# Preliminary Experiments with Ensembles of Neurally Diverse Artificial Neural Networks for Pattern Recognition

Adamu, A.S  Maul, T.H.  Bargiela, A.  Roadknight, C.

30th November 2014

**Abstract**

Although there have been a few approaches to achieve the goal of distributing the failure of the individual networks that make up a neural network ensemble, some of which include ensembles of neural networks of different sizes, and ensembles of different models of neural networks such as Radial Basis Function Networks and Multilayer Perceptron, there is yet to be an empirical study on hybrid neural networks that makes use of a diverse set of transfer functions, which we would expect to be able to exhibit diverse network architectures, and thus possibly more diverse error patterns. In this paper, we present an approach that uses transfer function diversity to achieve significant results on ensembles. The results show that even with relatively small networks having 5 hidden nodes, and a relatively small ensemble size of just 10 members, the ensemble is able to get competitive results on the Iris data set. It also able to able to get competitive results with 20 ensemble members of relatively small networks on other popular data sets such as the Diabetes, Sonar, Hepatitis, and Australian Credit Card problems. In addition to that, it is shown that these results can be achieved with a simple sorting and selection of the Top N solutions of the population, in contrast to other methods of selecting ensemble members that can be computationally expensive, such as selection of the pareto-front, or hill climbing methods of selection.

## 1 Introduction

A method of fault tolerance through redundancy is a concept that is used in many fields, for example in software engineering where one of the identified methods for achieving reliability is through redundancy that is diverse [18]. The aim is to minimize the failure of a system by combining several redundant, but diverse components designed for a single task in such a way that they are unlikely to all fail at once. This concept has also gained interest in the field of Artificial Neural Networks as Ensembles of Artificial Neural networks, where Neural Networks are designed and trained to be diverse in their behavior so that they have an improved generalization ability when combined. The research found by earlier studies [5, 18, 13, 19, 9, 14, 17] shows that neural networks combined as ensembles exhibit an improved generalization ability provided they are diverse. In addition to these empirical findings, it is also proven [5] that the error of an ensemble is guaranteed to be the same or better than the average error of its components.

The importance of diversity in ensembles is quite intuitive. If the individual solutions in an ensemble were all identical, the performance of the ensemble would not differ from any of the members of the ensemble. However, if all the members of the ensemble were different from each other such that there is little or no coincidence in their response to any given pattern, then one can expect improved performance from the ensemble particularly in the aspect of generalization. The significance also depends on some other factors such as the method used to combine the outputs of the members, the number of members in the ensemble, and the accuracy of the members in the ensemble.

Generally, most of the promising methods for creating diversity in ensembles could be categorized into three according to their area of focus [5]: data set, model, and training algorithm. In the first case, some approaches found in the literature use re-sampling and pattern distortion methods to achieve some variations in the training data set. These variations then implicitly cause behavioral differences in members of the ensemble. Popular re-sampling methods include bagging and boosting. In bagging, random samples of the data set are used to train each member. Boosting is similar to bagging, however it considers the distribution of the subsets while sampling. Another method used by some studies that was highlighted by Brown et. al. [5] is to re-sample the features of each pattern in the training data set. On the other hand, distortion methods used include the addition of Gaussian noise, or non-linear transformations of the training patterns in the data set. One of the non-linear transformation approaches found [5, 18] to be effective was to stimulate a randomly generated neural network with the training pattern and then use its output as the distorted pattern. Gaussian noise was also found to be helpful [5, 18]. In the case of diversity creation methods focusing on models [5], some methods used of a mixture of models. The most popular includes the use of homogeneous models with varied parameters [5], such as neural networks of different sizes within the ensemble, or neural networks with different types of architectures such as Multilayer Perceptrons and Radial Basis Function networks [5]. Other methods [5] use heterogeneous models such as the use of decision trees and neural network within an ensemble. In the final category are methods which focus on the training algorithm, some of which include the use of different training algorithms [18], and the introduction of an additional term in the objective function [14, 17], such as in neural network ensembles trained by evolutionary algorithms [5].

There are some areas that have been deficient with regards to experiments. One of such area is the topic of model diversification approaches for neural network ensembles. Intuitively, it makes sense that if we are aiming for error diversity within our neural network ensemble, an equally likely approach to the others that could yield significant diversity is an approach that is explicit. By *explicit*, we mean an approach that takes a direct method, such as combination of diverse architectural models of neural networks. According to our knowledge, there has been no experiments with ensembles use hybrid artificial neural networks implementing a diverse transfer functions set; which we would expect to increase error diversity in ensembles. This was also highlighted in the thorough survey of ensembles by Brown [5]. The only partially related work done so far was by Partridge who used pure models of Multilayer Perceptron's

(MLP) and Radial Basis Functions (RBF) in an ensemble to achieve diversity. However, even that work was suggested to be a preliminary study by [5].

In this paper, we experiment with hybrid neural network ensembles implementing a diverse transfer function set, also known as Neural Diversity Machine Ensembles(NeuDiME). This is unlike other approaches found in the literature which have used a mixture of pure models as reported by Brown et. al [5]. We explore the performance of this approach in different circumstances; specifically, we test it on popular pattern classification problems such as the Iris, Sonar, Hepatitis, Diabetes, and the Australian credit card problem, which have been used as benchmarks of choice in most literature. We also compare this approach with other approaches found in the literature, and also analyze two accurate yet different strategies evolved for some of the problems (i.e. diabetes and Iris).

The contributions of this paper are as follows: firstly, the paper presents the application of hybrid neural networks in ensembles and provides a study on some of the effects of transfer function diversity in neural network ensembles. Secondly, it shows that this neural network framework is able to develop different strategies for a problem that can be used in ensembles without explicit diversity maintenance that can be expensive, such as selection of the Pareto front for use as ensembles, or fitness sharing. It also shows that this approach can evolve a relatively smaller ensemble of compact networks that has a competitive performance. Finally, it shows how neurally diversity can result in diverse classifiers by analyzing two computational strategies evolved for the diabetes problem.

It should be noted that the term "weight function" is used interchangeably with the term "input combination function"and activation function. Likewise, the term, "node function" is used interchangeably with the term "output function". Additionally, the term "node" is used interchangeably with the term "neuron" and "unit".

## 2 Neural Diversity Machines

In this section, the Neural Diversity Machines approach proposed by Maul [15] is explained, starting from transfer function diversity - the core of the framework, then followed by weight and architecture evolution.

### 2.1 Transfer Function Diversity

A transfer function of a neural network is composed of the input combination function and an output function; the input combination function computes the input values being transferred into a neuron by other neurons in the network connected to it using weighted connections, while the output function can be considered as a mathematical model of its biological counterpart, which determines which determines the output value of the neuron.

Traditional and well established Artificial Neural Networks such as Radial Basis Functions (RBF) and Multilayer Perceptrons (MLP) use predetermined transfer functions for nodes of each layer as part of their distinct architectural properties. Often, the nodes of each layer are homogeneous in their transfer function. A Radial Basis

| Input Combination Functions | Color Code |
|---|---|
| Inner-Product ($ap_i = \sum_{i=0}^{n} w_i i_i + w_{bias}$) | Red solid Edge |
| Higher-Order Product ($ap_i = \prod_{i=0}^{n} cw_i * i_i$) | Yellow Solid Edge |
| Higher-Order Subtractive ($ap_i = \sum_{i=1}^{n} \lvert x_0 - x_i \rvert$) | Yellow Dash Edge |
| Euclidean Distance ($ap_i = \sqrt{\sum_{i=0}^{n} (w_i - i_i)^2}$) | Magenta Dash Edge |
| Standard Deviation ($ap_i = stdDev(w_i i_i, w_{i+1} i_{i+1} ... w_n i_n)$) | Blue Solid Edge |
| Min ($ap_i = min(w_i i_i, w_{i+1} i_{i+1} ... w_n i_n)$) | Gray Dash Edge |
| Max ($ap_i = max(w_i i_i, w_{i+1} i_{i+1} ... w_n i_n)$) | Black Dash Edge |

Table 1: List of some input combination and output functions used by Neural Diversity Machines and their visualization color codes.

| Output functions | Color Codes |
|---|---|
| Linear ($O_i = \alpha * ap_i$) | Yellow Node Outline |
| Hyperbolic tangent ($O_i = \frac{1-e^{-\alpha * ap_i}}{1+e^{-\alpha * ap_i}}$) | Cyan Node Outline |
| Sigmoid ($O_i = \frac{c}{1+e^{-\alpha * ap_i}}$) | Red Node Outline |
| Gaussian ($O_i = e^{\frac{-(ap)^2}{width}}$ ) | Blue Node Outline |
| Gaussian II ($O_i = e^{\frac{-(ap)^2}{width}} if\, O_i > \theta\, then\, O_i = 1.0$ ) | Dark-Blue Node Outline |

Table 2: List of Output functions for Neural Diversity Machines and their visualization color codes.

Function uses a combination of a distance-based input combination function(e.g. Euclidean distance) and a radial basis output function, which is typically a Gaussian, Multiquadratic, thin-plate-spline, or inverse Multiquadratic. In the case of a Multilayer Perceptron, its typically an inner-product input combination function accompanied with a sigmoid output function. Though it's proven that both RBF networks (RBFN) and Multilayer Perceptrons are able to approximate any function, provided that the complexity of the network's model is matched with the complexity of the problem [4, 7], it's not proven that the model will be optimal or practical. By *optimal*, we are specifically emphasizing on using a minimal number of hidden units to learn a problem.

There is no one-size-fits-all in the choice of transfer functions [8, 12, 15]. A certain problem might be more suited for an MLP unit, another require an RBF unit. Thus, there is a need for adapting the choice of transfer function.

Several studies have approached this problem using hybrid models [15, 8, 12] which implement different basis functions, either as a single hybrid layer, or as several pure layers and have been found to perform better than their canonical models (i.e. RBF or MLP).

This work differs because it is particularly motivated from the biological standpoint of the benefits of neural diversity found in biological neural networks, which includes increased representational capacity that contributed significantly to their efficiency. This was replicated by having different classes of functions in the transfer functions pool,such as radial-basis units, projection-basis units and higher-order units. However, due to the flexibility allowed in the combination of activation and output functions during optimization, other unconventional transfer functions were also evolved.

## 3 Neural Diversity Machine Ensembles (NeuDiME)

Neural Diversity Machine Ensembles is an ensemble made up of Neural Networks that conforms to the framework of Neural Diversity Machines [15]. It uses a set of diverse input combination functions (see input combination set in table 1) and output functions (see output function set in table **??**) which enables the optimization algorithm to find near optimal transfer functions for each node from the transfer function set, thus increasing the likelihood of having accurate ensemble members. In addition to that, the diverse transfer function set should result in relatively more diverse computational strategies [5] for a problem as compared to an approach using pure models. Subsequently, this is expected to result in the desired balance between bias and variance. After all, it is intuitive that an ensemble made up of different yet accurate models will likely yield more *useful* diversity. By *useful*, we are referring to the diversity that results in significant improvements in the generalisation ability of the ensemble. This differs from other approaches which have attempted the use of neural networks of different sizes within an ensemble or a mixture of pure models[5].

The stages for the optimisation algorithm are as follows:

Step i) Solutions (of various hidden node sizes) are created.

Step ii) Solutions are evaluated.

Step iii) Differential Evolution is applied.

Step iv) Other Evolutionary Operators are applied (i.e. Mutation and Cross-over).

Step v) Finally, select members of the ensemble from the population.

Step vi) Ensemble is evaluated.

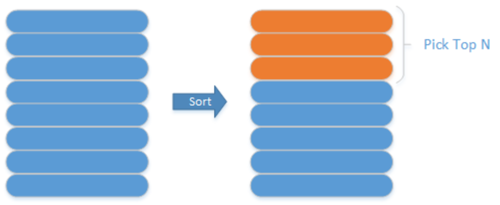
Step vii) Weak solutions are eliminated.

Step viii) Stopping condition is checked.

Step ix) If the stopping condition is met, the ensemble is evaluated on the test set and the error is used as its generalisation performance.

### 3.1 Ensemble Member Selection

In this study, we use the Top N solutions for convenience. It lacks the relatively higher computational cost of other selection methods such as the use of the pareto-front[2], or selection of members using a hill climbing approach while maintaining diversity with fitness sharing [6, 11]. In addition to that, it's relatively simple and has been used in later works by Opitz & Shavlik [16].



N.png

Figure 1: Top N - Selects the top N solutions of the population as the ensemble members after sorting according to their accuracies.

The method of selection selects the Top N solutions from the population after sorting by fitness, where *N* is the desired size of the ensemble. The sorting prior to selection helps in improving the chances of picking the *N* solutions with the most generalization ability.

However, this selection method assumes there is already diversity in the population such that the top *N* solutions are diverse after sorting by accuracy. In the case of NeuDiME, there is already diversity introduced by using a diverse set of transfer function combinations to pick from for all the neural networks. In addition to that, there is also the added diversity of neural networks of different sizes.

## 4 Experiments

In this section of the report, we present the benchmarks, performance measures, and finally the experimental setup.

### 4.1 Experimental setup

The benchmarks used consisted of some problems commonly used in the ensemble literature: Iris,Sonar, Australian credit card, Hepatitis and Diabetes retrieved from the machine learning repository [3]. In order to conform to the common measure of generalization ability found in the literature [7, 13, 9], cross-validation was used.

| Benchmarks | Features | Classes | Examples |
|---|---|---|---|
| Iris | 4 | 3 | 150 |
| Sonar | 60 | 2 | 208 |
| Card (Australian) | 51 | 2 | 690 |
| Hepatitis | 19 | 2 | 155 |

Table 3: List of some of the benchmarks retrieved from the UCI Machine Learning Repository[3].

| Benchmarks | Max Hidden units | Members(Ensemble) | Folds (K-fold CV) |
|---|---|---|---|
| Iris | 5 | 10 | 10 |
| Sonar | 5 | 10 | 10 |
| Diabetes | 5 | 20 | 10 |
| Hepatitis | 5 | 20 | 10 |
| Card (Australian) | 5 | 20 | 12 |

Table 4: Experimental setup for the datasets showing the maximum number of hidden units allowed per ensemble member, the number of members in the ensemble and the number of folds used for K-fold cross-validation.

The optimisation paramaters for the Global Stochastic Optimization (GSO) algorithm used to optimise the neural network ensembles is given as follows:

| Optimisation Parameter(s) | Value(s) |
|---|---|
| Max Iterations | 100 |
| Population size | 30 |
| Percent to eliminate | 0.3 |
| Min cost (elimination) | 0.66 |
| Min age (elimination) | 3 |
| Cross Over | True |
| Probability of Cross Over | 0.2 |
| Differential Evolution (DE) Iterations | 3 |
| DE alpha | 0.2 |
| Gene range | [-0.9, 0.9] |
| Probability of Mutation | 0.2 |
| Gaussian Mutation (Mean,Std) | (0.0, 0.2) |

Table 5: The optimisation parameters used for the neuroevolution of the ensemble members.

## 5 Results and Discussion

In this section we present the results of the neural network ensembles, NeuDiME, on Synthetic Data sets. This is then followed by the results on the popular data sets and a comparison with various other methods.

## 5.1 Performance on some popular benchmarks

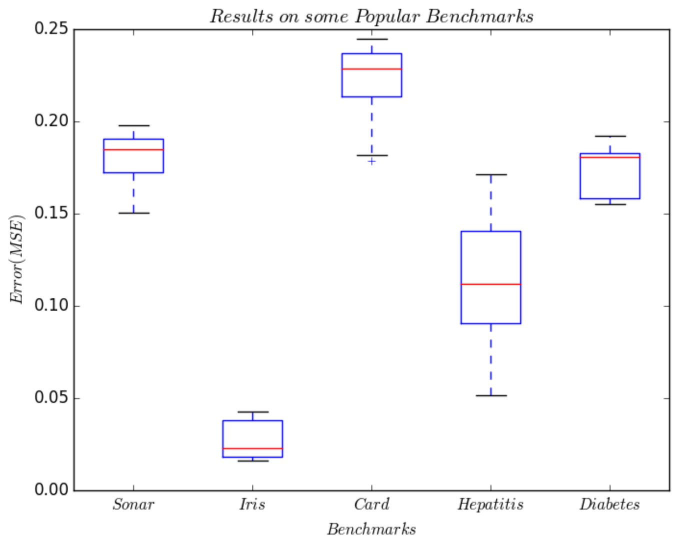


Figure 2: Results on the popular data sets of NeuDiME - Test error averaged over 10 folds cross-validation,except in the case of Australian credit card problem which was set to 12 folds to make a comparison with the known literature easier.

The results produced by the Neural Diversity Machine Ensemble were competitive. Though it didn't compare in performance for the Australian Credit Card problem, as it got the worst results. It was competitive for the rest of the problems: it got the best results for the Diabetes problem, which compensates for its relatively poor results on the Credit Card problem. In addition to that, it also got good results for the rest of the problems.

| | Australian Credit Card | Iris | Sonar | Diabetes | Hepatitis |
|---|---|---|---|---|---|
| MPANN | 0.135 | - | - | 0.23 | - |
| EENCL | 0.138 | - | - | 0.221 | - |
| CNNE | **0.092** | - | - | 0.198 | - |
| DIVACE | 0.138 | - | - | 0.226 | - |
| Opitz & Maclin[**?**] | 0.137 | 0.039 | **0.129** | 0.233 | 0.178 |
| NeuDiME | 0.221 | **0.027** | 0.181 | **0.174** | **0.115** |

Table 6: Comparison with other learning methods - Results of MPANN, EENCL, CNNE, DIVACE and Optiz et. al as reported in their findings. In the case of Optitz et. al the best results of the five techniques experimented with were used. The Testing Error of EENCL was used as reported in [10]. As for MPANN, the best of the three results as reported in [2], while results of CNNE and DIVACE were retrieved as reported in [11, 6],respectively.

## 5.2 Interesting Computational Strategies Evolved for Prediction

In this subsection of the results and discussion, we present statistics of transfer function use for some of the data sets used in the experiments. These include the probabilities of combining the possible input combination functions with the possible output functions in the fittest members of the ensemble, and the associated error with each combination. In addition to that, we also reveal some of the diverse strategies used by the members of the ensemble towards solving the diabetes problem by looking at their choice of transfer function, connectivity and weights.

### 5.2.1 Diabetes

The diabetes problem showed an emphasis on strategies that relied on using standard deviation and output function such as the identity function, and hyperbolic tangent. The least error was associated with the combination of Euclidean distance and Gaussian II, which is essentially a variant of a radial basis function unit. Higher product combined with identity function also showed a relatively lower associated error. However, considering these were some of the least likely to be evolved for the problem, they should be expected to have lower errors. The tables below present the statistics.

Table 7: Likelihood(%) of using combinations of input combination and output functions for the Diabetes problem - The most likely combination was standard deviation and identity, and standard deviation and hyperbolic tangent.

| | Identity | Sigmoid | Gaussian | Hyperbolic Tangent | Gaussian II |
|---|---|---|---|---|---|
| Inner Product | 0 | 0 | 0 | 3.57143 | 3.57143 |
| Euclidean Distance | 0 | 0 | 0 | 0.0 | 3.57143 |
| Higher Order Product | 3.57143 | 3.57143 | 0 | 0.0 | 3.57143 |
| Higher Order Subtractive | 0 | 0 | 0 | 0.0 | 7.14286 |
| Standard Deviation | **17.85714** | 0 | 0 | **14.28572** | 3.57143 |
| Min | 3.57143 | 7.14286 | 0 | 7.14286 | 0.0 |
| Max | 3.57143 | 7.14286 | 3.57143 | 3.57143 | 0.0 |

Table 8: Associated error of using combinations of input combination and output functions for the Diabetes problem

| | Identity | Sigmoid | Gaussian | Hyperbolic Tangent | Gaussian II |
|---|---|---|---|---|---|
| Inner Product | - | - | - | 0.01709 | 0.01161 |
| Euclidean Distance | - | - | - | - | **0.00125** |
| Higher Order Product | **0.00210** | 0.00644 | - | - | 0.01319 |
| Higher Order Subtractive | - | - | - | - | 0.02434 |
| Standard Deviation | 0.03476 | - | - | 0.05334 | 0.00405 |
| Min | 0.01357 | 0.00539 | - | 0.01066 | - |
| Max | 0.00500 | 0.02989 | 0.02224 | 0.00543 | - |

In the following discussion, we will present a study of two diverse strategies evolved for NeuDiME on the diabetes problem. This reveals the ability of NeuDiME to exhibit diverse neural computation strategies that are accurate.

One of the most accurate strategies evolved for the diabetes problem was a fully connected network consisting of four hidden units, implementing the following transfer functions found in Figure 3.

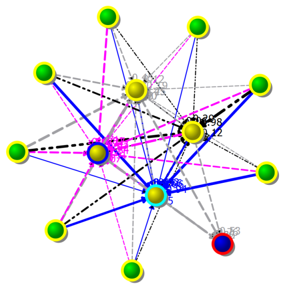


Figure 3: One of the member evolved in NeuDiME for the Diabetes problem with an individual training error of 0.18.

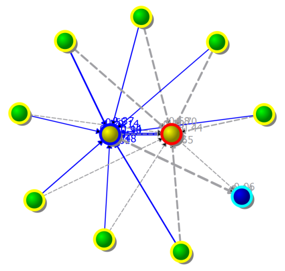


Figure 4: Another example of a hybrid neural network evolved as one of the members of NeuDiME for the Diabetes problem with a training error of 0.17.

It seemed that the range of the feature values (min and max) is somehow important in this strategy. Thus, why min and max functions were transferring information to the projection unit (i.e. the perceptron output unit). In an effort to understand why this is essential for this strategy, we compared the min and max of the raw data set: most of the time, the max value corresponds to the 2nd feature of the data set, i.e. *glucose concentration reading*. While the min value usually corresponded to either the 4th or the 5th feature, i.e. *skin fold thickness* and *serum insulin reading*, respectively. Based on the connection weights of these features to the hidden layers using min and max as a relay; the 4th feature (skin fold thickness) was given more weight as compared to the rest. The 2nd feature (glucose concentration) and 5th feature (serum insulin reading) were both given a medium weight, perhaps to normalize its values with the rest of the features, as they usually have the highest values. The most important feature relayed by these relay units based on their weights to them was *age*.

Interestingly, *age* and *skin fold thickness* are actually regarded be highly correlated to diabetes. The American Diabetes Association for example regards age as one of the leading contributors which increase the risk of a person having type-2 diabetes[1].

In general, this strategy seems to be taking advantage of the variety of input combination function and output functions to perform extract important features using unusual combinations of transfer functions such as the minimum feature value, weighted variance of between the features, proximity of the feature vector to the centre of the RBF unit, and maximum feature value. Then using these features to train a simple perceptron in the output layer. In other words, its using the hidden layer for feature selection, then taking advantage of the reduced dimensionality to train a simple hyperbolic-tangent perceptron.

Another strategy evolved was a fully connected network with two hidden units where one adopted a min input combination function and a sigmoid output function, while the other adopted a standard deviation output function with a hyperbolic tangent output function. The output unit differed from the other strategy, consisting of a max input combination which has a winner-take-all effect on the hidden nodes. Its hyperbolic tangent output function has another norming effect.

Once again, this strategy uses the min input combination function, however in this case, it is coupled with a sigmoid output function, which has a normalising effect on the output value - restricting it between 0.0 and 1.0. In addition to that, in this case it seems to act as a threshold for the other hidden node using the weighted variance. This is because of the choice of using max as the input combination function by the output unit.

In general, the normalising effect of the output functions of both hidden nodes allows the hidden node using the min input combination to essentially resemble the role of a bias node that is dependent on the features.

## 6 Conclusion

In conclusion, this paper has presented the application of hybrid neural networks in the field of ensembles and has shown that neural diversity in artificial neural network framework is able to exhibit different computational strategies for a problem that can be used in ensembles without explicit diversity maintenance. This is shown by the two different strategies for the diabetes problem. It has also been shown that this approach can evolve relatively smaller ensembles of compact networks that have a competitive performance.

## References

[1] Lower Your Risks: Age, Race, Gender & Family History, 2013.

[2] H.A. Abbass. Pareto neuro-evolution: constructing ensemble of neural networks using multi-objective optimization. In *The 2003 Congress on Evolutionary Computation, 2003, CEC '03.*, volume 3, pages 2074–2080, Cancun, 2003. IEEE.

[3] K Bache and M Lichman. {UCI} Machine Learning Repository, 2013.

[4] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* 1995.

[5] Gavin Brown, Jeremy Wyatt, Rachel Harris, and Xin Yao. Diversity creation methods: a survey and categorisation. *Information Fusion*, 6(1):5–20, March 2005.

[6] Arjun Chandra and Xin Yao. Ensemble Learning Using Multi-Objective Evolutionary Algorithms. *Journal of Mathematical Modelling and Algorithms*, 5(4):417–445, March 2006.

[7] PA Gutierrez, C Hervas, M Carbonero, and JC Fernandez. Combined projection and kernel basis functions for classification in evolutionary neural networks. *Neurocomputing*, 72(13-15):2731–2742, August 2009.

[8] P.A. Gutierrez and C. Hervas-Martinez. Hybrid Artificial Neural Networks : Models , Algorithms and Data. *Lecture Notes in Computer Science*, 6692(PART 2):177–184, 2011.

[9] LK Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[10] T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[11] Md M Islam, Xin Yao, and Kazuyuji Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 14(4):820–34, January 2003.

[12] Norbert Jankowski and Wlodzislaw Duch. Optimal transfer function neural networks. In *9th European Symposium on Artificial Neural Networks, ESANN 2001*, number I, pages 101–106, Bruges, 2001.

[13] Anders Krogh and J Vedelsby. Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems*, pages 231–238, 1995.

[14] Y Liu and X Yao. Ensemble learning via negative correlation. *Neural networks : the official journal of the International Neural Network Society*, 12(10):1399–1404, December 1999.

[15] Tomas Maul. Early experiments with neural diversity machines. *Neurocomputing*, 113:36–48, August 2013.

[16] DW Opitz and JW Shavlik. Generating Accurate and Diverse Members of a Neural-Network Ensemble. *Advances in neural information processing Systems*, 8:535–541, 1996.

[17] MP Perrone and LN Cooper. When networks disagree: Ensemble methods for hybrid neural networks. *Neural Networks for Speech and Image Processing*, pages 126–142, 1993.

[18] AJC Sharkey and NE Sharkey. Combining diverse neural nets. *The Knowledge Engineering Review*, 12(3):231–247, 1997.

[19] ZHZJX Wu and YJSF Chen. Genetic algorithm based selective neural network ensemble. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 201*, 1:797–802, 2001.