Neuroevolution for NPC Fighting

Asher Uman
Department of Computer Science
Smith College
Northampton, Massachusetts
auman@smith.edu

Jace Clowdus
School of Cinematic Arts
University of Southern California
Los Angeles, California
jclowdus@usc.edu

Michael Weeks
Department of Computer Science
Georgia State University
Atlanta, Georgia
mweeks@ieee.org

***Abstract*—This paper tests the flexibility of an already established neuroevolution method for improving the performance of non-player video game characters. We use the same neural network with weights evolved through a genetic algorithm on multiple non-player characters and across multiple computers. This allows us to compare the results and see if the same neuroevolution is effective across multiple situations in the context of one game.**

***Index Terms*—neural networks, genetic algorithm, neuroevolution, games**

I. Introduction

Neuroevolution, or the process of evolving an Artificial Neural Network (ANN) through genetic algorithms, has been proven to be a competitive, and in some cases superior [1], method of generating a strong ANN, even when compared to more popular gradient descent-based techniques, such as backpropagation. Neuroevolution is especially effective for AI problems with sparse rewards, such as games where the success of a single move is difficult to evaluate until a winner is determined. In order to further study the viability of neuroevolution in such an environment, we conducted experiments through a game where two characters must fight against each other, as in [2]. By varying the types of characters evolved and running the game across different machines, we attempted to see if there were any noticeable differences in how the ANN evolved in different environments.

The game is structured like a classic fantasy adventure game, where the player character must traverse a dungeon and fight various enemies in order to complete a goal. For the purposes of testing and training the neural network, we restricted the gameplay to a "tournament mode", where the player character is restricted to a single room, controlled by a simple, hard-coded strategy, and fights thousands of battles against an opponent character, who is controlled by the neural network. Over the course of a "tournament", the weights of the network are progressively evolved by means of a genetic algorithm, which evaluates each fight and identifies the most successful individuals from each generation. New weights for the neural network are then produced from these individuals by using crossover and random mutation. By repeating this process, the algorithm is able to evolve even more successful individuals.

Thanks to NSF Grant 1852516 for supporting this project.

Fig. 1. A screenshot of the game, with the "player" character, a wizard, fighting against the neural network controlled ogre.

II. Related Work

Video games are a well-established environment for reinforcement learning (RL) research. The Arcade Learning Environment, a framework which allows for the creation of AI agents that can play Atari 2600 games [3], established the ability to perform well in Atari as a benchmark problem for RL agents, and has been extensively used by others in the field in order to experiment with different RL algorithms, and attempt to create agents that can perform well in many Atari games [4]. Many RL agents trained to play these games have implemented gradient-based algorithms, but neuroevolution has been shown to be a competitive alternative. Genetic algorithms perform better than other established RL algorithms on some games, and are overall faster to train [1].

Neural networks have also been trained to play a variety of other games, often using neuroevolution as well. NeuroEvolution of Augementing Topologies (NEAT), has been applied to games such as Flappy Bird [5] and Top Gear [6] with successful results. The success of neuroevolution in SNES games such as Top Gear is especially promising, as even highly competent gradient-based algorithms that perform well on Atari have struggled with the more complex environments of SNES titles. Agents controlled by an algorithm that combined neuroevolution with backpropagation also performed better

and more efficiently than backpropagation alone in the game Quake II [7].

Neuroevolution also seems to be a promising solution for the problem of creating more engaging, human-like non-player characters in games. Genetically evolved agents behaved in a more human-like manner in the game Super Mario Bros [8], and were judged by human players to be more challenging and engaging opponents in the fighting game M.U.G.E.N. [9]. This is important because a more human-like non-player character is more capable of immersing the player in a game than one that is clearly artificial.

Previous work on this project involved the creation of an original game and game engine in which to implement neuroevolution, as opposed to training a neural network to play an existing game. Due to this, modifications can more easily be made to the game, including the editing or creation of different characters and their weapons. This also brings us closer to the eventual goal of most research in game neuroevolution: the purposeful use of convincing AI agents within a game, instead of simply training an agent to play or take on some other role within an existing game. Previously, we also experimented with the number of layers in the neural network, leading to our current setup of a five-layer network [10]. The inclusion of feeding the agent's previous output back to it as an input was also shown to improve results, and we further refined the inputs of the ANN within this paper. The main purpose of our current work was to expand on and continue to experiment with our previously created game and agent.

III. Research Activities

A. Hardware and Software Description

The source code for this experiment was created mainly in Java, with some functionality added through PHP and Bash scripts. The program was run on three machines. Machine 1 is a 2017 Apple Macintosh running macOS 10.14.6, with a 3.2 GHz Intel Core i5, and 8 GB of RAM. Machine 2 is a Lenovo ThinkBook 14 running Windows 11 with an x64-based AMD Ryzen 7 at 1.80 GHz, and 16 GB of RAM. Machine 3 is a Dell G15 5510 running Windows 11 with an x64-based Intel Core i7-10870H processor at 2.2GHz and 16 GB of RAM.

B. Running the Software

The tournament was run through a Bash script, which ran in "steps". Each step runs 100 generations of the tournament, and each generation is composed of 50 matches between the player and ANN. The top 50 results from each generation are then saved in a XML file. Each match, the neural network-controlled enemy is scored on its performance, which is based on the remaining health of both the player and the enemy. These scores are then used to assess the fitness of each population in order to find parents for the next generation of the genetic algorithm, thus improving the performance of the enemy.

```
#!/bin/bash
#

declare -i step=1
declare -i max=10
MAXT=50
MAXG=100

while (( step <= max ))
do
  # run tournament
  java maze 2d include_states no_graphics -t $MAXT -g $MAXG > $fname
  # sort the results
  java maze -s
  # keep only top 50
  grep -n "<RESULTS>" NNdataTop50.xml > NNrecstart.txt
  # What line number does record 51 start with?
  stopline=`awk -F: 'NR == 51 { print $1 }' NNrecstart.txt`
  # print all lines up to $stopline
  awk -v SL=$stopline 'NR < SL { print $0 }' NNdataTop50.xml > realtop50.xml
  # Make a backup copy of the old version
  mv NNdata.xml "NNdata_$step.xml"
  cp realtop50.xml NNdata.xml
  mv realtop50.xml NNdataTop50_$step.xml
  # Clobber NNdataTop50.xml so that we do not append more to it.
  rm NNdataTop50.xml
  (( step ++ ))
done
```

Fig. 2. An example of the Bash script that runs the tournament.

C. Constructing the Neural Network

We constructed our neural network with combat in mind, and as such the inputs and outputs are specifically geared towards combat, although not towards any specific character. For our inputs we used the opponent character's distance to the player, whether or not the player was recently hit, whether or not the opponent was recently hit, the opponent's remaining health, whether the opponent is currently shielding, the opponent's distance to an enemy projectile, whether the opponent is colliding with a target, what the opponent did last, and a bias weight. These inputs are then run through five processing layers (with nine neurons each) which then output the opponents next action. These are "Choose" (waits to find a different action), "Close" (gets closer to the player), "Charge" (gets closer to the player at double speed), "Backup" (gets away from the player), "Retreat (gets away from the player at double speed), "Check" (moves towards the player with shield up), "Special" (uses a slow but powerful attack), "Hold" (stays in place and shields), "Ignore" (ignores the player), "Zig" (moves towards the player at a +45 degree angle), and "Zag" (moves towards the player at a -45 degree angle). This significantly reduced the size of our previous network, but does not seem to impact training, suggesting that it is possible to create a competent agent for games with a minimal number of inputs.

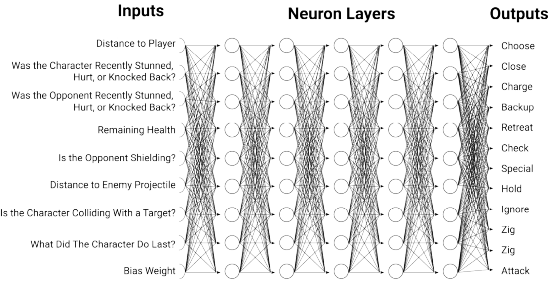

Fig. 3. A diagram of the inputs, processing layers, and outputs of our neural network.

D. Character Creation

For the purposes of testing neuroevolution on multiple characters, we created two new characters, the fire wizard and the bird, for a total of three when including the ogre. In order to properly test the flexibility of our neuroevolution, we designed these characters to have different strengths and weaknesses. Ideally, they would need to use different strategies in order to win, showing that our neural network is capable of creating different strategies depending on the scenario.

The first is one that already existed: the ogre. The ogre is a slow, large monster that is powerful up close. It wields a large hammer for close range and a sling for long range. Crucially, the sling is much less powerful than the ranged weapon the player has, meaning that relying on it alone would likely be an ineffective strategy for the ogre. The ogre's hammer, on the other hand, is a very powerful weapon, easily capable of out-damaging the player. In theory, the most effective strategy for the ogre would be to get in close and use its hammer to quickly dispatch the player.

The second character is the fire wizard. The fire wizard is a longer range character, but also has tools for up-close combat. They have a ranged fireball attack for distance and can surround themselves with flame (that only damages the player) at close range. The ideal strategy would be to shoot fireballs while far away and surround themselves with flame closer up. This is unique to our neural network because the character has two equally viable tools to deal with the enemy, so it must recognize which situations to use each one in instead of relying exclusively on one or the other.

The final character we added to the game is the bird. The bird is a small, fast character that can only fight up close. It wields a short spear that it can use to attack in front of itself. Unlike the other characters, the bird does not have a ranged attack, meaning its only option for combat is to put itself in a dangerous situation by getting close to the player. Its main advantage is its speed, which is higher than that of both the player and the other two enemies. The strategy we had in mind while designing it is one where it flies up close to the player, attacks, and retreats so that it does not get hit, over and over again. This challenges the neural network to go back and forth between different outputs as it repeats the attack cycle.

Fig. 4. (Left to right) The bird, fire wizard, and ogre as they appear in game.

E. Running the Tournament on Multiple Systems

After fixing some bugs, the tournament was able to run with identical performance on all three different machines. The program formats the results the same way regardless of machine, meaning that it is simple to take previous training from one machine and use it on another.

F. Issues and Limitations

We ran into a few issues while trying to run tests. Firstly, we had trouble even getting the tests to run at all, although that is now fixed. Because of the time it took to fix that, we ended up with much less time to actually run tests. This is an issue because properly evolving the neural network can take upwards of days and weeks. Because of this, we decided to limit our tests to only two characters: the ogre and fire wizard.

Additionally, we ran into an issue with the scoring very late into our research. The fire wizard is consistently getting scores far above what should be the maximum score. The character is winning consistently, but we cannot say with certainty that its scores are accurately reflecting its performance.

G. Initial Results

Due to the issues outlined in the previous section, and the fact that the majority of our research being spent on expanding the game and making it run successfully on different machines, we were unable to obtain conclusive results or data on all of our characters. However, initial results seem to suggest that while there is no noticeable difference between different machines, some of the new characters, and possibly the fitness function itself, may need to be re-balanced. Future work is needed to fully answer the question of how an agent's performance differs across characters, machines, and situations.

IV. Conclusion

This paper takes an already-existing neuroevolution method and improves it, laying the groundwork for future work testing its flexibility more thoroughly. We adapted our previous neural network to be more efficient by reducing the number of inputs. We then add two more characters: the fire wizard and the bird. We also modified the existing software to work effectively on multiple machines, and across operating systems. In this work, we lay the groundwork for a flexible, extensible way to test neuroevolution across machines, with easily addable and modifiable characters, and that is usable on multiple operating systems.

Continued research is needed to look into possible issues with evaluating the fitness of the characters, particularly the fire wizard. Given the fire wizard's strength, it likely should be balanced to be weaker. Future work should also include adding additional characters to see if neuroevolution works effectively for them as well. Additionally, we only tested each of our three non-player characters against a standard "human fighter" player character with a set strategy. An intriguing option for future study would be to see how different ANN characters evolve against opponents with differing abilities and strategies. Another interesting possibility would be to pit two ANN-controlled characters against each other, and evolve them both in response to their battles with each other.

REFERENCES

[1] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," 12 2017. [Online]. Available: http://arxiv.org/abs/1712.06567

[2] M. Weeks, D. Binnion, A. C. Randall, and V. Patel, "Adventure game with a neural network controlled non-playing character," *16th IEEE International Conference on Machine Learning and Applications*, vol. 2017-December, pp. 396–401, 2017.

[3] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 6 2013.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 12 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[5] T. Ramaswamy, Y. Pranati, C. V. S. Lahari, and S. Sanjana, "Teaching ai to play games using neuroevolution of augmenting topologies," *International Journal of Innovative Science and Research Technology*, vol. 7, pp. 1143–1145, 3 2022.

[6] S. Pham, K. Zhang, T. Phan, J. Ding, and C. L. Dancy, "Playing snes games with neuroevolution of augmenting topologies," *32nd AAAI Conference on Artificial Intelligence*, vol. 32, pp. 8129–8130, 4 2018.

[7] M. Parker and B. D. Bryant, "Lamarckian neuroevolution for visual control in the quake ii environment," *2009 IEEE Congress on Evolutionary Computation*, pp. 2630–2637, 2009.

[8] J. Ortega, N. Shaker, J. Togelius, and G. N. Yannakakis, "Imitating human playing styles in super mario bros," *Entertainment Computing*, vol. 4, pp. 93–104, 4 2013.

[9] G. Martínez-Arellano, R. Cant, and D. Woods, "Creating ai characters for fighting games using genetic programming," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, pp. 423–434, 12 2017.

[10] M. Weeks and D. Binnion, "Training a neural network controlled non-playing character with previous output awareness," *Proceedings of the 2019 ACM Southeast Conference*, pp. 202–205, 4 2019.