ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TOWARDS AN AUTOMATIC GENERATION OF NEURAL NETWORKS

[1]MAHA MAHMOOD, [2]BELAL AL-KHATEEB
[1]College of Computer Science and Information Technology, University of Anbar, Ramadi, Iraq
[2]College of Computer Science and Information Technology, University of Anbar, Ramadi, Iraq
[1]maha_882010@yahoo.com, [2]belal@computer-college.org

ABSTRACT
The automatic generation of neural network architecture is a useful concept that is used in many application areas despite the optimal architecture not being known a priori. Therefore, trial and error is often performed before a satisfactory architecture is found. Construction deconstruction algorithms can be used as an approach, but they have several drawbacks. Usually this approach is restricted to a certain subset of network topologies and as with all hill climbing methods, they often get stuck at local optima and may therefore not reach the optimal solution. In order to overcome these limitations, an evolutionary computation as an approach to the generation of neural network structures is used. The aim of this paper is to design an automatic generation of neural networks architecture that performs random operations within hidden layers. Associated operations include generating layer, add node, delete layer, delete node and keep the architecture with no change, at which all the weights are initialized randomly. These neural networks are able to adapt themselves with the reality, learn from the training of the various applications and adapt their architecture depending on the uses of the application. The automatically obtained neural network architecture is much better than all other architectures that are found during the evolutionary process. This network is tested in the game of tic tac toe and is played against selected tic tac toe computer programs and against selected human players and the obtained results are promising, suggesting many other research directions.

Keywords: *Neural Network, Evolutionary Algorithms, Genetic Programming, Genetic Algorithms*

1. INTRODUCTION

The neural network performance depends on the neural network's architecture and on the given task that the network performs. This performance includes some properties like the ability of generalization and learning speed. It is usual to use trial and error to find a suitable neural network architecture for a given problem but this method is time consuming and may not produce an optimal network. As with most of the applications of evolutionary computation in the generations of neural network architecture on our world, there is a significant influence on the performance of the network therefore using the evolutionary computation and it is considered to be a step towards automation of the neural network architecture's generation [1]. The aim of this research is to seek a preferable solution for any problem by performing random operations on the neural network architecture, of which operations include adding layer, adding node, deleting layer, deleting node, updating weights and keeping the architecture with no change. This paper focus on using evolution strategies to automatic generation of neural network architecture in order to evolve neural networks to play Tic Tac Toe. As mentioned before, the objective of this work is to propose a structure of learning methodologies for the game of Tic Tac Toe and to produce a better player.

The rest of the paper is organized as follows. In Section 2, background is presented. The experimental setup is described in Section 3. Section 4 presents our results and Section V5 concludes the paper together with some suggestions for future work.

2. RELATED WORK

The work on novel methods for the evaluation of machine translation systems were related to automatic ANN construction [23]. This was done by allowing the evolution of solutions to dynamic domains, a Dynamic Structured Grammatical Evolution (DSGE) solves a limitation of other Grammar based Genetic Programming (GGP) methodologies, such as ANNs, where there is the need to know the number of neurons available in previous layers so that valid individuals are generated. DSGE, is able to evolve the topology and weights of one hidden layered ANNs that perform statistically best than those evolved using GE or SGE. (DSGE) represents a new genotypic representation that overcomes the aforementioned limitations. The enabling of creation of dynamic rules that specify the connection possibilities of each neuron is done by using a methodology that enables more than one output neuron with the evolution of multi-layered ANNs. But difficult datasets are needed to better analyses the evolution of ANNs with more than one hidden layer.

Weizhong Yan.[24] attempted to develop an automatic ANN modeling scheme that is based on a special type of network, the Generalized Regression Neural Network (GRNN). The research used an automatic modeling scheme for time-series forecasting in effective manner, by introducing several design strategies. Because of the designing for automatically modeling is a large-scale time series, the main limitation of the proposed modeling scheme is that the performance of the model may not always be superior for certain time series over other models that are carefully handcrafted specifically for the series. Note that not all of the design parameters in the model are optimally chosen, but rather designed in ad hoc fashion. Nadi [25] used an evolution of neural network architecture and weights by mutation based genetic algorithm. The researcher presents a new approach for evolving optimized neural network architecture for a three-layer feedforward neural network with a mutation based genetic algorithm. This evolution optimized the weights and the network architecture simultaneously through a new presentation for the three layers feedforward neural network. Limitation of this work throw the testing of the algorithm on three data sets and comparing with Mutation-Based Genetic Neural Network (MGNN) shows that the used method has a higher resolution in finding the answer and a higher convergence speed. Jenkins [7] used a neural network weight training by mutation. The mutation in integer variables produces a progressive ‘shift’ of the center of the range of positive/negative values provided for selection in order to give the algorithm freedom to select weights from an unlimited range of values. At each iteration, the range of integer values offered to the algorithm is randomly selected. Vonk [1] used an evolutionary computation for the automatic generation of neural network architecture. The researcher presents a brief introduction to the field as well as an implementation of automatic neural network generation using genetic programming. The work reports an application of evolutionary computation in the automatic generation of neural network architecture. The using of evolutionary computation is a step towards automation of neural network architecture’s generation.

In this paper, Processing information of neural network is similar to human brain processing. It is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. The proposed neural network has the ability to learning with any application we design an automatic generation of neural networks architecture that perform random operations within hidden layer those operations include generating layer, add node, delete layer, delete node and keep the architecture with no change, at which all the weights are initialized randomly. The goal of this paper is to gain an optimal performance for neural networks architecture that is a kind of a fully connected feedforward neural network. This search deals with neural networks trained by an evolutionary algorithm in order to automatically generate the best possible neural network architecture that can perfectly play tic tac toe games. The most important operators are mutation, crossover and combination. The new population is produced and its fitness will be computed. This work continues until a desired answer or the maximum epoch assigned is reached.

3. BACKGROUND

Work on artificial neural networks, generally referred to as Neural Networks (NN), was motivated from the very beginning of its inception by the recognition that the brain computes in a fully different way from traditional digital computers [2]. A single neuron acts as fuse to change the flow of information propagating through the NN. To produce the final resulting output, neurons will strengthen some signals and limit others. In a brain, neurons gathering inputs like water flowing into a dam, it empties everything into its outputs and starts over again, when the water attains a certain predefined level [3].

In order to solve any specific problem, when there may be many different Neural Networks (NN) available, network designers usually face questions like "How could the size of a NN be found?", “Is the selected architecture an appropriate one?” and “How can an optimal network be designed?”. These questions usually lead the designer to optimization methods that can find the desired network. Scientists tend to use random search methods like evolutionary algorithms (EAs) and genetic algorithms (GA) to find an optimal network in order to avoid the local minima encountered in most of the optimization methods [4]. The use of evolutionary programming for adapting the design and weights of a multi-layer feed forward perceptron in the context of machine learning. Specifically, it is desired that the structure and weights of a single hidden layer perceptron to evolve such that it can achieve a high level of play in the game of tic-tac-toe without the use of heuristics or credit assignment algorithms So, this neural network is different from previous work which structurally having one hidden layer with (1-10) node; and in initial population network the previous network was initialized with 50 parents network [5].

4. NEURAL NETWORKS AND EVOLUTIONARY ALGORITHMS

Neural networks are widely used in applications such as pattern recognition, classification, clustering, prediction, among others. These networks are trained using application data. The ability of generalization in these networks depends on the training, architecture, the number of layers and the number of neurons in each layer. The network attracts to over fit the training set [6]. When the number of neurons in the network is increased, the interpolation capability will be decreased; in other words, the network cannot learn all the data if the number of neurons is less than the necessary number of neurons. Therefore, for every application, there are a particular number of neurons which maintain the best interpolation generalization balance. The designers need some methods for finding the suitable choice for keeping this balance [7].

EAs are types of random search algorithms that use natural evolution to solve optimization problems. There are different categories in EAs, such as genetic algorithm, evolutionary strategy, evolutionary programming and genetic programming [14]. An EA is applied to residents, which is a presentation of the optimization problem. The representation of problem could be as complicated as a computer program or as simple as a series of 0s and 1s that the primary residents could be defined as fully random or built on predated knowledge [16]. This algorithm will access the population based on specifying how much each agent is close to the goal of the problem and is based on a goal function as well. There is a difference for each individual problem in the goal function that should be defined by the user [7]. There are many ways to produce the next generation of the solutions from current population. One popular method for the next generation is to select the parents with better fitness to produce the next generation [12]. There are many operations, which are applied to chromosomes, they are called genetic operations. The most important operators are mutation, crossover and combination. The new population is produced and its fitness will be computed. This circle continues until a desired answer or the maximum epoch assigned is reached [14].

Random initialization of connection weights, when the estimate architectures are evaluated, is the first source of the noisiness because of to the fact that different random initial weights may generate different training outcomes. The training algorithms used for the evaluation generates the second source of noise. Even with the same set of initial weights, various training algorithms may generate various training results. To solving these Mistakes, synchronization evolution of both the architecture and weights is recommended [18]. There have been number of studies on evolving architectures and connection weights synchronization. An evolutionary system called NEAT. NEAT is based on three principles that work together to efficiently Development network topologies and weights. The first principle is Symmetry: NEAT encodes each node and connection in a network with genes. Whenever a structural mutation returns in a new gene, that gene receives a historical marking. Historical markings are Used to follow up Symmetric genes during crossover, and to Determine compatibility operator. The second principle is protecting innovation. A compatibility operator is used to spectate the population, which protects innovative solutions and prevents incompatible genomes from crossing over. Finally, NEAT follows the philosophy that search should begin in as Youngest space as possible and expand gradually. Evolution in NEAT always begins with a population of minimal structures. Structural mutations add new connections and nodes to networks in the population, leading to incremental growth. Topological innovations have a chance to realize their potential because they are protected from the rest of the population by speciation. Because only useful structural additions tend to survive in the long term, the structures being optimized tend to be the minimum necessary to solve the problem. NEAT‟s approach allows fast search because the number of dimensions being searched is minimized [18].

5. BAYESELO

The Elo rating system [7] was originally used for Chess, but it is now used for many other games like football (http://www.eloratings.net/system.html). Each player (or team) is given an initial rating, and after playing each other, their rating changes as a function of their current rating and whether they win, lose or draw. Elo says that the expected result of a game is a function of the difference in rating between two players.

$$E = 1/(1 + 10\hat{}(\frac{D}{400})) \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

Where E is the expected result and is the rating difference between two players and D is the rating difference between two players.

Elo assumes that there is a single value that can represent a player's strength and subsequently the expected result can be determined according to the above formula. One of the problems with Elo is that one cannot take a set of game results and produce a ranking of all relevant players.

In this paper, we use a modified version of Elo, called Bayeselo [8], to compare various evolved players. Bayeselo finds a likelihood of superiority (LOS), using a minimization–maximization algorithm [9]. Tables 1 and 2 show an example of Bayeselo estimates and LOS. Software can be downloaded from [7], which enables all of the necessary calculations to be made.

The disadvantage of the Elo rating is that it does not take in consideration the uncertainty that is found in the player's performance. While Elo advantage indicates the advantage of playing first. Elo Draw indicates how likely draws are. The default values in the program were obtained by finding their maximum-likelihood values over 29,610 games of Leo Dijksman's WBEC [9].

Table 1: BayesElo Ratings Example.

Rank	Name	Elo	+	-
1	P1	598	156	123
2	P2	345	45	98

Table 2: BayesElo Los Example.

Name	P1	P2
P1	-	87
P2	12	-

Table 1 shows the Elo rating for two players after they have played a number of games (not necessarily against just each other). The columns show the true rating at a 95% confidence level (one can change the confidence level in the program, but we leave it at 95% for all the results reported in this paper). Taking P1 as an example, its Elo rating is 598 and its true value is between (598- 123) 475 and (598+ 156) = 754, at the 95% confidence level.

The values in Table 2 shows how the supremacy of players is reported using Bayeselo as Table 2 shows that there is 87% likelihood that P2 is stronger than P1 and a 12% likelihood that P1 is stronger than P2. In this paper, we use the LOS as the main statistical measure of the superiority of one player over that of another. If this value is around 50%, we assume that the players' performances are equal. If the value is above 70%, we assume that the players are statistically different at the 95% confidence level. It is also worth noting (for the purposes of reproducibility) that we change the advantage parameter to zero when running Bayeselo. The default is +32, which is used for chess (representing the advantage white has) but this is not applicable to Tic Tac Toe so we set this parameter value to zero.

6. NEURAL NETWORK'S OPERATIONS

The neural network architecture has an input layer with any number of neurons that can be specified by the user (problem dependent). The work includes an initial network architecture starts with one to three hidden layers, each one with three to twenty of neurons. Finally, the network has an output layer with one neuron. ANN are used as nonlinear function approximators, which is the activation function (sigmoid transfer function), and the multilayer feedforward neural network is the most common type of ANN [10]. The type of structure used may be based on some knowledge of the problem domain but commonly a sufficient network structure is found by trial and error, in many cases the structure used will be a fully connected feed forward network and the user might try different numbers of hidden neurons to see how well the resulting structures will fit the task [13].

In order to get an efficient architecture of multi-layer perceptron neural network with high extension rate in making decisions, three types of operations (add, delete, and no change) were applied as shown in figure 1. These operations have as ever impact on the performance of neural network

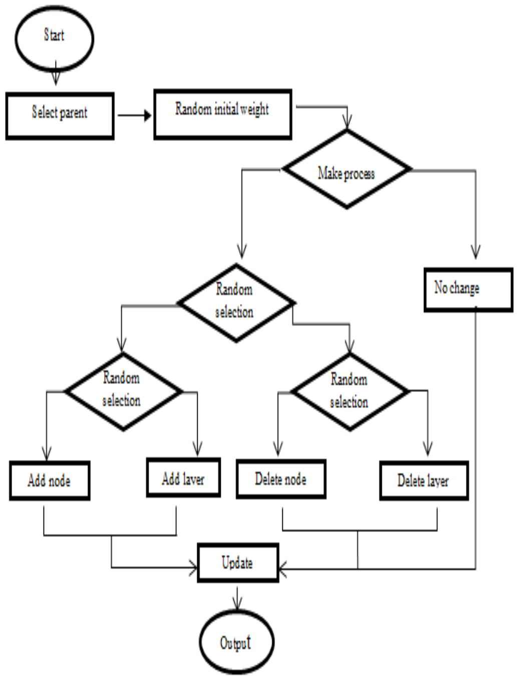


Figure 1: ANN Operations.

Algorithm 1 is proposed to get the neural network architecture automatically, while Algorithm 2 shows the training algorithm for the automatic generation of the neural network architecture that will be able to play the Tic Tac Toe game.

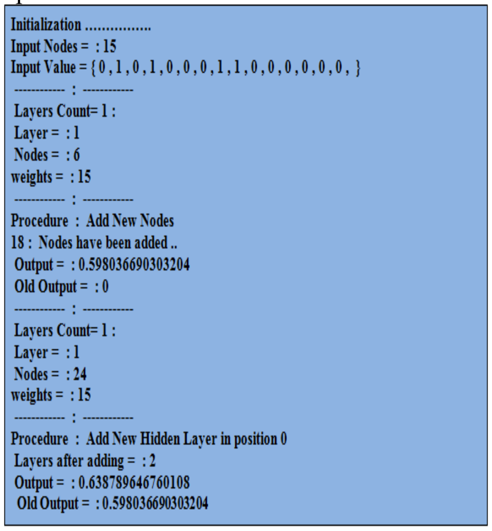
Algorithm 1:

1. For each network do the following:
   - With 1/3 probability choose the add operation then go to step 2.
   - With 1/3 probability choose the delete operation then go to step 3.
   - With 1/3 probability keep the architecture as it is then go to step 5.
2. - With 1/2 probability choose a hidden layer to be added to the neural network at a random position, such that the total number of the hidden layers should not exceed the maximum number of hidden layers in the network. If this is not the case, then do not add the new hidden layer. Then go to step 4.
   - With 1/2 probability randomly choose a hidden layer in order to add a number of nodes to it, such that the total number of the nodes does not exceed the maximum number of nodes in that hidden layer. If this is not the case, then only add the number of nodes that makes the total number of nodes equal to the maximum number of nodes in that hidden layer. Then go to step 4.
3. With 1/2 probability choose a hidden layer to be deleted from the neural network at a random position, such that the total number of the hidden layers should not be less than the minimum number of hidden layers in the network. If this is not the case, then do not delete the hidden layer.
   - With 1/2 probability randomly choose a hidden layer in order to delete a number of nodes from it, such that the total number of the nodes is not less than the minimum number of nodes in that hidden layer. If this is not the case, then delete the number of nodes that makes the total number of nodes equal to the minimum number of nodes in that hidden layer.
4. Reconnect the neural network's nodes and layers.
5. Update the neural network's weights.
6. Calculate the output (fitness) of each network.

Algorithm 2:

1. Population of 30 neural networks was initialized at random. All of the weights and biases of each network were initialized uniformly over [-0.5, 0.5].
2. Each strategy has an associated self-adaptive parameter vector Si, i=1,…,n initialized to 0.05.
3. Each neural network is playing with all other neural networks.
4. Games are played until either side wins, or a draw is declared.
5. For each game, the player receives a score of +1 for a win, -1 for a draw and 0 for a loss.
6. After completing all games, the best 15 players that have the highest scores are selected as parents and retained for the next generation. Those parents are then copied and mutated to create another 15 players.
7. Repeat step 3 to 6 for 1000 generation.

In order to get an efficient architecture of multi-layer perceptron neural network with high extension rate in making decisions, three types of operations (add, delete and no change) were applied. These operations have a severe impact on the performance of neural network. Figure 2 shows sample results of the used operations.



```
Initialization ...............
Input Nodes = : 15
Input Value = { 0 , 1 , 0 , 1 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , }
----------- : -----------
 Layers Count= 1 :
 Layer = : 1
 Nodes = : 6
weights = : 15
----------- : -----------
Procedure  :  Add New Nodes
18 :  Nodes have been added ..
 Output = : 0.598036690303204
 Old Output = : 0
----------- : -----------
 Layers Count= 1 :
 Layer = : 1
 Nodes = : 24
weights = : 15
----------- : -----------
Procedure  :  Add New Hidden Layer in position 0
 Layers after adding = : 2
 Output = : 0.638789646760108
  Old Output = : 0.598036690303204
```

Figure 2: Add Operation (Add New Nodes And Hidden Layer).

The sample of neural network's initialization as shown by figure 2, which contains one input layer, one hidden layer and one node output layer. All the weights of the neural network are initially assigned with random number with range [-0.5, 0.5], while all biases are set to 1. This is the process of adding layer in neural network concerning hidden layers only. The decision of adding new layer may occur at any stage of the processing. Each new layer will have a random number of nodes and random connection weights and it will be fully connected to previous and next layer. The new layer will be added only if the number of hidden layers is less than three. And In this case, a new node(s) is added to the neural network. It added a random node with random weight into randomly selected hidden layer. The new added node will be fully connected to the nodes in the next layer. The new node will be added only if the number of nodes in the specified hidden layer is less than twenty. In order to get better results, as in adding process, neural network may make decision of deleting a hidden layer and reconnect neural nodes again according to the number of the layer and nodes within each layer within the neural network. The layer will be deleted only if the number of hidden layers is greater than one. In some situation, deleting one node or more is required to get better result than deleting the whole layer of neural network, thus neural network can delete one randomly selected node or more from a randomly selected hidden layer and then reconnect neural network accordingly. The last decision on neural network is updating the weight for each one of neural network without adding or deleting processes. Finally, output is a layer of the neural network, which represents the value of the network after a number of mathematical operations on neural network to calculate the final product by using the activation function.

7. THE RESULTS OF THE EVOLUTIONARY TIC TAC TOE PLAYER

The algorithm for the automatic generation of the neural network architecture for the tic tac toe player is executed for 1000 generations and the best player from each 100 generations is taken. Each evolved neural player is played (starting first) ten tic tac toe games against all other players in order to ensure that the learning strategies that are used to construct neural network players are consistent. So, each player plays 10 games, against nine other players, making a total of 90 games for each player. Tables 3 through 5 show the obtained results.

Table 3: Number of Wins for the Evolved Tic Tac Toe Players (Row Player) Out of 90 Games.

Player	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	Total
N_1	-	2	1	2	1	1	1	0	1	1	10
N_2	2	-	1	2	0	1	1	1	1	0	9
N_3	2	0	-	1	0	1	2	1	1	0	8
N_4	2	2	2	-	1	1	1	1	1	0	11
N_5	3	1	1	2	-	1	2	2	0	1	13
N_6	2	2	2	3	1	-	1	1	1	1	14
N_7	1	3	2	1	2	2	-	1	1	1	14
N_8	3	3	2	2	2	2	1	-	1	2	18
N_9	2	3	2	2	2	1	2	1	-	1	16
N_{10}	3	3	2	3	2	2	2	1	2	-	20

Table 4: Number of Loses for the Evolved Tic Tac Toe Players (Row Player) Out of 90 Games

Player	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	Total
N_1	-	2	2	3	2	3	2	3	2	2	22
N_2	2	-	1	2	1	2	2	2	2	2	16
N_3	2	1	-	3	2	1	2	1	2	2	16
N_4	2	2	1	-	2	1	1	1	2	2	14
N_5	1	1	2	2	-	1	1	2	1	2	13
N_6	2	1	1	2	0	-	2	2	1	2	13
N_7	0	1	1	1	2	1	-	2	2	3	13
N_8	0	1	2	1	1	2	1	-	2	2	12
N_9	1	2	1	2	2	1	1	2	-	1	13
N_{10}	0	1	0	0	1	2	1	2	1	-	8

Table 5: Total Number of Wins, Draws and Loses

Player	Win	Draw	Lose
N_1	10	58	22
N_2	9	65	16
N_3	8	66	16
N_4	11	65	14
N_5	13	64	13
N_6	14	63	13
N_7	14	63	13
N_8	18	60	12
N_9	16	61	13
N_{10}	20	62	8

The results in tables 3 through 5 show variety of players' performances according to wins, loses and draws indicating that the 10th (N_{10}) neural network was the best player among all other players as it has the most number of wins and the least number of loses. Therefore, N_{10} is considered as the best obtained architecture. Also, there is one another interested thing to be noticed from the obtained results, that is the number of wins are almost gradually increased (except for few cases) and the number of loses are almost gradually decreased (except for few cases) within the learning time, which clearly indicates the occurring of the learning process.

In order to be more assured about those indications, the Bayeselo method is applied to the obtained results as shown in tables 6 and 7. Table 8 shows the number of layers in neural network and the nodes number for each layer.

Table 6: BayesianElo Ratings for the Evolved Tic Tac Toe Players.

Rank	Name	Elo	+	-	Games	Score	Oppo.	Draws
1	N_{10}	33	48	47	90	57%	-4	70%
2	N_8	23	48	47	90	55%	-3	70%
3	N_9	18	47	47	90	54%	-2	72%
4	N_6	8	47	47	90	52%	-1	72%
5	N_7	3	47	47	90	51%	0	70%
6	N_5	2	47	47	90	51%	0	72%
7	N_3	-17	47	47	90	46%	2	74%
8	N_4	-18	48	48	90	46%	2	68%
9	N_2	-26	47	48	90	44%	3	69%
10	N_1	-26	48	48	90	44%	3	67%

Table 7: LOS for the Evolved Tic Tac Toe Players

Player	N_{10}	N_8	N_9	N_6	N_7	N_5	N_3	N_4	N_2	N_1
N_{10}	-	60	66	76	80	80	92	92	94	94
N_8	39	-	55	66	71	71	87	87	91	91
N_9	33	44	-	61	66	67	83	83	88	88
N_6	23	33	38	-	55	56	76	75	82	82
N_7	19	28	33	44	-	50	71	71	78	78
N_5	19	28	32	43	49	-	71	71	78	78
N_3	7	12	16	23	28	28	-	50	59	59
N_4	7	12	16	24	28	28	49	-	58	58
N_2	5	8	11	17	21	21	40	41	-	50
N_1	5	8	11	17	21	21	40	41	49	-

Table 8: The Layers and Nodes for Neural Networks Training

Neural Network Number	Number of Layers	Number of Nodes in each layer
N_1	1	15
N_2	1	10
N_3	3	8-6-2
N_4	1	11
N_5	2	11-8
N_6	2	2-2
N_7	2	3-2
N_8	2	8-5
N_9	2	2-1
N_{10}	2	4-3

The results in table 6 indicate that N_{10} is clearly better than the first seven players (N_1-N_7) as the LOS between them is above 70%. Although N_{10} is 60% better than N_8 and 66% better than N_9, we consider this acceptable as N_8 and N_9 have more training (have closer performance) than N_1-N_7. So, based on the above results there is statistical difference between the players. We decided to choose N_{10} player (best neural network architecture) to be our baseline player. Also, the results show there was change in the architectures of the neural networks, as well as the changes in the training weights, during the training process. The results show that N_{10} is too better than N_1 with performance ratio reach of 94%, this is because that N_1 has not enough training (100 generations only). N_{10} is also better than the N_2 neural network, which their architect formed after 200 generations, have the same LOS as compared with N1 (94%) with a little change in their nodes and N_3 and N_4 start to have gradual training. In other word; it can increase their ability to choose the probable neural network architectural. N_{10} still has higher ratio than them (92%). Therefore, they have a little better result than N_1 and N_2.

The obtained results show that N_5, N_6 and N_7 have good efficiency by increasing the number of training cycles. Thus, the neural networks become more active in choosing the best architecture. N_9 is of comparable performance with N_{10}, the closet result to the previous tested neural networks, have the ratio 66%. Although the N_8 has lower training time than N_9 but it is better than the previous one in choosing their architectural design and how to play against the opponent. This is a clear indication that the changing in the architecture leads to better performance. N_{10} as shown from above is the best architectural of neural network which had achieved after 1000 training cycle through 7532 minutes of training time. The architecture of N_{10} is shown in figure 3.

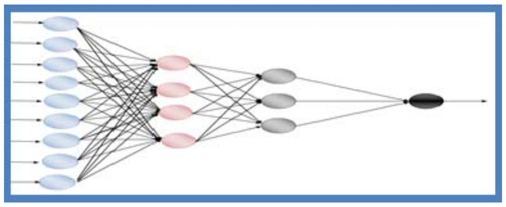


Figure 3: Architecture of N10 Neural Network.

The results in Table 7 show that N_{10} is too better than N_1 with performance ratio reach of 94%, this is because that N_1 has not enough training (100 generations only). Therefore, N_1 is considered as the first level of training for N_{10}. Figure 4 shows the architecture of N_1.

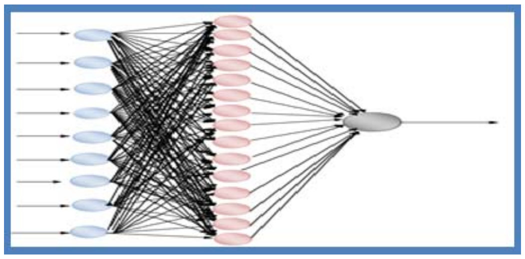


Figure 4: Architecture of N1 Neural Network

The obtained results show that N_5, N_6 and N_7 have good efficiency by increasing the number of training cycles. Thus, the neural networks become more active in choosing the best architecture. Figure 5 shows the N_5 neural networks architecture.

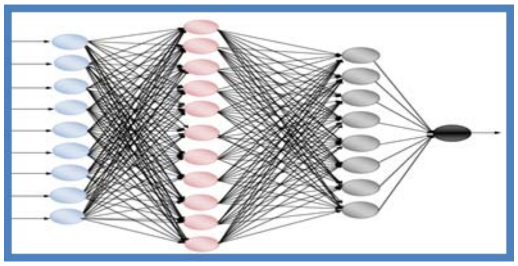


Figure 5: Architecture of N5 Neural Network

Table 7 shows that N_9 is of comparable performance with N_{10}, the closet result to the previous tested neural networks, have the ratio 66%. Figure 6 shows their architecture.

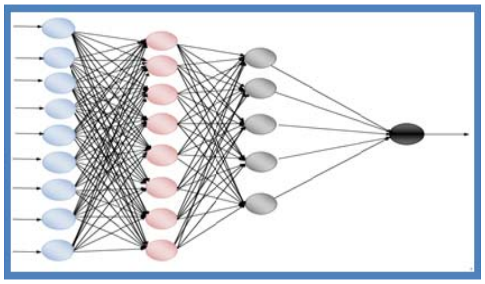


Figure 6: Architecture of N9 Neural Network

8. THE RESULTS OF N_{10} AGAINST ONLINE TIC TAC TOE PROGRAMS

The performance of N_{10} is tested against ten selected online tic tac toe programs. Those programs are with different playing levels (easy, medium and difficult). N_{10} is played 100 matches against each online program; Table 9 summarizes the results while tables 10 and 11 show the testing of those results using Bayeselo method.

Table 9: Number of wins for The Evolved N10 with online Players (Row Player) Out of 100 Games

	Opponent: N_{10}			
Player	Total	Win	Draw	Lose
O_1	100	12	81	7
O_2	100	10	85	5
O_3	100	12	83	5
O_4	100	11	85	4
O_5	100	11	83	6
O_6	100	8	78	14
O_7	100	13	78	9
O_8	100	10	84	6
O_9	100	12	78	10
O_{10}	100	12	81	7

Table 10: BayesianElo Ratings for the Evolved N10 with the Online Players.

Rank	Name	Elo	+	-	Games	Score	Draws
1	O_6	23	45	45	100	53%	78%
2	N_{10}	8	14	14	100	52%	82%
3	O_9	3	45	45	100	49%	78%
4	O_8	-1	44	44	100	48%	84%
5	O_7	-2	45	45	100	48%	78%
6	O_2	-4	44	44	100	48%	85%
7	O_5	-4	44	44	100	48%	83%
8	O_1	-4	44	44	100	48%	81%
9	O_{10}	-4	44	44	100	48%	81%
10	O_4	-8	44	44	100	47%	85%
11	O_3	-8	44	44	100	47%	83%

Table 11: LOS for the Evolved N10with the Online Players.

Player	O_6	N_{10}	O_9	O_8	O_7	O_2	O_5	O_1	O_{10}	O_4	O_3
O_6	-	72	71	75	75	77	77	77	77	81	81
N_{10}	27	-	57	65	65	68	68	68	68	75	75
O_9	28	42	-	55	55	57	57	58	58	63	63
O_8	24	34	44	-	50	52	52	52	52	58	58
O_7	24	34	44	49	-	52	52	52	52	57	57
O_2	22	31	42	47	47	-	50	50	50	55	55
O_5	22	31	42	47	47	49	-	50	50	55	55
O_1	22	31	41	47	47	49	49	-	50	55	55
O_{10}	22	31	41	47	47	49	49	49	-	55	55
O_4	18	24	36	41	42	44	44	44	44	-	50
O_3	18	24	36	41	42	44	44	44	44	49	-

The results in Table 11 show that N_{10} is statistically better than O_3 and O_4 as it has 75% level of superiority over them. Also, Table 9 shows that N_{10} is 68% better than O_1, O_2, O_5 and O_{10}, while it is 65% better than O_7 and O_8. As the 65% and 68% is closer to 70% (the threshold for determining the level of superiority) than 50% (the threshold for determining the level of equality), so N_{10} can be considered as statistically better than O_1, O_2, O_5, O_7, O_8 and O_{10}.

The results in Table 11 show that N_{10} is of comparable performance with O_9 as it is only 57% better than O9. Finally, Table 9 indicates that N_{10} is statistically worse than O_6 as O_6 has 72% level of superiority over N_{10}.

From all the above N_{10} is better than eight out of the ten selected online players, also N_{10} is of equal performance with one online player and finally only one player is better than N_{10}. So, this is a clear success for the aim of this paper.

9. THE RESULTS OF N_{10} AGAINST HUMAN PLAYERS

To measure the influence of the learning strategies that are used to construct the architecture automatically for N_{10}, the performance of N_{10} is tested against ten selected tic tac toe human players. Those players are with different playing abilities. N_{10} is played 100 matches against each player; Table 12 summarizes the results while tables 13 and 14 show the testing of those results using Bayeselo method.

Table 12: Number of wins for the Evolved N10 with Human Players (Row Player) Out of 100 Games.

Player	Opponent: N_{10}			
	Total	Win	Draw	Lose
H_1	100	16	74	10
H_2	100	11	80	9
H_3	100	13	76	11
H_4	100	6	84	11
H_5	100	14	78	8
H_6	100	11	82	7
H_7	100	13	77	10
H_8	100	15	76	9
H_9	100	11	83	6
H_{10}	100	13	79	8

Player	H_4	N_{10}	H_2	H_3	H_7	H_6	H_9	H_{10}	H_5	H_8	H_1
H_4	-	65	65	66	68	70	73	73	75	75	75
N_{10}	34	-	57	57	61	65	68	68	72	72	72
H_2	34	42	-	50	52	55	58	58	61	61	61
H_3	33	42	49	-	52	55	57	58	60	61	61
H_7	31	38	47	47	-	52	55	55	58	58	58
H_6	29	34	44	44	47	-	52	52	55	55	56
H_9	26	31	41	42	44	47	-	50	53	53	53
H_{10}	26	31	41	41	44	47	49	-	52	53	53
H_5	24	27	38	39	41	44	46	47	-	50	50
H_8	24	27	38	38	41	44	46	46	49	-	50
H_1	24	27	38	38	41	43	46	46	49	49	-

Table 13: BayesianElo Ratings for the Evolved N10 Players.

Rank	Name	Elo	+	-	Games	Score	draws
1	H_4	17	44	44	100	52%	84%
2	N_{10}	8	14	14	100	52%	79%
3	H_2	3	44	44	100	49%	80%
4	H_3	3	45	45	100	49%	76%
5	H_7	0	44	44	100	49%	77%
6	H_6	-2	44	44	100	48%	82%
7	H_9	-4	44	45	100	48%	83%
8	H_{10}	-4	45	45	100	48%	79%
9	H_5	-7	45	45	100	47%	78%
10	H_8	-7	45	45	100	47%	76%
11	H_1	-7	45	45	100	47%	74%

Table 14: LOS for the Evolved N10 Players.

The results in Table 14 show that N_{10} is statistically better than H_1, H_5 and H_8 as it has 72% level of superiority over them. Also, Table 14 shows that N_{10} is 68% better than H_9 and H_{10}, while it is 65% better than H_6 and 61% better than H_7. As the 61%, 65% and 68% is closer to 70% (the threshold for determining the level of superiority) than 50% (the threshold for determining the level of equality), so N_{10} can be considered as statistically better than H_6, H_7, H_9 and H_{10}.

The results in Table 14 show that N_{10} is of comparable performance with H_2 and H_3 as it is only 57% better than them. Finally, Table 14 indicates that N_{10} is worse than H_4 as H_4 has 65% level of superiority over N_{10}.

From all the above N_{10} is better than seven out of the ten selected human players, also N_{10} is of equal performance with two players and finally only one player is better than N_{10}. So, this is another clear success for the aim of this paper.

10. COMPARISON OF A RESULTING NEURAL NETWORK WITH OTHER NETWORKS

To determine if the evolutionary process is actually improving or not the neural networks concerning with their domain specific topologies, we compare a resulting net generated by different neural networks with the best random topology that generated by random operation on neural network. So, this neural network is different from previous work which structurally having random number of the hidden layer with random node.

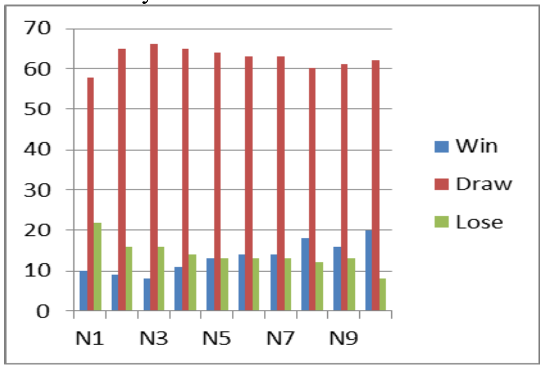


Figure 6: Total numbers of win, draw and loss for each Neural Networks

From figure 6, we see that the neural network generated by the random generation is able to learn better a new set of data than the other nets, the network generated by automatic generation of neural networks also has the best percentage for classifying examples not seen previously, as it is illustrated in figure 7.

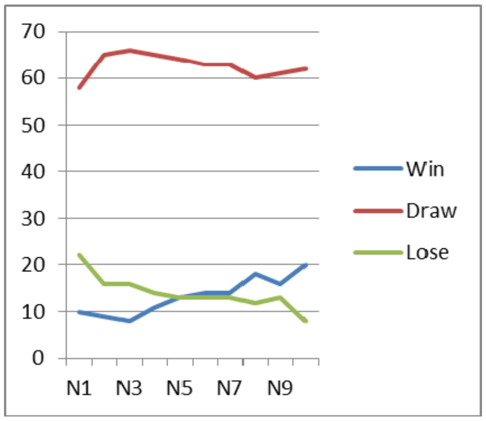


Figure 7: The number of win, draw and loss for the Neural Networks

11. CONCLUSIONS

The most important point of conclusions from this work can be summarized as follows:
Evolutionary computation has proven to be very useful optimization tool in many applications. Determining its efficiency as an optimization algorithm for feedforward neural network architectures and weights was the goal of this research.

The resulted neural network can be used in some applications like game playing (for example tic tac toe), pattern recognition, classification, clustering, and prediction. The user must set the number of inputs, in the input layer, in order to be adapted to the selected problem.

Using an automatic generation of neural network's architecture allow it to adapt their architecture according to the selected problem that deals with it.

Ten players were evolved to see the effects of training for a neural network in their architecture and. The obtained results showed that the performance of the players is increased by increasing the training of neural networks as shown in Tables 5 and 7.

The obtained results demonstrated that the obtained player (N10) was able to beat selected online tic tac toe programs (with different playing abilities), as shown in Tables 8 and 10. This can be considered as a success for the proposed approach.

The obtained results showed that the obtained player (N10) was able to beat selected human players (with different playing abilities), as shown in Tables 12 and 14 can be considered as a success for the proposed approach.

12. FUTURE WORKS

Investigate other evolutionary methods, like genetic programming, in order to automatically generate the neural network architecture for many applications. This will give a broad idea of which method is better for which application area.

Techniques for visualization in evolutionary computation may also prove very beneficial to the field, since in general the internal workings of the algorithms remain hidden to the user. With such techniques, it might even be possible for the user to intervene in the search and adjust certain parameters on the run.

For the purpose of generalization (test the success of the proposed method in many application areas), apply the proposed method that was developed in this paper to other computer games such as checkers or chess and also to other application areas such as pattern recognition.

REFRENCES:

[1] Vonk E. L.C. Jain, L.P.J. Veelenturf and R. Johnson, "Automatic Generation of a Neural Network Architecture Using Evolutionary Computation ",*IEEE*, 1995.

[2] Koza J. R. and Rice J. P I.,"Genetic Generation of both the Weights and Architecture for a Neural Network", *IEEE International Joint Conference on Neural Narrator's*, 1991.

[3] Koza J. R.," Genetic Programming, On the Programming of Computers by Means of Natural Selection", *MIT Press, Cambridge*, 1992.

[4] Freeman J.A .and Skapura D.M., "Neural networks Algorithms, Applications and Programming Techniques", *Addison-Wesley, Reading,* 1991.

[5] David B. Fogel, "using evolutionary programing to create neural networks that are capable of playing TIC-TACTOE", *ORINCON Corporation, San Digo*, CA92121, 1992.

[6] Fiszelew A., Britos P., Ochoa A., Merlino H., Fernández E. and García-MartínezR.,"Finding Optimal Neural Network Architecture Using Genetic Algorithms", *Research in Computing Science,* 2007.

[7] Jenkins W.M., "Used Neural Network Weight Training by Mutation", *School of Engineering, University of Hertfordshire, Hatfield, UK,* December, 2006.

[8] A. E. Elo, "The Rating of Chess Players, Past & Present", *New York: Arco*, 1978.

[9] C. R. Bayeselo, "Bayesian Elo rating," 2005 [Online]. Available: http://remi.coulom.free.fr/Bayesian-Elo.

[10] Vancouver BC, "Neural Information Processing Systems", *(NIPS) Foundation*, 2009.

[11] Kruse K., Borgelt, b., Klawonn, Moewes, Steinbrecher and Held, "Computational Intelligence: A Methodological Introduction", *Springer, ISBN,* 2013.

[12] Damas M., Salmeron M., DiazA., Ortega J.,Prieto, A. and Olivares, "Genetic Algorithms and Neuro-dynamic programming: application to water supply networks", *Proceedings of 2000 Congress on Evolutionary Computation. La Jolla, California, IEEE*, 2012.

[13] Vinicius G. Maltarollo, kathia M. honorio and Alberico Borges F. da silva,"Applications of Artificial Neural Networks in Chemical Problems", *maltarollo, In Tech*, 2013.

[14] Alejandro Correa, Andrés González and Banco Colpatria, "Genetic Algorithm Optimization for Selecting the Best Architecture of a Multi-Layer Perceptron Neural Network", 149, *SAS*, 2011.

[15] Steve Schaefer, "Math Rec Solutions (Tic-Tac-Toe)",*Math Rec.*, 2013.

[16] Goff, Allan, "Quantum tic-tac-toe: A teaching metaphor for superposition in quantum mechanics", *American Journal of Physics (College Park, MD: American Association of Physics Teachers),* 2006.

[17] Belal Al-Khateeb, "Investigating Evolutionary Checkers by Incorporating Individual and Social Learning, N-tuple Systems and a Round Robin Tournament", *The university of Nottingham, Nottingham, UK,* 2011.

[18] Krizhenvsky, I. Sutskever, and G. Hinton. ‘’ ImageNet classification with deep convolutional neural networks’’. *In Advances in Neural Information Processing Systems* 25, pages 1097–1105. 2012.

[19] Goodfellow, Y. Bengio, and A. Courville.’’ Deep Learning’’. *MIT Press, Cambridge, MA*, 2016.

[20] A. Graves, A. R. Mohamed, and G. Hinton. ‘’Speech recognition with deep recurrent neural networks’’. *In 38th IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pages 6645–6649.

[21] Y. Kim.’’ Convolutional neural networks for sentence classification’’. *Signal Processing Magazine*, 29(6):82–97, 2012. arXiv:1408.5882, 2014.

[22] X.-S. Wei and Z.-H. Zhou.'' An empirical study on image bag generators for multi-instance learning''. *Machine Learning*, 2016.,105(2):155–198.

[23] Filipe Assunc¸ao, Nuno Louren ˜ c¸o, Penousal Machado, Bernardete Ribeiro. "Towards the Evolution of Multi-Layered Neural Networks" ,*Conference 2017, Berlin, Germany*, 2017.

[24] Weizhong Yan ," Toward Automatic Time-Series Forecasting Using Neural Networks", *Machine Learning Laboratory, GE Global Research Center, Niskayuna, NY, USA* , *Published in***:** *IEEE Transactions on Neural Networks and Learning Systems* (Volume: 23, Issue: 7, July 2012).

[25] Nadi A., S. S. Tayarani-Bathaie and R. Safabakhsh, "Evolution of Neural Network Architecture and Weights Using Mutation Based Genetic Algorithm", *Proceedings of the 14th International CSI Computer Conference (CSICC'09)*, 2009.