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

Temporal Difference Learning of an Othello Evaluation Function for a Small Neural Network with Shared Weights

Edward P. Manning
Brookdale Community College, Lincroft, NJ, USA
emanning@brookdalecc.edu

***Abstract*—This paper presents an artificial neural network with shared weights, trained to play the game of Othello by self-play with Temporal Difference Learning (TDL). The network performs as well as the champion of the CEC 2006 Othello Evaluation Function Competition. The TDL-trained network contains only 67 unique weights compared to 2113 for the champion.**

Keywords: Othello, temporal difference learning, shared weights, neural network.

I. Introduction

The game of Othello has been of interest to the computational intelligence community for many years [1]. Its moderately large state space and moderate branching factor make it challenging enough such that it has not been solved. It is different from games such as chess and draughts (checkers) in that there is a large variation in board position from ply to ply. Computer programs using efficient look-ahead search algorithms have beaten the best human players [2]. Strong programs use evaluation functions with many weighted features; often the features used are selected by human designers. There is still much to be learned about how to make the most of limited resources, and how good features can be learned by programs.

The winning entry in the CEC 2006 Othello Competition . was a multi-layer perceptron with 32 hidden nodes and 2113 unique weights [3]. Also doing well was a weighted piece counter (WPC), with 64 weights, found by Co-Evolutionary Learning (CEL) [4]. Lucas and Runarsson showed how CEL can evolve WPCs by mutation and selection. An evolved WPC is selected to reproduce based on its ability to make winning sequences of moves against multiple WPCs (mutated from same parent). Additionally, they showed the advantage of CEL over TDL with self-play. The network used with TDL had 64 weights as in a WPC, plus a bias weight and hyperbolic-tangent activation function. They observe, "Note that, TD(0) is attempting to learn the probability of winning from a given state (when following the ε-greedy policy), while the ES is only learning the relative ordering of the set of game states." Their networks trained using TDL did not perform as well as WPCs evolved by CEL.

This paper analyzes the obstacles in training simple networks to play Othello using TDL, describes the architecture of a network that was trained by TDL and self-play to beat the champion of the CEC 2006 Othello Competition, and discusses potential reasons for its success.

II. CEC 2006 Competition [3]

A. Player Constraints

Entries consisted of evaluation functions. The function input was a board position consisting of 64 elements. The function output was a single real number. The functions were "feedforward", with no recurrency or memory of prior board positions.

B. Board Position

Board position was communicated to the player as a 64-element vector. Each element (0 … 63) represented a location (square) on the 8x8 game board as shown in Fig. 1. The value of each element was +1 to indicate a Black piece in the square, 0 to represent an empty square, and -1 to represent a White piece.

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Fig. 1. Square numbering

C. Move selection

To select among legal moves, the board position after each possible legal move was presented to the player's evaluation function and the output recorded. Outputs were interpreted as larger positive magnitudes are favourable to Black, larger negative magnitudes favourable to White. If the player was assigned the color Black, then the move selected was the one with the most positive output corresponding to the resulting board position. If two or more positions had identical best evaluations, the move was selected randomly from among that subset. The environment provided the capability to, with probability ε, force a uniformly random choice among legal moves.

D. Tournament format

Three round-robin tournaments were held, one each for $\varepsilon = 0$, $\varepsilon = 0.01$, and $\varepsilon = 0.1$. With $\varepsilon = 0$, each player played each other player twice (once with each color). With other values of ε, players played each other ten times for each color.

E. Architectures provided for

There were two standard architectures provided for entry of evaluation functions. An entry for a weighted piece counter (WPC) consisted of 64 weight values, one for each square. The output of the WPC is the sum, over the squares, of the product of the input value and the weight. A "heuristic" WPC was made available as an opponent for testing entries.

The other standard architecture was a fully-connected multi-layer perceptron (MLP). The number of hidden units and the number of layers were not restricted. A sample MLP with one hidden layer of 32 units was provided as an example. This MLP had been trained using TDL.

F. Champion entry

The champion entry had an architecture identical to the sample MLP. The champion was the result of evolution over 100 generations. The population of 50 networks was initialized to the sample MLP weights and mutated. Probability of mutation of each of the 2113 weights was 1%. Mutation of a weight was achieved by adding or subtracting 0.01 with equal probability. The fitness function was performance over 1000 games, with $\varepsilon = 0.1$, against the heuristic WPC. Roulette-wheel selection was used to populate the next generation.

The champion entry won all three of the tournaments in the competition.

III. ANALYSIS

A. Temporal Difference Learning and WPCs

Temporal Difference Learning attempts to learn the expected value of reward based on the current state from the expected value of reward associated with succeeding states [5]. Perfect learning is dependent on the ability to access the current evaluations for all states without interference from other states, and on the experience of visiting all states an infinite number of times. In contrast, updating the weights of a WPC for one state affects the future evaluations of all states. In a trained WPC of reasonable skill, the moves selected at early plies lead to states that will have BOTH a high evaluation and a high expected value of reward. At many game states, the evaluations of winning and losing actions may be very close. With a greedy or ε-greedy policy, performance will be reasonable as long as the winning state has a higher evaluation. With continued training, a losing action may receive a higher evaluation for a state or set of states. When this happens for an early game state, the expected value of reward for following states may have no relation to that learned for common, but similar, states reached by the previous policy. Even if the advent of a losing streak causes a return to the former move selection at early plies, the evaluations at later plies may have changed, and learning will have to start from a place of much lower performance. This is known as "catastrophic forgetting" [6]. The learning environment enables this by:

- Training on one pattern affects the evaluation of all patterns. The value of the learning rate determines the degree of change for interference as well as for learning.
- Target values are based on expected value of reward, resulting in a distribution dense near zero for evaluations of early plies. Closely-spaced evaluations are more susceptible to interference.
- Weights are changed to move the network output toward the target without regard for the resulting performance.

In contrast, in the CEL learning environment:

- Patterns are not used in learning.
- Weights are not constrained, so resulting evaluations can be far apart.
- A good set of weights only has to rank winning moves higher than losing moves (as often as possible). The set does not have to track expected reward – a winning move can have a low evaluation as long as it is higher than the losing moves available from the same state.
- A set of mutated weights only reproduces if it has already performed well.

So why choose TDL?

IV. DESIGN CONSIDERATIONS

The original intent of this research was visualization of Othello games. One concern in visualization is the number of dimensions. An image of the game board allows visualization of a single position in 64 dimensions (one for each square), but does not allow visualization of the path of positions through the course of a game in a single image. The ouput of an evaluation function, such as used in the CEC 2006 Othello Competition, is one-dimensional – a path can be plotted of function output versus ply number. However, it would be difficult to distinguish dissimilar positions that have similar evaluations. The symmetry of the Othello game board suggests that a 4-dimensional representation is possible.

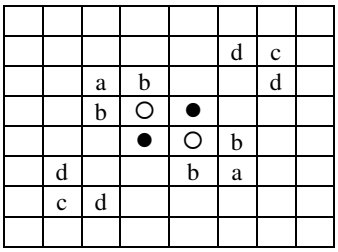

Fig. 2. 4-way symmetry

Fig. 2 shows the position at the start of the game. There are axes of symmetry on the two largest diagonals. The legal opening moves are labeled 'b'. Having chosen a first move and resulting board position, note that there are three other equivalent board positions that can be obtained by 1) reflection about the upper-left-to-lower-right diagonal, 2)

reflection about the lower-left-to-upper-right diagonal, 3) both reflections 1 and 2 in any order. Squares on the diagonals ('a' and 'c') are mapped to themselves by reflections about the same diagonal. Off-diagonal squares ('b' and 'd') clearly show the 4-way symmetry.

Four dimensions may be harder to interpret than the two or three usually used for visualization, but may overcome the disadvantages of either 64 dimensions or a single dimension.

MLPs have been used for reducing the dimensionality of representation [7]. This approach was selected because there is an existing Othello environment [3] that accepts MLPs. The 4 visualization dimensions can be produced by a 4-unit hidden layer. The 4-way symmetry can be enforced by weight-sharing throughout the network. The software used allows for trivial encoder and decoder layers (pass-through connections). This option was selected in order to "start small". Unlike [7], the ability to reproduce the inputs from the reduced representation was not a design consideration.

For an example of the use of symmetry and weight-sharing in Othello evaluation functions, see [8], which makes use of 8-way symmetry (disregarding the starting position). In this work, four dimensions were chosen instead of eight because eight dimensions would be harder to visually interpret. Also, board positions resulting from reflection about additional axes of symmetry (horizontally or vertically through the board's center) are not reachable early in a legal game.

An additional design consideration was the ability to visualize positions (and paths) in terms of the expected value of the game result (+1 for a Black win, -1 for a White win, 0 for a draw). The existing Othello environment accepts MLPs with one output, which can be used to approximate the expected value of the result. This can be achieved using TDL, as described in the ANALYSIS section above.

V. ARCHITECTURE

The architecture is shown in Fig. 4. Hidden and output units use a hyperbolic tangent activation function.

A. Input Units

There are 64 input units, one for each square of the 8x8 Othello board. Input units, like board squares in Fig. 1, are labeled from 0 to 63. A Black disc on a square is represented as +1 on the corresponding input, with 0 for empty, and -1 for White. (Unit numbering and input representation match that described in [4].)

B. Hidden Units

There are four hidden units, one for each dimension to be displayed. All hidden units share a common bias weight. Each hidden unit is connected to all the input units. Only 64 weights are used to make the full 256 connections with the input units. The weights are ordered differently for each unit according to the four directions of symmetry of the Othello board ("Patterns of shared input weights" in Fig. 4). Inputs representing off-diagonal squares are each connected by different weights to each hidden unit (HU). For example, Input 1 is connected to HU0 by w_1, to HU1 by w_8, to HU2 by w_{55}, and to HU3 by w_{62}. Input units representing squares on the diagonals are associated with two weights, each connecting to two hidden units.

C. Output Unit

The output unit has a bias weight and one other weight which is used to connect to all four hidden units. The purpose of the output unit is to propagate game-result information to the hidden units. The intent is to satisfy the design consideration that the visualization will provide information about the expected value of the game result. As a side effect, the network is also able to provide an evaluation function.

D. Comparison to other weight-sharing architectures

In [8], the eight axes of symmetry divided the board into eight triangles of board locations (inputs). For each hidden unit ("feature map"), each set of eight similar inputs shared a single weight. Multiple hidden units were used. Because of the weight-sharing within hidden units, there could be no hidden units with different weights among a group of similar inputs. This would limit the ability of each hidden unit to recognize patterns across the entire board.

The architecture of the "visualization tool" does not share weights within a hidden unit. This allows a hidden unit to recognize a pattern across the entire board. Since all hidden units are connected with the same weight to the output unit, an advantageous pattern can be recognized in any of four orientations. The same approach is taken in [9] for an agent for the game of Cellz. The agents have 8-way radial symmetry in their sensory inputs. A neural network-based controller module is associated with each direction of symmetry. Each module takes input from all sensors. Weights are shared across modules for inputs that have the same relative angle to the module. For example, the weight from the sensor at 135° to the module at 0° is shared with the weight from the sensor at 90° to the module at -45° (and six others with the same relative angle of 135°). This arrangement takes advantage of symmetry in that a successful set of weights for one module should be successful for all modules.

VI. LEARNING ENVIRONMENT

The weights were initialized to random values uniformly in the range $-\alpha$ to $+\alpha$, where α is the learning rate. The learning rate was set to 0.0001 in an attempt to reduce inter-pattern interference. There was no decrease in the learning rate over time as I did not intend a long run. If two or more potential moves have equal evaluations higher than all others, selection decisions are made randomly among them. Random moves were forced at a rate ε of 0.1. Weights were stored as single-precision floating-point numbers and updates were done with single-precision arithmetic. The network was trained for 250,000 games with samples of the

network taken every 10,000 games.

VII. Results and Discussion

A. Competition

Because the visualization tool provides an evaluation function output, it can also act as an Othello player. Training with self-play provides no indication of how the player will perform against other competition. To test the "Othello knowledge" of the visualization tool, each of the sampled networks played 1000 games against the CEC 2006 Othello Competition champion. (All games described in this section were played with training turned off and $\varepsilon = 0.1$.) The best performing sample (number 25 of 25) then played 10,000 games against the champion; with the results shown in Table I:

TABLE I
Competition results

Player Color	Player wins	Draws	Champion wins
Black	2478	147	2375
White	2476	154	2370
Total	4954	301	4745

Why did TDL succeed for this architecture while it did not in [4] for an architecture with a similar number of weights? Some hypotheses were investigated:

- Shared weights may cause cancellation of some interference. Each weight is updated using values for more than one input in the presented pattern.
- Symmetry of weights may allow exploitation of weaknesses in an asymmetric opponent. In addition, any position which matched the symmetry of the network could have two equivalent actions which would be decided randomly – an asymmetric network might not have a counter-play for both actions.
- The learning rate may have been small enough relative to the distance between typical first- and second-choice move evaluations such that interference was reduced.

B. Weight Values

The weight values of the Othello player/visualization tool are shown in Fig. 3. The hidden-to-output weight is negative, so squares with more negative input-to-hidden weights are advantageous to be occupied by the player to move. The corner squares have the most-negative weights. The top row of weights, including C-squares, is negative; allowing hidden unit 0 to encourage ownership of the top edge; the other hidden units would encourage ownership of the other edges. The other C-squares are positive, discouraging them from being taken when the corresponding edge is not available.

VIII. Investigation

A. Cancellation of Interference

The symptom of catastrophic forgetting is loss of past performance against one opponent with continued training against a different opponent. With self-play, the "different opponent" is the player-in-training. For a network that has achieved stable performance against a non-learning opponent and forced random moves, the number of wins in N games will approximately follow a binomial distribution (only exact if both players do not learn). With catastrophic forgetting, performance over a series of N game samples will have a standard deviation much higher than that expected from a binomial distribution ($[Np(1-p)]^{1/2}$, where p is the probability of winning). In this experiment, the "heuristic" player of [3] and [4] is used as the non-learning opponent to allow comparison with [4]. The sampled networks from the original training (see Learning Environment) were each played 1000 games against the heuristic opponent and their performance was recorded. The first three samples were not used as their performance was much lower than the others', indicating the network was not yet trained. Table II compares the standard deviation of the set of samples to that expected from a binomial distribution.

TABLE II

	Standard deviation
Binomial estimate	15.8
Sampled networks	31.5

Standard deviation of the player's performance (based on a mean 508 wins in 1000 games) is significantly ($< .01$) higher than expected from a static network. It compares with the performance of TDL in [4].

B. Weight symmetry

To test the effect of symmetry, the sampled network ("Player" as in Table I) played eight games with $\varepsilon = 0$ against the champion – one for each combination of color and location of first move. The player won all four games as Black (moving first), and two of the four games as White. The same symmetric position occurred at ply 4 of both lost games (giving the player a choice of moves). Both choices resulted in a loss. There was no evidence that the player was able to exploit symmetry of position. Note that the fitness function used to create the champion was based on a symmetric "heuristic" player. The champion demonstrated success in all orientations by winning the CEC 2006 competition.

C. Learning rate

The training as in Learning Environment was repeated for several values of learning rate α. The sampled networks each played 1000 games against the champion. Table III

shows the number of wins achieved by the best network in each run, and the standard deviation across the "trained" networks in each run.

TABLE III
Comparison by learning rate

α	Wins vs. Champion [Best Network]	Std. Dev. Of Wins [Trained Networks]
.01	552	46.2
.0025	513	42.3
.0005	506	38.3
.0001	517	42.3

In each case, training with TDL found a network with good playing ability despite the presence of catastrophic forgetting.

D. Comparison to CEL

CEL was not selected as the learning mechanism because it did not match with the design goals for the visualization tool. As discussed in the ANALYSIS section, CEL has some advantages over TDL in finding a good set of weights for playing Othello. Without the limitation of network outputs tracking expected reward, CEL might be able to find a better set of weights (for playing) for the same architecture.

The learning environment was a (1, 10) ES as in [4]. The mutation function was N(0, 1/67). The fitness function was round-robin play, with ε = 0.1. Each mutated network played each other network twice each generation, the second time with colors reversed. The parent network was replaced by a network with each weight equal to 95% of the weight from the parent plus 5% of the weight from the highest-performing mutated network. Evolution took place over 25,000 generations, with samples taken every 1000 generations. The eighth sample played the best against the champion (Table IV). The weights that evolved are shown in Fig. 5.

TABLE IV
CEL player vs. Champion

CEL Player Color	CEL Player wins	Draws	Champion wins
Black	2771	150	2079
White	2547	156	2297
Total	5318	306	4376

The standard deviation of wins in 1000 games against the champion across the CEL samples was 34.4, also significantly (< .01) more than the binomial standard deviation.

The CEL player did not perform as well as the TDL player with the same architecture (Table V). It was also beaten by the heuristic WPC from [3]and [4].

TABLE V
CEL Player vs. TDL Player

CEL Player Color	CEL Player wins	Draws	TDL Player wins
Black	2219	191	2590
White	2328	198	2474
Total	4547	389	5064

The above CEL player was selected for its performance against the CEC 2006 Othello champion. Other samples from the same CEL run were found that outperformed the TDL player.

IX. VISUALIZATION

Further investigation of visualization has been held up while the playing ability of the visualization tool is being researched. Some early examples of potential uses are shown below. The game being visualized is between the champion network and its ancestor network, with ε = 0. Fig. 6 shows the how the hidden unit outputs vary by ply number. The outputs of hidden units 0 and 2 start to fall at ply 45, indicating Black is strengthening its position in those dimensions.

The four dimensions can be followed in two 2-D images (Fig. 7 and 8). In both examples, positions favourable to Black are in the lower left. Positions favourable to White are in the upper right. Fig. 7 shows the path traced by the course of the same game as in Fig. 6 using the outputs of hidden units 0 and 2 as the visualization dimensions.The path begins near the origin – there is maximum uncertainty about the outcome at the beginning of the game. Fig. 8 shows the path traced by the course of the same game using the outputs of hidden units 1 and 3 as the visualization dimensions.

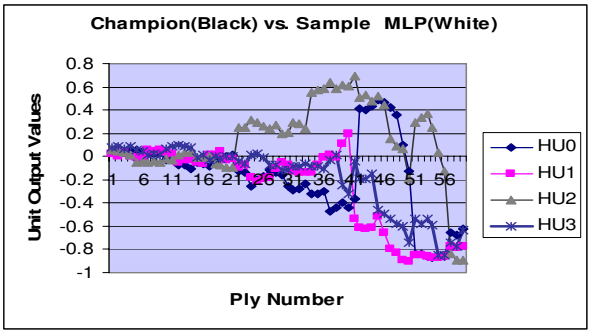

Fig. 6

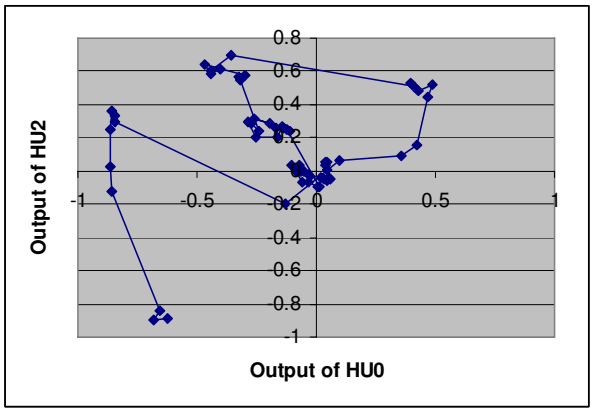

Fig. 7

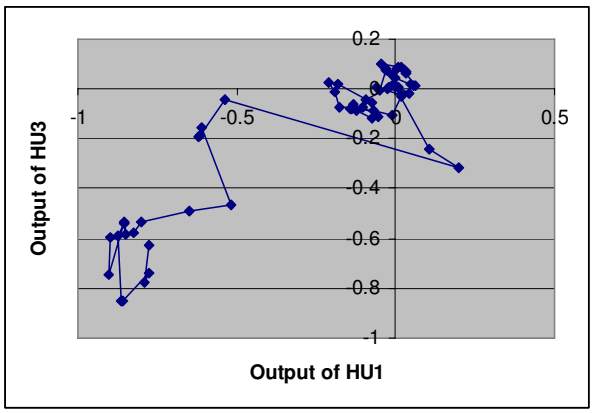

Fig. 8

X. Conclusion

The presented network architecture can be trained by TDL or evolved by CEL to outperform the CEC 2006 Othello Competition champion, despite having fewer weights and hidden units. The architecture does not reduce or eliminate catastrophic forgetting. This confirms the caveat from [4] that the network taken from the end of a training run may not be the best one that appeared in training. Finding a successful set of weights resulted from sampling from the sequence of networks produced by training, then selecting between the samples based on a performance measure. Using a single opponent as a performance standard is unreliable, as the simple Othello players are intransitive (e.g. the CEL player beat the champion, which beat the heuristic WPC, which beat the CEL player). A different choice of opponent may result in a different sample being selected.

There is a match between this architecture and the Othello game that allows "good" sets of weights to appear during training or co-evolution. A good set of weights selects moves such that the entire path through the game tree results in a win against a variety of opposing strategies. The hidden layer and non-linearities allow multiple inputs to be considered together as a feature. The weight-sharing according to board symmetry enables orientation-invariant features to be found and exploited.

Acknowledgment

Thanks to Simon Lucas for his encouragement. Thanks to Kyung-Joong Kim for his champion entry as described in [3]. Thanks to the anonymous reviewers for their insights.

References

[1] P. B. Maggs, "Programming strategies in the game of Reversi", *BYTE*, Vol. 4, Issue 11, pp. 66-79, 1979.

[2] M. Buro, "LOGISTELLO – a strong learning othello program," 1997, http://www.cs.ualberta.ca/ mburo/ps/log-overview.ps.gz.

[3] S. M. Lucas,and T. P. Runarsson, "*CEC 2006 Othello Competition*", 2006, http://algoval.essex.ac.uk:8080/othello/html/Othello.html.

[4] S. M. Lucas, and T. P. Runarsson, "Temporal Difference Learning Versus Co-Evolution for Acquiring Othello Position Evaluation", *IEEE Symposium on Computational Intelligence and Games* (2006), pages: 52-59.

[5] R. S. Sutton, "Learning to Predict by the Method of Temporal Differences", *Machine Learning* 3:9-44, 1988.

[6] R. M. French, "Catastrophic Forgetting in Connectionist Networks", *Trends in Cognitive Sciences*, 3(4) 128-135, 1999.

[7] D. DeMers and G. Cotrell, "Non-Linear Dimensionality Reduction", *Advances in Neural Information Processing Systems*, vol. 5, pp. 580-587, 1993.

[8] A. Leouski, "Learning of Position Evaluation in the Game of Othello", Master's Project, Dept. of Computer Science, University of Massachusetts, 1995.

[9] J. Togelius and S. M. Lucas, "Forcing neurocontrollers to exploit sensory symmetry through hard-wired modularity in the game of Cellz", *IEEE Symposium on Computational Intelligence and Games* (2005) , pp. 37– 43.

Hidden unit bias: 0.046755

Input-to-hidden weights:

Row \ Col	0	1	2	3	4	5	6	7
0	-0.412767	-0.086716	-0.137243	-0.116847	-0.108441	-0.132597	-0.082691	-0.447977
1	0.095415	0.066026	0.081115	0.039152	0.037808	0.075500	0.060762	0.098185
2	-0.014942	0.006509	-0.002831	-0.007430	-0.010424	-0.003539	0.018977	-0.011253
3	0.018324	0.002287	0.014223	0.007309	0.006887	0.008938	-0.002179	0.025855
4	0.008673	0.001184	-0.004982	-0.003349	-0.003205	-0.011319	-0.001403	0.014132
5	-0.030202	-0.019950	-0.000659	-0.000770	-0.002035	-0.002382	-0.024284	-0.018793
6	0.107028	0.068674	-0.015182	0.002003	0.002427	-0.016564	0.080611	0.101164
7	-0.185074	0.113824	0.012961	0.036921	0.034189	0.020531	0.106863	-0.190973

Output bias: 0.051627

Hidden-to-output weight: -0.608910

Fig. 3. Connection Weights of TDL-trained player

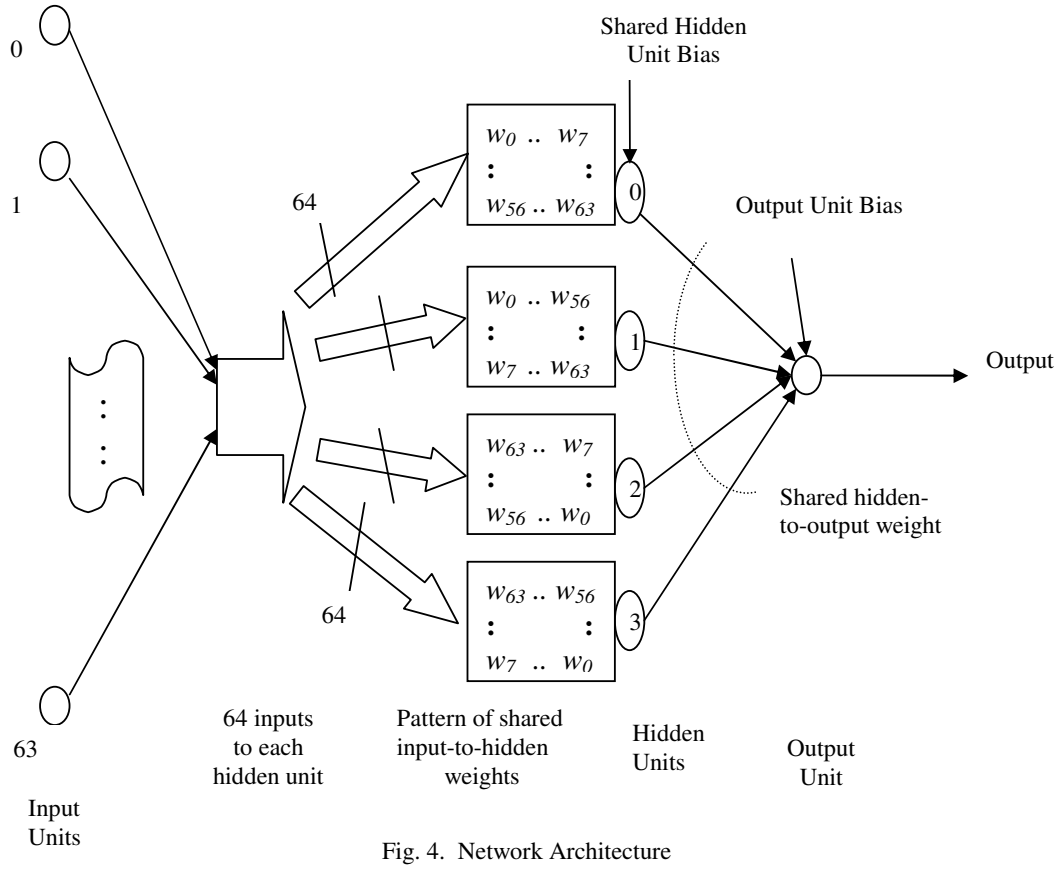

Fig. 4. Network Architecture

Hidden unit bias: 0.074982

Input-to-hidden weights:

Row \ Col	0	1	2	3	4	5	6	7
0	0.176211	-0.016419	0.065518	-0.055454	-0.057461	0.100878	-0.098303	0.254570
1	-0.104481	-0.130790	0.0233256	-0.009091	-0.067571	0.065519	-0.152240	-0.108126
2	0.148707	-0.008260	-0.017373	-0.067466	-0.028830	-0.013478	-0.099959	0.097666
3	0.017459	-0.038781	0.064864	-0.040968	0.000638	-0.056187	0.012138	0.029352
4	-0.090786	-0.039387	-0.013869	0.003110	0.040638	-0.023122	0.034377	-0.068545
5	0.050908	0.057751	0.060624	0.107575	0.059930	0.043094	-0.088655	0.083757
6	-0.167291	-0.164509	-0.105543	0.007365	-0.052622	-0.074962	-0.062038	-0.054924
7	0.276004	0.064116	0.121930	0.061563	0.026529	0.063474	-0.009005	0.142357

Output bias: 0.054569

Hidden-to-output weight: 0.063101

Fig. 5. Connection Weights of CEL-evolved player