Neuroevolution of Augmented Topology in a Snake Video Game

MC Cizungu 222008349 (November 2024)

CSCO3P3

Computer Science and Informatics with AI,

University Of Johannesburg,

Abstract

Implement a Snake video game, using a reinforcement method, NEAT (Neuro evolution of augmented topology) such that the agent(snake) can learn to navigate its environment, targeting food in a dynamic, real-time manner. The snake will be able to locate its food source without user input, with the aid of the most optimized neural network over generations.

Keywords

NEAT, TWEANN, neural network, neuro-evolution, 2D grid snake game, reinforcement learning

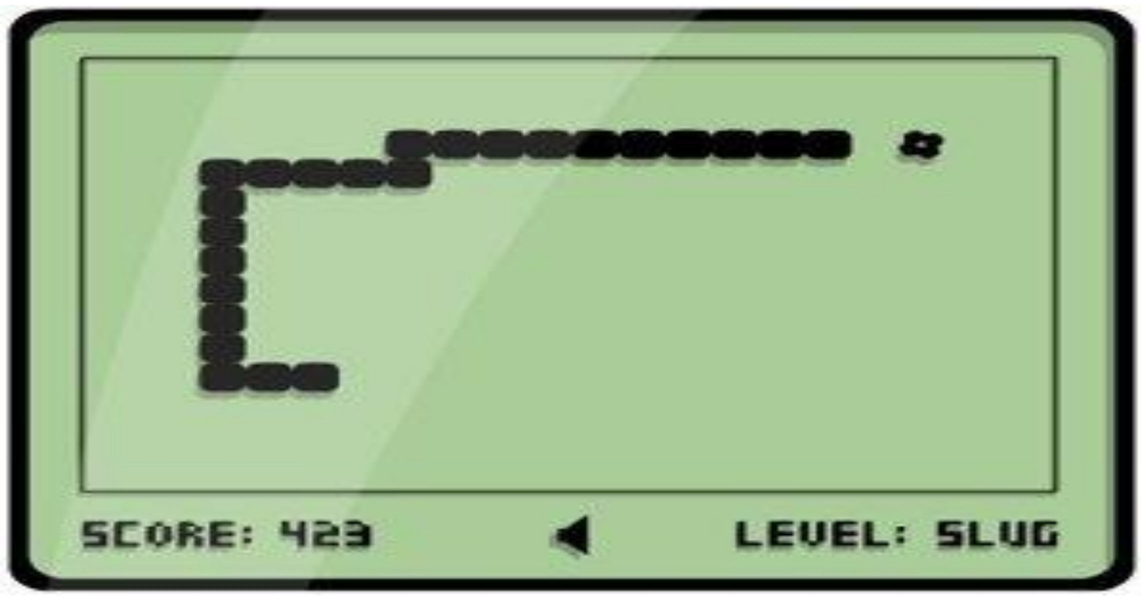
Introduction

NEAT is a combination of evolutionary algorithm and neural network infrastructure. Like natural selection in nature, which is driven only by feedback from reproductive success, neuroevolutionary is guided by some measure of overall performance. Whereas the most common artificial neural network learning algorithms operate through supervised learning and therefore depend upon a labelled corpus of input-output pairs, a main advantage of neuroevolutionary is that it allows learning even when such corpora are not available, based only on sparse feedback. For example, in game playing, vehicle control, and robotics, the optimal actions at each point in time are not always known; it is only possible to observe how well a sequence of actions worked, e.g. resulting in a win or loss in the game. Neuro evolution makes it possible to find a neural network that optimizes behaviour given only such sparse feedback, without direct information about what exactly it should be doing. (Kenneth O. Stanley, Bobby D. Bryant, & Risto Miikkulainen, Real-Time Neuroevolution in the NERO Video Game, 2005)

Furthermore, neuro-evolution generalizes to a wide range of network architectures and neural models; applying neuro-evolution requires only that the performance of networks can be evaluated over time, and that the behaviour of the networks can be modified through evolution. While most neural learning methods focus on modifying only the strengths of neural connections (i.e. their connection weights), neuro evolution can additionally optimize other parameters, such as the structure of the network (e.g. adding neurons or connections), the type of computation performed by individual neurons, and even learning rules that modify the network during evaluation.

In this paper, we will show how NEAT is used in a video game to evolve a snake agent from infant to expert in forage, making future games more interesting and less predictive and hence improving game longevity. In addition, reinforcement learning using Q-Learning (Quality learning) will be used as a comparison ground to the NEAT algorithm, contrasting the parameters and overall performance. (Kenneth O. Stanley, Bobby D. Bryant, & Risto Miikkulainen, Real-Time Neuroevolution in the NERO Video Game, 2005)

Game Background



The snake game is a classic video game that dates to the 1970s and has since appeared in various forms on different platforms. Its core gameplay revolves around controlling a line or "snake" that grows longer as it consumes objects, often represented as food, displayed randomly on the game environment.

In the game of Snake, the player uses the arrows (up, down, left and right) to control the snake's movement in the two-dimensional environment. The snake grows larger as it finds and eat the food. The game terminates when the snake either eats itself or bumps into a barrier(wall). The goal is to make the snake as large as possible before any termination conditions. (Jia-Fong Yeh, Pei-Hsiu Su, Shi-Heng Huang, & Tsung-Che Chiang)

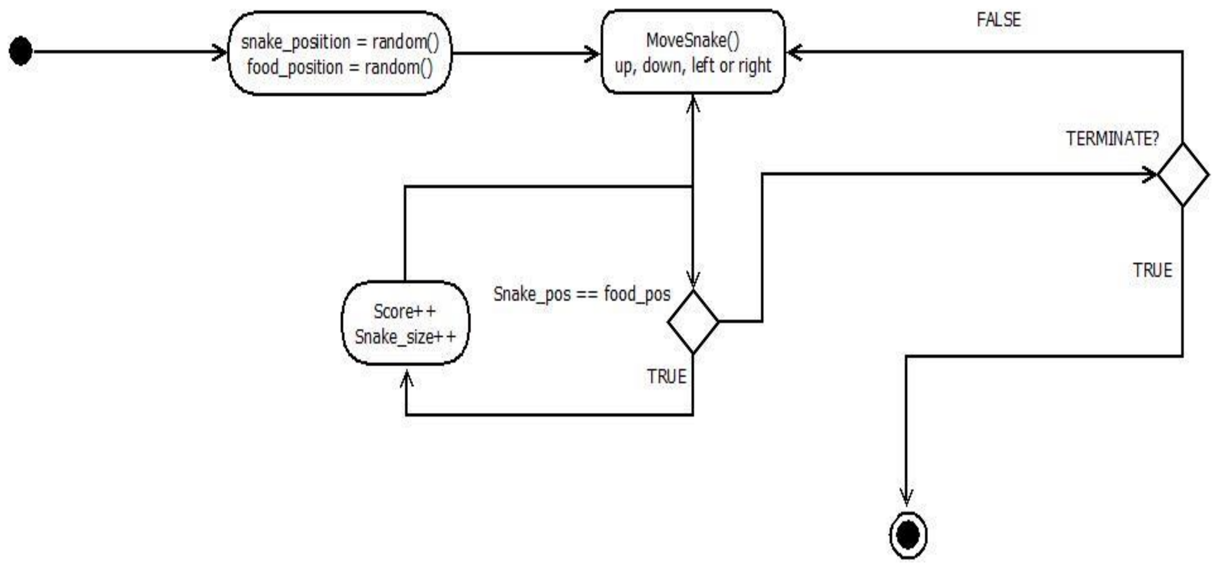


Figure 1 Snake Game State Diagram

In the above Figure 1 shows the snake state diagram. Initially, the snake and food occupy a random location respectively. The agent or player makes a move as shown by the transition arrow (one of the four possible moves). If the snake position is the same as the food position, then the player gains some points, and the snake size increases as a result. However, if the termination conditions are met: the snake moves into itself or the snake hits the walls of the two-dimensional world, then the game is over; the player must start from the beginning.

The game ends when any of the following four conditions is satisfied:

(1) The snake hits its own body or the wall.

(2) The game score does not change after a fixed MAX time units

Neural Network

A class of machine learning models called neural networks is modelled after the architecture and operations of the human brain. They are made up of layers of interconnected nodes, or neurons, that process information and may identify patterns, anticipate the future, and carry out various functions. (Kenneth O. Stanley & Risto Miikkulainen, Evolving Neural Networks through)

Structure

Neurons*:* Fundamental units that receive input, process it, and produce an output. Neurons in a network mimic the biological neurons in the human brain.

Layers*:* A basic neural network will contain usually 3 layers-

1. Input Layer: Receives raw input data and passes it to the network.
2. Hidden Layers: Intermediate layers between input and output. They process the input through weighted connections and apply activation functions to produce outputs.
3. Output Layer: Produces the final output or prediction.

***Connections**:* Each neuron in one layer is connected to every neuron in the subsequent layer through weighted connections. These weights represent the strength of the connections and are adjusted during the learning process. (Roopal Tatiwar)

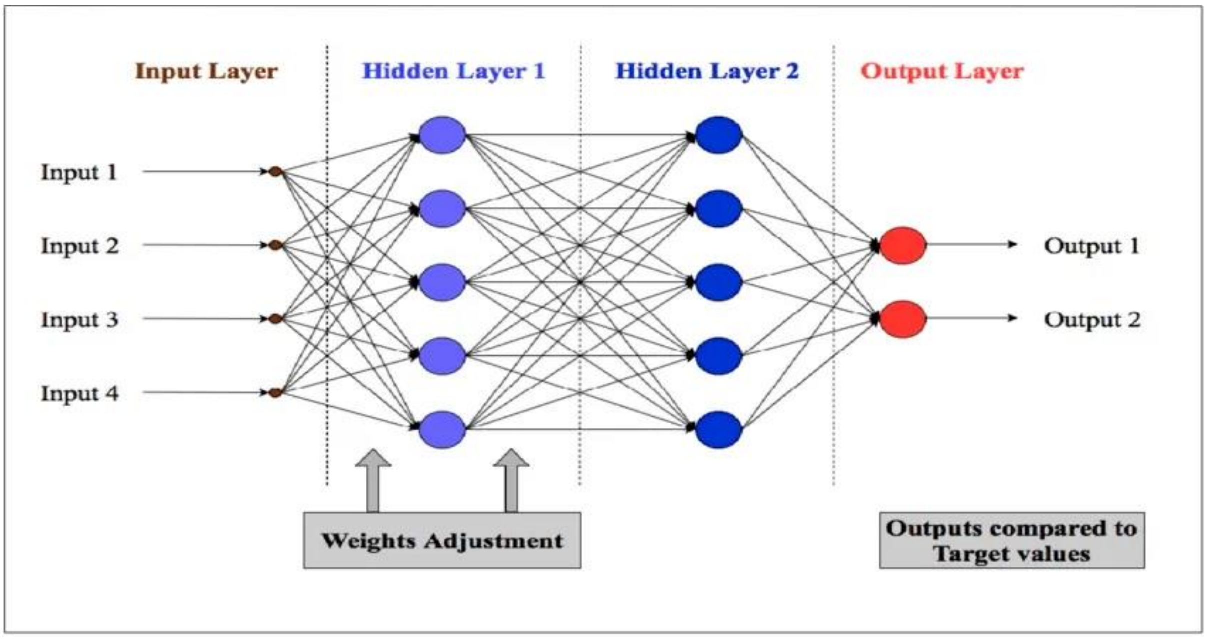


Figure 2 Artificial Neural network structure

Steps within the training phase of a neural network:

1. **Forward Propagation**: During feedforward propagation, input data is passed through the neural network from the input layer to the output layer. Each layer performs

computations using weighted connections and activation functions, producing the final output of the network. This step calculates the predicted output based on the current set of weights in the network. (Roopal Tatiwar)

2. **Backpropagation**: Following feedforward propagation, backpropagation is the process of computing the gradient of the loss function with respect to the weights. This gradient is then used to update the weights of the network, adjusting them in the direction that minimizes the loss. Backpropagation is crucial for training the neural

network. By iteratively adjusting weights based on the calculated gradients, the network learns to make more accurate predictions over time. NEAT does not use this feature, as weights are adjusted by other evolutionary mechanisms. (Roopal Tatiwar)

Genetic Algorithm

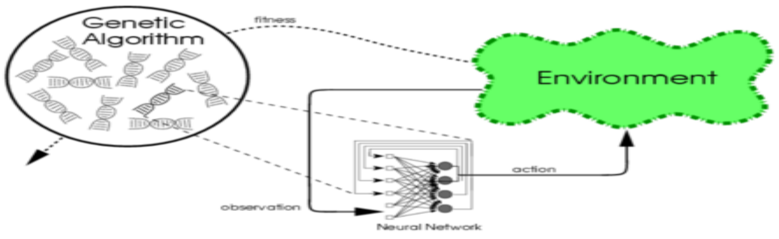


Figure 3

The computer methods known as genetic algorithms (GAs) are derived from the ideas of natural selection and evolution. They are employed for optimization and search issues and are members of the evolutionary algorithm family. GAs, which were created by John Holland in the 1970s, mimic natural selection to produce answers to a variety of issues. (Lehman & Miikkulainen, 2013) (Roopal Tatiwar)

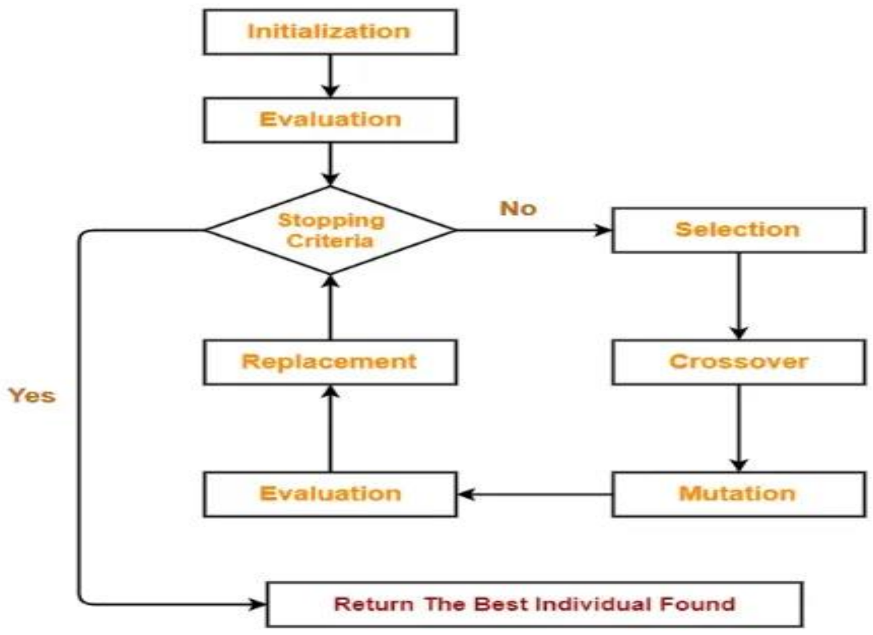


Figure 4

1. ***Chromosomes and Genes:*** Solutions to a problem are represented as chromosomes, typically encoded as strings of binary digits (0s and 1s) or a matrix.
   Each position in the string represents a gene, which holds a piece of information

contributing to the solution. With respect to the snake game. Chromosomes are represented as a two-dimensional matrix.

2. ***Initialization:*** The process starts by creating an initial population of snake agents' solutions (neural networks) , randomly generated with varied number of hidden layers(topology) and random number of neurons per layer.
3. ***Fitness Evaluation:*** Each solution's fitness, i.e., how good it is at solving the problem, is evaluated using a fitness function. The fitness function measures how close a solution is to the optimal solution. In this case, an agent's fitness is measured on how well it plays the game by avoiding collisions with the environment and itself, with an aim to get closer to the food location, being rewarded when it minimizes distance to food, avoid repeated moves, and duration of survival.
4. ***Selection:*** Solutions are selected probabilistically, favouring solutions with higher fitness scores. Common selection techniques include roulette wheel selection, tournament selection, or rank-based selection. So based on the points that each snake agent receives, they are selected to move on to the next generation. I use Elitism as my selection mechanism
5. ***Reproduction:*** Selected solutions (parents) undergo genetic operations — crossover and mutation — to create new solutions (offspring). Crossover involves exchanging information between parent solutions to produce new solutions. Mutation introduces small random changes to individual solutions to maintain diversity. So, the topology of two parents is exchanged and weights are tweaked by a random, small numbers to promote exploitation.
6. ***Termination:*** The process iterates through generations until a stopping criterion is met, such as reaching a maximum number of generations or achieving a satisfactory solution: A snake that accurately navigates the environment for a longer period, with little to no terminations or mistakes, hence the best snake is the one with highest fitness value. (Roopal Tatiwar)

Neuro Evolution of Augmented Topology

Neuro Evolution of Augmented Topology, also referred to as NEAT is a method we shall use to evolve artificial neural networks (ANNs), that not only optimize the weight of a fixed topology but also evolves the topology itself. (Lehman & Miikkulainen, 2013)

Initialization

The process begins by generating an initial population of neural networks. Each network starts with a simple topology, typically consisting of only input and output nodes without hidden layers

Genome Representation

Each neural network is encoded as a genome, which is a list of genes. There are two types of genes:

Node Genes: Represent the neurons (input, hidden, and output nodes) in the network.

- **Connection Genes**: Represent the connections (with associated weights) between the nodes. Each connection gene includes:
  - **Innovation Number**: A unique identifier for the gene, crucial for tracking and aligning genes during crossover.
  - **Weight**: The strength of the connection between two nodes.

Speciation

Speciation: NEAT organizes networks into species based on their structural similarity. This process helps preserve diversity in the population by protecting innovative structures from being prematurely discarded.

Distance Metric: The similarity between two genomes

g_1 and g_2

is calculated using a distance metric

$$d\left(g_1,\ g_2\right)$$

By formula:

$$d\left(g_1,\ g_2\right) = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot W$$

Where:

E – number of excess genes

D – number of disjoint genes

W – average weight difference of matching genes

C1, C2, C3 are coefficients

N – is the number of genes in the larger genome (used to normalize the distance)

Crossover

Recombination: NEAT uses crossover to combine the genomes of two parent networks to create offspring. Matching genes are inherited randomly from one parent, while disjoint and excess genes are inherited from the more fit parent.

$$g_c = combine(g_1, g_2)$$

Where g1 and g2 are two parent genomes, and child g_c is formed by the combined method which aligns genes correctly.

Mutation

Weight Mutation: The weights of the connections in the network can be mutated by adding a small random value:

$$w^{`} = w + \alpha \cdot w$$

Structural Mutation: NEAT allows the structure of the network to change via two types of mutations:

- **Add Node Mutation**: A new node is added by splitting an existing connection. The original connection is disabled, and two new connections are created.
- **Add Connection Mutation**: A new connection is added between two previously unconnected nodes.

Result of the system

$$
\begin{aligned}
gridSize &= 10\\
Generations &= 100\\
Population &= 100\\
\max Steps &= 500\\
SpeciesElitism &= 2\\
recentActionsHistory &= 5\\
initialMutationRate &= 0.075
\end{aligned}
$$

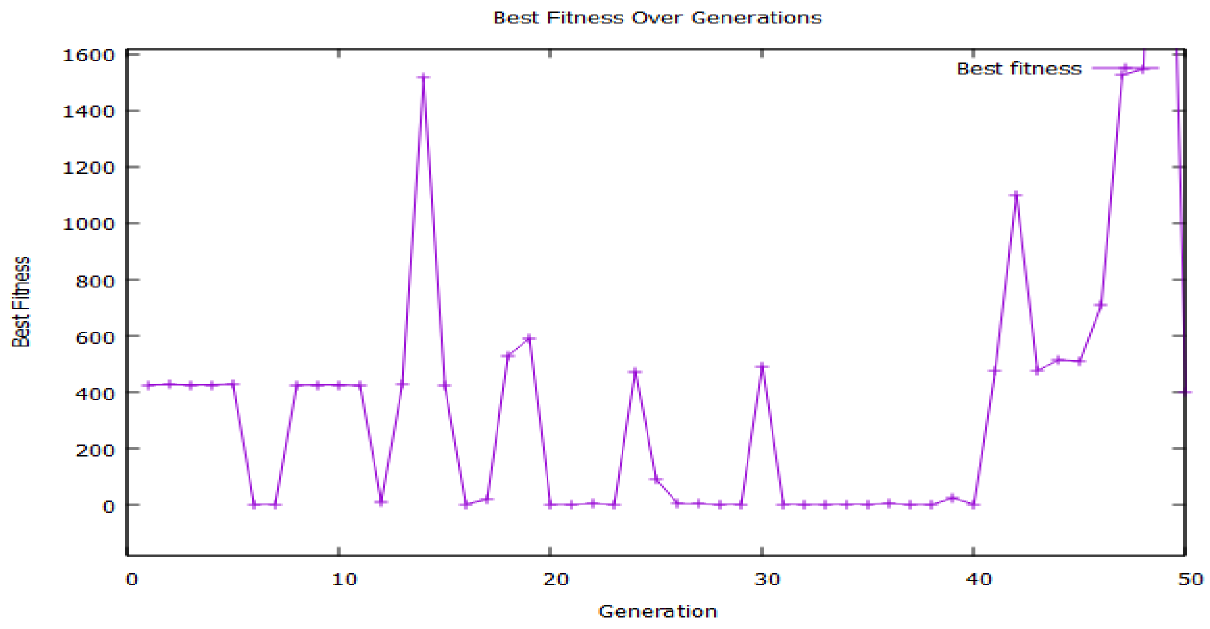


Figure 4 Best fitness visual

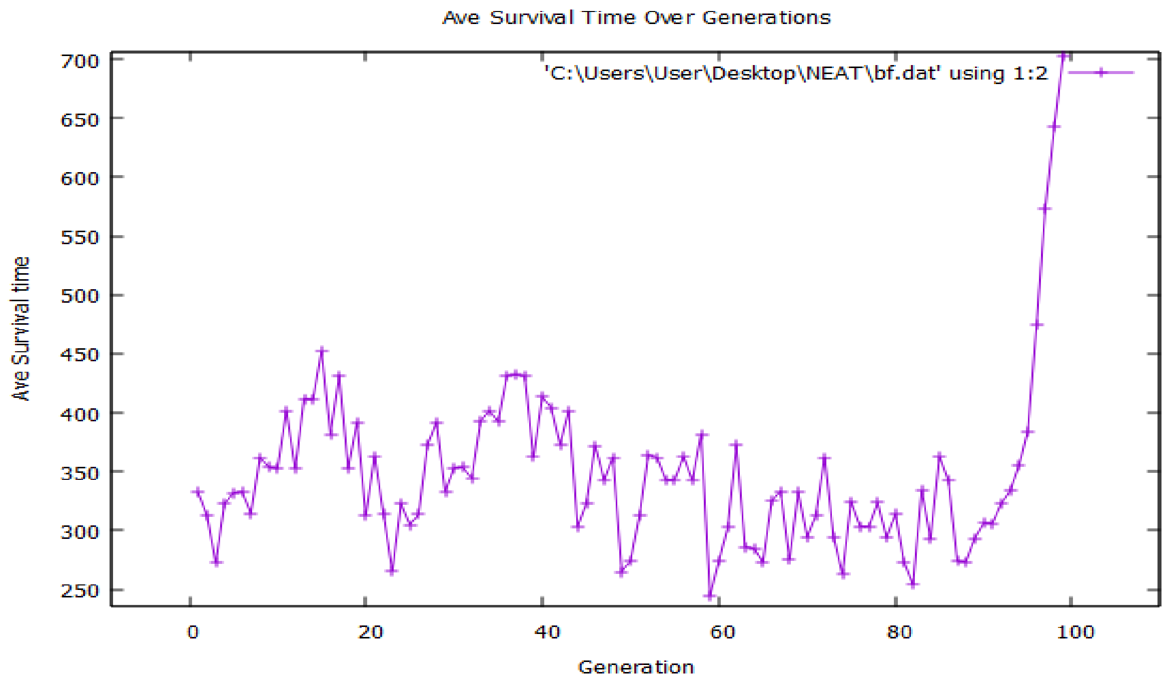


Figure 5 Ave survival time

Analysis of the results

Figure 4 illustrates the correlation between the best fitness achieved and the generation count throughout the evolutionary process. The diagram reveals significant fluctuations in fitness during the initial phases, primarily due to the random initialization of weights within the neural networks. In these early stages, the agents exhibit a wide range of performance levels, which is a natural consequence of the diversity introduced by these random weights.

As the evolutionary process progresses, the speciation mechanism further influences this variability. Speciation promotes the emergence of distinct subpopulations that can develop specialized traits, leading to varied fitness scores across generations. This dynamic interplay allows for a rich exploration of the solution space, contributing to the observed fluctuations in fitness levels.

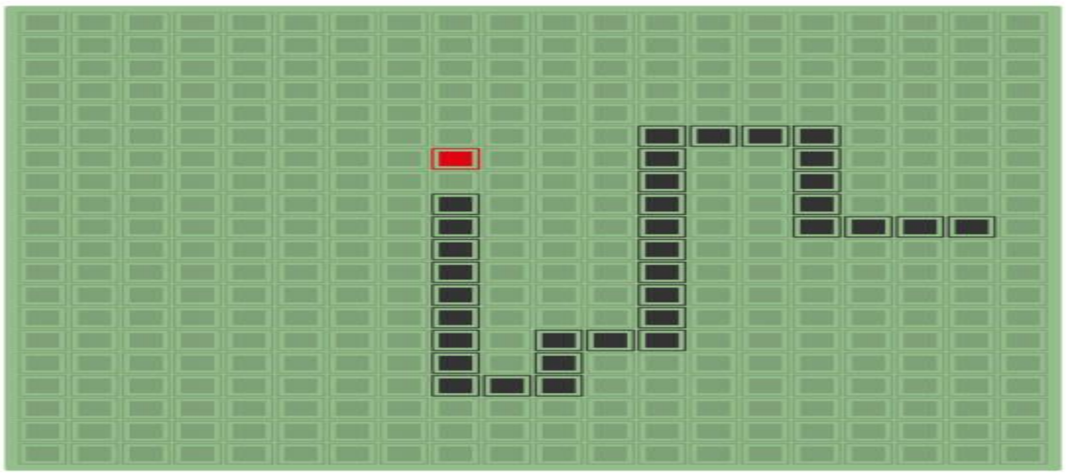
The consistent enhancement of fitness values over time can be attributed to several factors. One crucial aspect is the evolution of network topology, which allows the agents to adaptively modify their architecture based on the challenges presented by the environment. Additionally, crossover mechanisms, which combine genetic material from parent genomes, facilitate the sharing of successful traits among agents, fostering a collaborative evolution process. Such interactions enable the system to converge toward more effective solutions as generations advance, ultimately leading to a better representation of NEAT (Neuroevolutionary of Augmenting Topologies) fitness values.

Figure 5 depicts the average survival duration of the "snake" genomes within the game environment. A longer gameplay duration, measured in "Max steps," translates into higher rewards for the snake. The reward system is intricately designed to incentivize not only movement toward food but also strategic avoidance of self-collisions and wall impacts, reflecting a nuanced understanding of the game's objectives. This dual reward structure encourages the agents to prioritize safe navigation while actively seeking out resources.

Furthermore, the NEAT fitness function plays a pivotal role in minimizing the average distance between the snake's head and the food source, promoting effective foraging behaviour. By rewarding the snake for successfully closing the gap between itself and the food, the system drives it to develop strategies that optimize its movement patterns. Additionally, the fitness function encourages diversity in movement sequences by providing rewards for avoiding repetitive actions. This aspect is crucial for preventing infinite loops, where the snake might get trapped in a cycle of movements without progressing in the game. By promoting varied behaviour, the evolutionary algorithm fosters a more adaptive and resilient agent that can handle the complexities of the game environment.

To sum up, the insights gained from Figures 4 and 5 highlight the intricate relationship between evolutionary dynamics and agent performance. The fluctuations in fitness scores reveal the challenges inherent in early-stage evolution, while the gradual improvements underscore the effectiveness of NEAT's adaptive mechanisms. Together, these figures paint a comprehensive picture of how evolutionary strategies can enhance the capabilities of agents, ultimately leading to more sophisticated gameplay experiences.

Conclusion



In this study, we have demonstrated the efficacy of the Neuro Evolution of Augmenting Topologies (NEAT) algorithm in evolving a snake agent, progressing from a novice to an expert forager in a dynamic gaming environment. The results highlight several key advantages of NEAT:

1. *Adaptability to Sparse Feedback*: Unlike traditional supervised learning approaches that rely on labelled datasets, NEAT effectively learns and optimizes behaviour through sparse feedback. This characteristic is particularly beneficial in game scenarios where the optimal actions are not always apparent. Our snake agent successfully adapted its strategy based solely on performance outcomes, illustrating the power of neuro evolution in complex, real-time decision-making contexts.

2. *Optimization of Network Structure*: NEAT's ability to evolve not only the connection weights but also the network topology and neuron parameters allowed for a more flexible and sophisticated learning process. This adaptability facilitated the emergence of diverse strategies in the snake agent, enabling it to outperform conventional methods that focus solely on weight adjustments.

3. *Enhanced Game Longevity and Engagement*: The evolved snake agent demonstrated improved gameplay performance, making the game experience more engaging and less predictable. This evolution not only enhances player interest but also contributes to the longevity of the game, as the agent continues to develop novel strategies over time.
4. *Comparison with Q-Learning*: When contrasted with reinforcement learning using Q-learning, NEAT has shown promising results, demonstrating that neuro evolution can achieve competitive performance without the need for extensive training data. While Q-learning requires a substantial amount of trial-and-error to converge to optimal strategies, NEAT's evolutionary approach accelerates the learning process by leveraging performance feedback across generations.

The promising results of our NEAT implementation pave the way for further exploration into hybrid models that combine the strengths of neuro evolution and traditional reinforcement learning techniques. Future studies could focus on optimizing parameters and integrating more complex neural architectures to enhance agent performance in increasingly challenging environments.

All in all, our findings validate NEAT as a powerful tool for evolving intelligent agents in gaming and other domains, showcasing its potential to enhance both gameplay experiences and the development of adaptive artificial intelligence systems.

(Jia-Fong Yeh, Pei-Hsiu Su, Shi-Heng Huang, & Tsung-Che Chiang)

Reference

Jia-Fong Yeh, Pei-Hsiu Su, Shi-Heng Huang, & Tsung-Che Chiang. (n.d.). *1 Snake Game AI: Movement Rating Functions and Evolutionary Algorithm-based Optimization*.

Kenneth O. Stanley, & Risto Miikkulainen. (n.d.). Evolving Neural Networks through. *The MIT Press Journals*, 30.

Kenneth O. Stanley, Bobby D. Bryant, & Risto Miikkulainen. (2005). *Real-Time Neuroevolution in the NERO Video Game.* Austin,.

Lehman, J., & Miikkulainen, R. (2013). Neuroevolution.

Roopal Tatiwar. (n.d.). Neuroevolution: Evolving Neural Network with Genetic Algorithms. *Medium*.