Evolving and training of Neural Network to Play DAMA Board Game Using NEAT Algorithm

Banaz Anwer Qader[1], Kamal H. Jihad[2] and Mohammed Rashad Baker[3]
E-mail: banaz_2017@uokirkuk.edu.iq, kamal.jihad@uokirkuk.edu.iq, mohammed.rashad@sadiq.edu.iq

[1]College of Computer Science and Information Technology, University of Kirkuk, Kirkuk, Iraq

[2]College of Science, University of Kirkuk, Kirkuk, Iraq

[3]Collage of Information Technology, Imam Ja'afar Al-Sadiq University, Baghdad, Iraq

Keywords: dama game, NEAT algorithm, evolutionary algorithm, genetic algorithm, neuroevolution

Received: December 25, 2021

Neuroevolutionary algorithms, such as NeuroEvolution of Augmenting Topologies (NEAT) in Machine Learning (ML) methods, are utilized for training and playing computer games due to increased research in Artificial Intelligence (AI) field. NEAT is a genetic algorithm for the generation of evolving artificial neural networks. In this paper, a new study is presented. A Dama board game is designed, and the NEAT algorithm is implemented to develop and train the populations of neural networks for playing the game efficiently. Different inputs and outputs for the network are used, and various network sizes are tried for the game to reach or pass the human level. This paper aims to make a neural network that plays a Dama game like humans or is close by training different neural networks for many generations. The experimental results show that neural networks have been trained for several thousands of generations, and they have played more than one million games. It is concluded that using more input to handle information is better for the learning process. It is also found that a set of values for NEAT parameters is suitable for extensive neural networks like those used in this paper.

Povzetek: je predstavljena nova študija. Zasnovana je tradicionalna igra, imenovana "Dama", in implementiran je algoritem NEAT za razvoj in usposabljanje populacij nevronskih omrežij, da igrajo igro, prvič, na način, ki presega človeške zmožnosti.

1 Introduction

Since the 1940s, digital computer evaluation has been evident that computers can be programmed to execute very convoluted jobs such as proofs discovery of mathematical theories [1] and to play chess with excellent skills. Artificial Intelligent (AI) is the technique that is profusely used to apply and develop systems that deal with the different mental processes of humans, such as the ability to analyze and find the causes, generalize, discover meaning, or learn from past experiences [2, 3].

Machine Learning (ML) is a subfield of AI. ML is generally used to understand the data structure and fit that data into models, which people can utilize and understand [4]. Furthermore, ML algorithms allow the computers to train data inputs and use statistical analysis to output values that become within a specific range [5, 6]. Therefore, ML facilitates the processes of building models by computers from data samples to assist in decision-making processes relying on data inputs [4].

In the middle of the 20th century, there were attempts to simulate the human brain that finally arrived at something now known as the Artificial Neural Network (ANN) [7]. ANN is a generalized decision-making system, and it is a base for processing the brain's essential elements functions [8].

Simultaneously, some researchers were working to evolve the learning of computers depending on essential evolution keys in natural networks evolution. They arrived at evolutionary programming or Genetic Algorithms (GAs) [1, 9, 10].

At the beginning of the current century, there have been attempts to combine Neural Networks with GA. One of the successful attempts was Stanley's algorithm NeuroEvolution of Augmenting Topologies (NEAT) in 2002 [11, 12]. The NEAT algorithm works by creating the neural network in its simplest possible structure and letting it grow through evolving and recording its inherited genes information [3, 11, 13]. One of the most superficial neural networks is Feedforward Neural Network (FNN), where the input data travels only in one direction. The data passes through the input nodes and exits through the output nodes [14]. FNN does not have a back-propagation feature; therefore, there is no way to go back to the beginning and correct itself in case of any error in the output answer [15]. Since creating and developing ML, its algorithms have been used and tested on games, especially board games [6]. ML is a typical environment whose rules are clear and simple to test a theory and offers critical information to its algorithms or the neural networks [16]. In computer games, Neuro-Evolution is applied over a wide area more than supervised ML algorithms, requiring a correct tactic of input-output pairs [13].

Our contribution is to apply the NEAT algorithm on the traditional Dama game for the first time to make a neural network plays like a human or close to them by training different neural networks for many generations.

In this paper, a neural network using the NEAT algorithm was proposed to train and learn how to play the traditional Dama game. The proposed algorithm uses variables such as population, adding node chance, altering connection's weight, adding connection chance, other variables set for speciation, and mutating new generations.

The remainder of this paper will be organized as follows: section 2 will present related works, section 3 will explain the methodology of game board design and algorithm training, section 4 will discuss and display the experimental results of implementing and playing the game, and section 5 will present the conclusion and future works.

2 Related works

In this section, we show research and projects done with Checkers because there is almost no project that used neural networks on the Dama board game, and Dama and Checkers are similar games.

In 1999, a neural network was implemented on the checker's board game and was able to reach the human skill of the game. Using a minimax search strategy, they used Multilayer feed-forward neural networks to evaluate alternative positions of the board and games played. The additional information extraction required to play at this level depends on the competency evolution process. The best evolved neural network obtained after 250 generations was played against humans through 90 games online; it could defeat two expert players and draw against a master [17].

Neuroevolution was the best approach to general video game playing in a 2014 research of 61 Atari games that used four neuroevolution algorithms and three alternative state representations. The NEAT algorithm is used to play an available Atari 2600 game with indirect network encoding (HyperNEAT) [3]. Also, in 2014 a framework named Modular Multiobjective NEAT was used for modular neural networks evolution to play Pac-Man video games and treated the game as multiple tasks that required various modes of behaviour to succeed. The number of modules was learned by using mutation to duplicate and evolve these existing modules [18]. Then in 2015, the deep convolutional neural networks were trained to play the Go game by predicting the moves that the expert players of it has made. As a result, the trained networks were acquired improved performance and high levels of skill. For example, it could achieve 41.1% and 44.4% of move prediction accuracies on two different datasets of the Go game, passing other previous states of the art [19].

In 2016, the researchers used the NEAT algorithm to make a neural network to play the game 2048 and reach well behaviour. Different board mappings were developed trying to find the appropriate encoding for NEAT. All mappings got similar scores. Altering training parameters did not significantly impact the evolution, but recurrent edges introduced looked better [20]. In 2019, a study proposed a minimal training method to develop independent virtual players utilizing the NEAT evolutionary algorithm for evolving an agent to play the Flappy Bird game. The fitness function is a weighted average that relies on multiple scenarios with scenario-specific components. The neural network almost achieved perfect behaviour in the game, in which it took around 20 generations [21]. But in 2020, a new Statistical Forward Planning (SFP) method was presented. Different versions of Rolling Horizon NeuroEvolution of Augmenting Topologies (rhNEAT) were explored in a collection of 20 GVGAI games. This method was compared with other SFP methods and other results [22].

Table 1 summarises the examined publications in applying existing models used in game-playing. We highlighted our findings to focus on applied games utilized models, benefits, limitations, and the authors' accuracy of the applied proposed method.

3 Methodology of proposed method

This section will briefly discuss Dama game rules, strategies, and game design methodologies.

3.1 Dama game rules and strategies

Dama game is shown in figure 1, consisting of 64 equal areas squares. Each player has 16 coloured stones that differ in colour from the opponent stones. It has several rules: The role player moves the stone one square forward. At the beginning of the game, he must choose to move it from among the eight possible moves, then play continues, one movement for each player, in turn, the direction of playing one square forward. The order of eating the stone is one square after eating, whether in front, right, or left. The king plays in all directions and distances. The goal is to draw a strategy on setting up a trap for the opponent to eat the largest amount of stones and transfer at least one stone to any square in the last row of the opponent squares to turn this stone into a king.

Capture in the Dama game is obligatory, and it is not possible to eat the minority and leave the majority, whether it is taking a king or a pawn (stone). The stone gets the title of king (Dama) when it reaches the end of the last row of the competing team squares. The king advantage in the Dama game is that he moves anywhere vertically or horizontally. Therefore, he can eat more than one opposite stone without specifying the distance;

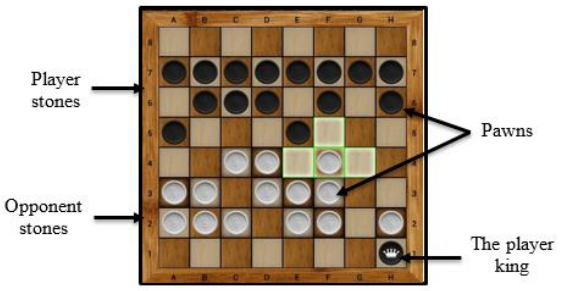

Figure 1: Dama game after few moves.

Study	Game	Model(s) used	Characteristic	Drawbacks	Accuracy
Hausknecht, et. al, (2014) [3]	Atari 2600 video game	Four neuro-evolution algorithms (CNE, CMA-ES, NEAT, and (HyperNEAT)	An empirical investigation of a variety of representations and algorithms in this domain.	The applied models did not solve some games complexity like Pitfall.	NEAT highest average z-scores across the object is 0.67 and noise is 0.09, but the Hyper NEAT highest scaling of pixels is -0.46
Chellapilla and Fogel, (1999) [17]	Checkers	Multilayer feed-forward neural networks	The best ENN can defeat expert-level players and even master.	The speed is limited to 3500 board positions per second. And using minimax to choose the optimum move.	1901.98 average wining rate at 90 games
Schrum and Miikkulainen, (2014) [18]	MS. Pac-Man	MM-NEAT	Modular networks find the multimodal activity more often than single module networks.	It is not rewarding multiple players module directly.	32,647 The average post-learning score for the FourMaze game and 65,447 Average post-learning score for MPMvs game
Clark and Storkey, (2015) [19]	Go	DCNN	predicting the moves made by expert Go players.	The limitation of network size	41.1% and 44.4% on two different Go datasets
Boris and Goran, (2016) [20]	2048	NEAT	Neuroevolution models can be used to solve game problems	Board mappings produce similar scores. Parameter modifications had less influence than planned.	An average score is below 600 till 5000Th generations
Cordeiro, Serafim and Neto (2019) [21]	Flappy Bird	NEAT	Using minimal training to create optimal scoring agents	It was not applied on simple platform games	An average score is below 2.0 with 100 generations
Liebana, Alam, and Gaina, (2020) [22]	10 General Video Game Playing	rhNEAT	rhNEAT's ability to adapt to changing game elements has helped it to set new world records in games where other approaches have struggled	Overall the results are not better than other Statistical Forward Planning methods	Overall winning rate 36.5%

Table 1: The summary table of related works.

considering the horizontal and vertical situation when the stone reaches the title of king, the color of the stone changes and becomes distinguished among the other stones. Kish (Go away) in Dama is optional and done by an opponent, then you can leave your stone in its current square or take it away, and sometimes it is mandatory when judged, for example (which stone will you leave? king or pawn). The Kish: This is a warning given by a stone or a king, and the player can use it defensively to level the playing field or offensively to win. Equalize is declared if the game ends between two players because each has 15 stones or one king. The game ends if one of the players wins the most stones or a king. If there are two kings for the player and one king for the competitor, the player who has two kings is considered the winner.

3.2 The proposed method

The proposed methodology contains three main phases: design, implementation, and application and evaluation. Figure 2 illustrates the methodology flowchart, which consists of designing the Dama board, implementing the algorithm on the game, training the algorithm, and testing and evaluating the game application by playing with trained generations of Neural Networks (NNs). The application design explains how it operates and its functionality in detail in the following subsections.

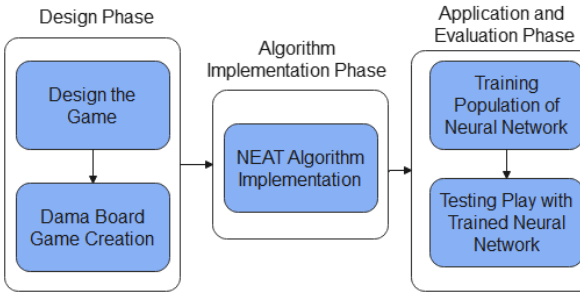

Figure 2: Flowchart of the proposed method.

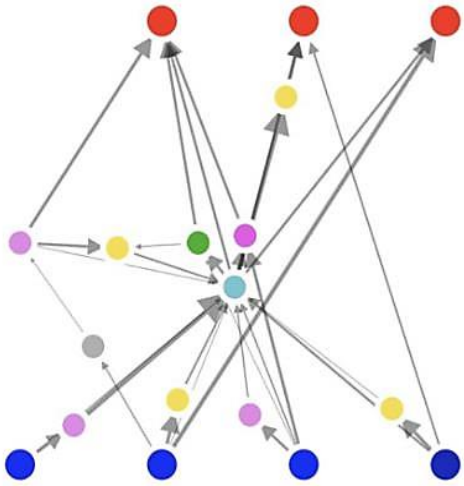

Figure 3: NEAT algorithm structure.

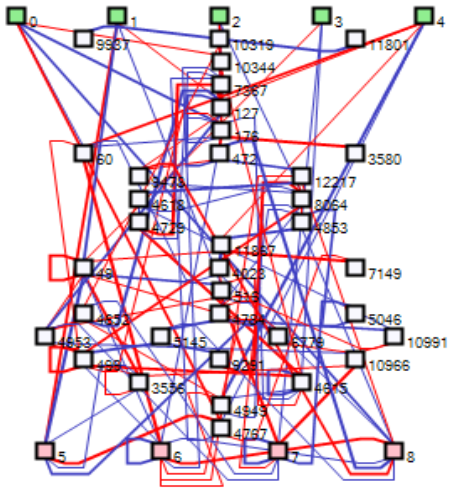

Figure 4: Growing an AI agent with NEAT.

3.2.1 Design phase

In this phase, a new board game Dama uses Windows Presentation Foundation (WPF) for the first time. The WPF is used for designing the application and the game board. First, the application interface is designed using XAML programming Language and built using C# programming language. Then the structure of the game is developed using XAML, where the board and Dama pieces are dynamically added using OOP with C#.

3.2.2 Algorithm implementation phase

NeuroEvolution is the most successful technique to be applied to the games. Evolutionary algorithms (EAs) are robust tools used to design and train neural networks [23]. EAs consist of several evolutionary processes: selection, mutation, and crossover (also known as recombination), as well as encoding and speciation. Recently, Stanley proposed the NEAT algorithm shown in figure 3, which has solved a series of topology problems that the other EAs could not solve, such as competing conventions [11]. The NEAT algorithm is built to evolve topologies and connection weights of ANNs, which simultaneously learns weight values and a suitable topology for a neural network. It starts with the most superficial structure networks in the initial population with less input, output nodes, and a series of connection genes between them, but with no hidden nodes. Thus, as time progresses, the complexity of networks' topology is developed, as illustrated in figure 4, combined with the idea of speciation if it is found necessary or valuable [24]. In addition, NEAT uses a more complex direct encoding methodology than other binary and simple graph encoding systems.

In NEAT, mutation can transform existing structures or add new structures (nodes or connections) to a network. Adding a new connection between a start and end node is associated with a randomly assigned weight. But adding a new node must be placed between two already-connected nodes, leading to disabling the previous connection [24]. NEAT algorithm tackles the competing conventions, which are one of the evolving issues of neural networks topologies, by marking the historical number of the gene. Competing conventions means blindly crossing over the genotypes of two neural networks, resulting in horribly and non-functional mutated [23]. Speciation is used to support the tackling process. It divides the population into several species (classes) depending on the topology and connections similarity.

3.2.3 Application and evaluation phase

The NEAT algorithm was implemented using the C# programming language. Then, a connection between the Dama game code and the NEAT algorithm code is created to train using the game. Finally, different encoding systems are designed that encode an 8 by 8 game board into a row of numbers to feed it into the neural network, as shown in figure 5.

The NEAT algorithm must start from the simplest neural network structure that consists of only input and output nodes and be fully connected. It means that each input node should be connected to each output node. In this work, two neural network structures are tried to get the best suitable NNs for the game training. The neural network structure in the first attempt did not enhance performance, leading to a second neural network structure in the second attempt. The latest one gets enhancement in comparison to the first one. The second structure was defended as a basic structure in the NEAT algorithm to generate populations of neural networks.

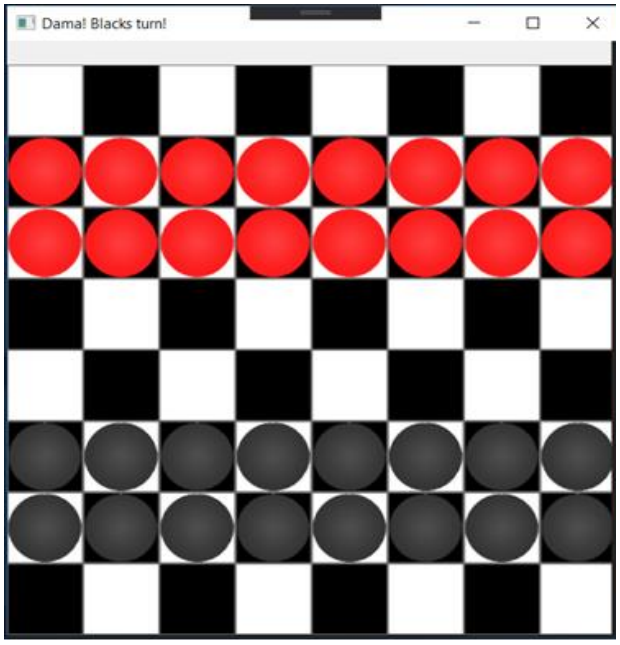

Figure 5: Designed Dama game board with neural network.

First Attempt

In the first structure, we used 64 input nodes. Each node represents a block on the board that takes different input values for each state on that block. Where, Input (0) for empty block, input (1) for the regular piece, input (3) for the king piece, input (-1) for the regular opponent piece, and input (-3) for the opponent king piece.

Whereas, for the output, we used 256 output nodes that represented as table 64 by 4. Each row represents the piece that should move, and each of those 4 columns represents a direction that the piece should move.

In this structure, there were 320 input-output nodes and 16,384 connections. Unfortunately, after months of training, it was found that the neural networks did not improve and did not work well because the input nodes were taking too much different information that caused bad connection weights. As a result, the algorithm could not get a good weight to work well with these inputs. Therefore, it is moved to try another neural network structure as a second attempt.

Second Attempt

To solve the previous problem in the first attempt, we changed the structure of the networks to be better. The output nodes were kept the same, but the input nodes were changed. In this attempt, the 192 input nodes are used that are represented as a table of 64 by 3. Each row represents a block on the board, and the columns represent the state of that block. Now the input nodes take values as a binary number. Where, Input (0,0,0) is for empty block, Input (1,0,0) is for the regular piece, Input (1,1,0) is for the king piece, Input (0,0,1) is for the opponent regular piece, and Input (0,1,1) is for the opponent king piece.

After months of training, this structure performs better than the previous structure. There were 448 input-output nodes and more than 49,000 connections in this structure.

Along with the training process, the training parameters had to be changed many times to get better values for our structure. Unfortunately, at the first attempt, the set of parameters values was too low, making the neural networks evolve too slowly, showing no progress.

The next time, the values of the parameters were increased. The neural networks size grew very fast because these values were too big for the neural network structure. Also, the values of the big parameters were too bad for the algorithm because it depends on real evaluation of nature, and things evolve slowly. After many attempts, we reached suitable parameters for the second structure and balanced the neural network between evolving and size growth.

The application is designed to execute two functions: training the NNs and evaluating the trained NNs. The training function includes selecting a path in the Personal Computer (PC) to save the trained NN, then starting the training process. The best-trained NNs are held, and the others are aborted. The evaluation function is performed by selecting one of the saved trained NNs, then playing with it. The application's main window shown in figure 6 is split into two parts. The first part on the left side of the window is used to select the neural network and play with it. In the second part on the right side, the parameters train neural networks with the NEAT algorithm. First, parameters are set to their default values. Then, select a location path to save trained neural networks.

The parameters setting and controlling shown in figure 6 are explained below:

- Population: it is the population (number) of neural networks. It has some values to be chosen from the drop-down list.
- Coefficients: The coefficients illustrated by equation (1) are used to tell how much two neural networks are compatible with each other. These coefficients control how much we depend on Excess and Disjoint Genes and their weight differences.

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \,.\bar{W} \quad (1)$$

- Compatibility Threshold determines the threshold we set as a factor for neural network species.
- Survival Chance refers to the number of neural networks that will remain and pass directly to the new generation.
- Weight Mutation Chance: it will set the chance to slightly mutate the value of the connections in the new generation's neural networks.
- Random Weight Chance: it will set the chance to generate entirely new value to connections in the new generation's neural networks.
- Add Node Chance: it will set the chance to add a new hidden node to the new generation's neural network.
- Add Connection Chance: it will set the chance to add a new connection between two nodes in the new generation's neural networks.

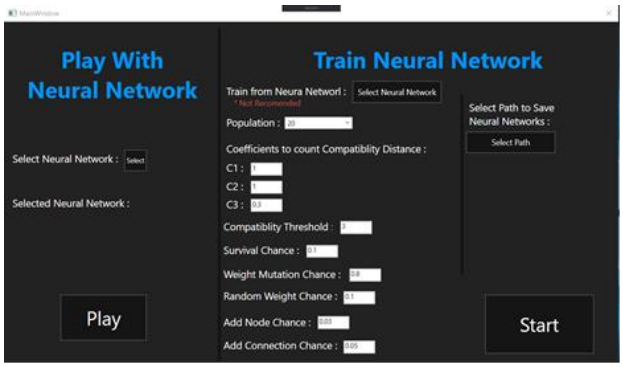

Figure 6: Application main window.

```
Average Moves: 44
Game Played: 601   Time: 5/20/2020 12:43:06 AM
-------------------------------------------------
Gen: 5048, Population: 100, Species: 1,     5/20/2020 12:43:13 AM
Start at: 5/20/2020 12:47:22 AM
Average Moves: 60
Game Played: 901   Time: 5/20/2020 12:48:49 AM
-------------------------------------------------
Gen: 5049, Population: 100, Species: 1,     5/20/2020 12:48:57 AM
Start at: 5/20/2020 12:53:37 AM
Average Moves: 78
Game Played: 1201   Time: 5/20/2020 12:55:28 AM
-------------------------------------------------
Gen: 5050, Population: 100, Species: 1,     5/20/2020 12:55:36 AM
Start at: 5/20/2020 12:59:53 AM
Average Moves: 95
Game Played: 1501   Time: 5/20/2020 1:02:10 AM
-------------------------------------------------
Gen: 5051, Population: 100, Species: 1,     5/20/2020 1:02:18 AM
Start at: 5/20/2020 1:06:42 AM
Average Moves: 111
Game Played: 1801   Time: 5/20/2020 1:09:21 AM
-------------------------------------------------
Gen: 5052, Population: 100, Species: 1,     5/20/2020 1:09:27 AM
Start at: 5/20/2020 1:14:05 AM
Average Moves: 114
Game Played: 2101   Time: 5/20/2020 1:16:52 AM
-------------------------------------------------
Gen: 5053, Population: 100, Species: 1,     5/20/2020 1:17:00 AM
```

Figure 7: Training results on console application.

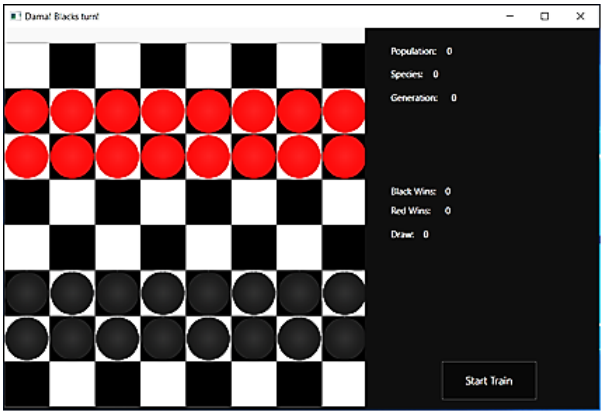

Figure 8: Training window of the game application.

4 Experimental results and discussion

This section will discuss the experimental setup, the results gained from the experimental, and related works.

4.1 Experimental setup

The PC used to implement the Dama game application has 8 GB RAM, 256 GB Hard Disk, Intel CORE i7 CPU @ 2.40 GHz, Windows 8.1 operating system of type 64-bit based processor. The WPF is used for the training process, which is a GUI Application. But due to using a low-performance PC that had to perform all the neural network calculations and visualize the game simultaneously. So a console application is used without visualizing the game so that the used PC could focus more on neural network calculations. The console application evolves NNs for generations to play as human behaviour or better. This approach is much better than training on WPF application but still slow for training such big-sized neural networks. After running console applications, training for months, playing more than one million games, and approaching more than 5000 generations, as shown in figure 7, the neural network has become better. However, because of the used low-performance PC and lack of training time, the neural network could not be powerful enough to defeat a professional human. The algorithm needs a few months to get to the human level with this PC performance available.

The training window of the game application is shown in figure 8. The neural networks can be trained by clicking the “Start Train” button. The neural networks are created and attached to the AI agent to start the training process. The training window shows live training with basic information about the training process. It also shows every match played by the neural networks and how they are playing.

The application will not store all the trained neural networks in the text files but only holds the best neural network from each species in every generation. The saving format for trained neural network data (nodes and connections) depends on connection information to store the whole neural network. As illustrated in figure 9, each row represents a connection from left to right. The first part of the row represents the input node to the connection, storing a node type and node id. The second part represents the output node from the connection. The third part describes the connection weight. Where (i) is the input node, (h) is a hidden node, and (o) is the output node.

In addition, the application will only store enabled connections data and ignore those connections that are disabled in the training process.

The left side of the main application window, shown in figure 6, is used to play with the trained NNs and evaluate the Dama game. First, a neural network file is selected to play with, and press the “Play” button to start play, as demonstrated in figure 10. Then, the Dama game board window shown in figure 5 appears and will be ready to play with humans. Human plays as Black, and the neural network play as Red; when the game is started, humans will have the first move to play. Also, he has to take out all the pieces from his opponent to win.

4.2 Experimental Results

The result of playing many peoples with many selected, trained neural networks is shown in table 2. As appeared in the table, each person has played more than once with five selected AI agents (stored trained neural networks) in different generations: AI1, AI2, AI3, AI4, AI5. In the beginning, the results proved that the winning rate of the machine was little with spending much playing time

```
i92 h646 0.4482778
h646 o312 0.1163678
i107 h647 0.02192972
h647 o430 0.9810541
h636 o262 -0.6545767
i63 h648 1.507824
h648 o373 0.3718817
i151 h649 0.08862734
h649 o289 0.3902431
h595 o427 -0.07256421
h540 o336 0.1770918
i98 h650 -0.7408667
```

Figure 9: Stored neural network data.

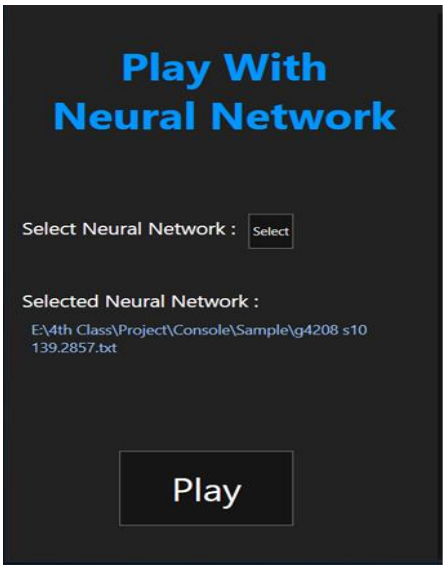

Figure 10: Play with selected neural network.

reaches to a few minutes. But the winning rate started increasing with decreasing the playing time by evolving the ANNs and training for progressed generations from AI1 towards AI5. The evaluation and playing results showed that the winning rate of the AI1 agent is 31.25%, spending much time, and it could not reach the human skills. While AI2's winning rate increased to 43.75% with less playtime, AI3's winning rate increased to 56.25 %, similar to AI2. Also, it is found that the winning rate of AI4 is 62.5%, with Lessing playtime being between less than 2 minutes and 53s. Finally, there is an apparent increase in AI5's win rate, 81.25%, with less playtime ranging from 1.5 minutes to several seconds. Increasing epochs and generations and progressing in time demonstrate the ability to reach AI5 to human levels and beat them due to advancements in the training process.

4.3 Discussion of the result

Here we will not go into standard criteria for comparing our work and other recent relevant studies that have used direct and indirect neuroevolution algorithms due to the apparent range of applicable games.

More than one million Dama games were played in our study, and NNs were evolved using NEAT for more than 5000 generations, yielding the top trained and grown neural network AI5 with an 81.25 % winning rate. However, the study [17] used 90 checkers games to develop NNs over 250 generations, resulting in a mean winning rate of 1901.98 for the best trained and evolved neural network and a peak winning rate of 1975.8 for game 74. The results demonstrate that the NEAT algorithm is superior to multilayer feed-forward neural networks when used in the gaming industry. HyperNEAT was incapable of learning based on object representation and noise representation when tested on a complete collection of 61 Atari video games (see table 1). Direct encoding also demonstrates the NEAT impairment of learning to play based on a visual, raw-pixel representation. In contrast, our research indicates the NEAT power of speed NNs learning to reach the most significant winning rate of 81.25 % in about 50 seconds.

In contrast to study [18], our work used the Dama game, which has many predetermined behaviors for each Dama piece ideal for direct neuroevolution algorithms like NEAT, whereas study [18] used MS. Pac-Man, which is a multi-behavior game with more than ten. As a result, indirect neuroevolution techniques such as Modular neural networks and Module Mutation networks are better for detecting multimodal behavior. On the other hand, the poorest modular networks do not utilize many modules, showing that the modular architecture's promise is not always realized. The Dama board, on the other hand, has a specific size that can be utilized using direct neuroevolution algorithms in our research.

5 Conclusions and future works

This paper contributed to implementing the NEAT algorithm on the Dama game, finding out how well it goes and the best structure for the neural networks. The game board should be encoded to give good clean information to the neural networks. The novelty of our work is using the NEAT algorithm to train and evolve NNs to play, for the first time, a traditional game named "Dama" with the human in a way that surpasses human skills. Furthermore, many different structures of neural networks and parameters' values of the NEAT algorithm were tried to get the best structure and set of suitable parameters.

We have trained the NEAT algorithm for two months. The result shows that training large neural networks with a large population, such as the one created in this paper, requires powerful computers for the training to reach high generations and provide results in a reasonable time. Furthermore, it was inferred that using console applications for PCs is much better and faster to train a significant population of big-sized neural networks because it does not require much CPU effort. Lastly, it is found that separating information into a more substantial amount of inputs instead of using a few inputs is better for the network and can be used easily with a good encoding system.

In the future, the Dama game can be trained and played faster by using a high-performance PC with

Player Name	AI1		AI2		AI3		AI4		AI5	
	Winner	Time	Winner	Time	Winner	Time	Winner	Time	Winner	Time
Ibrahim S. H.	Ibrahim	3:17m	AI2	2:05m	AI3	1:31m	Ibrahim	1:00m	AI5	1:03m
Ismail S. H.	Ismail	2:24m	AI2	2:12m	AI3	3:32m	AI4	2:25m	Ismail	3:07m
Hawraz R. A.	Hawraz	2:43m	Hawraz	2:07m	AI3	3:13m	AI4	58s	Hawraz	2:12m
Harman Y. I.	Harman	2:39m	Harman	1:56m	Harman	2:41m	Harman	1:09m	AI5	1:24m
Dylan A. A.	Dylan	3:11m	Dylan	2:21m	AI3	2:53m	AI4	2:01m	AI5	2:17m
Ibrahim R. S.	Ibrahim	2:11m	Ibrahim	1:44m	AI3	2:36m	AI4	1:54m	AI5	1:29m
Ahmed O. A.	Ahmed	3:17m	AI2	2:28m	Ahmed	3:03m	Ahmed	1:41m	Ahmed	2:28m
Ali H. B.	AI1	2:31m	Ali	2:37m	Ali	2:40m	AI4	55s	AI5	54s
Sirwan A. J.	AI1	2:50m	Sirwan	2:27m	AI3	2:15m	Sirwan	59s	AI5	57s
Maryam F. K.	Maryam	3:01m	Maryam	2:08m	Maryam	3:00m	AI4	1:38m	AI5	50s
Noor A. T.	Noor	2:49m	AI2	1:48m	AI3	1:29m	Noor	1:48m	AI5	1:02m
Amira D. L.	AI1	2:00m	Amira	1:59m	Amira	1:52m	AI4	1:05m	AI5	58s
Emad I. A.	Emad	3:12m	AI2	1:46m	AI3	1:42m	AI4	1:12m	AI5	1:10m
Fatma SH. E.	AI1	2:19m	Fatma	2:25m	Fatma	2:38m	AI4	53s	AI5	50s
Aya F. M.	AI1	2:03m	AI2	2:23m	Aya	2:18m	AI4	56s	AI5	1:19m
Zainab M. A.	Zainab	3:14m	AI2	1:30m	AI3	1:14m	Zainab	59s	AI5	1:30m

Table 2: The evaluation result by playing humans with neural networks.

considerable resources for training the neural networks because this process needs a lot of CPU effort and big-size RAM and HD. Also, finding a better encoding technique gives the neural networks better information. And eventually, using other ML algorithms that train the neural networks faster.

References

[1] E. Z. Elfeky et al. (2021). "A Systematic Review of Coevolution in Real-Time Strategy Games." *IEEE Access*, vol. 9, pp. 136647–136665, doi: 10.1109/ACCESS.2021.3115768.

[2] H. L. J. van der Maas, L. Snoek, and C. E. Stevenson (2021). "How much intelligence is there in artificial intelligence? A 2020 update". *Intelligence*, vol. 87, pp. 101548, doi: 10.1016/j.intell.2021.101548.

[3] M. Hausknecht, J. Lehman, R. Miikkulainen, and P. Stone (2014). "A neuroevolution approach to general atari game playing". *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 4, pp. 355–366, doi: 10.1109/TCIAIG.2013.2294713.

[4] S. Risi and J. Togelius (2017). "Neuroevolution in games: State of the art and open challenges". *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 1, PP. 25-41, doi: 10.1109/TCIAIG.2015.2494596.

[5] M. Bichler, M. Fichtl, S. Heidekrüger, N. Kohring, and P. Sutterer (2021). "Learning equilibria in symmetric auction games using artificial neural networks". *Nature machine intelligence*, vol. 3, no. 8, pp. 687–695, doi: 10.1038/s42256-021-00365-4.

[6] S. M. Miraftabzadeh, M. Longo, F. Foiadelli, M. Pasetti, and R. Igual (2021). "Advances in the application of machine learning techniques for power system analytics: A survey†". *Energies*, vol. 14, no. 16, pp. 11–14, doi: 10.3390/en14164776.

[7] X. Yang et al. (2021). "Research and applications of artificial neural network in pavement engineering: A state-of-the-art review". *Journal of traffic and transportation engineering (English Ed.)*, vol. 8, pp. 1-22, doi: 10.1016/j.jtte.2021.03.005.

[8] Simonov, A. Zagarskikh, and V. Fedorov (2019). "Applying Behavior characteristics to decision-making process to create believable game AI". *Procedia Computer Science*, vol. 156, pp. 404–413, doi: 10.1016/j.procs.2019.08.222.

[9] P. García-Sánchez, A. Tonda, A. M. Mora, G. Squillero, and J. J. Merelo (2018). "Automated playtesting in collectible card games using evolutionary algorithms: A case study in hearthstone". *Knowledge-Based Systems*, vol. 153, pp. 133–146, doi: 10.1016/j.knosys.2018.04.030.

[10] T. M. Martins and R. F. Neves (2020). "Applying genetic algorithms with speciation for optimization

of grid template pattern detection in financial markets". *Expert Systems with Applications*, vol. 147, pp. 113191, doi: 10.1016/j.eswa.2020.113191.

[11] K. O. Stanley and R. Miikkulainen (2002). "Evolving neural networks through augmenting topologies". ," *Evolutionary Computation*, *MIT Press*, vol. 10, no. 2, pp. 99–127,, doi: 10.1162/106365602320169811.

[12] K. C. Chatzidimitriou and P. A. Mitkas (2013). "Adaptive reservoir computing through evolution and learning". *Neurocomputing*, vol. 103, pp. 198–209, doi: 10.1016/j.neucom.2012.09.022.

[13] S. Lang, T. Reggelin, J. Schmidt, M. Müller, and A. Nahhas (2021). "NeuroEvolution of augmenting topologies for solving a two-stage hybrid flow shop scheduling problem: A comparison of different solution strategies". *Expert Systems with Applications*, vol. 172, doi: 10.1016/j.eswa.2021.114666.

[14] M. Şahin and R. Erol (2017) "A Comparative Study of Neural Networks and ANFIS for Forecasting Attendance Rate of Soccer Games". *Mathematical and computational applications*, vol. 22, no. 4, p. 43, doi: 10.3390/mca22040043.

[15] S. Murat H. (2006). "A brief review of feed-forward neural networks," Commun. *Fac. Sci. Univ. Ankara*, vol. 50, no. 1, pp. 11–17, doi: 10.1501/commua1-2_0000000026.

[16] M. Giannakos, I. Voulgari, S. Papavlasopoulou, Z. Papamitsiou, and G. Yannakakis (2020). "Games for artificial intelligence and machine learning education: Review and perspectives". *In Lecture Notes in Educational Technology*, Springer Science and Business Media Deutschland GmbH, pp. 117–133.

[17] K. Chellapilla and D. B. Fogel (1999). "Evolving neural networks to play checkers without relying on expert knowledge". *IEEE transactions on neural networks*, vol. 10, no. 6, pp. 1382–1391, doi: 10.1109/72.809083.

[18] J. Schrum, R. M.-P. (July 2014). "Evolving multimodal behavior with modular neural networks in Ms. Pac-Man". *in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Canada, dl.acm.org, pp. 325–332, doi: 10.1145/2576768.2598234.

[19] C. Clark and A. Storkey (July 2015). "Training deep convolutional neural networks to play go". *In 32nd International Conference on Machine Learning (ICML)*, France, vol. 3, pp. 1766–1774, Accessed: Dec. 10, 2021. [Online]. Available: http://proceedings.mlr.press/v37/clark15.html.

[20] T. Boris and S. Goran (2016). "Evolving neural network to play game 2048". *In 2016 24th Telecommunications Forum (TELFOR)*, Belgrade, Serbia, doi: 10.1109/TELFOR.2016.7818911.

[21] M. G. Cordeiro, P. B. S. Serafim, Y. L. B. Nogueira, C. A. Vidal, and J. B. Cavalcante Neto (Oct. 2019). "A Minimal Training Strategy to Play Flappy Bird Indefinitely with NEAT". *In 2019 18th Brazilian Symposium on Games and Digital Entertainment, (SBGAMES*), Rio de Janeiro, Brazil, vol. 2019-Oct., pp. 21–28, doi: 10.1109/SBGames.2019.00014.

[22] D. Perez-Liebana, M. S. Alam, and R. D. Gaina (Aug. 2020). "Rolling Horizon NEAT for General Video Game Playing". *In IEEE Conference on Games (CoG)*, Osaka, Japan, pp. 375–382, doi: 10.1109/CoG47356.2020.9231606.

[23] M. Wittkamp, L. Barone, and P. Hingston (2008). "Using NEAT for continuous adaptation and teamwork formation in pacman". *In 2008 IEEE Symposium on Computational Intelligence and Games (CIG)*, 2008, pp. 234–242, doi: 10.1109/CIG.2008.5035645.

[24] A. L. A. Paulino, Y. L. B. Nogueira, J. P. P. Gome, C. L. C. Mattos, and L. R. Rodrigues (2020). "On the Use of Cultural Enhancement Strategies to Improve the NEAT Algorithm". *In IEEE Congress on Evolutionary Computation (CEC)*, doi: 10.1109/CEC48606.2020.9185847.