Brain Simulator: Snake Brain with Genetic Algorithm



Brain is a neural network simultor tool implemented in Java using the Processing environment. The project aims to provide a visual representation of the decision-making process of a neural network controlling the movements of a snake. The tool incorporates a genetic algorithm to optimize the neural network's performance. Through real-time visualization, users can gain insights into the neural network's behavior, identify patterns, and witness the improvement achieved through the genetic algorithm.

**1- Problem Statement**

The objective of Brain is to develop a snake brain simulation tool that showcases the decision-making process of a neural network (how the brain of the snake work) controlling the snake's movements. The neural network is optimized using a genetic algorithm, in addition to being designed from scratch and implemented in Java using the Processing environment. The tool provides a real-time or iterative visualization of the network's decision-making during brain activity.

**2- Neural Network Architecture**

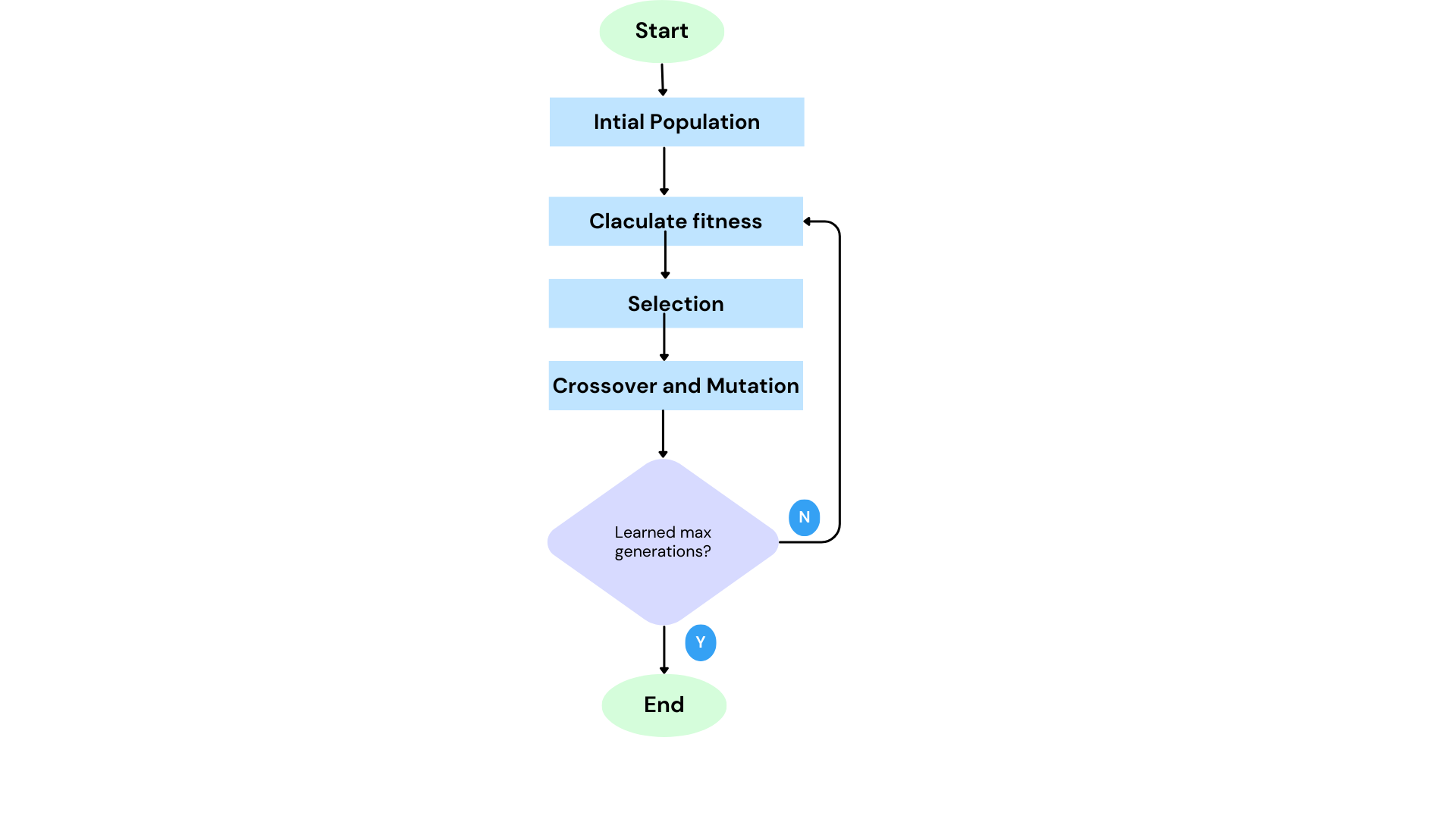
The neural network architecture for Brain is as follows:

- Input Layer: The input layer consists of 24 nodes representing distances from the snake's body, walls, and food in eight directions. Each distance is encoded as a floating-point value

- Hidden Layers: Brain includes two hidden layers, with each layer comprising 18 nodes. The Rectified Linear Unit (ReLU) activation function is used for the hidden layers

- Output Layer: The output layer consists of four nodes representing the possible directions that the snake can move: up, down, left, and right. Each output node produces a value between 0 and 1, indicating the probability of selecting the corresponding direction.

**3- Genetic Algorithm**

****

Brain incorporates a genetic algorithm to optimize the neural network's performance. The genetic algorithm involves the following steps:

1. Initialization: A population of neural networks is randomly generated, each with its own set of weights.

2. Evaluation: Each neural network in the population is evaluated by playing multiple snake games and measuring its fitness based on performance metrics such as survival time or score.

3. Selection: Neural networks with higher fitness are more likely to be selected for reproduction. Selection methods such as tournament selection or roulette wheel selection can be used.

4. Reproduction: Selected neural networks undergo crossover and mutation operations to create new offspring. Crossover combines the genetic material of two parent networks, while mutation introduces random changes to the offspring's weights.

5. Replacement: The offspring replace a portion of the previous population, ensuring the population size remains constant

6. Iteration: Steps 2 to 5 are repeated for multiple generations, allowing the population to evolve over time

**4- Simulation Features**

Brain offers the following visualization features:

- Overlay Simulation: The simulation overlay is presented on the snake's brain structure, providing a clear representation of the neural network's position, food cues, and other relevant information.

- Activated Neurons: Activated neurons are highlighted during the decision-making process. This visual distinction can be achieved through color or size variations, making it easier to observe the active parts of the neural network.

- Connection Visualization: Connections between neurons, representing the weights of the neural network, are visually displayed. This visualization may utilize lines, arrows, or other graphical elements to represent the strength and direction of the connections.

- Real-time or Iterative Visualization: The visualization updates in real-time or at regular intervals to depict the decision-making process of the neural network as the snake moves and interacts with its environment.

- Genetic Algorithm Metrics: The tool displays relevant metrics related to the genetic algorithm, such as the current generation, best fitness value, average fitness value, and other statistics to track the progress of the optimization process.

-----------------------------

**6- Contributing**

Contributions to the Brain project are highly appreciated. If you encounter any issues or have suggestions for improvements, please submit a pull request or open an issue in the project repository.

When contributing, kindly adhere to the existing coding style and follow the project's guidelines.