Optimizing Flappy Bird Gameplay: A Genetic Algorithm and Neural Network Approach

Lakshit Unadkat
Electircal and Computer Engineering
University Of Waterloo
Waterloo, Canada
Student Id - 21112424

***Abstract*—The rapid advancements in artificial intelligence (AI) have led to significant strides in automating complex tasks, with games serving as an ideal testing ground for machine learning (ML) algorithms. This paper explores the application of Genetic Algorithms (GA) and Neural Networks (NN) in automating the gameplay of Flappy Bird, a deceptively simple yet challenging game. A hybrid approach combining GA for optimizing NN weights and NEAT (NeuroEvolution of Augmenting Topologies) for dynamic neural architecture evolution is used. The AI agent learns and adapts its decision-making strategies over multiple generations to navigate the game's dynamic obstacles.**

I. Introduction

The rapid advancement of artificial intelligence (AI) has opened up new possibilities for solving complex, dynamic problems. One of the exciting areas where AI has made significant progress is in the automation of games, providing a testing ground for machine learning (ML) algorithms. Game automation not only challenges AI agents to perform under real-time conditions but also offers valuable insights into the optimization of decision-making processes.

Flappy Bird, a simple yet highly engaging mobile game, presents a unique challenge for AI agents, requiring them to navigate through dynamic obstacles with minimal input. Traditional methods of game-playing AI often rely on pre-programmed strategies or simple rule-based systems, which are limited in their adaptability. In contrast, this work seeks to develop an intelligent agent that learns and evolves its gameplay over time using a combination of Genetic Algorithms (GA) and Neural Networks (NN).

Genetic Algorithms, inspired by Darwinian principles of natural selection, have been used effectively to optimize solutions to complex problems by mimicking evolutionary processes. When combined with Neural Networks, which can learn patterns and make predictions based on data, this hybrid approach provides a powerful mechanism for evolving intelligent agents. The aim of this paper is to explore how GA and NN can be used together to automate the Flappy Bird game, evolving the agent's decision-making capabilities and improving its gameplay over time.

II. Literature Review

[1] Genetic Algorithms (GA) have been successfully applied to optimize game-playing agents. Togelius et al. (2007) demonstrated the use of evolutionary algorithms in evolving controllers for racing games, showcasing the potential of GA in optimizing AI behavior in dynamic game environments. Their work highlighted how evolutionary techniques could fine-tune game-playing strategies. Similarly, [2] Stanley and Miikkulainen (2002) introduced the NeuroEvolution of Augmenting Topologies (NEAT) algorithm, which has been widely adopted in evolving both the structure and weights of neural networks. NEAT has proven to be particularly effective in complex game environments, such as those requiring adaptive behavior and strategy optimization.

[3] Neural Networks (NN) have proven to be powerful tools for decision-making in game environments. Mnih et al. (2015) demonstrated the capabilities of deep reinforcement learning in mastering Atari games, including those with mechanics similar to Flappy Bird. Their approach, which combined convolutional neural networks with Q-learning, achieved human-level performance in several games, providing a robust framework for building intelligent agents capable of handling complex game scenarios.

[4] In the specific domain of Flappy Bird automation, several researchers have explored different AI approaches. Chen (2015) applied Q-learning to train an agent to play Flappy Bird. Their results showed that after sufficient training, the agent could learn to navigate the game environment effectively, demonstrating the potential of reinforcement learning for game automation. Additionally, Ebner and Tiede (2009) applied genetic programming to evolve strategies for a similar 2D scrolling game, highlighting the potential of evolutionary computation for optimizing decision-making in games with simple but challenging mechanics.

The combination of GA and NN, as utilized in this project, builds upon these prior works by leveraging the optimization capabilities of genetic algorithms with the pattern recognition and decision-making abilities of neural networks. This hybrid approach allows for the evolution of both the network structure and its weights, which enhances adaptability and efficiency in game-playing agents. By combining these two powerful techniques, we aim to develop a more versatile agent capable of navigating the Flappy Bird environment more effectively than traditional methods.

III. Theoritical Foundation

A. Flappy Bird

Flappy Bird is a widely recognized game that presents a simple yet challenging task for players, navigating a bird through a series of pipes without colliding with them. The bird constantly falls due to gravity, and the player must control its flapping to stay afloat while avoiding obstacles. [4] The game's simplicity makes it an ideal candidate for automation.

In context of game automation, the challenge lies in developing an AI agent capable of effectively responding to environmental changes such as the movement of pipes and the bird's vertical position. The agent must make real-time decisions about when to flap or remain stationary to maximize its survival time and score. Given the limited inputs, the task requires an efficient decision-making process, which makes the game a suitable problem for applying AI techniques like Genetic Algorithms and Neural Networks.

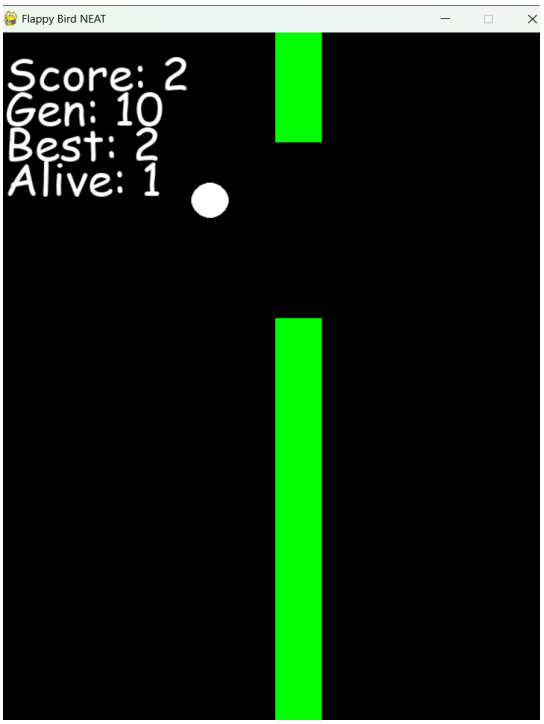


Fig. 1. Flappy Bird Game window

B. Neural Networks

A Neural Network is a computational model inspired by biological neural systems, designed to recognize patterns, make decisions, and learn from experience. [2] In the context of game automation, neural networks are used to enable an agent to make decisions based on environmental inputs.

In this project, a feedforward neural network is used to control the actions of the AI agent in the Flappy Bird game. The network takes input from three nodes representing the game environment and outputs a decision through a single node: whether the bird should flap or remain stationary.

To allow the network to effectively model complex relationships within the game environment, a non-linear activation function is crucial. In this project, the sigmoid function is used due to its ability to output values in the range of 0 to 1.

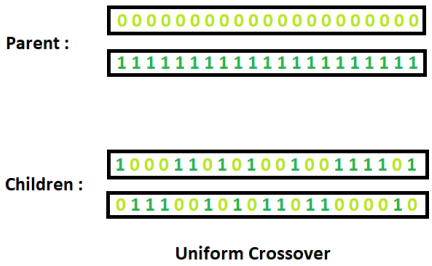
C. Genetic Algorithm

The Genetic Algorithm (GA) is a search heuristic inspired by the principles of natural selection and genetics. It is used to optimize the neural network's weights, enhancing the AI agent's ability to navigate the game. The GA works by evolving a population of candidate solutions through a series of operations, including selection, crossover, mutation, and evaluation based on a fitness function.

1) Selection: Selection is the process of choosing individuals from the population to reproduce and create offspring. The goal of selection is to ensure that the most promising solutions, based on their fitness, are passed on to the next generation. Several selection methods exist, each with its advantages and trade-offs:

1) **Tournament Selection**: A small group of individuals is selected randomly, and the best individual is chosen to reproduce. This method helps maintain diversity and is simple to implement.
2) **Roulette Wheel Selection**: Individuals are selected based on their fitness, with fitter individuals having a higher probability of being chosen.
3) **Rank Selection**: Individuals are ranked according to their fitness, and selection occurs based on their rank rather than raw fitness scores.
4) **Random Selection**: Individuals are selected randomly, without considering their fitness, which helps preserve diversity.

2) Crossover: Crossover is the genetic operator that combines the genetic material of two parent individuals to create offspring. The purpose of crossover is to explore new solutions by recombining successful traits from different parents.



3) Mutation: Mutation introduces random changes to an individual's genetic material, providing a mechanism to explore new solutions and maintain diversity in the population. The mutation process involves randomly adjusting one or more genes in the chromosome, such as changing the value of a weight by a small amount. In this implementation, the mutation rate is set to 0.5, meaning that there is a 50% chance of mutation occurring for a gene.

Mutation ensures that the algorithm does not become stuck in local optima by introducing new genetic material that could lead to better solutions. It also prevents the loss of diversity, which is crucial for the GA's ability to explore the solution space effectively.

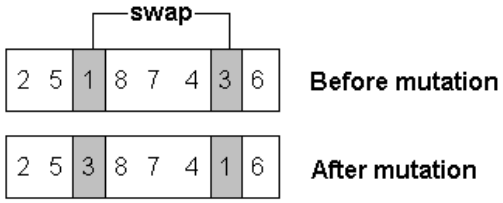


Fig. 2. Mutation

4) Fitness Function: The fitness function is a critical component of the genetic algorithm, used to evaluate the performance of individuals in the population. It assigns a fitness score based on how well an individual solves the given problem or meets predefined objectives. The fitness function guides the selection process, favoring individuals that perform better and are more likely to reproduce.

IV. EXPERIMENTATION

A. The Game Making

The first step in automating the Flappy Bird game is to create a functional game environment that mirrors the original gameplay. This involves developing the bird, obstacles, gravity mechanics, and collision detection using Python. In the game, the bird is represented as a white circle, and its movement is governed by physics principles such as gravity and the bird's flap action. The bird's position, velocity, and acceleration are updated each frame based on these physics. Obstacles in the game consist of moving pipes with a fixed gap that the bird must navigate through. The pipes move horizontally from right to left, and the gap between them remains constant. If the bird collides with a pipe or the ground, the game ends. The game also tracks the score and resets if the bird fails to avoid obstacles. This game environment is designed to interact with an AI agent that controls the bird. The agent receives game state information, such as the bird's position and the distance to the nearest pipe, which is fed into a Neural Network to determine the bird's actions. [6] The game loop continuously updates the state, checks for collisions, and provides feedback to the agent, which uses this information to improve its performance over multiple generations via a Genetic Algorithm.

The following flowchart outlines the game's operational logic, including the interaction between the game environment and the AI agent.

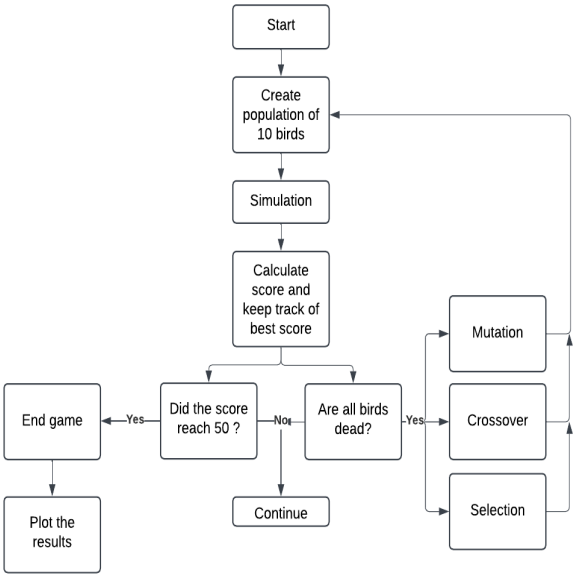


Fig. 3. Game Operation

B. Neural Netwrok

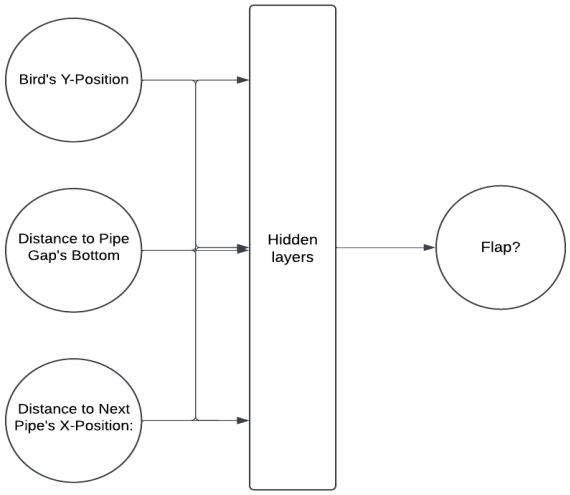


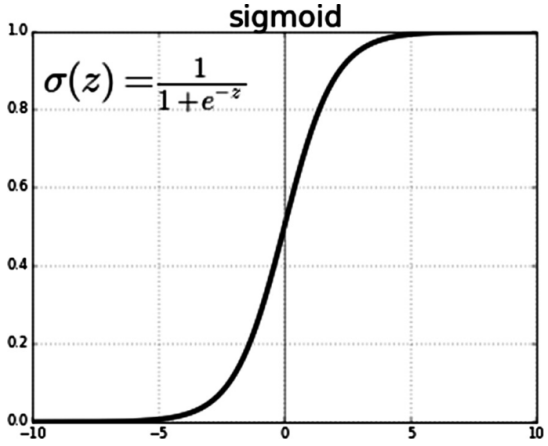
Fig. 4. Neural Network

1) Layers: The Neural Network in this project consists of an input layer, one or more hidden layers, and an output layer [7]

a) Input Layer: The network receives three inputs:

1) **Bird's Y-Position**: The bird's vertical position relative to the window height.
2) **Distance to Pipe Gap's Bottom**: The vertical gap between the bird and the bottom of the next pipe.
3) **Distance to Next Pipe's X-Position**: The horizontal distance to the next pipe.

b) Output Layer: The output layer has one node activated by a sigmoid function, producing a value between 0 and 1. [7]If the output exceeds 0.5, the bird decides to jump (bird.jump()); otherwise, it stays in place. This output directly controls the bird's decision-making process during the game.



c) Hidden Layer: The number of hidden layers and nodes in the network are dynamically adjusted by the NEAT (NeuroEvolution of Augmenting Topologies) algorithm. NEAT evolves the network's architecture by modifying node counts, connections, and activation weights during the learning process, allowing the network to optimize its decision-making capabilities over generations.

2) Neat: In this project, NEAT (NeuroEvolution of Augmenting Topologies) was applied to evolve the neural network that controls the bird in the game.The NEAT algorithm plays a crucial role in evolving both the structure and weights of the neural network throughout the learning process. Unlike traditional training methods, NEAT dynamically adjusts the network's topology, enabling it to grow in complexity as it learns. [2] Initially, NEAT starts with simple networks and gradually introduces new nodes and connections as generations evolve. This enables the network to tackle more complex challenges and optimize its decision-making.

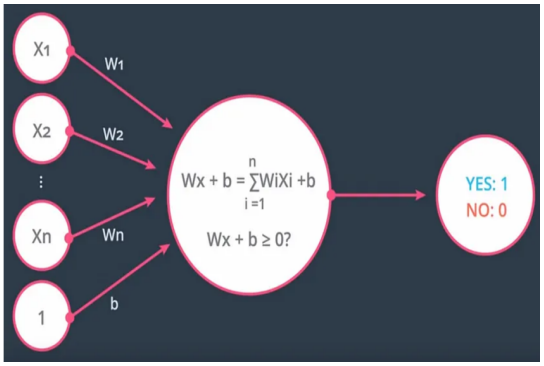


Fig. 5. How a Perceptron Works

C. Genetic Algorithm

1) Selection: In Tournament Selection, individuals are selected by first randomly choosing a subset of the population, referred to as a species, and then selecting the individual with the highest fitness score within that species. The probability of an individual being selected in a tournament is proportional to its fitness relative to the others in the species. It can be mathematically expressed as:

$$P_{\text{selected}} = \frac{f(x)}{\sum_{i=1}^{T} f(x_i)}$$

Where:

- $f(x)$ is the fitness of the individual being considered.
- $f(x_i)$ is the fitness of individual i in the tournament.
- T is the tournament size.

2) Crossover: In Two-Point Crossover, two parents from the population are selected, and a new offspring is created by swapping segments of their genetic material between two randomly chosen crossover points. The process begins by selecting two parent individuals, which are then divided at two points along their genome (which represents the neural network parameters, such as weights and biases). The sections of the parents' genomes between these points are swapped to create two new offspring, each inheriting parts of the parents' genetic material.

The two-point crossover method helps in maintaining a balance between exploration and exploitation in the search space. By combining parts of different parents' genomes, it introduces diversity into the population while potentially allowing beneficial traits to be passed on. This helps the population evolve more efficiently toward optimal solutions.

3) Fitness Function: The fitness function evaluates how well a bird performs in the game and guides the evolution of its neural network. It is based on the following criteria:

- **Survival Time**: The bird earns 0.1 fitness points for each frame it stays alive.
- **Pipes Passed (Score)**: The bird gains fitness equal to the number of pipes it successfully passes.
- **Penalty for Collision**: A penalty of -2 points is applied for collisions with pipes or going out of bounds.

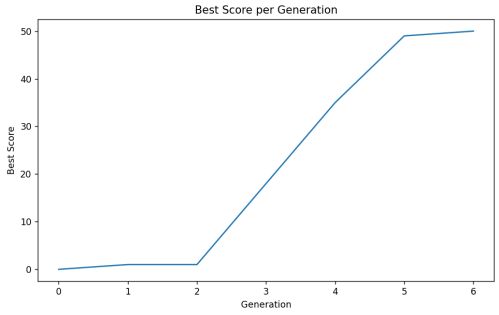
The fitness is calculated using the formula:

$$\begin{aligned}\text{Fitness} = {} & (\text{Survival Time} \times 0.1) \\ & + \text{Pipes Passed} \\ & - \text{Penalty for Collision}\end{aligned}$$

This encourages birds to focus on survival and efficient navigation through pipes.

V. Results

The experiment was conducted using a population of 50 individuals across multiple generations. The game continued until an agent reached a score of 50, which determined the end of the experiment.



Above graph describes the best scores achieved across generations. The curve illustrates the improvement in performance as the agents evolved and adapted to the environment. Initially, there were slow advancements, but by later generations, the agents could consistently achieve higher scores

Game Statistics

Generation	Score	Best Score
1	0	0
2	1	1
3	1	1
4	18	18
5	35	35
6	49	49
7	50	50

The table further highlights key statistics, including the generation number, the best score achieved in each generation, and the Score of the population. These results validate the success of the Genetic Algorithm in optimizing the behavior of agents within the Flappy Bird environment.

VI. Conclusion

This project demonstrates how a combination of Genetic Algorithms (GA) and Neural Networks (NN) can effectively automate gameplay in Flappy Bird, a game known for its deceptively simple mechanics and challenging decision-making requirements. By using GA to optimize weights and NEAT (NeuroEvolution of Augmenting Topologies) to adapt network architectures dynamically, the AI agent showed consistent improvements over multiple generations. [7] The agent evolved strategies that enhanced both its survival time and its ability to navigate obstacles, showcasing the strength of evolutionary learning in dynamic and unpredictable environments.

This work underscores the practical potential of combining optimization and learning-based methods for tasks requiring real-time adaptability. The results not only highlight the effectiveness of this hybrid approach but also point to its applicability in areas beyond gaming, such as robotics, autonomous systems, and real-time optimization problems.

VII. Future Work

The proposed methodology has demonstrated promising results; however, there are several avenues for further optimization. One potential improvement involves enhancing the genetic algorithm's adaptability through context-sensitive mutations. Specifically, weights influencing critical decisions could undergo larger adjustments based on the nature of failure events. For instance, collisions with the top pipe could trigger stronger mutations in the weights associated with vertical distance calculations, enabling more effective learning. Additionally, scaling the neural network by increasing the number of hidden layers or adjusting layer configurations could further enhance the agent's capacity to model complex relationships in the game environment. Breeding strategies could also be refined by exploring alternative mechanisms.

References

[1] J. Togelius, R. Nardi, and S. M. Lucas, "Towards automatic personalised content creation for racing games," in Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG), 2007.

[2] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," Evolutionary Computation, vol. 10, no. 2, pp. 99-127, 2002.

[3] V. Mnih et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529-533, 2015.

[4] K. Chen, "Deep reinforcement learning for flappy bird," CS229 class project report, Stanford University, 2015.

[5] M. Ebner and T. Tiede, "Evolving behavior for a 2D scrolling game," in Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, 2009, pp. 2615-2622..

[6] Mishra, Yash Kumawat, Vijay Kamalanathan, Selvakumar. (2019). Performance Analysis of Flappy Bird Playing Agent Using Neural Network and Genetic Algorithm.

[7] Lai, T. D. Genetic Algorithm and Application in training Multilayer Perceptron Model.