Flappy Bird AI using a NEFT Genetic Algorithm

Section A0 - Rahat Rahman & Shyam Kannan

Problem Statement:

In the realm of artificial intelligence and game development, creating intelligent agents that can adapt and respond to dynamic environments is a pivotal challenge. Flappy Bird, a game characterized by its simplicity and the dynamic nature of its gameplay, provides an ideal platform for exploring machine learning algorithms. The primary objective of this project is to **develop an AI that can proficiently play Flappy Bird by autonomously navigating through the series of obstacles** presented in the game. To achieve this, we propose the implementation of a neural network model, specifically a single-layer perceptron, which is a foundational building block in neural network theory.

However, the conventional training methods for neural networks, such as the gradient descent algorithms, may not be directly applicable or efficient for a game like Flappy Bird where the learning feedback is discontinuous and binary (either the bird passes through the gap or it doesn't). To address this, we turn to an alternative approach using the Neuroevolution Fixed Topologies(NEFT) genetic algorithm. This algorithm offers a robust method to optimize the perceptron by treating the problem as one of evolutionary survival, where only the best-performing AI through successive generations is developed.

This project aims to **explore the effectiveness of the NEFT genetic algorithm** in rapidly evolving a perceptron that can handle the game's requirements. By focusing on this method, we anticipate not only achieving a high level of play but also demonstrating the feasibility of using genetic algorithms for training neural networks in settings where traditional backpropagation is less effective. The successful implementation of this project could provide insights into the broader applicability of genetic algorithms in neural network training, particularly in real-time decision-making scenarios.

Network Architecture:

The neural network designed for this project comprises three input nodes, one bias node, and a single output node, all contained within one layer.

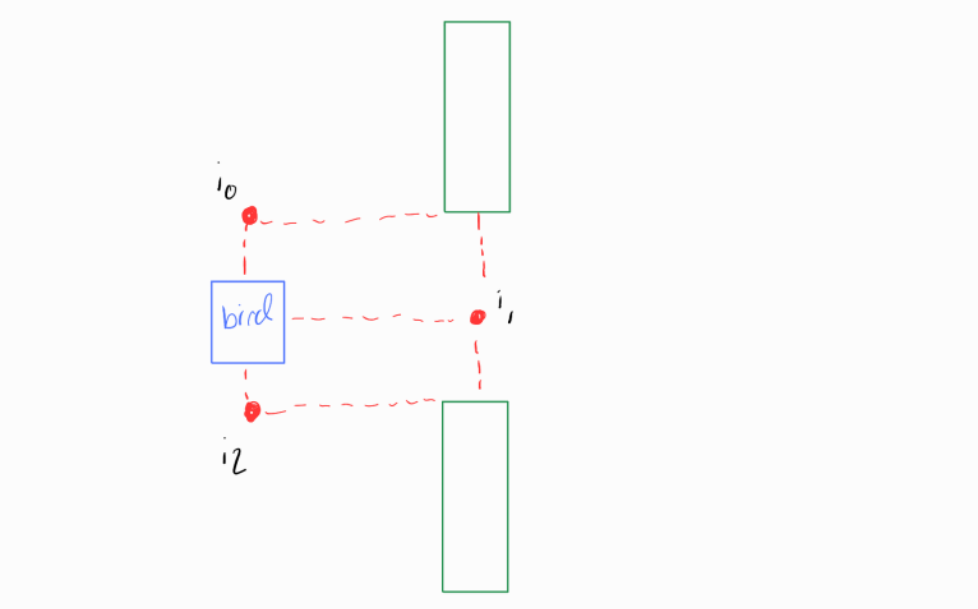
Input Nodes:

- **i_0 and i_2:** These nodes are crucial for assessing the vertical distances of the bird from the top and bottom pipes, respectively (see Figure 1). This spatial information is critical as it influences the bird's decisions to flap at moments that will avoid obstacles. Initially they are set to 0.5 each.
- **i_1:** This node evaluates the horizontal distance between the bird and upcoming pipes (see Figure 1). Monitoring this distance helps the AI determine the timing of the flaps to maintain a flight path that navigates through the gaps between the pipes successfully. Initially it is set to 1.
- **i_3 or the bias node:** The inclusion of a bias node (always set to 1) is instrumental in our network. It allows the model to learn an appropriate threshold more effectively during the

training phase, making the network adaptable to various game scenarios without manual recalibration.

Output Node:

- The single output node processes inputs from the three data nodes and the bias node to decide whether the bird should flap or not. The decision is based on a threshold activation function; if the output exceeds 0.73 (value found from error and trial), the bird flaps (this happens only when the bird is not already flapping). This threshold was determined to optimize the bird's response time and agility, enhancing its ability to navigate through tight spaces.



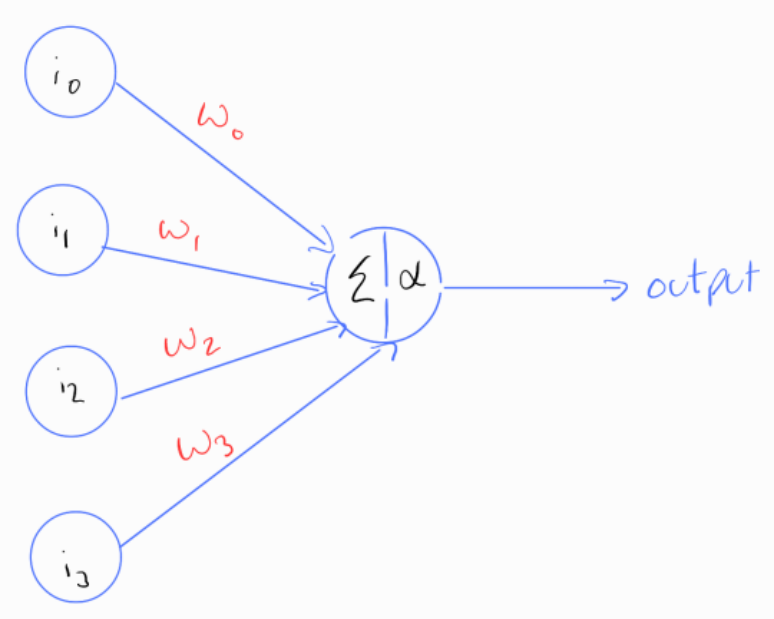
[Figure 1] Input Nodes (Genes)

Weights and Input Normalization:

- Each input node is connected to the output through four randomly generated weights - w_0, w_1, w_2, w_3 - with values ranging between -1 and 1, allowing for a comprehensive modulation of input signals.
- Input values are normalized between 0 and 1 to ensure the neural network processes consistent and scaled data, facilitating more stable and predictable learning outcomes.

Activation function and Output Calculation:

- The activation function $\alpha = 1/(1 + e^{-x})$, also known as the $sigmoid(x)$, is used to calculate the *value for the output node* located in a bird's brain. Here the variable x is the summation of the product of the input nodes and their respective weights
- The output node value is calculated as: $output = \alpha(\sum_{n=0}^{3} i_n * w_n)$ (Figure 2).
    - When *output* >= 0.73 the bird flaps



[Figure 2] Output Calculation

Training and Optimization Using the NEFT Algorithm: [Main Algorithm]

The training and optimization of our neural network utilize the NEFT genetic algorithm, a variant of genetic algorithms tailored for fixed-topology neural networks. This section details the operation of genetic algorithms, the specifics of NEFT, and the parameters used in our project.

Overview of Genetic Algorithms:

Genetic algorithms (GAs) are a subset of evolutionary algorithms inspired by natural selection and genetics. These algorithms mimic the process of natural evolution, enabling solutions to optimization and search problems to evolve. A typical genetic algorithm includes the following phases:

- **Initialization:** Generate an initial population of individuals randomly. Each individual, or genome, represents a potential solution to the problem, encoded as a set of parameters or genes (in our case, neural network weights).
- **Evaluation:** Each individual is assigned a fitness value which is used to compare and find superiority between individuals.
- **Selection:** Individuals are selected for reproduction based on their fitness, where higher fitness values indicate a high likelihood of being selected for future generations.
- **Crossover:** Selected individuals undergo crossover (mating) to produce offspring. This involves swapping parts of their genome to create variation.
- **Mutation:** Offspring are subjected to random mutations, which alter one or more values in their genome. This introduces new traits into the gene pool.
- **Replacement:** New individuals replace some or all of the old population, and the process repeats for several generations.

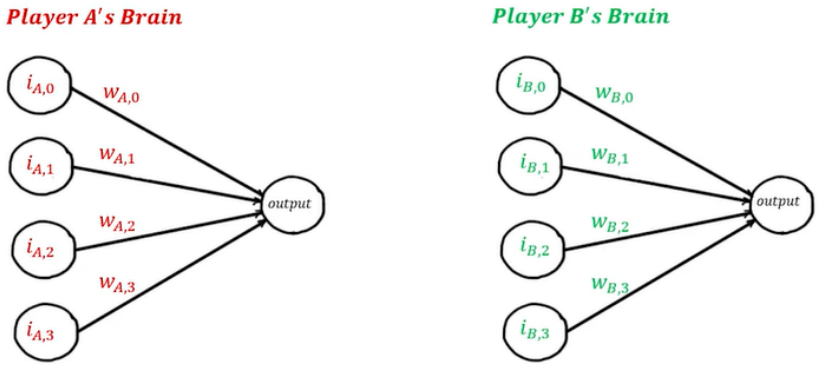
NeuroEvolution of Fixed Topologies (NEFT):

NEFT is a specialized form of a genetic algorithm that is specifically designed for optimizing neural networks with a fixed structure (topology). Unlike algorithms that also evolve the structure of the network (e.g., NEAT), NEFT focuses solely on optimizing the weights and biases within a predefined architecture. This simplification can lead to faster convergence and less computational overhead, making it ideal for applications like our Flappy Bird AI where the network architecture is predetermined and simple.

Specific Parameters and Calculations:

- **Population Size:** We initialized a population of 100 birds, ensuring a diverse genetic pool for robust evolutionary outcomes.
- **Fitness Evaluation:** A bird's fitness is equivalent to the lifespan of the bird. The fittest bird in a species would have the highest lifespan value, which is the number of pipes it successfully avoided. The more pipes they cross, the more superior genes they have.
- **Speciation Threshold:** The speciation threshold is an arbitrary value necessary for categorizing birds into species. Birds are categorized based on the sum of the differences between the weights of the species benchmark bird and the bird that we need to categorize. By error and trial, we found calculated the disparity threshold value of 1.2 maintains optimal diversity through niche solution retention and consistent crossover prevention between significantly different birds.

$$total\_weight\_difference = |w_{A,0} - w_{B,0}| + |w_{A,1} - w_{B,1}| + |w_{A,2} - w_{B,2}| + |w_{A,3} - w_{B,3}|$$



[Figure 3] Speciation Process

- **Selection Strategy:** For the next Generation, We select the best bird from each species, then add a random bird from each species(excluding the best bird) with high chances mutations and fill the remaining population with random birds from the best species.
- **Mutation Mechanics:** In our algorithm, mutations occur when we produce offspring from a species. The mutation rate is set at 80%, this high rate encourages a broad exploration of the solution space, critical for avoiding local minima.

Results and Discussion:

Overview of Results

The training process spanned multiple generations, each resulting in incremental improvements and adaptations. Key metrics tracked across generations included:

- **Generation Score:** The highest score achieved by any individual within the generation.
- **Number of Species:** The total number of distinct species identified in each generation, reflecting the diversity of solutions evolved.
- **Best Species Fitness Average:** The average fitness score of the best-performing species.
- **Best Bird Fitness:** The highest fitness score achieved by any individual (the best bird).

These metrics were tabulated and are presented in the accompanying screenshots. Each table captures the dynamics and evolution of the network's performance, providing a clear view of progress and the effectiveness of our evolutionary strategy.

Generation	Score	Number of Species	Best Species Fitness	Best Bird Fitness
0	0	35	328	504
1	0	37	601	617
2	2	41	648	967
3	2	40	679	998
4	13	40	1110	3198
5	5	40	409	3198
6	3	39	321	3198
7	2	39	193	3198
8	4	41	273	3198
9	6	19	213	3198
10	17	22	683	3998

[Figure 4] Results 1

Generation	Score	Number of Species	Best Species Fitness	Best Bird Fitness
0	0	31	324	497
1	0	33	381	592
2	0	35	278	598
3	0	36	361	598
4	0	36	333	601
5	0	36	134	601
6	19	36	486	4470
7	3	37	198	4470
8	2	37	216	4470
9	0	15	135	4470
10	4	21	240	4470
11	2	20	315	4470

[Figure 5] Results 2

Generation	Score	Number of Species	Best Species Fitness	Best Bird Fitness
0	0	34	376	491
1	0	38	505	598
2	0	37	461	598
3	0	38	480	598
4	2	40	732	1002
5	3	43	676	1195
6	12	43	918	2998
7	2	44	634	2998
8	2	47	598	2998
9	13	21	968	3198
10	18	23	761	4198
11	10	26	939	4198
12	2	27	642	4198
13	3	29	690	4198
14	2	33	652	4198
15	3	32	674	4198
16	1	31	645	4198
17	3	32	751	4198
18	9	28	756	4198
19	1	26	597	1207
20	6	24	585	1798
21	4	28	1131	1798
22	9	28	1639	2398
23	10	31	866	2598
24	21	32	790	4798
25	25	34	1008	5598
26	8	33	507	5598
27	9	33	487	5598
28	16	32	1936	5598
29	15	31	1016	5598

[Figure 6] Results 3

Observations and Trends

From the data collected, several key observations were made:

- **Consistency of Best Bird Fitness:** Across all generations, the fitness of the best bird rarely decreased. This indicates a consistent improvement or at least maintenance of peak performance, which suggests that our genetic algorithm effectively preserves and propagates advantageous traits. The best bird fitness decreased only when there were no improvements in the metrics for many generations.
- **Fluctuation in Best Species Average Fitness:** The fitness of the best species varied across nearly all generations. This fluctuation is indicative of the exploratory nature of the genetic algorithm, as it constantly searches for new and potentially more effective strategies.
- **Correlation Between Species Improvement and Score Increase:** Notably, increases in the game score from one generation to the next were observed only when there was an increase in the average fitness of the best species. This pattern highlights the importance of overall species improvement to achieving higher scores.

- **New Maxima Achievement:** Whenever the best species reached a new maximum in fitness, corresponding increases in both the score and the best bird fitness were observed. This correlation underscores the critical role that top-performing species play in pushing the boundaries of what AI can achieve.

Time Complexity

Perceptron Neural Network:

- The most important part of the model, i.e the calculation of the output value, happens in **O(n)**, where ***n*** is the number of nodes in the network. Here as the number of nodes doesn't change, we can assume that this is a constant time operation

NEFT Genetic Algorithm Functions:

- **Speciate:** Compares the current bird's brain to another bird for each bird in the population. Adds the other bird to the current bird's species if the speciation threshold is lesser than the threshold value of disparity. The time complexity is **O(b*n*s)**, where ***b*** is the population of birds, ***s*** is total no of species in the generation, and ***n*** is the total number of connections or the input nodes. This is because we check the similarity of each bird with every existing species.
- **Calculate Fitness:** Updates the fitness level for each bird, then finds the average fitness for every species. The time complexity is **O(b + (b+s)) = O(b+s)**, where ***b*** is the population of the birds and ***s*** is the number of species. This is because, the time taken to calculate the fitness(which is updated when a bird dies) for every bird is the total population of birds operations and time to taken to calculate average fitness of each species is total no of birds in each species + 1 operation time for each species (this equates to total population of birds + the number of species).
- **Sorting by Fitness:** Sort the birds in each species by fitness. Then, update the *benchmark fitness* value to the fitness value of the first bird. Then sort the species according to the *benchmark fitness* value of each species. The time complexity for this is **O(s*b*log b + s*log s)**, where ***s*** is the number of species, ***b*** is the number of birds in **each species**.
- **Children Generation:** The function adds the fittest bird of each species to the next generation's children, then adds random offspring of each species, and finally for the remaining capacity in the population, we add random offspring from the best species. The time complexity is **O(b)** , where ***b*** is the total population of the birds. This is because adding twice the total number of species times the constant operation of fetching + filling the extra with random best offsprings, is still going to result in the same population of birds in the next generation.

References

https://svitla.com/blog/modern-methods-of-neural-network-training#:~:text=The%20standard%20method%20for%20training,stochastic%20gradient%20descent%20(SGD).