Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Games (CIG 2007)

Hybrid Evolutionary Learning Approaches for The Virus Game

M.H.Naveed, P.I.Cowling and M.A. Hossain
MOSAIC Research Centre, Department of Computing,
University of Bradford
Bradford, BD7 1DP, UK.
E-mail: {M.H.Naveed, P.I.Cowling and M.A.Hossain1}@bradford.ac.uk

***Abstract*—This paper investigates the effectiveness of hybrids of learning and evolutionary approaches to find weights and topologies for an artificial neural network (ANN) which is used to evaluate board positions for a two-person zero-sum game, the Virus Game. Two hybrid approaches: evolutionary RPROP (Resilient backPROPagation) and evolutionary BP (BackPropagation) are described and empirically compared with BP, RPROP, iRPROP (improved RPROP) and evolutionary learning approaches. The results show that evolutionary RPROP and evolutionary BP have significantly better generalisation performance than their constituent learning and evolutionary methods.**

Keywords: Gradient-based learning, evolutionary learning, hybrid learning techniques, The Virus Game.

I. Introduction

In hybrid training of artificial neural networks, evolutionary algorithms and gradient-based techniques work together. In these training methods, evolutionary algorithms are used to find an appropriate neural network topology, to explore good initial weights of the neural networks and to explore appropriate values of learning parameters while learning methods are used to tune the connection weights. In this paper, we aim to investigate whether a hybrid of learning and evolution is more effective than learning and evolution alone. The paper describes two hybrid methods and compares them with evolutionary and gradient-based learning methods. These methods are applied to a board game where artificial neural networks learn to evaluate board positions to provide a (highly non linear) deterministic evaluation function. Mapping or generalising a non linear evaluation function is challenging task for a learning algorithm. This paper describes hybrid approaches which combine evolutionary and gradient-based learning methods for better generalisation performance.

[1] and [2] applied Genetic Algorithms to find good initial weights for a neural network and then applied backpropagation to tune these weights in the direction of minimum error. Their results show that a hybrid of a Genetic Algorithm and BP performs better than BP alone. [3] used evolutionary algorithms to find an appropriate topology of a neural network and the results in this case show that a neural network designed by an evolutionary method has better generalisation performance than a fixed size neural network when trained using BP. Harp et al [4] also evolved both neural network topologies and learning parameters for backpropagation. In this case, the results show that the simultaneous evolution of both learning parameters and topologies improves the performance of backpropagation significantly. In [5], the authors evolved learning parameters for backpropagation with predefined neural network topologies. In this case, Evolutionary Programming (EP) is used to evolve the learning rates of BP where a population contains a list of different learning rates. The fitness of each chromosome is measured as the inverse of Total Sum Square Error while offspring are generated by adding a Gaussian perturbation to the selected parents. In this case, the performance of BP with evolution of learning parameters is shown to be better than a number of variants of BP without evolution. [6] used hybrid of Quickprop [7] and GAs where a GA finds good initial weights and evolves topologies and learning rates while Quickprop tunes the connection weights. The results show that their approach has better performance than both RPROP and Quickprop.

Stanley et al [8] mutated topologies by adding a new connection between existing nodes, adding a new node or splitting up an existing connection into two different connections. In case of adding a new node (hidden neuron) along with a new connection, Stanley et al [8] assigned one as the value of the weight of new connection.

Gradient-based learning methods for neural networks estimate the derivative of the network error with respect to the weight of each link and use this to modify weights at each iteration. Backpropagation (BP) uses a delta rule to modify neural network weights according to the estimated derivatives [9]. Resilient Backpropagation (RPROP) [10] uses the sign of partial derivatives rather than their values to tune the connection weights and biases of an artificial neural network. The amount of weight update is determined by a weight specific factor called the "update value" [10]. Improved resilient backpropagation (iRPROP) [11] modifies RPROP with the consideration that those weight updates that have caused changes to the signs of the corresponding estimated partial derivatives are reversed.

Games, particularly two-person zero sum games such as chess [12] and draughts [13] have been fertile ground for AI research for many years. In this paper we will consider the Virus game [14] [15] [16]. Virus is a two-person, zero-sum game of perfect information played on a square board. This report will consider only an 8×8 board. The Black player always starts the game and the other player is the White player. Players alternate and each player can move only one piece (of the player's colour) per turn. There are two kinds of moves. In a *grow* or *step* move (as shown in figure 1), a

piece is added to an empty position adjacent to one of the player's pieces. Note two squares are adjacent if their borders adjoin or they share a corner. A *jump* or *two-step* move (as shown in figure 2) occurs when a player moves one of his pieces via an adjacent empty square to a square two steps away from the piece's original position. For both types of moves, if the squares adjacent to the new position of the moved piece contain pieces of the opposite colour then these grown pieces change their colour to that of moved piece. This operation is called *capturing* or *infecting* the opponent's pieces. The aim of each player is to capture more squares than the opponent at game end, which occurs when neither player can move.

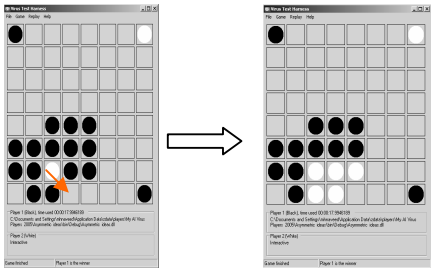

Fig 1: This figure represents a one-step or grow move. The White player is to move the white piece at position (3, 2) moves to position (4, 1). This move gives another white piece at (4, 1) and captures the pieces of opposite colour at positions adjacent to (4, 1).

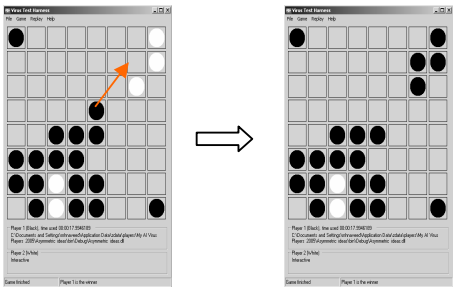

Fig 2: This figure shows a jump move. Black is to move and chooses to move the piece at (5, 5) to (7, 7) which captures white pieces in the squares surrounding square (7, 7).

In the paper, two hybrid learning models (Co-Evo-RPROP and Co-Evo-BP) are presented and compared to Backpropagation, RPROP, iRPROP and an evolutionary algorithm. Our hybrid methods modify neural network structures and learning parameters of RPROP/BP using evolutionary methods while neural networks weights are tuned using RPROP/BP.

The rest of the paper is structured as follows. The learning methods used in this paper are described in the Section "Learning Models". The section "Experimental Design" provides the experimental framework for our learning methods. In the Section "Results and Analysis", the hybrid learning methods are investigated experimentally and compared to non-hybrid methods. We then present conclusions and describe possible future research directions.

II. Learning Models

In this paper, we investigate three learning models: Gradient-based learning, Evolutionary learning and our new evolutionary Gradient-based learning approach. These models are discussed briefly in the following sub sections.

A. Gradient based learning Model

The gradient-based learning models that we consider use BP, RPROP and iRPROP to train neural networks using known examples. The accuracy and speed of BP are highly sensitive to a single learning rate parameter [10] while RPROP and iRPROP use five learning parameters which are Δ_o, Δ_{max}, Δ_{min}, η_+ and η_-. Δ_o is used to initialise the values of partial derivatives and weight update values; and the weight update value is constrained to lie between Δ_{min} and Δ_{max}. If the partial derivative of a connection does not change its sign then the weight update value for that connection is multiplied by η_+ ($\eta_+>1$) and if that partial derivative changes sign then the value of weight update is multiplied by η_- ($\eta_-<1$) [11].

The topologies of the neural networks, used in the gradient-based learning methods are 64-7-3-1, 64-10-5-1, 64-13-7-1, 64-17-8-1, 64-20-10-1, 64-23-12-1, 64-27-13-1, 64-30-15-1, 64-33-17-1, 64-37-18-1, 64-40-20-1 and 64-43-22-1. Here, each network topology is represented as 64-N_1-N_2-1 where there are 64 input units, N_1 and N_2 are the number of hidden neurons in first and second hidden layers respectively. There is 1 output neuron in all these architectures.

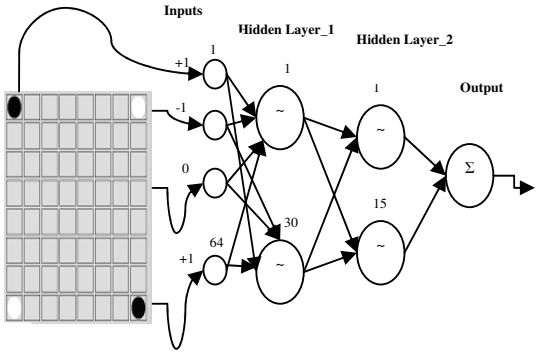

Fig 3: An Artificial neural network of 64-30-15-1 topology with an starting board position of the Virus game. The figure also shows the input encoding of board the Virus board

Figure 3 shows a graphical description of a neural network with topology of 64-30-15-1. The selected neural networks have around twice as many hidden neurons in the first hidden layer as compared to the second hidden layer. All hidden neurons have sigmoid activation functions. This arrangement of hidden neurons in the topologies of neural network is based on the work of [11] for a problem of Cancer classification and that of [13] for draughts/checkers. Each input unit of a neural network represents a square on the Virus board. If the Virus board has a black piece on a

given square then its corresponding input unit receives value '1' and for a white piece it receives value '-1' while an empty square is encoded as '0'. The output neuron has a linear activation function. Positive values of output function indicate positions where black is ahead, while negative values indicate positions where white is ahead. The magnitude of output value indicates the size of the advantage in the given board position.

TABLE 1
THE LEARNING PARAMETERS FOR RPROP AND IRPROP EXPERIMENTS

Δ_O	Δ_{MAX}	Δ_{MIN}	η^+	η^-
0.1	50.0	0.000001	1.2	0.5
0.001	50.0	0.000001	1.2	0.5
0.001	80.0	0.000001	1.2	0.5
0.0001	40.0	0.000001	10.2	0.5
0.0001	90.0	0.0000001	15.2	0.1
0.05	40.0	0.00001	1.0	0.01

In the initial experiments, we used 1000 iterations of training as the stopping criterion and root mean square error at the 1000th iteration measures the accuracy of each neural network with respect to its initial weights. We investigated initial weights uniformly distributed at random in ranges [-015, 015], [-0.30, 0.30], [-0.40, 0.40], [-0.50, 0.50] and [-0.60, 0.60]. BP and RPROP have slightly better performance for initial weights in the range [-0.50, 0.50]; so we use this initial range in all of our subsequent experiments. For BP, we use learning rates 0.0001, 0.001, 0.01, 0.1, 1.0 and 5.0. The learning rates for backpropagation are based on the work of [11]. The selection of the first set of five learning parameters (shown in table 1) for RPROP and iRPROP is made according to the recommended values by authors in [10] and [11] and other sets are variations of the first parameter set.

B. Non Hybrid Evolutionary Model

The evolutionary learning model we use here, applies a Genetic Algorithm (GA) to evolve the weights of neural networks in the direction of minimum error using a set of training examples. The chromosome encodes the weights of a fixed topology network as real numbers. The fitness of an individual in the population is measured as the reciprocal of the root mean square error (RMSE) [18] of the encoded neural network on the training data. Rank selection [19] is used to select parent chromosomes. The top 33% chromosomes of a population are selected as parents to generate a new population and are used to generate the remaining 77% of new chromosomes through crossover and mutation operations. A two-point crossover operation [19] is applied to two randomly selected chromosomes from the parent chromosomes to combine the genetic material of both individuals and create two offspring. During crossover, each pair of parent chromosomes can reproduce at most once per generation. A Gaussian mutation operator [13] is applied to the offspring chromosome weights. The standard deviation σ_i of each weight W_i is then updated as shown in equations (1).

$$W_i^m = W_i + N(0,\sigma_i),$$
$$\sigma_i \leftarrow \sigma_i \times \exp((\sqrt{L})^{-1} \times N(0,1)), \quad i = 1,2,..L \tag{1}$$

Where W_i is the i^{th} weight of a given chromosome before mutation and W_i^m is the i^{th} weight of that chromosome after mutation. Each component σ_i ($i=1,2,..L$) is a standard deviation related to the weight of an individual gene. According to [20] and [21] the Gaussian mutation operator with self-adaptation of σ_i performs better than the Gaussian mutation with out self-adaptation.

Two neural network topologies are used for experiments with a non-hybrid evolutionary learning approach. The network topologies are 64-20-10-1 and 64-25-15-1. The population size is 12. Three values of crossover operator (0%, 1.2% and 2.4%) and three values of mutation standard deviation parameter (0.1, 0.01 and 0.001) are used. The selection of these empirical parameters is made based upon our intuition and testing of a wider range of parameters. Each experiment is run for 1000 generations.

C. Hybrid Models

In this paper, we present two novel hybrid learning models which combine coevolutionary and gradient-based learning methods. These two methods are Co-Evo-RPROP and Co-Evo-BP. Co-Evo-RPROP uses a Genetic Algorithm to evolve the learning parameters of RPROP. It uses evolutionary programming to evolve the topologies of neural networks and tunes the connection weights and biases using RPROP. The topology of a neural network is encoded as a chromosome (n_1, n_2), which contains n_1, hidden neurons in the first hidden layer and n_2, hidden neurons in the second hidden layer. The learning parameters are encoded as a 5-tuple chromosome, with each learning parameter encoded as a floating point number. This approach differs from the hybrid methods proposed by [4] and [6] in that two different populations (of learning parameters and network topologies) are maintained and

Π = Population of RPROP parameter sets. N_{Π} = parameter set population size. c_{Π} = crossover probability for parameter sets. s_{Π} = mutation standard deviation for parameter sets. The p fittest parameter sets pass unchanged to the next generation.
T = Population of network topologies. n_T = network topology population size.

Repeat for g generations

for each $(\pi, \tau) \in \Pi \times T$
train an ANN with topology τ starting from random weights, using RPROP with parameters π, and obtain RMSE $\rho(\pi, \tau)$

for each $\pi \in \Pi$, calculate fitness:

$$f(\pi) = \frac{n_T}{\sum_{\tau \in T} \rho(\pi, \tau)}$$

for each $\tau \in T$, calculate fitness:

$$f(\tau) = \frac{n_\Pi}{\sum_{\pi \in \Pi} \rho(\pi, \tau)}$$

Π_f = {the p fittest parameter sets $\pi \in \Pi$}
$\Pi \leftarrow \Pi_f$
While $|\Pi| < n_{\Pi}$
select distinct π_1, π_2 uniformly at random from Π.
Apply 2-point crossover with probability c_{Π}.
Add Gaussian mutation with standard deviation s_{Π} to each parameter to generate children π_1', π_2'
$\Pi \leftarrow \Pi \cup \{\pi_1', \pi_2'\}$
T_f = {the (n_T/3) fittest parameter sets $\tau \in T$}
$T \leftarrow T_f$
for each $\tau = (t_1, t_2) \in T_f$
randomly choose one of t_1 and t_2 and Add 1 to it to create topology τ'
Subtract 1 from the other of t_1 and t_2 to Create topology τ''if τ'or τ'' has a hidden layer with more than 60 or less than 2 nodes then shift half of the nodes in the larger layer to the smaller layer
$T \leftarrow T \cup \{\tau', \tau''\}$

Fig 4: Pseudocode for Co-Evo-RPROP

coevolved using two different evolutionary methods.
In the Co-Evo-RPROP model (as shown in figure 4), we coevolve network topologies and learning parameters and use RPROP to learn network weights. The population of initial neural network topologies is initialised randomly with integers representing the number of neurons in the first and second hidden layer, chosen uniformly at random in the range [10, 60]. The initial weights of all neural networks in a population are selected uniformly at random in the range [-0.5, 0.5]. The population of parameter sets (Δ_o, Δ_{max}, Δ_{min}, η^+, η^-) is also initialised randomly using a different uniform distribution for each parameter as shown in table 2.

TABLE 2
RPROP PARAMETERS (PARAM) AND THEIR DISTRIBUTION RANGES.

PARAM	RANGE	PARAM	RANGE
Δ_o	[0.0001, 5]	Δ_{min}	[0.0000001,0.00001]
Δ_{max}	[20, 120]	η^+	[1.0, 20.0]
η^-	[0.01, 0.5]		

The fitness of each individual in the population of parameter sets is measured as the reciprocal of the average RMSE of this parameter set over all neural networks of the current neural network population when trained using RPROP. The fitness of a neural network topology is measured as the reciprocal of the average RMSE of this neural network over all parameter sets of the given population of parameter sets when trained using RPROP. The use of average fitness of a neural network is made to find a neural network topology which is more consistent over the whole population of parameter sets and vise versa.

Rank selection is used to select the parents from the populations of parameter sets for reproduction. The top 33% of individuals from the population of parameter sets sorted by decreasing fitness are selected to reproduce offspring by performing two-point crossover and Gaussian mutation to replace the 77% of the chromosomes which are not selected as parents. The parameters of genetic operators for evolving learning parameters are shown in table 3.

TABLE 3
GA PARAMETERS FOR EVOLVING LEARNING PARAMETERS OF CO-EVO-RPROP.

CROSSOVER RATE	MUTATION PARAMETER	POP SIZE
0%	0.01	12
30%	0.001	12
60%	0.00001	12

The selection of parents for the population of neural network topology also uses the Rank selection where the top 4 individuals are selected for reproduction to reproduce 8 offspring using a mutation operator. These 8 offspring replace the chromosomes which are not among top 4 individuals. During the mutation operation, one chromosome produces two offspring; adding one hidden neuron in a randomly chosen hidden layer creates the first offspring and removing one hidden neuron from randomly selected hidden layer creates the second offspring. The connection weights of all new born neural networks are initialised uniformly at random in the range [-0.5, 0.5].

In Co-Evo-BP scheme, each chromosome in the population of learning parameters contains simply a learning rate. These learning rates are initialised uniformly at random in the range [0.0001, 5]. The evolution of learning parameters is implemented by evolutionary programming using a rank selection method where the top 4 individuals are selected as parents to reproduce 8 offspring to replace the 8 individuals of population which are not parents. The evolution of neural network topologies is the same as used in the Co-Evo-RPROP model. Order of the evolution of parameters and topologies is same as in Co-Evo-RPROP.

The size of topology and learning parameters populations is 12 for both Co-Evo-RPROP and Co-Evo-BP. 100 generations are used in each evolutionary experiment as stopping criteria. The mutation self-adaptive parameters (σ), which are explored with this approach, are 0.0001, 0.001, 0.01 and 0.1.

III. Experimental Framework

The experimental setup includes the initialisation of connection weights, network topologies, evolutionary and learning parameters. A set of board positions whose values are given by effective heuristics AI player is divided into two subsets, a training dataset and a test dataset. All learning methods used the same training and test data sets. The performance of each method is measured using the RMSE on training and test datasets. The training data used in all learning procedures contains 2000 training instances, of Virus board positions obtained in actual play and their evaluation values as given by an effective hand-crafted AI, "the Teacher". "The Teacher" is a highly effective AI player which came top out of 45 players who competed in an AI Virus tournament [16]. The board evaluations are scaled between [-64, 64]. Training data board positions occur in a 1-ply search of the game tree when playing "the Teacher" against the 10 other hand-crafted AI players. The neural networks are trained using pattern mode or online learning [17], and each instance of the training data is presented many times to the neural network. In our case, a single training instance is presented to a neural network 10 times. In initial experiments, we used 3-15 epochs of training and 10 epochs were found to provide the best trade-off between CPU time and learning ability in most of our experiments. The test data is used to investigate the generalisation performance of learning methods. The test data contains approximately 1000 board positions, generated in 1-ply searches in games between "the Teacher" and five different, effective hand-crafted AI players. Again "the Teacher" provides a board evaluation for each position. The results using test data measure the generalisation performance of the neural network in evaluating positions which were unseen during training. A large difference between training error and test error is indicative of overfitting [18].

IV. Results And Analysis

Table 4 shows the RMSE on test data for the best networks produced by each learning method. The RMSE of trained neural networks over test data shows that neural networks trained with evolutionary gradient-based learning model have far better generalization accuracy than the neural networks trained with evolutionary model or gradient-based learning model alone. The large difference between training RMSE and test RMSE shows overfitting especially in the case of BP and GA where test RMSE are orders of magnitude higher than training RMSE. Among the gradient-based learning methods, iRPROP shows the best generalization performance, but still is much worse than our evolutionary gradient based hybrid methods. The hybrid methods perform well on both training data and test data due to their searches for an appropriate topology of neural network and a set of learning parameters which work over a range of topologies. Hybridisation of evolution and learning in this way minimises the probability of converging to poor local minima. The topologies explored by Co-Evo-RPROP and Co-Evo-BP perform significantly better than neural networks with 64-2n-n-1 topologies.

TABLE 4
SUMMARY OF TEST RMSE WITH ALL LEARNING MODELS

Learning Method	Best Network Topology	Best Training data RMSE	Test data RMSE
BP	64-27-13-1	0.08207	1.88740
RPROP	64-27-13-1	0.0383	0.41567
iRPROP	64-13-7-1	0.0475	0.10014
Evolutionary Model	64-20-10-1	0.00008	0.43102
Co-Evo-RPROP	64-41-29-1	0.00013	0.00142
Co-Evo-BP	64-28-23-1	0.00090	0.00460

TABLE 5
SUMMARY OF MEAN TRAINING RMSE OF THE NEURAL NETWORK USING CO-EVO-RPROP LEARNING MODEL(X RATE IS CROSS OVER RATE AND M VALUE IS MUTATION VALUE)

X Rate	M Value	Network Topology	RMSE	Δ_o	Δ_{MAX}
0%	0.1	64-13-29-1	0.00014	0.084	56.22
30%	0.1	64-31-9-1	0.0032	0.182	55.36
60%	0.1	64-18-7-1	0.00089	0.113	79.63
0%	0.01	64-33-19-1	0.0002	0.001	131.2
30%	0.01	64-12-15-1	0.0280	0.1	53.21
60%	0.01	64-31-11-1	0.0051	0.21	57.02
0%	0.001	64-17-31-1	0.0172	0.1	56.72
30%	0.001	64-9-16-1	0.0025	0.079	84.40
60%	0.001	64-13-15-1	0.0606	0.02	125.6
0%	0.0001	64-41-29-1	0.0001	0.004	94.23
30%	0.0001	64-21-14-1	0.0541	0.02	125.8
60%	0.0001	64-37-23-1	0.0073	0.14	71.85

Table 5 contains average training RMSE of the neural networks over 10 runs using the Co-Evo-RPROP learning method. The average RMSE shown are the mean for the network with the best RMSE in the final evolved population over 10 runs. Table 5 also shows the evolved initial update value parameter and maximum update value parameter

along with the evolved topology. The mean RMSE of different neural networks with the same crossover rate are shown in figure 5. According to figure 5, the average training RMSE of Co-Evo-RPROP without crossover are smaller than the average training errors with crossover; 60% crossover has lower average RMSE than 30% crossover; However there is no significance difference between the performance of different crossover rates.

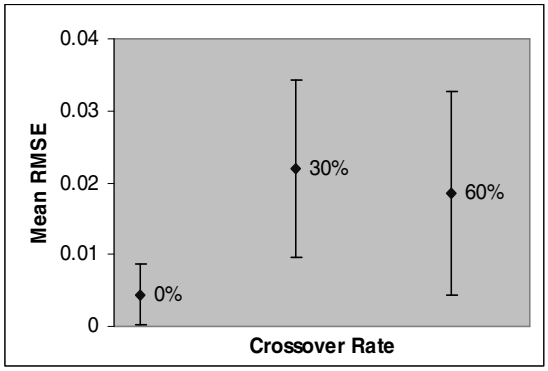

Fig 5: Mean RMSEs of neural networks with different Crossover Rates using the Co-Evo-RPROP learning method.

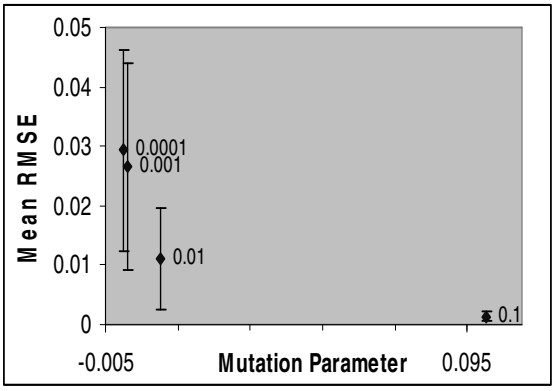

Fig 6: Mean RMSEs with Standard mean error of ANN with Mutation parameters using Co-Evo-RPROP.

Figure 6 graphs the average RMSE with standard mean error for different initial values of the mutation parameter. We can see from figure 7 that initial value of 0.1 of mutation parameter produces better and more consistent results than other values of mutation parameter. The graph also suggest that larger values of mutation parameter than 0.1 may produce even better results.

Table 5 shows that Co-Evo-RPROP method explores a wide range of network topologies and learning parameters for each topology. In most runs we see that evolved values of learning parameter Δ_o converge at around 0.01, much smaller than the value of 0.1. it seems that starting from an initial value of 0.1 (recommend by [10] and [11]) and reducing over time gives the best performance. The evolved values of Δ_{max} jump about rather than converging to a single value. The results show that Co-Evo-RPROP has good accuracy generally for larger neural networks, and evolved population contain mostly larger neural networks. Large networks with small values of Δ_o yielded the best accuracy. We note that value of evolved learning parameters is highly dependent on the number of hidden neurons in a network and their configuration. The best evolved neural network topologies of evolutionary RPROP do not have the structure of 64-2N-N-1 as used in our experiments of gradient-based methods and explored in [10], [11] and [15].

TABLE 6
SUMMARY OF MEAN TRAINING RMSE WITH CO-EVO-BP

MUTATION PARAMETER	LEARNING RATE	NETWORK TOPOLOGY	RMSE
0.0001	0.240	64-29-21-1	0.001436
0.001	0.334	64-30-21-1	0.003015
0.01	0.300	64-28-23-1	0.000902
0.1	0.331	64-37-19-1	0.00275

Table 6 shows the mean of training RMSEs in the final population over 10 runs using Co-Evo-BP with different values of the mutation parameter. The best mean training RMSE has a very small value (0.00014) and is obtained with initial value of 0.01 for the mutation parameter. As for Co-Evo-RPROP, Co-Evo-BP showed better accuracy with larger neural networks. For all networks, we see convergence to a similar topology and a learning rate of around 0.3. The evolution of the learning rate in Co-Evo-BP is highly dependent on the value of the mutation self-adaptive parameter. Smaller values of the mutation parameter have slow speed of convergence and with the higher values of mutation parameter, the results of convergence started jumping about although in both cases a mean value around 0.3 was finally observed. The converged topologies in evolutionary BP have minor differences between the numbers of hidden neurons in each hidden layer and these topologies do not follow the configuration of best reported network topologies used in the non-hybrid gradient-based methods. The experimental results shown in table 6 demonstrate that training RMSE of neural networks with Co-Evo-BP is not very sensitive to the initial value of the mutation parameter. When comparing tables 5 and 6, we see that Co-Evo-RPROP produces smaller training RMSE than Co-Evo-BP for training and test data..

Table 7 shows the best training RMSE, learning parameters and topology of neural networks obtained from the experimental results of BP, RPROP, iRPROP and evolutionary learning method on training data. These training RMSE values are the best of the results of each learning method with the learning and evolutionary parameters discussed in previous section. Each learning method is run 10 times and average results are shown.

TABLE 7
SUMMARY OF AVERAGE TRAINING RMSE WITH BP, RPROP, IRPROP AND EVOLUTIONARY METHODS

	BEST NETWORK TOPOLOGY	BEST TRAINING RMSE	PARAMETERS VALUES
BP	64-27-13-1	0.08207	η=0.1
RPROP	64-27-13-1	0.0383	Δ_o=0.1, Δ_{max}=50
iRPROP	64-13-7-1	0.0475	Δ_o=0.1, Δ_{max}=50
Evolutionary (GA)	64-20-10-1	0.00008	Crossover=0%, σ=0.01
BP	64-27-13-1	0.08207	η=0.1
RPROP	64-27-13-1	0.0383	Δ_o=0.1, Δ_{max}=50

According to the results of table 7, evolutionary learning has better performance than gradient-based learning methods on the training data. The training data RMSE with evolutionary method are slightly smaller than for Co-Evo-RPROP (shown in table 5). Evolutionary learning has better performance with low crossover rates while a mutation parameter value of 0.01 produces the smallest RMSE over training data.

V. CONCLUSION

The paper provides empirical analysis of three different types of supervised learning models: gradient-based, evolutionary and two new hybrids of gradient-based and evolutionary techniques. The performance of these models is measured by RMSE on training and test datasets which evaluate board positions for a two player zero-sum game. Evolutionary learning performs much better on training data than the training results of gradient-based learning methods. The hybrid methods we have introduced show similar (but slightly worse) performance on training data. However, our hybrid of evolutionary and gradient-based methods perform much better on test data than either evolutionary or gradient-based methods alone and suffer less from overfitting. In evaluating a highly nonlinear function to map board positions in the Virus game to winning chances for the black player, we have shown that a hybrid of evolution and gradient-based method is far greater than the sum of its parts. We have also provided evidence that the 64-2N-N-1configuration formed by [7, 10, 21] may be less effective than other topologies if network topology is allowed to evolve.

As future work, it will be interesting to train networks using our hybrid methods evaluating their fitness based on their performance on test data to avoid further overfitting, to investigate the effectiveness of our neural network as a component of an effective game player and to investigate the performance of hybrid learning methods for other game environments.

REFERENCES

[1] Lee, S.W., "Off-line recognition of totally unconstrained hand-written numerals using multiplayer cluster neural network", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol .8, 1996, pp: 648-652.

[2] Omatu, S., and Yoshioka, M., "Self-tuning neuro-PID control and applications", in the proceedings of IEEE International Conference on Systems, Man, and Cybernetics No. 3, 1997, pp: 1985-1989.

[3] Schaffer, J.D., Caruana, R.A., and Eshelman, L.J., "Using genetic search to exploit the emergent behavior of neural networks", Phys. D, Vol. 42, 1990, pp: 244-248.

[4] Harp, S.A., Samad, T., and Guha, A., "Toward the genetic synthesis of neural networks", In the proceedings of 3rd International Conference on Genetic Algorithms and Their Applications 1989, San Mateo, CA, pp.360-369.

[5] Kim, H.B., Jung, S.H., Kim, T.G., and Park, K.H., "Fast learning method for back-propagation neural network by evolutionary adaptation of learning rates", Neurocomputing Vol. 11, No.1, pp: 101-106.

[6] Castillo, P.A., Rivas, V., Merelo, J.J., Gonzalez, J., Prieto, A. and Romero, G., "G-PROP-II: Global Optimization of Multilayer Perceptrons using Gas", in the Proceedings of the Congress on Evolutionary Computation, 1999, Mayflower Hotel, Washington D.C.: IEEE Congress on Evolutionary Computation, pp: 2022-2028.

[7] Fahlman, S.E., "Faster-Learning Variations on Back-Propagation: An Empirical Study", In the proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann. CA.

[8] Stanley, K.O., Bryant, B.D. and Miikkulainen, R., "Realtime Neuroevolution in NERO Video Game", IEEE Transactions on Evolutionary Computation, Vol.9, No. 6, 2005.

[9] Rumelhart, D.E., McMlelland, J.L., and the PDP Research Group., "Parallel Distributed Processing", Exploration in the Microstructure of Cognition, Vol.1, MIT Press, 1986..

[10] Riedmiller, M., and Braun, H., "A direct adaptive method for faster backpropagation learning: the Rprop algorithm", in the Proceedings of the IEEE International Conference on Neural Networks 1993, pp: 586-591.

[11] Igel, C. and Husken, M., "Empirical Evaluation of the Improved RPROP Learning Algorithms", Neurocmputing , Vol. 50 (C), 2003, pp: 105-123.

[12] Campbell, M., Jr. Haone, A.J., Hsu, F-h., "Deep Blue", Artificial Intelligence, Vol.134, 2002, pp: 57-83.

[13] Fogel, D.B. and K, Chellapilla., " Verifying Anaconda's expert rating by competing against Chinook: experiments in co-evolving a neural checkers player", Neurocomputing, Vol. 42, 2002, pp: 69-86.

[14] Cowling, P.I., "Board Evaluation for the Virus Game", in the Proceedings of CIG 2005, Graham Kendall and Simon Lucas (editors), Essex.: IEEE Symposium on computational Intelligence and Games 2005.

[15] Cowling, P.I., Naveed, M.H., and Hossain, M.A., "A Coevolutionary Model for the Virus Game", IEEE Symposium on computational Intelligence and Games 2006, 22-26 May, 2006, University of Nevada, Reno, USA

[16] Cowling, P.I. Fennell, R. Hogg, R. King, G. Rhodes, P. Sephton, N., "Using Bugs and Viruses to Teach Artificial Intelligence", in the proceedings of 5th International Conference on Computer Games: Artificial Intelligence, Design and Education, Reading.: The International Computer Games Conference, 2004, pp:360-364.

[17] Haykin, S. Neural Networks: A Comprehensive Foundation. NewYork: Macmillan College Publishing Company, 1994.

[18] Witten, I.H. and Frank, E. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. San Francisco: Morgan Kaufmann Publishers, 1999.

[19] Mitchell, M. An Introduction to Genetic Algorithms, MIT Press, 1998.

[20] Fogel, D.B. Evolving Artificial Intelligence. PhD diss, University of California, San Diego, 1992.

[21] Bäck, T. and Schwefel, H.-P., "An overview of evolutionary algorithms for parameter optimization", Evolutionary Computation 1993; Vol.1, No.1, pp: 1-23