% of the test set, it is more likely that the errors when made will not overlap than it would be if each net made errors on 50% of the training set. Taking smaller samples from a larger data set is likely to reduce the level of generalisation achieved by each net in an ensemble. Smaller training sets are likely to result in less good generalisation, unless we know enough about the problem to select the training set very carefully. A similar point is made by Tumer and Ghosh (1996) when they explored different methods of varying training sets, but found that the benefits of the resulting ensembles were reduced by the effect of using small training sets.

In summary, the argument made in the preceding paragraphs is that high levels of diversity are more likely to be achieved through the use of methods that promote different patterns of

generalisation, whilst not resulting in lower levels of generalisation. We suggest that the methods such as boosting, data from different sensors, and transformations of the input are most likely to fulfill these requirements.

We shall conclude this section with a more detailed description of two of these methods However, we should first make a further important point about ensemble creation, and that is that it can be facilitated through the use of selection. Rather than simply generating a set of nets through the application of some method, and then combining them, it makes sense to select amongst a larger pool of candidate members for a set of nets that can be effectively combined.

The idea of selecting nets for ensemble combination was raised by Perrone and Cooper (1993), when they suggested not including near identical nets. There are different ways in which such selection can be undertaken; for instance, different approaches to selection can be found in the following papers: Hashem (1996), Opitz and Shavlik (1996), Partridge and Yates (1995) and Sharkey and Sharkey (1997). The selection of nets for effective combination can be based on performance on a test set, and its effectiveness checked on a further test (or validation) set.

Our approach to selection has been to base it on estimates of the levels of diversity as outlined earlier, and to repeat the selection process until an ensemble is found that exhibits the required level of diversity. As opposed to trying all possible combinations, this process can be usefully guided by examining the correlations between pairs of nets, where the product-moment correlation coefficient between n pairs of observations, whose values are (x_i, y_i) is calculated as follows.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{[\sum_{i=1}^{n}(x_i - \bar{x})^2][\sum_{i=1}^{n}(y_i - \bar{y})^2]}}$$

Where correlations are close to zero, nets are likely to share a minimum number of coincident failures.

We have used the correlation matrix as a heuristic guide to which nets can be effectively combined. It is possible, on the basis of this matrix, to select pairs of nets which show a low correlation between their failures. Further testing can then be carried out to find a set of three nets which show a minimum (or an absence) of coincident failures.

3.1 Two methods of ensemble creation

Data from different sensors or the DDS method, provides a way of obtaining a set of training sets, each of which generalise well, but differently. Like boosting, this method is appropriate only under certain circumstances; that is, when the same output classification can be computed on the basis of different sensor readings. The input-output relationships in nets trained on data from different sensors will differ, even if they are trained on the same fault classification. There is therefore reason to suppose that the functions extracted by nets trained on different sensory data will differ, and lead to different patterns of generalisation. Evidence in support of this conclusion is provided by Sharkey *et al.* (1996), and is summarised in the next section of this paper.

Non-linear transformations of data (NLT): an effective way of creating nets which generalise well, but differently, is to start with a training set which permits good generalisation. Further training sets can then be created by distorting that training set in different ways. The NLT method involves creating a new training set by distorting the inputs (for example, by testing them on a randomly initialised net, and treating the outputs as new versions of the inputs). This approach is best used in conjunction with selection methods; eliminating distorted training sets which cannot be trained on the original classification, or which can be trained, but result in the same patterns of generalisation. The method can be thought of as being equivalent to preprocessing the data in different ways. A similar effect can be obtained by adding noise to different samples of data (Raviv and Intrator, 1996), or by pruning the inputs (Tumer and Ghosh, 1996). An advantage of the NLT method is that, unlike boosting, it does not require large amounts of data to be available, and unlike the DDS method, does not require the presence of data from different sensors.

Thus, in terms of guidelines about which techniques are likely to result in the most reliable forms of diversity with respect to a test set, our recommendation is to use the DDS method when it is possible to assemble good training sets using data from more than one source. Failing that, techniques like the NLT method, which rely on preprocessing, or distorting a training set in different ways, are likely to be particularly effective. Both these methods, or any others which might be adopted, should be used in conjunction with testing and selection of potential ensembles on the basis of the number of coincident errors they make. Admittedly this process takes time and is computationally expensive (although not as expensive as the process of generating independent versions of conventional software). Nonetheless, where nets are being applied in a safety-critical, or safety-related area, the extra time and expense needed to produce a more reliable system can readily be justified.

Having argued that two methods (DDS and NLT) are likely to provide effective means of creating diversity in ensembles, in the next section we shall report an example of their application in a particular case study. In this case study, the two methods are shown to provide effective means of creating diverse ensembles. They produce better results than training from different sets of initial weights, or randomly assembling disjoint training sets.

4 Applying diversity-promoting methods in a diesel engine case study

In a diesel engine case study (Sharkey *et al.*, 1996), nets were trained to diagnose two combustion faults on the basis of simulated data from a ship's engine corresponding to in-cylinder pressure and temperature. The data were generated from the MERLIN diesel performance simulator (shown to produce results which are almost identical to those of a real engine; Banisoleiman, 1993). The simulator was used to generate data corresponding to two combustion faults (i) Retarded injection of fuel, (ii) Advanced injection of fuel, and to (iii) Ideal combustion.

An early aim of this research was to examine the feasibility of replacing the standard method by which a skilled marine engineer examines indicator diagrams for discrepancies with ideal indicator diagrams, with a neural net system which could be used to monitor combustion quality during every engine cycle, and to provide immediate warnings of faults (Gopinath, 1994). The rapid detection of combustion condition in a marine engine is crucial since undetected faults can rapidly become compounded and even result in total breakdown. In the subsequent research, described here, the same data formed the basis of an investigation of the effectiveness of *ensembles* of nets at increasing generalisation performance.

In initial experimentation, nets were trained to perform this classification on the basis of a sample of the available data, and then tested for their ability to generalise to two sets of previously unseen set of data. Training, and test sets were generated using the MERLIN simulator based on data corresponding to in-cylinder pressure, in-cylinder temperature and a combination of the two (for further details see Sharkey *et al.*, 1996). The two master data sets of pressure and temperature were divided into a Test Set 1 (a test set of 314 example pairs), Test Set 2 (a second test set of 100 example pairs), and nine training sets of 150 example pairs (50 from each class).

For three types of data (Pressure, Temperature, and Combined Pressure and Temperature), nets were trained on nine (non-overlapping) training sets, from nine different sets of initial weights, resulting in a total of 3×81 trained nets. When the trained nets were tested on Test Set 1, the average generalisation performance for nets trained on pressure data was 98.33% (where the best net got 99.4% of Test Set 1 correct, and the worst got 96.8% correct), whilst for nets trained on temperature data the average correct generalisation performance was 98.15%, and for nets trained on combined data it was 98.45%. It should be apparent that these generalisation figures leave room for improvement through the application of diversity-creating methods.

We shall describe the application to this fault diagnosis task of two ensemble-creation methods: DDS, or *Data from more than one sensor*, and NLT, or *Non-linear transformations of input*.

4.1 Data from different sensors

Looking at the effect on ensemble diversity of using data from different sensors arose naturally in the context of the Diesel Engine case study, since data corresponding to both in-cylinder pressure, and in-cylinder temperature were available. As described above, nets could be trained to perform the fault diagnosis on the basis of either Pressure, Temperature, or a combination of the two. It was therefore possible to test these trained nets on the appropriate version [4] of Test Set 1, and to draw up the correlation matrix for nets trained on these measures. These are shown in Tables 1, 2 and 3. For

Table 1 Correlations between nets trained on Temperature and Pressure data. Temperature labelled from T1 to T9, corresponding to nine different training sets used. Pressure also labelled from P1 to P9, corresponding to nine different training sets used

	T 1	T 2	T 3	T 4	T 5	T 6	T 7	T 8	T 9
P 1	−0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01
P 2	−0.03	0.16	0.29	0.01	0.01	0.01	0.11	0.03	0.01
P 3	−0.02	0.22	0.37	0.01	0.01	0.01	0.15	0.02	0.01
P 4	−0.03	0.02	0.01	0.02	0.01	0.01	0.03	0.03	0.02
P 5	−0.03	0.02	0.01	0.02	0.01	0.02	0.03	0.03	0.02
P 6	−0.03	0.02	0.01	0.01	0.01	0.39	0.02	0.11	0.01
P 7	−0.02	0.23	0.40	0.01	0.01	0.01	0.16	0.02	0.01
P 8	−0.03	0.02	0.01	0.02	0.01	0.25	0.03	0.06	0.02
P 9	−0.02	0.01	0.01	0.01	0.01	0.01	0.02	0.02	0.01

Table 2 Correlations between nets trained on Pressure and Combined data

	P 1	P 2	P 3	P 4	P 5	P 6	P 7	P 8	P 9
C 1	−0.01	0.01	0.01	0.01	0.01	0.46	0.01	0.01	0.02
C 2	−0.03	0.20	0.23	0.02	0.60	0.25	0.18	0.05	0.03
C 3	−0.02	0.26	0.24	0.02	0.16	0.32	0.24	0.01	0.02
C 4	−0.03	0.02	0.02	0.02	0.09	0.28	0.02	0.02	0.03
C 5	0.11	0.02	0.01	0.00	0.41	0.31	0.02	0.03	0.03
C 6	0.23	0.02	0.02	0.02	0.01	0.62	0.02	0.01	0.03
C 7	−0.02	0.29	0.26	0.02	0.04	0.31	0.26	0.01	0.02
C 8	−0.03	0.02	0.03	0.01	0.56	0.45	0.02	0.04	0.03
C 9	−0.02	0.01	0.01	0.02	0.05	0.36	0.01	0.01	0.02

Table 3 Correlations between nets trained on Temperature and Combined data

	T 1	T 2	T 3	T 4	T 5	T 6	T 7	T 8	T 9
C 1	0.48	0.38	0.02	0.03	0.02	0.02	0.13	0.02	0.29
C 2	0.06	0.30	0.30	0.02	0.01	0.01	0.30	0.01	0.02
C 3	−0.01	0.50	0.77	0.01	0.01	0.01	0.77	0.01	0.02
C 4	0.07	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02
C 5	0.06	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02
C 6	−0.01	0.01	0.01	0.01	0.01	0.69	0.01	0.01	0.02
C 7	0.08	0.26	0.50	0.03	0.02	0.01	0.50	0.02	0.01
C 8	0.03	0.02	0.02	0.03	0.02	0.22	0.03	0.02	0.04

[4] For comparisons to be meaningful, it is important that all versions of the test set corresponded to the same set or sequence of engine cycles (containing both faulty and normal data). This ensures that what is being done is to take an engine cycle and to classify it into one of the three combustion categories, on the basis of three different input measurements (Pressure, Temperature and Combined).

Table 4 Correlations between nets trained from different Random Initial Conditions

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

Table 5 Correlations between nets trained on different training sets

	Train2	Train3	Train4	Train5	Train6	Train7	Train8	Train9
Train1	0.268	0.319	0.675	0.536	0.368	0.467	0.446	0.760
Train2		0.832	0.477	0.684	0.173	0.705	0.727	0.499
Train3			0.567	0.563	0.196	0.902	0.486	0.608
Train4				0.629	0.262	0.574	0.594	0.871
Train5					0.626	0.551	0.733	0.703
Train6						0.250	0.453	0.314
Train7							0.465	0.650
Train8								0.649

the purposes of comparison, Table 4 shows the correlations between nets trained on the same Pressure data from different initial weights, and Table 5 shows the correlations obtained between nets trained on different subsets of the Pressure data.

The correlations shown are the correlations between the output failures of nets trained under different conditions and were computed using the product-moment correlation coefficient (see section 3). Correlations take values between -1 and $+1$. Using binary data (presence/absence of error), a correlation close to $+1$ indicates that there is a linear relationship between the two variables such that an error made by one kind of net is usually accompanied by an error on the other kind of net. By contrast, a correlation close to -1 implies that the two variables produce opposite results, and that when one net fails the other succeeds, and *vice versa*. When the correlations in a correlation matrix are close to zero, this suggests that the two variables fail independently. The ideal situation, where two methodologies both result in good generalisation and show few, or no, shared errors will be represented by correlations that are close to zero. Therefore the tables can be examined to see the extent to which they show high correlations, or low correlations, bearing the ideal situation in mind.

The rows and columns of Tables 1, 2 and 3 are labelled either with a "T" for Temperature, a "P" for Pressure or a "C" for combined. Each entry represents the correlation between two methodologies (data sources), each methodology being represented by nine different versions (nets trained from nine different random seeds), using statistical methods derived from Littlewood and Miller (1989). Table 1 shows the correlations between the outputs of nets trained on Temperature data, and the outputs of nets trained on Pressure data; Table 2 shows the correlations between the outputs of nets trained on Combined data with nets trained on Pressure data; and Table 3 shows the correlations between the outputs of nets trained on Combined data with nets trained on Temperature data. In Table 4 the correlations between nets trained on Pressure data from different random initial conditions are shown; the rows and columns being labelled RIC 1–9 to indicate differences in the random initial conditions used. In Table 5 the rows and columns are labelled Train 1–9 to indicate differences in the mutually exclusive training sets used, all of which consisted of Pressure data.

Examination of Tables 1, 2 and 3 shows that low correlations were obtained when the comparison was between nets trained on different sources of data. These low correlations imply that, as discussed above, the two methodologies fail independently. The correlations seem particularly low when contrasted to those shown in Table 4, where the comparison was between nets trained on different initial conditions, and the correlations are all greater than 0.9. These high correlations imply that when inputs result in errors on one methodology, they also result in errors on the other methodology. Similarly (although to a lesser extent), higher correlations were also obtained when the comparison was between randomly selected different training sets, all based on Pressure data (Table 5). Here the lowest correlation obtained as a result of using non-overlapping training sets was 0.173, with the highest being 0.871. By contrast, 75% of the correlations in Tables 1, 2 and 3 were within 0.05 of zero correlation. Using the correlation matrices as a guide (i.e., focusing on those nets whose pairwise correlations are close to zero) it was possible to select three nets trained on different sources of data which together made no coincident errors on Test Set 1 (Level 1 Diversity), and thus showed 100% correct generalisation performance. When combined by means of a majority voter they similarly showed 100% correct generalisation to Test Set 2.

4.2 Non-linear transformations

The NLT method of using non-linear transformations of the input in order to promote diversity in an ensemble is particular to our laboratory (Sharkey *et al.*, 1996). It is best thought of as analogous to using different preprocessing methods on a set of data, although it bears some similarity to the idea of adding noise to the inputs (Raviv and Intrator, 1996). The method involves creating new training sets from an original, by means of transforming the inputs, and then constructing an ensemble which consists of a net trained on the original data, and nets trained on transformed data. In an application of the NLT method of nonlinear transformations to the case study of fault diagnosis of a ship's engine, the starting point was to take a net trained on a set of pressure data. When tested this net generalised correctly to 99.3% of Test Set 1. Two different transformations of this training set, *Transformation A* and *Transformation B*, were then selected from amongst a number of others. They were selected because when nets trained using these transformations were tested on Test Set 1, they showed no coincident failures with either each other, or the original training set.

Transformation A: the pressure data input patterns were transformed by using another ANN with two layers of weights, called a transformation net (TN), shown as Transformation *A* in Figure 3. The transformation of the inputs was accomplished by training the TN to autoencode the pressure data by reproducing the input vector as output. Once trained, only the input layer of weights in the TN was used to transform the input data, as shown in Figure 3. Thus ANN_1 was trained on the classification task with the TN hidden unit activations as input. After training, ANN_1 exhibited 98.1% correct generalisation on Test Set 1. To enable testing the test sets have to be similarly transformed by passing their inputs across the relevant transformation net. In this case, the test set was transformed by passing the inputs across the autoencoding net, and treating the hidden unit representations as the transformed test inputs.

Transformation B: the input for ANN_2, was the output from the set of untrained randomly selected weights that constituted a TN (rather than the hidden unit activation as in Transformation *A*). After training, ANN_2 exhibited 95.7% correct generalisation on the appropriately transformed Test Set 1.

These transformations can be thought of as equivalent to taking a single data set, and creating different versions of it by using different preprocessing methods.

An ensemble put together as shown in Figure 3. One net (ANN_3) was trained on a set of Pressure data; two other nets (ANN_1 and ANN_2) were trained on the same set of data after it had been transformed as described above. These transformations are illustrated in Figure 3; for ANN_1 the inputs were transformed by training them on another task and then treating the resulting hidden

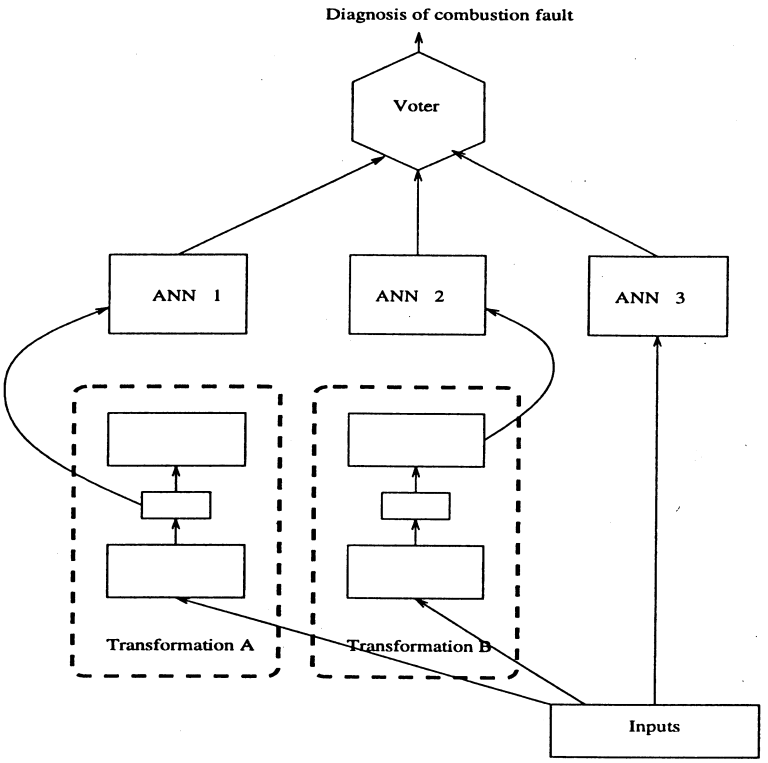

Figure 3 Health monitoring in a ship's engine

unit representations as the inputs. For ANN_2, the inputs were transformed by passing them across a random, untrained net, and treating the resulting outputs as inputs. What this heuristic methodology requires is that several transformations are attempted, and selected amongst on the basis of their ability when tested on a test set to produce outputs which show a low correlation with the original or other transformations. Many such transformations either result in data sets which nets cannot learn, or which are highly correlated with the original. Once nets which show a low correlation with one another have been identified, selection can then be further refined so as to choose a set of nets which when combined in an ensemble yield 100% performance on a verification set. The performance can be further tested by looking at the performance of the ensemble on a further test set not used for this selection (i.e., Test Set 2).

When considered in isolation, each of the three ANNs describe here exhibited high generalisation performance, but none were at 100%. However, when combined by means of a majority voter as shown in the system shown in Figure 3, there were no coincident failures on either Test Set 1 or Test Set 2, and thus the majority of ANNs was always correct. This is because the transformations were selected on the basis that they resulted in good generalisation, and did not show any coincident failures with the original net when tested on the Test Set 1. Thus, it has been shown that this transformation methodology can lead to an estimated Level 1 Diversity with respect to a test set. The pairwise correlations, and absence of coincident failures, can be seen in Table 6.

These results, and those reported under the preceding heading "Data from different sources", provide an illustration of the application of diversity-promoting techniques to a fault diagnosis task. They demonstrate that substantially improved performance can be obtained as a result of employing a neural net system that consists of an ensemble of neural nets. The two ensembles considered here were each combined by means of a simple majority voter, and exhibited 100% performance on two test sets, representing an improvement of 1.67% over the average performance of nets trained on the basis of pressure data and an improvement of 1.85% over the average performance of nets trained on the basis of temperature data. Of course, the nets could have been combined by some other method (e.g., stacked generalisation, Wolpert, 1992; weighted averaging, Hashem and

Table 6 Pairwise correlations between nets from three methodologies; Original, Transformation *A* and Transformation *B*. Number of coincident failures shown in brackets

	Orig	TM A	TM B
Orig		−0.0306 (0)	−0.0173 (0)
TM A			−0.0302 (0)
TM B			

Schmeiser, 1993), but the point is that their "error independence" allows them to be effectively combined through the simple method of majority voting.

5 Conclusions

If neural nets are indeed "imperfect estimators", the question might be raised as to whether there was any reason to use them at all in areas in which reliability and accuracy were important. The first answer to this question is to suggest that neural computing techniques should only be employed for tasks where they can outperform alternatives. Such tasks are likely to be those that involve physical data, with their typical characteristics of noise and incompleteness. Taking this approach implies a subsidiary role for neural nets in the safety-critical industry, in which neural computing techniques are used to monitor or diagnose individual components, rather than to provide a complete system. This approach would benefit from further work on the most effective ways of integrating neural computing solutions to control, monitoring and sensing tasks into conventional systems. Another possibility, considered elsewhere, is to use neural computing as a means of increasing the diversity of a set of solutions to a problem by including a neural net version alongside conventional ones.

A second way of responding to concerns about the use of neural computing techniques in the Safety-Critical industry is to concentrate on ways of increasing the reliability of neural computing. In this paper we have considered the recent research emphasis on combining nets for the formation of more reliable ensembles. The argument was made that better results will be obtained from neural net ensembles when the generalisation patterns of the member nets differ, and there are few coincident errors. A hierarchy of four levels of diversity was presented. In this hierarchy, the diversity exhibited by an ensemble with respect to a test set can range from Level 1, where the member nets do not share any failures and the majority is always correct, to the minimal diversity of Level 4, where the correct output is not obtained for all the patterns in the test file, but some improvement in performance is still achieved as a result of combining.

An account of this hierarchy was followed by a consideration of the requirements which need to be fulfilled by an ensemble that exhibits a high level of diversity. First, the component nets in the ensemble will need to show different patterns of generalisation. It was argued that nets trained on different training sets are more likely to generalise differently than nets created through variations in their initial conditions, or their topology. Second, high levels of diversity are more likely to be achieved when the nets in an ensemble each generalise well. Fewer errors reduces the chance of overlap. Third, it was pointed out that better results can be obtained if an ensemble-creation method is used in conjunction with *selection* such that nets are only included in an ensemble if they show the required level of diversity with respect to a test set.

Two methods of ensemble creation, DDS and NLT are discussed in more detail. Both of these methods encourage different patterns of generalisation without relying on sampling methods that can reduce the size of the training set with a consequent reduction in generalisation ability. Both can also be used in conjunction with selection. Their efficacy is supported by the account of the Level 1 Diversity obtained when they were applied to a fault diagnosis task. The engine fault diagnosis case

study also provides an illustration of the benefits of making use of insights about combining nets in an applied domain.

In considering the possible role of neural computing in the safety-critical industry, it is important to take a realistic view of its strengths and weaknesses. An admitted weakness of the present approach is that assessment of the effect of combining nets, and the determination of diversity level, depends on the composition of the particular test set(s) used. Clearly care needs to be taken to ensure that this test set is representative of the target function, and that it does not overlap in content with the training set. Nonetheless, a reliance on testing methods is a problem common both to all neural computing approaches, and to software engineering. Since the ensemble-based approach reviewed here can result in significantly improved performance, it makes sense to adopt it as an essential component of the appropriate use of neural computing techniques.

Acknowledgement

We would like to thank the EPSRC Grant No.GR/K84257 for funding the preparation of this paper.

References

Adams, JM and Taha, A, 1992. "An experiment in software redundancy with diverse methodologies" *Proc. Twenty-Fifth Hawaii International Conference on Systems Sciences.*

Banisoleiman, K, Smith, LA and Matheieson, N, 1993. "Simulation of diesel engine performance" *Trans. Institute of Marine Engineers* **105**(3) 117–135.

Boek, MJ, 1991. "Experiments in the application of neural networks to rotating machine fault diagnosis" *Proc. IEEE International Joint Conference on Neural Networks* Singapore, 769–774.

Breiman, L, 1992. "Stacked Regression" Dept. of Statistics, Berkeley, Technical Report no 367.

Breiman, L, 1994. "Bagging predictors" Technical Report, UC Berkeley, 1994.

Croll, PR, Sharkey, AJC, Bass, JM, Sharkey, NE and Fleming, PJ, 1995. "Dependable intelligent voting for real-time control software" *Engineering Applications of Artificial Intelligence* **8**(6) 615–623.

Denker, J, Schwartz, D, Wittner, B, Solla, S, Howard, R, Jackel, L and Hopfield, J, 1987. "Large automatic learning, rule extraction and generalisation" *Complex Systems* **1** 877–922.

Drucker, H, Cortes, C, Jackel, LD, LeCun, Y and Vapnik, V, 1994. "Boosting and other ensemble methods" *Neural Computation* **6** 1289–1301.

Duyar, A and Merrill, W, 1992. "Fault diagnosis for the space shuttle main engine" *Journal of Guidance, Control and Dynamics* **15**(2) 384–389.

Geman, S, Bienenstock, E and Doursat, R, 1992. "Neural networks and the bias/variance dilemma" *Neural Computation* **4**(1) 1–58.

Gopinath, OC, 1994. "A neural net solution for diesel engine fault diagnosis" *MSc thesis*, University of Sheffield.

Granger, CWJ, 1989. "Combining forecasts — twenty years later" *Journal of Forecasting* **8**(3) 167–173.

Hansen, LK and Salamon, P, 1990. "Neural network ensembles" *IEEE Tran. Patterns Anal. Machine Intelligence* **12**(10) 993–1001.

Hashem, S, 1996. "Effects of collinearity on combining neural networks" *Connection Science* **8**(3/4) 315–336.

Hashem, S and Schmeiser, B, 1993. "Approximating a function and its derivatives using MSE-optimal linear combinations of trained feedforward neural networks" *Proc. Joint Conference on Neural Networks* vol 87, 617–620, New Jersey.

Knight, JC and Leveson, NG, 1986. "An experimental evaluation of independence in multiversion programming" *Trans. on Software Eng.* **12**(1).

Kolen, JF and Pollack, JB, 1990. "Backpropagation is sensitive to initial Conditions" *TR 90-JK-BPSIC.*

Krogh, A and Vedelsby, J, 1995. "Neural network ensembles, cross validation and active learning" In: G Tesauro, DS Touretzky and TK Leen, eds, *Advances in Neural Information Processing Systems 7*, MIT Press.

Lincoln, WP and Skrzypek, J, 1990. "Synergy of clustering multiple back propagation networks" In: *Advances in Neural Information Processing Systems 2* 650–657.

Littlewood, B and Miller, DR, 1989. "Conceptual modelling of coincident failures in multiversion software" *IEEE Trans Software Engineering* **15**(12) 1596–1614.

MacIntyre, J, Smith, P, Harris, T. and Brason, A, 1993. "Application of neural networks to the analysis and

interpretation of off-line condition monitoring data" *Sixth International Symposium on Artificial Intelligence* Monterrey (Mexico).

Marko, KA, James, JV, Feldkamp, TM, Puskorius, GV and Feldkamp, LA, 1996. "Signal processing by neural networks to create 'virtual' sensors and model-based diagnostics" *Proc. ICANN 1996* Bochum, Germany.

Nilsson, NJ, 1965. *Learning Machines: Foundations of Trainable Pattern-Classifying Systems* McGraw Hill.

Opitz, DW and Shavlik, JW, 1996. "Actively searching for an effective neural network ensemble" *Connection Science* **8**(3/4) 337–354.

Partridge, D and Griffith, N, 1995. "Strategies for improving neural net generalisation" *Neural Computing and Applications* **3** 27–37.

Partridge, D and Yates, WB, 1996. "Engineering multiversion neural-net systems" *Neural Computation* **8**(4) 869–893.

Perrone, MP and Cooper, LN, 1993. "When networks disagree: Ensemble methods for neural networks" In: RJ Mammone, ed., *Neural Networks for Speech and Image Processing* Chapman Hall.

Raviv, Y and Intrator, N, 1996. "Bootstrapping with noise: an effective regularization technique. *Connection Science* **8**(3/4) 355–372.

Ramamoorthy, CV, Mok, YR, Bastani, EB, Chin, GH and Suzuki, K, 1981. "Application of a methodology for the development and validation of reliable process control software" *IEEE Trans. Software Eng.* **7** 537–555.

Rogova, G, 1994. "Combining the results of several neural network classifiers" *Neural Networks* **7**(5) 777–781.

Rumelhart, DE, Hinton, GE and Williams, RJ, 1986. "Learning internal representations by error propagation" In: DE Rumelhart and JL McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol 1: Foundations* MIT Press.

Sharkey, AJC and Sharkey, NE, 1997. "Diversity, selection and ensembles of artificial neural nets" In: *Proc. Third International Conference on Neural Networks and their Applications: IUSPIM* Marseilles, France, pp. 205–212.

Sharkey, AJC, 1996. "On combining artificial neural nets" *Connection Science* **8**(3/4) 299–314.

Sharkey, AJC, Sharkey, NE and Chandroth, GO, 1996. "Diverse neural net solutions to a fault diagnosis problem" *Neural Computing and Applications* **4** 218–227.

Sharkey, AJC and Sharkey, NE, 1995a. "Cognitive modelling: psychology and connectionism" In: MA Arbib ed., *The Handbook of Brain Theory and Neural Networks* Bradford Books/MIT Press, pp. 200–203.

Sharkey, AJC and Sharkey, NE, 1995b. "How to improve the reliability of artificial neural networks" *Research Report CS-95-11*, Department of Computer Science, University of Sheffield.

Sharkey, NE and Sharkey, AJC, 1995c. "An analysis of catastrophic interference" *Connection Science* **7**(3 & 4) 301–329.

Sharkey, NE, Neary, J and Sharkey, AJC, 1995. "Searching weight space for backpropagation solution types" In: LF Niklasson and MB Boden, eds., *Current Trends in Connectionism: Proceedings of the 1995 Swedish Conference on Connectionism*, Lawrence Erlbaum, pp. 103–120.

Sharkey, NE and Partridge, DP, 1992. "The statistical independence of network generalisation: an application in software engineering" In: PG Lisboa and MJ Taylor, eds., *Neural Networks: Techniques and Applications* Ellis Horwood.

Tumer, K and Ghosh, J, 1996. "Error correlation and error reduction in ensemble classifiers" *Connection Science* **8**(3/4) 385–404.

White, H, Hornik, K and Stinchcombe, M, 1992. "Multilayer feedforward networks are universal approximators" In: H White, ed., *Artificial Neural Networks: Approximation and Learning Theory* Blackwell.

Wolpert, DH, 1992. "Stacked generalisation" *Neural Networks* **5**(2) 241–259.

Worden, K, 1997. "Structural fault detection using a novelty measure" *Journal of Sound and Vibration* **201**(1) 85–101.

Xu, L., Krzyzak, A and Suen, CY, 1992. "Methods of combining multiple classifiers and their applications to handwriting recognition" *IEEE Trans. Systems, Man and Cybernetics* **22**(3) 418–435.