Applications of Artificial Neural Networks in Games; An Overview

Joonatan Mänttäri
Mälardalen University
School of Innovation, Design and Engineering
Box 883, Västerås, Sweden
jmi08001@student.mdh.se

Jonas Larsson
Mälardalen University
School of Innovation, Design and Engineering
Box 883, Västerås, Sweden
jln08010@student.mdh.se

ABSTRACT
An artificial neural network is a system that tries in various degrees to emulate a human brain in order to perform tasks that other computer systems are usually not fit to handle. Artificial neural networks are used in many different areas due to their ability to learn and adapt to many different tasks and make complex predictions. In gaming, computer controlled opponent behavior is usually rule based and dependent on specific conditions and can thus be predictable to a certain degree. As the field of AI and learning systems using artificial neural networks is being developed and expanded, it is inevitable that its use in gaming will be explored thoroughly. This short survey looks at the attempts of using artificial neural networks for opponents in board games and modern computer games, as well as other uses in gaming throughout the last 20 years.

Keywords
Artificial Neural Networks, Artificial Intelligence, Games, Computer Games, Evolutionary Algorithms

1. INTRODUCTION
Artificial Neural Networks(ANNs) have been in wide use since at least the 1980s for, among other things, complex modeling and various recognition, prediction and filtering tasks. Their ability to learn and evolve has made them attractive to many different fields of research and innovation, including gaming. As an opponent AI has a huge impact on a game's enjoyability, the ability for such an opponent to learn and get better over time is intriguing. Usually, the actions of non-player-characters (NPCs) in games are either completely pre-determined or rule-based and dependent on different conditions being met (as shown in figure 1). This can be done in such a manner that the NPC will still appear very intelligent and life-like to the player, but often there will be predictable and/or exploitable patterns which will make the game easier than intended, or remind the player too much that they are playing against a computer. This can ruin the game for the player looking for an immersive experience or a good challenge. This is why the idea of a learning and evolving NPC, by means of an ANN, is very interesting.

No.	*IF (Condition)*	*THEN (Behavior)*
1	see the enemy, has better weapons, and is being shot by others	*ChangeToBetterWeapon*
2	see the enemy, has weapon loaded, and is being shot	*Engage*
3	has weapon loaded, is shooting and being shot, and is pursuing enemy	*StopShooting*
4	is pursuing enemy, has weapon loaded, and gets damaged	*ResponseToHit*
5	has weapon loaded	*Pursue*
6	has weapon loaded, see some item he want, but colliding on the path	*Walking*
7	spots some item and want it	*GrabItem*
8	has weapon loaded, and health level is weak	*GetMedicalKit*

Figure 1. The rules for the Hunter bot in Unreal tournament 2004, an example of rule-based AI behavior [20].

Over the last 20 years, computer scientists, engineers and programmers have experimented with the possibilities of the ANN in games. Different kinds of ANNs have been used, to create life-like and intelligent opponents for classic board games such as Checkers and Othello, as well as for modern PC games such as Unreal Tournament. ANNs have also been implemented for other tasks, such as detecting cheaters in online games and for dynamic gesture recognition in dance games. This short survey attempts to summarize the most significant attempts and their results, as well as provide a brief look at the future possibilities of ANNs in gaming.

Section 2 provides a more thorough explanation of the Artificial Neural Network and its different uses. Section 3 will address the earlier gaming implementations, discussing implementations for classic board games. Section 4 will continue into modern computer games and section 5 discusses the various other gaming related tasks that ANNs have been used for. Section 6 provides a summary, conclusions and discussion of the subject matter. Finally, section 7 shortly describes possible future work.

2. BACKGROUND
In a strictly analytical sense, artificial neural networks are mathematical models for non-linear functions. Trained ANNs are used to provide desired output from a set of input parameters without the need for an exact function or model for the problem. ANNs can be seen as simplified replica of their biological counterparts. All ANNs just as the brain, consist of neurons, also called nodes in ANNs, but while the human brain encases around 100 billion neurons with thousands of connections between them, the typical ANNs are on a much smaller scale. This paper will briefly discuss the basic elements of ANNs.

2.1 The Typical ANN
The typical ANN consists of nodes that are connected to each other and exist in several different layers, resulting in it being often referred to as a Multi Layered Perceptron (MLP) network. These layers are the input layer, the hidden layer, and the output layer. Each of these layers has a design specific amount of individual nodes in them. An individual node works much like its biological counterpart the neuron. It receives input from a multitude of different weighted input connections, sums these inputs and then produces an output that serves as input for other nodes. This output is generally normalized to be between -1 and 1 and typically a sigmoid function of the

type discussed in [11] can be used for this.

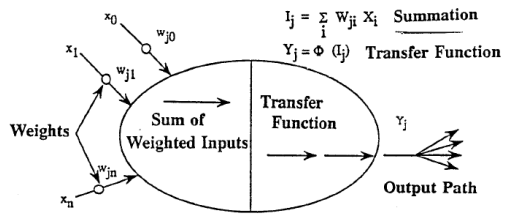

Figure 2. An abstract model a neuron, displaying inputs, weights and outputs [24].

These nodes are used to construct the input, hidden, and output layers. The nodes existing in the input layer receive the input parameters that are fed to the system as their inputs. The input signals propagate through the ANN through the neurons in the input, hidden and output layer, each layer's neurons producing output used as input for the next. Ultimately the output layer produces an output as a result of the given input to the network and the weights used.

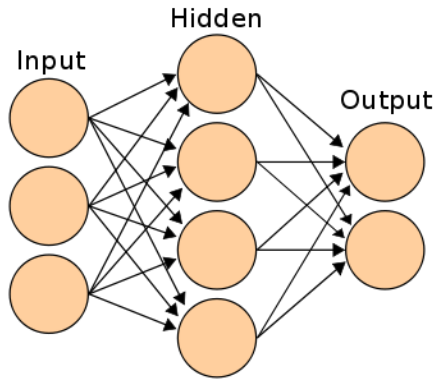

Figure 3. A simple model of an ANN.

2.2 ANN Training

ANNs are capable of learning to produce more correct and desirable output which is one of the main reasons applications of ANNs are so favored. The process of training an ANN is configuring the weights or even architecture of the ANN to produce the correct output based on the input. A few methods of ANN training will be discussed here.

2.2.1 Back propagation

Back propagation requires specific knowledge of the desired output of the ANN. It calculates the error between the desired output for the network and the actual output of the network. Each node gets a total error from all of its outputs, and then adjusts the weights to its input connections accordingly. This process is propagated backwards from the output layer all the way through the input layer of the network, giving it its name "back propagation". A more detailed description of back propagation is explained in [14] along with alterations of the algorithm to improve performance, which is also discussed in [11].

2.2.2 Evolutionary Algorithms

Evolutionary Algorithms (EAs) and Genetic Programming (GP) are used to optimize parameters to find the best possible solution to a problem [9]. To use EAs, a population of possible solutions as individuals must be acquired. In the case of ANN training, an individual would be an ANN, with its associated weights and architecture. Individuals are evaluated with a fitness function, depicting how well they solve the problem. Then, the most fit solutions generate "offspring" which are solutions whose parameters are copies or combinations of the parent solutions', usually with a degree of random mutation. This way, the genetic algorithm strives to find the best solution by letting the population of individual solutions evolve over a multitude of generations. In the case of ANN training, this would result in finding an optimal set of weights to be used in the neural network, or if specified, even the optimal structure.

The specific way to represent the individual solutions can differ from application to application. For example it is possible to represent individuals as just values of the weights and number of nodes and connections in each layer of an ANNs, or use bit strings as discussed in the method in [12] which is associated with Genetic Algorithms, a type of EA.

3. ANNS IN BOARD GAMES

The application of ANNs to serve as artificial intelligence that plays board games was a quite natural direction for ANNs, given the pattern-learning nature of ANNs and the growing computational power of the 1980's. Since board games often include a simple rule set, relatively small playing environment and clear success criteria, application of machine learning in board games has been a popular field. Several implementations of ANNs to serve as trainable AI for games were made popular, including ANNs as AI for Othello, Tic Tac Toe, five in a row, and checkers, which will be discussed in this section.

3.1 ANNs in Othello

A popular implementation of ANNs for AI is for the game of Othello, also known as Reversi**.**

One implementation [12] was done by David E. Moriarty and Risto Miikkulainen in 1995. Moriarty and Miikkulainen created and trained an ANN capable of discovering complex strategies and playing on par with an experienced human.

The ANNs which received the 8x8 board configuration and produced the next best placement were optimized using an EA. Moriarty and Miikkulainen used a population of 50 ANNs represented using the Marker-Based Encoding scheme, explained in detail in [12] , allowing for evolution of weights and architecture, resulting in a dynamic architecture for the ANNs.

The ability to restructure the architecture of ANNs is a common theme when using ANNs for AI in games, as this is utilized in a multitude of implementations including**,** [4],[20], [19]. According to Moriarty And Miikkulainen, this is an important implementation, as they replicated the experiment using static architecture ANNs and did not receive the same level of performance at all. They stated that the dynamic architecture allows the ANNs to not discard new strategies that do not give great performance in the start of the game, such as the advanced mobility strategy.

Moriarty and Miikkulainen started by training the ANNs against an AI opponent that simply made random moves, and the ANNs quickly developed a positional strategy based on controlling corners and edges to create static chips. This strategy is often used by novice players that are quite new to the game but have started to develop a sense of strategy. Within 100 generations of evolution, the best trained ANNs could defeat a random moving opponent 97% of the time.

After this, the ANNs were pitted against an AI using the alfa-beta pruned minimax search algorithm [8]. This algorithm searches through a defined "ply depth" of moves and resulting possible next moves to choose the best possible action in a situation. In the beginning, the win rate of the ANNs dropped drastically, but after 2000 generations of evolution, a new strategy was being perfected by the ANNs, resulting in a win rate of around 70% against the search algorithm based AI. This strategy emphasizing a low chip count and high amount of possible moves until the end (shown in figure 4) is consistent with what is known as the mobility strategy, used by players of experienced level in tournaments. The ANNs also continued to use this strategy when playing against the random mover AI.

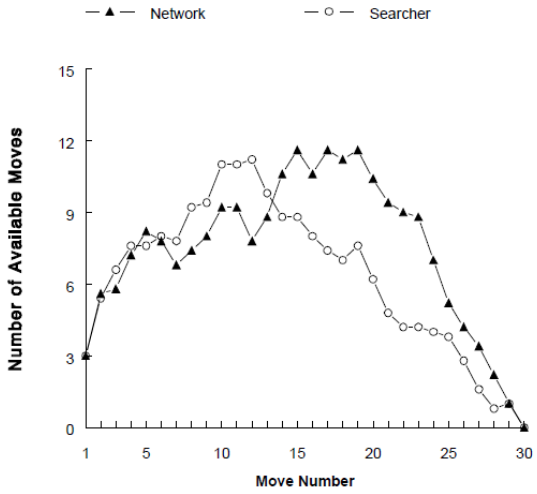

Figure 4. A graph showing the amount of possible moves at each turn for the ANN AI and the minimax AI [12].

David Shaman, the 1993 Othello champion gave his description of the ANNs strategy as someone who has been introduced to mobile strategy recently, often making good moves but not always the best moves. Moriarty and Miikkulainen mention that it is important to note that the ANNs were not "introduced" to mobile strategy as is the case with almost all human players, but instead discovered it on its own, displaying the strength of using genetically evolved ANNs as AI in games.

In [2] Siang Y. Chong, Mei K. Tan, and Jonathon D. White examine another use of genetically evolved ANNs in AI for Othello. Rather than letting the ANN decide the next move, the mini-max algorithm was used for decision making, but instead of the usual board configuration evaluation function accompanied with the algorithm, ANNs were trained to evaluate the board instead. The ANNs received each square as an input, and in the first hidden layer distributed these squares as all possible subsets of the squares to the hidden layer. The output is then the "value" of the board configuration, between -1 and +1. Chong, Tan, and White chose to use a static ANN architecture due to its simplicity.

Instead of playing against other types of AI opponents, the ANNs were trained using co-evolution, meaning that their fitness was evaluated depending on their performance against each other. Observation of the performance was done by pitting the evolving algorithms against the standard mini-max algorithm. In relation to the level expected by humans, from the beginning of the evolution to generation 400, the majority of the ANN supported algorithms play at novice level, increasing their performance to intermediate at generation 400 to 800 and finally performing at master level after generation 800.

Chong, Tan and White received in general similar results as Moriarty and Miikkulainen in [12], producing ANNs that develop strategies that depict positional strategy at the beginning and later evolve to the more advanced mobile strategy.

3.2 ANNs in Checkers

Chellapilla and Fogel [1] utilized a similar approach as Chong Tan and White to incorporate ANNs as AI in Checkers. The mini-max algorithm was again used for decision making for moves this time with majorly a ply-depth of 6, and ANNs were trained to be the evaluation function used for the minimax algorithm. For the evolutionary algorithm, a population of 15 individual static-structured ANNs was chosen to be evolved over 250 generations with co-evolution. Again, the ANNs received the board layout as input and provided the minimax algorithm the board value as output.

To test the trained AI, Chellapilla and Fogel let 90 human opponents play against it on the internet over a course of 2 weeks. According to a ranking system applied on the website which is explained in [1], the AI that was developed received a rating of "Class A", the fourth best possible ranking. It was also capable of defeating 2 "Expert" class opponents (third highest possible ranking) and reaching a draw against a "Master" (second highest possible ranking). It is worth mentioning that the top 10 players on the website all had the "Master" ranking. Chellapilla and Fogel suggest that if the AI had been given a higher ply-depth for the minimax algorithm it could have increased its performance, but nevertheless their results show the strength of evolved ANNs as AI, challenging greatly adept human players without being given any expert knowledge, or any a priori knowledge besides the rules.

3.3 ANNs in Tic Tac Toe and Five in a Row

Tic Tac Toe is an exemplary application for testing AI due to its simple rules and small number of possible moves due to the small playing area. Fogel [3] also incorporated evolving ANNs for AI in Tic Tac Toe. Fogel used a population of 50 dynamic structured ANNs evolved over 800 generations against a rule-based AI which played perfect tic tac toe apart from having a 10% chance to move randomly. The best ANN never lost to the rule-based AI, but neither was it able to win unless the random element of the AI forced it to make a mistake. Fogel mentions that the trained AI result could have also been obtained by co-evolution. In a study done by Yau and Teo [26] they show that evolution performance could

be increased by using adaptive EAs such as self-adaptive co-evolution [6].

Friesleben [4] took a slightly different approach to incorporating ANN based AI for 5 in a row. Instead of using evolutionary algorithms, he instead used a variant of comparison learning for the training of the ANN [23], A method training the ANN to play like its (ideally) better opponent. As mentioned by Friesleben, this method of learning involves disadvantages, since if the ANN were to lose to an ultimately inferior opponent, it would learn the inferior play style of the opponent. This method therefore obviously requires experienced well performing opponents in order to train the ANN. Frieslebens ANN was composed of two sub ANNs, each viewed unoccupied squares from the one player's perspective. With a weighted sum of these subnetworks as the output, a bias for attacking or defending could be evolved. The best trained ANN won 9 out of 10 games against a commercial AI and 7 out 3 against advanced human players.

4. AI FOR MODERN COMPUTER GAMES

After ANNs were used successfully for board game AI opponents, it was a natural consequence that ANNs would then be applied to modern computer games. This can be a bigger challenge than implementing AI for board games, as the rules may not be as clear and it may be hard to determine what behavior is suitable. Different games have different levels of complexity for implementing AI, and some games are more suitable as test subjects for ANNs than others.

4.1 Using ANNs in Unreal Tournament 2004

Unreal Tournament 2004 is a first person shooter game that can be played among humans or with computer opponents in different "free-for-all" or team based scenarios. In [20] Shu Feng and Ah-Hwee Tan used two classes of self-organizing neural models to try to emulate the behaviour of computer controlled opponents.

The first class that was used was Self-Generating Neural Networks(SGNN). The neurons in these networks are arranged in a tree structure where the training data creates "leaf nodes". A consequence of using SGNNs is a huge number of generated neurons, which is why Feng and Tan used a "pruning" method to check for redundant neurons and remove them. The second class that was used was Fusion Architecture for Learning and Cognition (FALCON), an Adaptive Resonance Theory based ANN with Temporal Difference learning incorporated. Feng and Tan's report includes more details about SGNNs and FALCON.

The AI opponents ("bots") in UT2004 have a set of possible behaviors that are adapted depending on state attributes, such as if the bot sees an enemy and how many health points it has. These rules are programmed manually. This experiment tried to emulate the behaviors of the so called "Hunter Bot" with SGNNs and FALCON by both offline training and testing, and online testing. In the offline training, the internal states, external states and behavior patterns of the Hunter Bot were used as training and testing data for both the two SGNN bots (one without the pruning) and the FALCON bot. There were 8000 training samples and 8000 test samples. Feng and Tan also measured the efficiency of the three solutions in terms of time and size.

As the training went on, the learning time of the SGNNs increased at a seemingly exponential rate as it created more and more neurons. The pruning did help with the efficiency but it was still not close to the efficiency of the FALCON bot that created a very small number of neurons and had an almost constant and very low learning time.

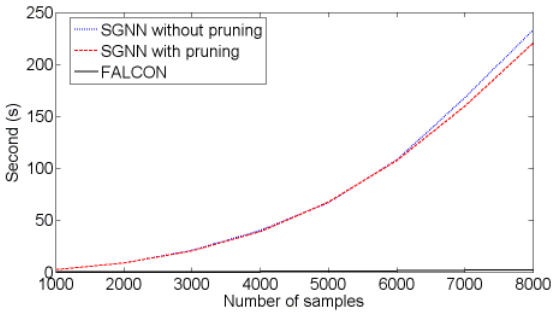

Figure 5. A graph showing the learning time as a function of the samples gone through for training. As shown. [20]

The online training consisted of 10 one-on-one "deathmatches" between the Hunter Bot and the three newly trained bots, one after the other. The first bot to kill the other one 25 times was the winner of the match. The SGNN with pruning showed the weakest results, losing by about an average of 4 points to the Hunter Bot. The SGNN without pruning was slightly better, only losing by about 2-3 points. The FALCON bot however, had basically the same capabilities as the Hunter bot and tied with it in games. Thus it was shown that while both SGNN and FALCON could be used to create bots in UT2004, Falcon was the most suitable.

No.	*IF (Condition)*	*THEN (Behavior)*
R_5	weapon loaded, 95.3% health, enemy spotted, and medical kits around	*Persue*
R_7	weapon loaded, 21.6% health, see some needed item, and it can be obtained	*GetMedicalKit*

Figure 6. Two examples of rules created by the FALCON bot [20]

Shu Feng also used SGNNs together with S. Chaudhari to create less predictable and more chaotic behavior of soldiers in a strategy game [19].

4.2 Using ANNs in racing games

ANNs have also been used to control cars in racing games. H. Tang, C. H. Tan, K. C. Tan and A. Tay compared the possibilities of creating AI for racing games by using neural networks or Behavior Based Artificial Intelligence (BBAI) [22]. The racing simulator used required two players to drive through waypoints in a decided sequence for a set amount of time. There are only two waypoints visible at any time, where one is active and one is not. When the active waypoint is driven through by any of the two players, it disappears, the next waypoint becomes active and a new inactive waypoint appears somewhere on the playing field. The player that had driven through most waypoints when the time was up had won.

The neural network was trained with an evolutionary algorithm, running for 200 generations with a population size of 100. The individuals were evaluated by comparison with the HeuristicSensibleController, which was one of 3 basic heuristic controllers, similarly to how the Hunter Bot was used in the UT2004 experiment.

When compared to the heuristic controllers, the Neural Network controller won in both solo races and one-on-one races, with a convincing win margin. The Neural Network controller did however get bested by the BBAI, of which more details are enclosed in [22]

4.3 Games centered around ANNs and training

There are a few games where the goal is to train ANNs for different tasks and then compete against other ANNs, either pre trained by the game creators or trained by another player. These games serve multiple purposes as they are both educational and entertaining, while exploring the possibilities of using ANNs in this manner.

The game NERO (NeuroEvolving Robotic Operatives) is a game that simulates a battle between two armies of robots. These armies' behavior is controlled by ANNs that are trained by the player in different training levels. In these levels the player can "teach" his army different behaviors by setting up increasingly difficult scenarios and determining which behaviors should be rewarded and which should be punished. The player can then choose to battle either one of the armies that come with the game or an army that a friend has trained. The method used for evolving these ANNs in real-time is called real-time Neuroevolution of Augmenting Topologies (rtNEAT)[21], and allows for a different gaming experience as well as a more visual presentation of machine learning, as the learning and evolution of the robots is visible in real-time.

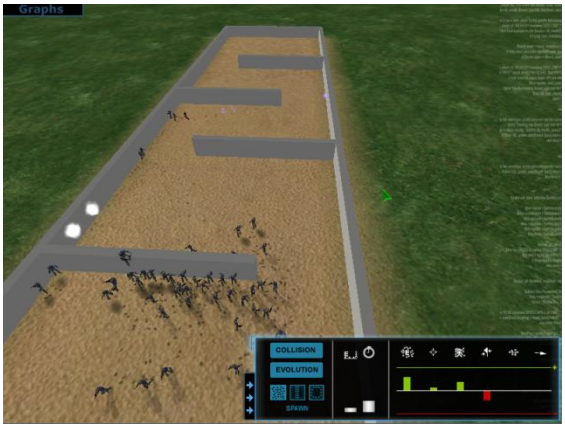

Figure 7. A screenshot from the NERO game. At the bottom of the screen an army is learning and attempting to navigate through the maze to the opposing army. At the bottom right of the screen are the sliders which control the reward system (for example the reward for finding the enemy, shooting the enemy, not getting hit etc.) and thereby the learning of the ANNs.

Another example of a game centered around training ANNs can be found at [16] where Charoenkwan, Fang, and Wong evaluated training an "avatar AI" to compete in mini games.

5. OTHER APPLICATIONS THAN AI

The nature of ANNs allows them to be applied to other fields than just to serve as AI players in games. Applications of ANNs to provide predictions of sports and games, detecting cheating in games and educational and medical interests.

5.1 Predictions

ANNs have been widely used for predictions throughout their development, due to their ability to decipher patterns [16]. Purucker in his work [16] predicted winning teams in the NFL. He used different ANNs that were trained with Back Propagation (BP), Adaptive Resonance Theory (ART) and Kohonen Self Organizing Map (SOM), to output the likelihood of a team winning based on the teams' statistics, and compared the training results. His work showed that Back Propagation used with ANNs resulted in the best performance, producing an ANN that could correctly predict 78.6% of all games in week 16 of the NFL in 1994.

The same method was used by Huang and Chang in [7] to predict the ability of a team winning in each football match in the 2006 World Cup and similar work has also been done by Rotshtein, Posner and Rakityanskya in [17] with genetic algorithms for training instead of BP.

Interestingly enough, not only the outcome of sports can be predicted but in a study by Chris Pedersen, Julian Togelius and Georgios N. Yannakakis in [14] predictions of player emotions (challenge, frustration and fun) based on the design of different levels in the game Super Mario Brothers are done using a ANN with a single perceptron evolved with an evolutionary algorithm. Depending on the number of inputs chosen, the best ANNs could predict the emotion fun with 69.18%, frustration with 88.66% and challenge with 77.77%. The authors mention that this ability could have great applications as to create the best possible level designs.

ANNs for prediction in video games are also shown in work by Kim, An, and Cha in [13], utilizing ANNs trained with BP to recognize player poses in a dancing game. After learning each basic move one at a time, the ANN could correctly recognizing 84% of the basic gestures.

5.2 Sociopolitic applications

ANNs can also be used to evaluate player performance in games, and if applied to certain types of games this performance information can be used for educational interests and medical diagnostics.

In [10] Lawrence, Carter and Ahmadi evaluate the use of an ANN based AI as an opponent in an educational Mathematics game. In the game where the objective is to correctly assess the area of a triangle, the AI acts as an adaptive "rival". The AI receives the dimensions of the triangle and the answer of the player, and is trained with BP to adapt to the player's level and thereby behave as a sufficiently challenging and evolving opponent. The aim with this method is to create a fun and challenging educational environment for students of different levels, and the authors mention future work including the addition of exercises including angles and possibly algebra.

Puzenat and Verlut evaluate the possibility to analyze young children's performance in games in order to

predict the age of a child [26]. Using 4 games that include simple instructions and "click and drag" actions, a BP trained ANN predicts the age of the child with a median error of 3 months based on the child's performance in just 10 minutes. This is on par with the best psychometric tests, but more importantly the authors suggest that given a larger training set, the program could be used to detect mental deficits for a large test group at a low cost. It is worth mentioning that this ANN was developed using the free open source library "Fast Artificial Neural Network" (FANN), a framework for quickly and easily creating and handling ANNs.

5.3 Cheat detection

In order to detect "speed cheating" in the Massively Multiplayer Online Game (MMOG) called Hoverkill, Gasperato, Barone and Schneider [5] evaluated the use of ANNs for this application. An MLP and Focused Time Lag Feedforward (FTLF) Network were evaluated and their performance was compared. The authors provided 800 training examples and an epoch limit of 1000, and after training the MLP network had the best results. It could correctly detect 21.14% of illegal moves, and produced 0.98% false positives. The FTLFN while being able to correctly detect 59.27% of illegal moves, unfortunately generated a 4.4% for false positives, and was deemed inferior because of this. According to the authors the ANNs gave significant advantages in cheat detection and showed good results in accuracy.

6 SUMMARY AND CONCLUSIONS

The ability to evolve and learn presents great possibilities for the use of ANNs as AI in games. In the earlier stages during the late 1980s and early 1990s it was discovered that ANNs could be used to create adaptive and creative AIs that were not rule based and did not require expert knowledge or significant a priori knowledge to train.

These AIs were first implemented on board games such as Reversi or Tic Tac Toe, given their simple rules and playing environment. Through different experiments evolving ANNs through evolutionary algorithms or other training methods, AIs that played at levels rivaling top human players and AIs based on expert knowledge were developed on several fronts. With modern day computational power, variations of evolutionary algorithms and implementation of fuzzy ANNs have been used to evaluate ANNs as AI for modern games. These AI operate in a much more advanced environment than the playing ground of a board game, and must be trained for more advanced tasks. Time is also a factor. In board games, the AI often has time to compute various different moves and outcomes and choose the most favorable action accordingly. In computer games played in real-time, there are much harder time constraints. Luckily, the nature of the ANN makes it very suitable for these sorts of real-time tasks as well, as has been demonstrated by its use in first person shooters and racing simulators**.** ANNs have also shown applications in predicting player experience in games, the outcome of sports matches, and helping diagnose cognitive disorders.

The results of AIs utilizing evolved ANN have showed promising results thus far in the development for new, adaptive and less predictable AI. With the availability of modern computational power and evolutionary algorithms game developers should find this field well worth investing time and training in.

Usually, artificial neural networks are studied by computer scientists, and not by game developers. This might be part of why ANNs have not been utilized more in gaming already, despite their obvious advantages. As the field of AI research extends its reaches, and game AI becomes more advanced, we should see more use of the ANN in gaming in the future.

7 FUTURE WORK

Since the use of ANNs in modern games is still in the evaluation phase, concrete future goals are not yet widespread. It is possible to speculate however, that given the results displayed so far by ANN based AI, it is a field that will with all likelihood become increasingly popular among game developers. If more developers become knowledgeable in ANNs, and if more complete frameworks and support software such as FANN are developed and become more common, the change to ANN based AI will with certainty occur faster.

Games in the same genre as NERO are still in their infant phases as well, and it is possible that games making machine learning the main aspect instead of a supporting role in the game, will create a new breed of strategy games. Meanwhile, developers will continue to strive to create more versatile and adaptive AIs using ANNs.

8. REFERENCES

[1] Chellapilla K., and Fogel, D.B, "Evolving Neural Networks to Play Checkers Without Relying on Expert Knowledge". IEEE Transactions on Neural Networks, Vol. 10, No. 6, November 1999

[2] Chong, S.Y., Tan, M.K., and White. J.D., "Observing the Evolution of Neural Networks Learning to Play the Game of Othello". IEEE Transactions on Evolutionary Computation , Vol. 9, No. 3, June 2005

[3] Fogel, D.B. "Using Evolutionary Programming to Create Neural Networks that are Capable of Playing Tic-Tac-Toe". IEEE International Conference on Neural Networks (ICNN), Vol. 2, 28 March – 1 April 1993, p. 875 – 880.

[4] Freisleben, B. "A Neural Network that Learns to Play Five-in-a-row". Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, 1995, 20-23 November 1995, p. 87-90.

[5] Gaspareto, O.B., Barone D.A., and Schneider. A.M. "Neural Networks Applied to Speed Cheating Detection in Online Computer Games". Fourth International Conference on Natural Computation, Volume 4, 18-20 Oct. 2008, p.526 – 529.

[6] Hinterding, R., Michalewicz, Z., and Eiben, A.E., "Adaptation in evolutionary computation: a survey", Evolutionary Computation, 1997, IEEE International Conference on 13-16 April 1997, pp 65-69.

[7] Huang, K.Y., Chang, W.L., "A neural network method for prediction of 2006 World Cup Football Game". The 2010 International Joint Conference on Neural Networks (IJCNN), 18-23 July 2010.

[8] Kaindl, H. “Tree searching algorithms,” in Computers, Chess, and Cognition, T. A. Marsland and J. Schaeffer, Eds. New York: Springer-Verlag, 1990, p. 133–168

[9] Koza, J.R. “Survey of Genetic Algorithms and Genetic Programming”, WESCON/'95. Conference record. 'Microelectronics Communications Technology Producing Quality Products Mobile and Portable Power Emerging Technologies', 1995

[10] Lawrence, W., Carter J., and Ahmadi, S. "A GCSE Maths Tutoring Game using Neural Networks". 2010 2nd International IEEE Consumer Electronics Society's Games Innovations Conference, 21-23 Dec. 2010.

[11] Lee, H. M, Huang, T.C., and Chen, C.M., “Learning Efficiency Improvement of Back Propagation Algorithm by Error Saturation Prevention Method”. International Joint Conference on Neural Networks 1999 (IJCNN ’99), Volume 3, 10-16 July 1999, p. 1737 – 1742.

[12] Moriarty, David E, Miikkulainen, Risto. “Discovering Complex Othello Strategies Through Evolutionary Neural Networks”, Connection Science, Volume 7, 1995, p. 195-209.

[13] Namho, K., YoungEun A., ByungRae, C. “Gesture Recognition Based on Neural Networks for Dance Game Contents”. Internation Conferencec on New Trends in Information and Service Science, 2009. (NISS '09). June 30 2009-July 2 2009. p. 1139 – 1139.

[14] Ng, S.C, Leungh, S.H, and Luk, A. “the generalized back-propagation algorithm with convergence analysis”. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, 1999 (ISCAS ’99), Volume 5, May 30 – June 2 1999, p. 612-615.

[15] Pedersen, C., Togelius J and Yannakakis, G.N. "Modeling Player Experience in Super Mario Bros". IEEE Symposium on Computational Intelligence and Games, 2009. CIG 2009, 7-10 Sept. 2009, p. 132 – 139.

[16] Phasit Charoenkwan, Shih-Wei Fang, Sai-Keung Wong. "A Study on Genetic Algorithm and Neural Network for Implementing Mini-Games". 2010 International Conference on Technologies and Applications of Artificial Intelligence, 18-20 Nov. 2010, p.158 – 165.

[17] Purucker, M. C., "Neural networks quarterbacking - how different training methods perform in calling the games," IEEE Potentials, August/September 1996, p.9-1S

[18] Rotshtein, A. P., Posner M., and Rakityanskaya A. B., "Football predictions based on a fuzzy model with genetic and neural tuning," Cybernetics and Systems Analysis, Vol.41, No.4, 2005, pp.619-630

[19] Shu, F., Chaudhari, N. S., "A Chaotic Behavior Decision Algorithm Based on Self-Generating Neural Network for Computer Games". 3rd IEEE Conference on Industrial Electronics and Applications, 2008. ICIEA 2008, 3-5 June 2008, p. 1912 – 1915.

[20] Shu Feng and Ah-Hwee Tan. "Self-Organizing Neural Networks for Behavior Modeling in Games". The 2010 International Joint Conference on Neural Networks (IJCNN), 18-23 July 2010.

[21] Stanley, K.O., Bryant, B.D., and Miikkulainen, R. "Real-Time Neuroevolution in the NERO Video Game". IEEE Transactions On Evolutionary Computation, Volume 9, number 6, Deccember 2005, p. 653-668.

[22] Tang, H., Tan, C. H., Tan K. C., and Tay, A. "Neural Network versus Behavior Based Approach in Simulated Car Racing Game". IEEE Workshop on Evolving and Self-Developing Intelligent Systems, 2009. ESDIS '09, March 30 2009-April 2 2009, p. 58 – 65.

[23] Tesauro, G. Connectionist Learning of Expert Preferences by Comparison training. in *Advances in Neural Information Processing Systems I,* pp. 99-106, Morgan Kauhan, 1989

[24] Uhrig, R.E. “Introduction to Artificial Neural Networks”. Proceedings of the 1995 IEEE IECON 21st International Conference on Industrial Electronics, Control, and Instrumentation, Vol. 1, 6-10 November 1995, p. 33 – 37.

[25] Verlut, I., Puzenat, D., “Behavior Analysis through Games Using Artificial Neural Networks”. Third International Conference on Advances in Computer-Human Interactions, 2010 (ACHI '10). 10-15 February 2010, p. 134 – 138.

[26] Yau, Y.J., Teo, J. “An Empirical Comparison of Non-adaptive, Adaptive and Self-Adaptive Co-evolution for Evolving Artificial Neural Network Game Players”. 2006 IEEE Conference on Cybernetics and Intelligent Systems, 7-9 June 2006.