ANNEbot – An Evolutionary Artificial Neural Network Framework.

M.S.S. Mathotaarachchi, D.C. Perera, L. Udawatte, S. Perera
Department of Computer Science and Engineering,
University of Moratuwa
Moratuwa, Sri Lanka.

***Abstract* — A novel method for selecting the appropriate architecture and learning rule of an artificial neural network for a given application is discussed in this paper. Evolutionary Artificial Neural Networks (EANN) use the adaptation capabilities of genetic algorithms in which the natural selection process is used to attain the optimum network structure and learning algorithm for a specific task. ANNEbot is a framework which allows the combined powers of learning and adaptation of EANNs to be applied in various machine learning tasks. The framework was tested on the Iris Classification problem and the Wisconsin Breast Cancer Diagnosis problem, both of which provided results with above 90% accuracy. ANNEbot was also successfully applied on a robotic application for obstacle avoidance.**

***Index Terms*—Evolutionary Artificial Neural Networks, GA, ANN.**

I. Introduction

Over the years, the strong interest in artificial neural networks (ANNs) amongst members in the scientific community has been fueled by the many successful and promising applications in optimization problems, speech recognition, pattern recognition, signal processing, control problems, classification problems, etc. However, due to various reasons like inappropriate selections of network architecture and learning algorithms, the level of performance produced by these ANNs has been marginal. Even though there are claims from time to time that a new training algorithm has been proposed that can significantly increase performance and are better than others, there is no guarantee of the validity of these statements because most of these algorithms are proved by using them on specific applications in which the given algorithm would provide optimum performance. Therefore, finding a mechanism to select the appropriate network architecture and learning algorithm for a given task is of vital importance in order for ANNs to be useful in the field of machine learning. This paper attempts to address this issue.

It is apparent that finding the optimum ANN for a specific problem relies heavily on human expertise and practical knowledge about the different characteristics of particular networks and the problem domains for which they are most suited. However, as the problem domain becomes more complex and amount of prior knowledge reduces, this method becomes infeasible and unmanageable. In such situations, the solution is an Evolutionary Artificial Neural Networks (EANN). With no human intervention, Evolutionary Algorithms are used in EANNs to adapt the various parameters of ANNs such as connection weights, network topology and learning rules in order to find the ANN that best suits a given problem. Since the evolutionary algorithms go through all possible combinations of ANN parameters, the optimum ANN can be obtained without any prior knowledge of the problem domain or ANN characteristics. In addition to being able to adapt to any unknown problem environment, the diverse adaptability of EANNs enable it to also adapt to continuously changing, dynamic environments as well. EANNs attain these capabilities by combining the two fundamental forms of adaptability, learning and evolution.

Another advantage of EANNs over standard ANNs is that it is able to minimize if not eliminate, the presence of multiple stationary points, including multiple minima. How likely are we to encounter a sizable number of local minima? Empirical experience with training algorithms show that different initializations yield different resulting networks. Thus indicating the existence of multiple minima which in turn implies that the minima obtained is local. If the minima were global, there would be only one resulting network. Hence the issue of multiple minima is very much real. However, by using evolutionary algorithms to determine the optimum ANN, we are able to reduce the probability of this occurring. When compared with gradient descent or second-order optimization techniques that can only find local optimum in a neighborhood of the initial solution, evolutionary algorithms (EAs) always try to search for a global optimal solution.

Through ANNEbot we hope to bring the capabilities of EANNs to the above mentioned applications of ANNs. ANNEbot is an EANN framework that can be imported as a library or run as a standalone application which contains its own GUI component. Because EANNs require a significant amount of processing power and memory, ANNEbot is specifically designed to be lightweight. Due to this reason, using existing frameworks for ANNs which were more focused on performance and not weight, were nor viable [16,17,18]. Therefore ANNEbot consists of its own ANN implementation.
The framework was tested for performance and scalability using two standard classification problems, the iris classification data set and the much larger Wisconsin breast cancer diagnosis data set. ANNEbot was also applied to a robotic application in which a robot had to learn to traverse an obstacle course without colliding with any of the obstacles.

II. Evolutionary Artificial Neural Networks

A. Evolution in Artificial Neural Networks.

Evolutionary Artificial Neural Networks combine the two fundamental forms of adaptation - learning and evolution. The learning aspect is provided by the Artificial Neural Network and the evolution is provided by the evolutionary algorithm. While the ANN learns and adapts, the EA finds the optimum ANN that would be most suited for the problem and would complete the task with more efficiency. In order to achieve such a level of adaptation, the EA evolves the ANN architecture, the learning rule and the connection weights [3,4,14,15]. Currently ANNEbot is the only available EANN framework in which the architecture of the ANN is modified to best suit the application. The ANN frameworks available today, only provide weight evolution using evolutionary algorithms. The literatures available on EANNs either focus only on the theoretical aspect or uses experimental EANN implementations which are not available to the public.

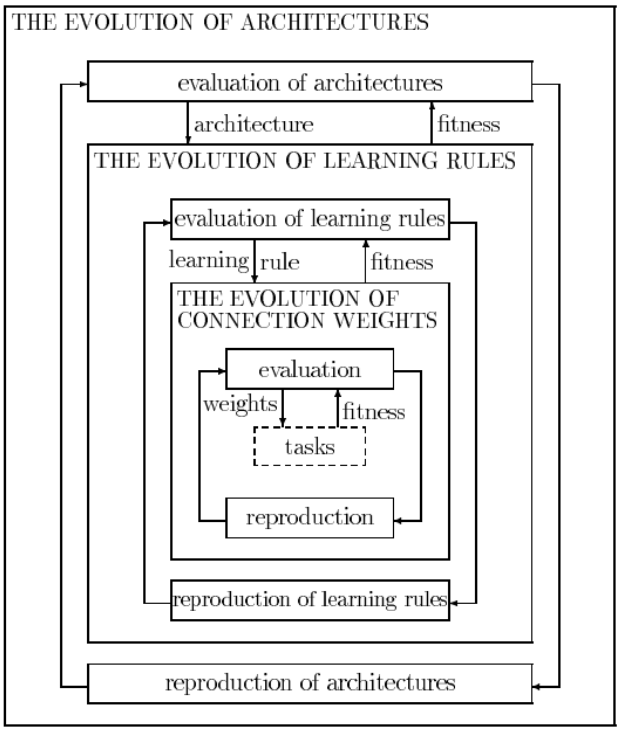

Figure 1 : Evolution in Artificial Neural Networks

There are 3 main parts in the evolution of an evolutionary artificial neural network. The evolution of connection weights act as the most primitive of them all. Its aim is to find a near optimal set of connection weights for a given ANN with a fixed architecture. The evolution of the learning rule is the evolution of the learning itself, which is similar to 'learning to learn', which in turn help us to improve EANN's learning ability. The evolution of architecture is the process of deciding its own architecture by the EANN according to the task at hand, which provides us with a powerful adaptive system.

These levels are illustrated in Figure 1 [3]. The outermost circle is the evolution of the architecture. The architecture has to be decided in order to select the Learning Rule. The connection weight evolution, which is the inner most circle, can only happen once the Architecture and Learning Rule are both fixed. The selections that are made in each of these layers affect different components of the ANN.

1) Evolution of Connection Weights.

The method in which the ANN is trained is by adjusting of connection weights. Evolutionary Algorithms can be used to get a near-optimal set of connections. The connection weights can be encoded before adding to the chromosome for the Genetic operation or can be added as real values.

2) Evolution of Learning Rule.

The learning rule is the weight updating rule which decides how connection weights are changed. The evolution of the learning rule has to be implemented such that the evolution of weight chromosomes are evolved at a faster rate, i.e. for every learning rule chromosome, there will be several weight chromosomes evolving at a faster time scale.

3) Evolution of Architecture.

The selection of the architecture effects the ANN as it may change the structure of the topology as well as various attributes of the ANN such as the number of input and output neurons, the possible existence of hidden layers etc.

However, it must also be mentioned that the order of the two outer most layers of evolution can be interchanged depending on the nature of the problem. The next section will discuss the typical cycles and how to choose the order of levels of evolution.

B. Typical Cycles of Evolution in EANNs.

The order of cycles of evolution varies depending on the nature of the problem and amount of prior knowledge available [3]. From the point of view of engineering, the decision on the level of evolution depends on what kind of prior knowledge is available. If there is more prior knowledge about EANN's architectures than that about their learning rules or a particular class of architectures is pursued, it is better to implement the evolution of architectures at the highest level because such knowledge can be used to reduce the search space and the lower level evolution of learning rules can be more biased towards this kind of architectures [4]. On the other hand, the evolution of learning rules should be at the highest level if there is more prior knowledge about them available or there is a special interest in certain types of learning rules. Therefore, as illustrated in Figure 2 [4] the highest level of evolution can be either the evolution of the Architecture or the learning rule.

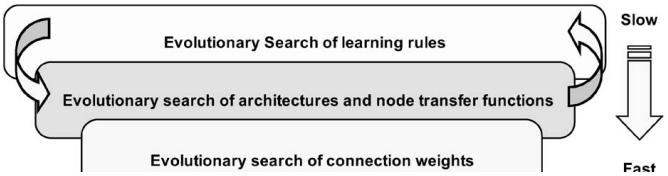

Figure 2 : Levels of Evolution

1) Evolutionary search of connection weights

A typical cycle of the evolution of connection weights:

1. Generate an initial population of N weight chromosomes. Evaluate the fitness of each EANN depending on the problem.
2. Depending on the fitness and using suitable selection methods reproduce a number of children for each individual in the current generation.
3. Apply genetic operators to each individual child generated above and obtain the next generation.
4. Check whether the network has achieved the required error rate or the specified number of generations has been reached. Go to Step 2.
5. End.

2) Evolutionary search of Architecture

A typical cycle of the evolution of architectures

1. The evolution of architectures has to be implemented such that the evolutions of weight chromosomes are evolved at a faster rate.
2. Generate an initial population of N architecture chromosomes. Evaluate the fitness of each EANN depending on the problem.
3. Depending on the fitness and using suitable selection methods reproduce a number of children for each individual in the current generation.
4. Apply genetic operators to each individual child generated above and obtain the next generation.
5. Check whether the network has achieved the required error rate or the specified number of generations has been reached. Go to Step 3.
6. End.

3) Evolutionary search of Learning Rule

A typical cycle of the evolution of Learning Rule

1. The evolution of learning rules has to be implemented such that the evolution of architecture chromosomes is evolved at a faster rate.
2. Generate an initial population of N learning rules. Evaluate the fitness of each EANN depending on the problem.
3. Depending on the fitness and using suitable selection methods reproduce a number of children for each individual in the current generation.
4. Apply genetic operators to each individual child generated above and obtain the next generation.
5. Check whether the network has achieved the required error rate or the specified number of generations has been reached. Go to Step 3.
6. End.

III. METHODOLOGIES

A. Structure of the ANN

The structure of the ANN which is used in this project is a fully feed forward network. It consists of one input layer, one output layer and a set of hidden neurons. The difference between the architecture used in ANNEbot and the standard neural network architecture is that there is no layer to which the hidden neuron belongs. The hidden neurons will organize themselves through the architecture evolution process. The differences between the two architectures are illustrated in Figure 3. This evolution in architecture happens dynamically during run time as the neural network evolves. Therefore, instead of using a readily available Neural Network library, a custom Neural Network module had to be implemented as the readily available tools lacked the support for the evolution of such dynamic architectures.

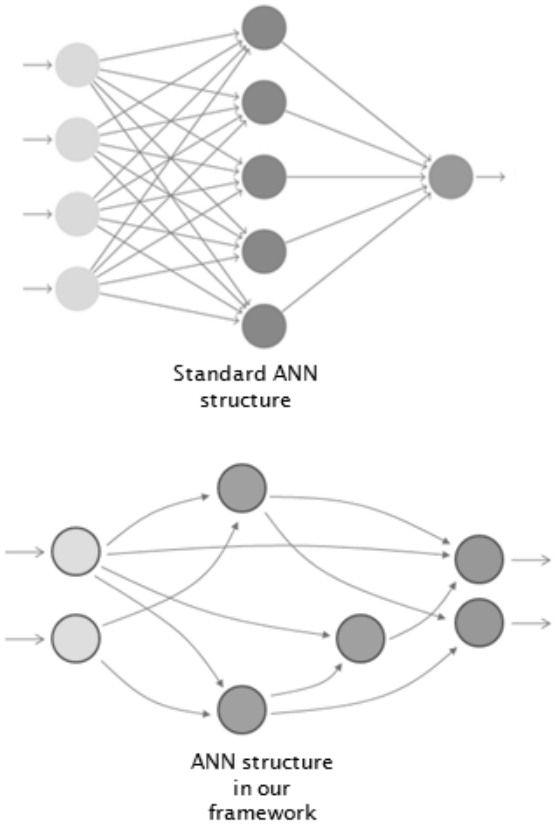

Figure 3 : Differences between the standard ANN structure and the ANN used in the framework

B. Integration of the Genetic Algorithm

JGAP [20] Java Genetic Algorithm tool was used for the implementation of the GA component of the framework. JGAP is used in such a way that the users of the EANN frame work will only have to provide the fitness function needed to evaluate the fitness of one particular chromosome. The fitness function is specific to the problem which the EANN framework is used to solve and is thus left as a component of the framework that is to be user defined.

C. Evolution of the ANN.

The activity diagram in Figure 4 illustrates the overall evolution process of the EANN in ANNEbot. The main iterations are centered on the main operations described in figure 2. ANNEbot implements two levels of evolution i.e. weights evolution and architecture evolution. Architecture evolution consists of two sub parts, namely connection reduction and hidden node addition.

The first step is to initialize the ANN with the minimum number of hidden neurons i.e. one, and make the structure a fully connected feed forward network. In the next step, this initial ANN is sent to the GA for the weights search. After the optimum weight set is obtained, it moves to the connection reduction stage in which the least weighted connection is removed and the ANN is retrained. The reasoning behind this heuristic is that the least weighted connection is assumed to be the connection with the least significance to the ANN. By removing it and obtaining the fitness, it is possible to identify if the ANN needs that connection or not. If the fitness is the same or it has improved, the ANN without that connection is passed on to the next stage of evolution. The objective of this stage is to obtain the simplest network structure for a given set of neurons that can provide the required functionality for a given problem.

After the connection reduction stage is complete the ANN goes to the hidden neuron addition stage. The heuristic used for hidden node addition is splitting the highest connected hidden node. The highest connected node is assumed to be the node representing the most number of features and so dividing these features and representing them using more hidden nodes improves the ANNs ability to converge to a more optimum solution. After a hidden neuron is added it moves to the weight search in which the optimum weights are obtained for the new ANN. After the weight search it goes to the connection reduction stage in which the optimum structure for the new set of neurons are searched. Then it compares with the old ANN and evolution continues only if the fitness of the new ANN is better than the old ANN. If the fitness is less than or equal to the old ANN, the evolution process terminates.

The user can specify a maximum number of hidden nodes. But if the fitness does not improve with the addition of a hidden node the evolution process terminates. The same applies for the connection reduction stage. If the fitness was reduced by removing a connection, removing more connections would only further reduce the fitness.

D. Simulation of a Robot using the Framework

To simulate a robot which travels from a start position to an end position, Simbad, the Java 3D robot simulator was used. When used as an application, Simbad runs as the main program while the EANN framework was used as a library.

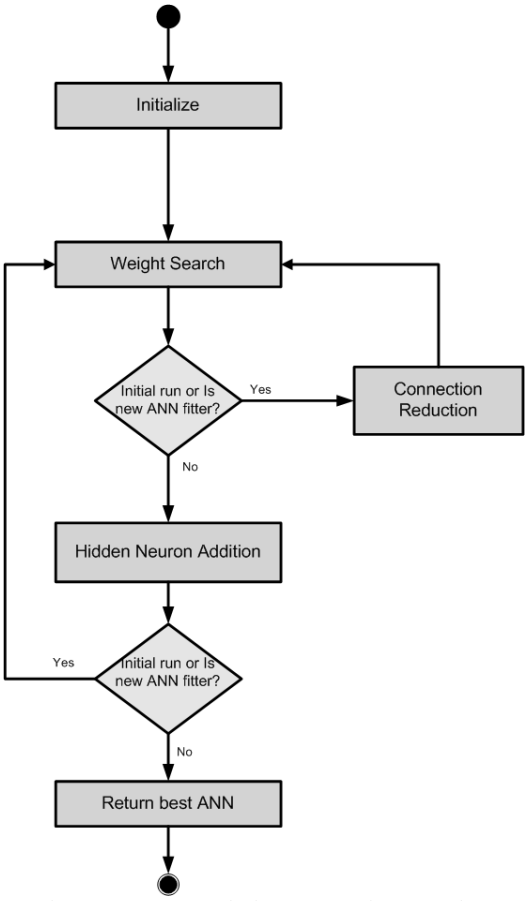

Figure 4 : ANN evolution process in ANNEbot

IV. RESULTS

A. Iris Classification Test

Fisher's Iris classification dataset[20] is perhaps the best known database to be found among pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. The data set contains 3 classes of 50 instances each (total 150), where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

Predicted attribute: Class of iris plant

Input Variables

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm

Output Classes

1. Iris Setosa
2. Iris Versicolour
3. Iris Virginica

Before the test, the full dataset was divided in to 3 sections, namely, the training set, validation set and test set with 60, 45 and 45 data instances respectively. Out of these sets, training and validation sets were used in the training process of the ANN using ANNEbot and the test set was used to measure the final performance of the resultant network.

Figure 5 illustrates the final fitness scores of the respective network. Fitness value is calculated as a percentage value from the number of correctly classified instances against the total number of data instances. The x-axis of the graph represents the corresponding network architecture. 'HN' stands for Hidden Neuron count and 'Con' stands for the number of connections in the network.

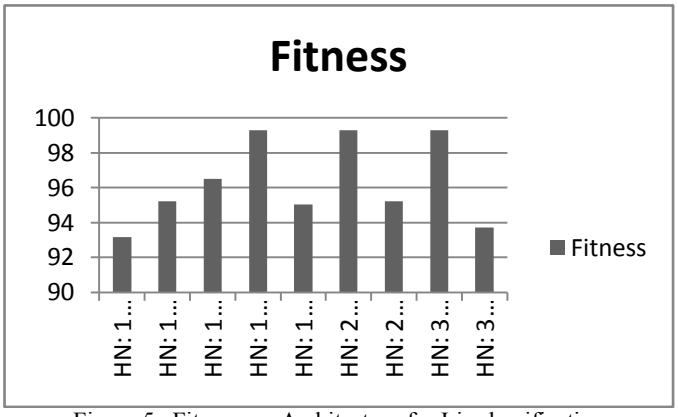

Figure 5 : Fitness vs. Architecture for Iris classification

According to figure 5, the best fitness value was reached with 1 hidden neuron with 16 connections. Although the same fitness value is reached with a higher number of hidden neurons and connections, the network with 1 hidden neuron and 16 connections is chosen as the most optimal network as it produces the best fitness with the least complexity.

Figure 6 illustrates the training, validation and testing scores of the networks against their architecture. These scores are generated as a percentage of the correctly classified instances against the total number of data instances in their respective data sets.

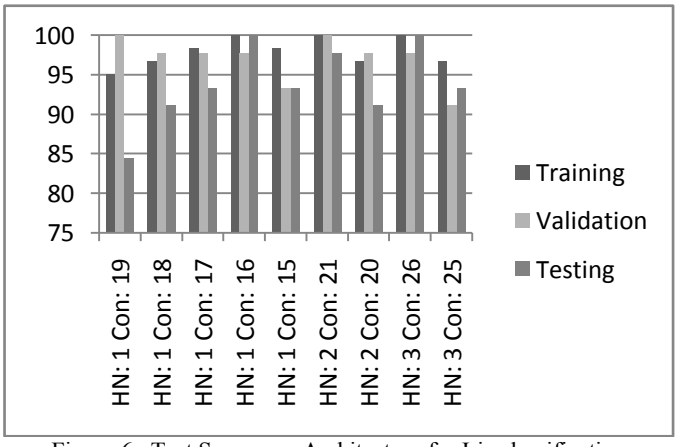

Figure 6 : Test Scores vs. Architecture for Iris classification

Figure 7 illustrates the fitness variation against the generation count of the best network which has only 1 hidden neuron and 16 connections. This graph is taken during the training process of the network which uses the genetic algorithm, and shows us the convergence of the fitness value over the number of generations.

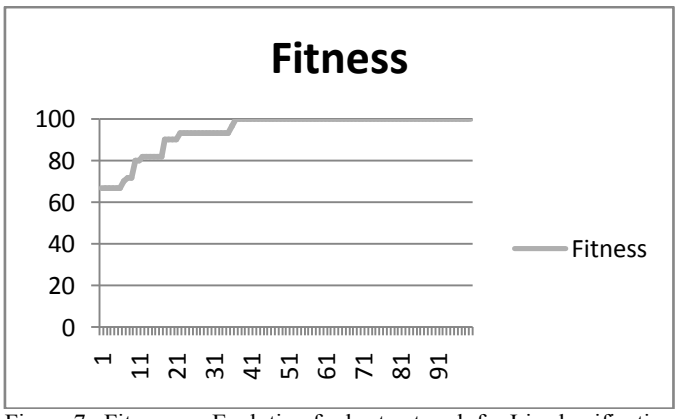

Figure 7 : Fitness vs. Evolution for best network for Iris classification

The network structure of the ANN for the best network for Iris classification is displayed below in Figure 8. From this diagram we can see that not all neurons are connected to each other and one of the inputs directly connects to the output layer without connecting to the hidden neuron in the middle. This result is in accordance with the actual mathematical properties of the iris petal classification problem.

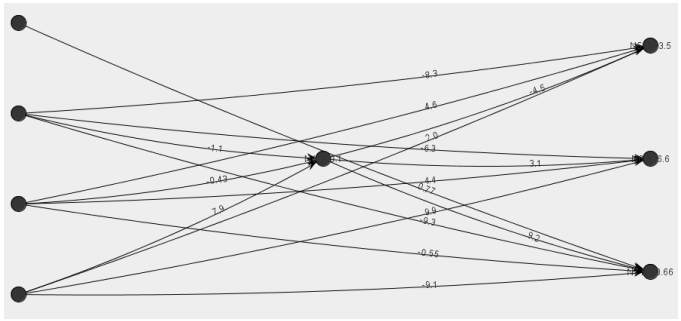

Figure 8 : Network diagram for the best network for Iris classification

Using evolutionary algorithms and heuristic methods, ANNEbot could successfully classify the Iris classification problem with satisfactory accuracy of over 99%, resulting in the optimal structure which cannot be retrieved using any other classical neural network method.

B. Wisconsin Breast Cancer Classification

For the Wisconsin dataset[20] the input features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. There are 32 attributes in this dataset including the ID number and the desired output.

Predicted attribute: Diagnosis i.e. malignant or benign

Attribute Information:

1. ID number
2. Diagnosis (M = malignant, B = benign)
3. 3-32

   Ten real-valued features are computed for each cell nucleus:

   a) radius (mean of distances from center to points on the perimeter)

   b) texture (standard deviation of gray-scale values)

   c) perimeter

d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

This dataset was also divided into three datasets as training, validation and testing with 369, 100 and 100 data instances. This test was conducted to measure the performance of ANNEbot with complex networks.

Figure 9 illustrates the final fitness scores of the respective network for the Wisconsin classification. The fitness value is calculated as a percentage of the number of correctly classified instances out of the total number of data instances.

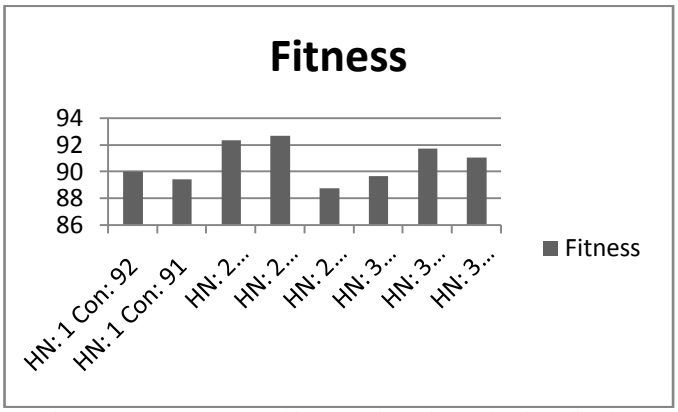

Figure 9 : Fitness vs. Architecture for Wisconsin classification

According to figure 5, the best fitness value was reached with 2 hidden neurons with 121 connections.

Figure 10 illustrates the training, validation and testing scores of the networks against their architecture for the Wisconsin classification. These scores are generated as a percentage of the correctly classified instances against the total number of data instances in their respective data sets.

Although in some architectural configurations there are individual scores (validation score of 3 hidden neurons with 151 connections) higher than that of the optimal (2 hidden neurons with 121 connections), the collective score of that configuration is lower than the optimal as shown in figure 9.

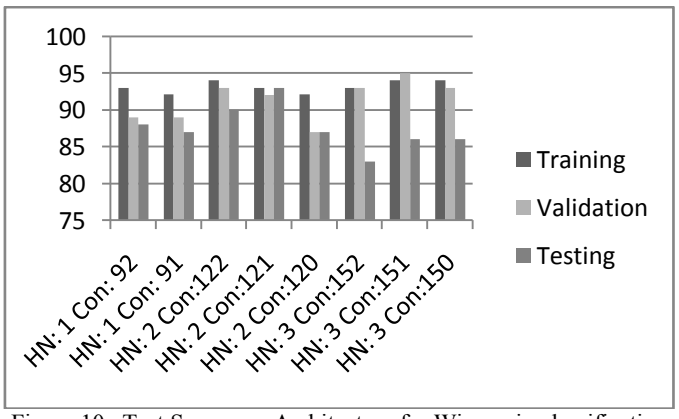

Figure 10 : Test Scores vs. Architecture for Wisconsin classification

Figure 11 illustrates the fitness variation against the generation count of the best network which has only 2 hidden neuron and 121 connections.

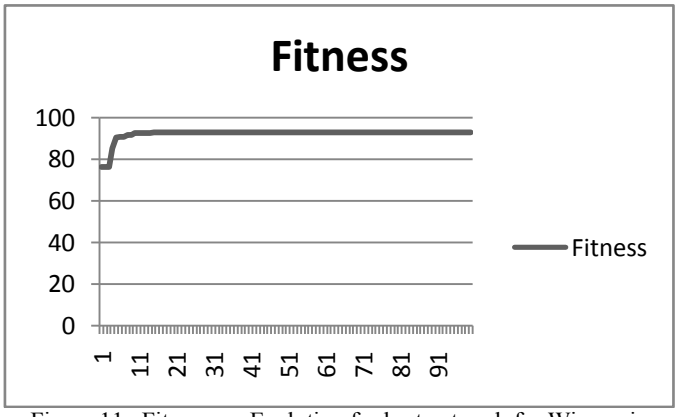

Figure 11 : Fitness vs. Evolution for best network for Wisconsin classification

The network structure of the ANN for the best network for Wisconsin breast cancer classification is displayed below in Figure 12.

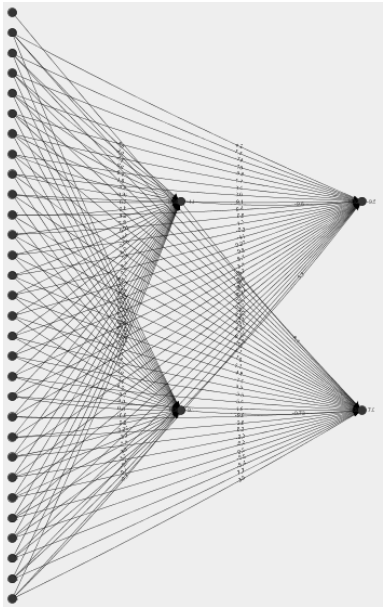

Figure 12 : Network diagram for the best network for Wisconsin classification

Although the complexity of the problem is higher than the Iris classification, for this Wisconsin breast cancer classification, ANNEbot could produce the optimal network structure with a desirable accuracy of over 90%.

C. *Robot Application Simulation*

ANNEbot was tested for its performance on robotic applications using the Simbad robot simulator. After the training of the neural network, the robot was tasked to avoid obstacles using this trained neural network. The robot did not hit any obstacles on its way around the environment. Also with a different environment, the robot gave a similar performance which indicated that the neural network is not specific to a single environment. The final neural network structure used in the robot is shown in Figure 13.

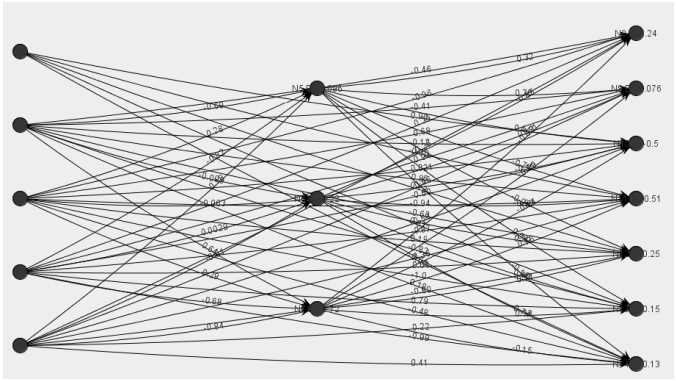

Figure 13 : Optimal Network Structure for Obstacle Avoidance in Robot Application

V. CONCLUSION

The paper discusses the use of Evolutionary Artificial Neural Networks together with heuristics to find the optimal neural network for tasks such as classification and other machine learning tasks. The results discussed in section IV, suggests that the approach can deliver highly satisfactory results when considering classification. Also the results obtained from the robot simulation suggest that an area like robotics can also benefit largely from the inherent adapting capability of evolutionary neural networks.

ANNEbot can adopt some methodologies discussed in some of the literature ([5], [8]) on evolutionary artificial neural networks, which might lead to better performance. Since these methods are still very much experimental and new, adaptation needs to be carried out with proper techniques and testing.

ACKNOWLEDGMENT

We would like to express our heartfelt gratitude to our supervisors, Prof. Lanka Udawatte and Dr. Shehan Perera, and to our project coordinator Dr. Shantha Fernando for all the support and guidance provided during the development of ANNEbot.

REFERENCES

[1] I.A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design and application," Journal of Microbiological Methods, 43, 2000

[2] S. Haykin, Neural Networks – A Comprehensive Foundation 2nd Edition: Pentice Hall, 1999, pp. 11-105.

[3] X. Yao: "Evolving Artificial Neural Networks," Proc. of the IEEE, vol. 87, no. 9, pp. 1423-1439, Sept. 1999.

[4] A. Abraham, "Meta Learning Evolutionary Artificial Neural Networks" Science Direct: Neurocomputing 56 (2004), pp. 1-38, Mar 2003.

[5] L.M. Almeida and T.B Ludermir, "Tuning Artificial Neural Networks Parameters Using an Evolutionary Algorithm," presented at Eighth International Conference on Hybrid Intelligent Systems, 2008. HIS '08, vol., no., pp.927-930, 10-12 Sept. 2008

[6] L. Fausett, Fundamentals of Neural Networks – Architecture, Algorithms and Applications: Pearson Education, 2009, pp. 30-59.

[7] Y. Katada and J. Nakazawa, "Investigation of simply coded evolutionary artificial neural networks on robot control problems," presented at IEEE congress on Evolutionary Computation, 2008. CEC 2008. vol., no., pp.2178-2185, 1-6 June 2008

[8] S. Minghui, W. Pan, H. de Garis and K. Chen; , "Approach to controlling robot by artificial brain based on parallel evolutionary neural network," presented at 2nd International Conference on Industrial Mechatronics and Automation (ICIMA), 2010, vol.2, no., pp.502-505, 30-31 May 2010

[9] L. Jian-juan, "Application of evolutionary neural networks in integrated navigation system," presented at 2nd International Symposium on Systems and Control in Aerospace and Astronautics (ISSCAA), 2008, vol., no., pp.1-5, 10-12 Dec. 2008

[10] Y. Guo, L. Kang, F. Liu, H. Sun and L. Mei, "Evolutionary Neural Networks Applied to Land-cover Classification in Zhaoyuan, China," IEEE Symposium on Computational Intelligence and Data Mining, 2007. CIDM 2007, vol., no., pp.499-503, March 1 2007-April 5 2007

[11] D. Floreano and F.Mondana, "Automatic creation of an Autonomous agent: Genetic evolutions of a Neural Network driven robot ," unpublished.

[12] S. Baluja, " Evolution of an Artificia1 Neural Network Based Autonomous Land Vehicle Controller," lEEE Transactions on Systems, Man, and Cybernetics-part B Cybernetics, vol. 26, no. 3, June 1996

[13] S. Barber, "AI: Neural Network for beginners," (2007, May 19). [Online] Available: http://www.codeproject.com/KB/recipes/NeuralNetwork_1 [Accessed: 2010, Sept 30].

[14] S. Pal, S. Vipsita and P.K. Patra, "Evolutionary approach for approximation of artificial neural network," presented at IEEE 2nd International Advance Computing Conference (IACC), 2010, vol., no., pp.172-176, 19-20 Feb. 2010

[15] X. Yao and M.M. Islam, "Evolving artificial neural network ensembles," Computational Intelligence Magazine, IEEE , vol.3, no.1, pp.31-42, February 2008

[16] J. Pitt, "OpenCog Wiki - The Open Cognition Project," (2010, Aug 22). [Online] Available: http://wiki.opencog.org/w/The_Open_Cognition_Project [Accessed: 2010, Sept 29].

[17] J. Heaton, "Encog Java and DotNet Neural Network Framework,"(2009) [Online] Available: http://www.heatonresearch.com/encog [Accessed: 2010, Sept 29].

[18] D. Hudson and M. Cohen, Neural Networks and Artificial Intelligence for Biomedical Engineering : Wiley IEEE press, 1999, pp. 14-90.

[19] Klaus Meffert, "JGAP: Java Genetic Algorithms Package,". [Online] Available: http://jgap.sourceforge.net/ [Accessed: 2010, Sept 27].

[20] A. Asuncion and D. Newman. (2007) UCI Machine Learning Repository. [Online]. http://archive.ics.uci.edu/ml, [Accessed: 2010, Oct 30]