Optimization of Flappy Bird using NEAT

by SAYAN BANERJEE

Submission date: 11-Apr-2024 07:33AM (UTC+0530)
Submission ID: 2344216969
File name: Optimization_of_Flappy_Bird_using_NEAT_Report.pdf (2.28M)
Word count: 6454
Character count: 37401

PROJECT REPORT

on

"Optimization of Flappy Bird using NEAT"

Submitted to

KIIT Deemed to be University

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN

Computer Science and Engineering

BY

Sayan Banerjee	21051087
Shubham Patel	21051094
Anjali Raj	21051032
Anil Padhan	21051031

UNDER THE GUIDANCE OF

Mr. Amiya Kumar Dash

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

April 2024

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

"Optimization of Flappy Bird using NEAT"

submitted by

Sayan Banerjee	21051087
Shubham Patel	21051094
Anjali Raj	21051032
Anil Padhan	21051031

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during the year 2023-2024, under my guidance.

Date: 10/04/2024

Mr. Amiya Kumar Dash
Project Guide

Acknowledgements

We are profoundly grateful to **Mr. Amiya Kumar Dash, Assistant Professor** in School of Computer Engineering for his expert guidance and continuous encouragement throughout to see that this project meets its target since its commencement to its completion.

Sayan Banerjee

Shubham Patel

Anjali Raj

Anil Padhan

ABSTRACT

Neuroevolution—the process of evolving artificial neural networks using genetic algorithms has proven very successful in reinforcement learning tasks, especially when the states are hidden. NEAT leverages evolutionary principles to automatically develop the structure and connection weights of neural networks. Unlike traditional ANNs where the architecture is predefined, NEAT allows the network to grow and adapt its complexity during the training process.

Game learning is currently one of the most popular topics being researched in the field of artificial intelligence and is an efficient way to measure the progress of AI. AlphaGo, Agent57 and DecodeChess are some AI agents which have even defeated human players. These AI agents need to go through a training phase. In this project, we propose a minimal training strategy to develop an Artificial Intelligence agent using the NeuroEvolution of Augmenting Topologies (NEAT) algorithm to play the Flappy Bird Game. The agent was not provided any prior information about the surroundings and in order to play the well-known "Flappy Bird" game as best it can, our agent learns how to carefully avoid all of the obstacles and flap its way through them.

Keywords: *Artificial Intelligence; Artificial Neural network; Genetic algorithm; Flappy bird; Neuro-Evolution*

Contents

List of Figures

1. Introduction

An intelligent agent is anything that can detect its surroundings, act independently to accomplish goals, and learn from experience or use knowledge to execute tasks better. The method is a general one, capable of being applied to an extremely wide range of problems. One such approach is NEAT algorithms which are numerical optimization algorithms inspired by both natural selection and natural genetics.

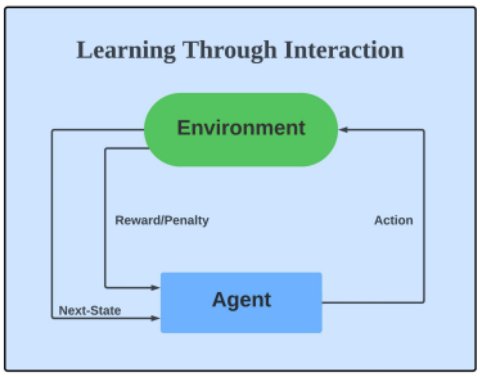

Figure1: Self-Learning Agent Process

This project explores the application of NEAT to evolve a neural network capable of controlling a virtual bird in the classic Flappy Bird game. Flappy Bird presents a challenging environment due to its fast-paced nature and the need for precise decision-making to avoid obstacles.

Existing AI approaches to Flappy Bird often rely on pre-defined rules or hand-crafted algorithms. While these methods can achieve success, they lack the ability to learn and adapt to unseen scenarios. NEAT, on the other hand, offers an evolutionary approach where a population of neural networks compete and evolve over generations. This allows the system to discover effective strategies for navigating the Flappy Bird environment without explicit programming.

This report details the development of a NEAT-based solution for Flappy Bird. It outlines the limitations of current approaches and explains the rationale behind using NEAT. The subsequent sections will delve into the specific implementation details, including the game environment, the NEAT configuration, and the evaluation process. Finally, the report will present the results obtained and discuss the potential for further improvements.

Why Flappy Bird?

1. **Simple Ruleset and Environment:** Flappy Bird has a straightforward objective (avoiding pipes) and a well-defined environment (scrolling background, fixed-size pipes). This allows NEAT to focus on evolving the control strategy (neural network) without dealing with complex environmental factors.
2. **Clear Fitness Function:** The fitness function in Flappy Bird is easily defined as the duration a bird survives (higher score signifies better performance). This provides a clear signal to NEAT for evaluating the performance of different network configurations.
3. **Continuous Action Space:** Flappy Bird requires a continuous action (tapping the screen) rather than discrete choices (like left/right movement). This aligns well with the way neural networks typically output continuous values.
4. **Visually Engaging and Accessible:** The game's visual nature and simple mechanics make it easy to understand and observe the bird's behavior controlled by the evolved network. This allows for clear visualization of how the network's decisions translate to actions in the game.
5. **Representative Challenge:** Despite its simplicity, Flappy Bird still presents a challenging task for a neural network, requiring good timing, coordination, and obstacle avoidance. This makes it a good example of how NEAT can evolve networks to handle complex control tasks.

2. Literature review

The model proposed by Kevin Chen employs Deep Q-Learning with experience replay to train an agent for playing Flappy Bird. It formulates the game environment as an MDP, utilizing a neural network to approximate the Q-function. Techniques like target networks and exploration strategies are employed for stability and learning efficiency. The training pipeline involves iteratively updating the network with batches of experiences sampled from a replay memory. Through this approach, the agent learns to navigate the game environment autonomously, demonstrating the effectiveness of reinforcement learning in gaming scenarios.

In the paper by Iveta et al. utilizes a dueling deep Q-network (DQN) approach and a DeepMind Reverb database server for effective learning and model sharing. It employs ray casting (LIDAR) for measurements, a motion transformer for feature extraction, and introduces private zones to enhance agent safety. Episodic memory stores past measurements, while penalties for approaching obstacles within the private zone encourage safe navigation

The method proposed by Tai and Leon involves baseline algorithms like random policy and an infinite score oracle, alongside modifications to SARSA and Q-Learning for Flappy Bird training. Techniques include discretization, epsilon-greedy exploration, and function approximation using linear regression, feed-forward neural networks (FFNN), and convolutional neural networks (CNN). Preprocessing involves image resizing and grayscale conversion. Forward/backward updates are explored for efficiency. Overall, the method aims to train an effective Flappy Bird-playing agent by combining reinforcement learning techniques and tailored modifications

Atkan et al. explore a Type-2 Fuzzified Flappy Bird Control System, aiming to autonomously navigate the bird through the game environment. The methodology involves utilizing a Matlab replica of Flappy Bird, defining game parameters, and designing a control system comprising a reference generator and a SIT2-FLC (Single Input Type-2 Fuzzy Logic Controller). The SIT2-FLC structure is tuned using Control Curve generation, resulting in Smooth and Aggressive CCIT2 designs. Performance evaluation includes analyses of control performance and game performance, demonstrating the effectiveness of the proposed control system in navigating the Flappy Bird environment autonomously.

To control Flappy Bird autonomously three methods were proposed by Mathew et al. : a manual controller, an optimization-based approach, and model predictive control (MPC). The manual

controller determined the bird's action based on its altitude relative to a set-point above the upcoming pipe. The optimization-based method minimized flapping over a specified time horizon using a linear optimization program, adjusting parameters heuristically. MPC formulated an optimization problem over a prediction horizon, implementing part of the control policy and repeating until a termination criterion was met. These methods aimed to navigate the bird effectively through the game environment.

3. Requirement Specifications

3.1. Problem Statement

The aim of this project is to implement an artificial intelligence (AI) agent capable of learning to play the popular game Flappy Bird using the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. Flappy Bird is a simple yet challenging side-scrolling game where the player controls a bird by tapping the screen to make it flap its wings and navigate through a series of obstacles, aiming to achieve the highest possible score.
Traditional approaches to developing AI for Flappy Bird typically involve handcrafted rules or heuristics and defined along the x-axis, which may not generalize well to various game scenarios and can be time-consuming to design. By contrast, NEAT offers a powerful solution by evolving neural networks that control the bird's actions, allowing for adaptive and autonomous learning. Our project optimizes the game scenario in y-axis which outputs a much better result along with scores which are the highest ever recorded by any human user on this game.

The following are the key components of this project:

1. Game Environment Setup: Develop a Flappy Bird game environment where the AI agent can interact with the game world. This includes rendering graphics, managing game physics, handling user inputs, and defining game mechanics such as obstacle generation and scoring.
2. NEAT Integration: Implement the NEAT algorithm, a form of evolutionary computation, to evolve neural networks that control the bird's actions. NEAT starts with a population of randomly generated neural networks and evolves them over successive generations through processes such as selection, mutation, and crossover, aiming to improve performance on the task.
3. Fitness Function Definition: Define a fitness function that evaluates the performance of each neural network based on its ability to play the game. The fitness function should consider metrics such as the distance traveled by the bird, the number of obstacles passed, and the survival value.
4. Training and Evaluation: Train the NEAT algorithm by repeatedly simulating Flappy Bird gameplay sessions for different generations. Evaluate the performance of each generation of networks and track improvements over time.
5. Performance Analysis: Analyze the performance of the trained AI agent, comparing its performance to human players and assessing its ability to adapt to different game scenarios and challenges.

The ultimate goal of this project is to demonstrate the effectiveness of NEAT in training an AI agent to master the Flappy Bird game through autonomous learning, showcasing its potential for solving complex tasks in a variety of domains.

3.2. Software Requirements Specification

3.2.1. Functional Requirements

- Gameplay Mechanics
  - The primary objective is to prolong the bird's survival by guiding it through gaps between pairs of pipes without collision.
  - The game operates within a loop, with each iteration representing a frame. During each frame, game elements move, the bird jumps as required, and scores are tallied.
- User Interface
  - The game offers a test environment within a virtual space, aiding in the optimization of learning agents. This feature allows agents to learn the behavior of non-deterministic phenomena.
- Obstacles
  - Pipes are randomly generated with varying heights and gaps for the bird to navigate through.
  - Pipes move horizontally from right to left, emulating the bird's forward flight.
  - Collision detection is integrated to detect any instances where the bird collides with pipes or the ground.

3.2.2. Non-Functional Requirements

- Performance
  - Fitness scores are computed for each population to evaluate their quality.
  - Genetic Algorithm (GA) operators are utilized to determine the next generation.
  - The top fitness scores from the current population are selected.. Random mutations are applied to each offspring to introduce variability. This iterative process continues until the bird masters the avoidance of all pipes.
- Components
  - Bird
  - Base
  - Pipe
  - Background

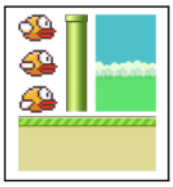

Figure2: Game Components

- Customizability
    - The code structure is modular and highly adaptable, enabling developers to adjust parameters such as game speed, obstacle generation rate, and neural network architecture to suit their preferences and requirements.

3.3. Data Flow Diagram

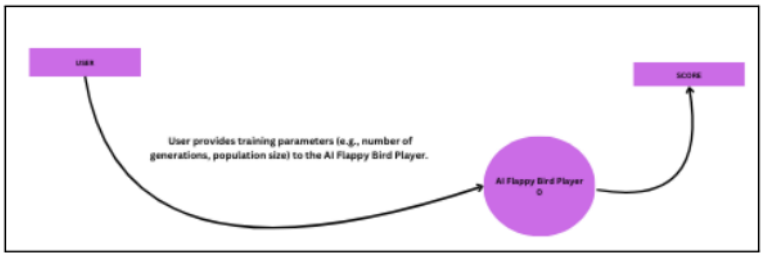

Figure3: Level 0 DFD: Context Diagram

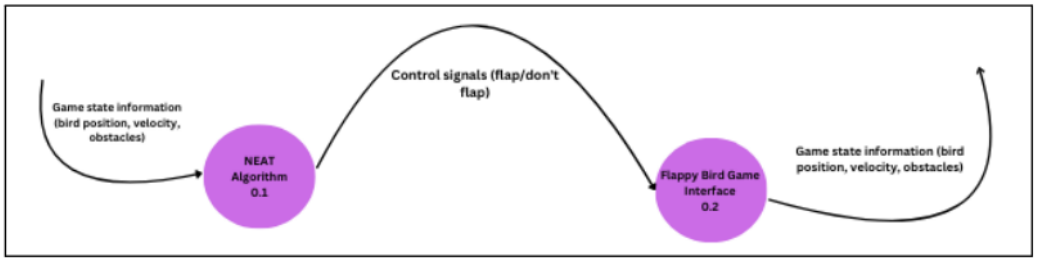

Figure4: Level 1 DFD

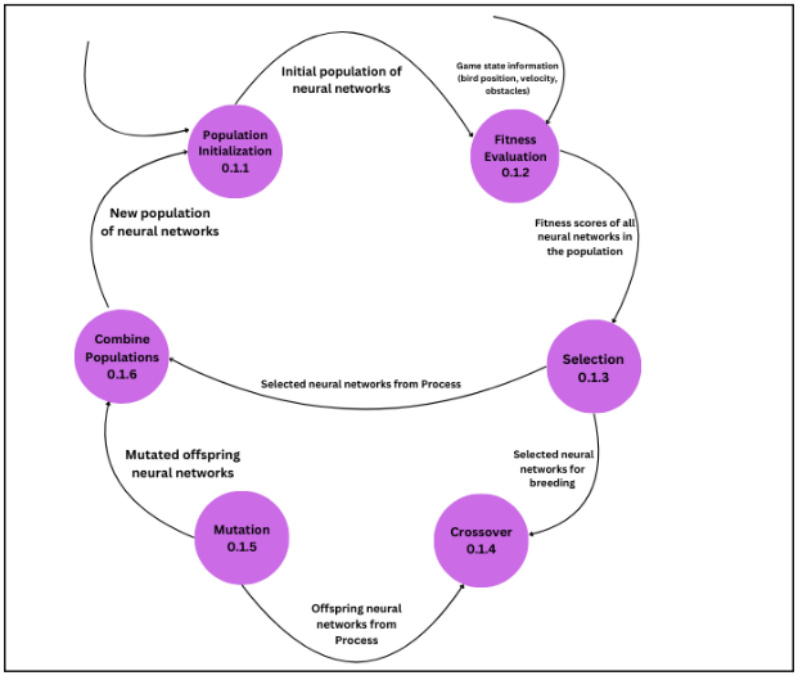

Figure5: Level 2 DFD

3.4. Unified Modified Language Diagram

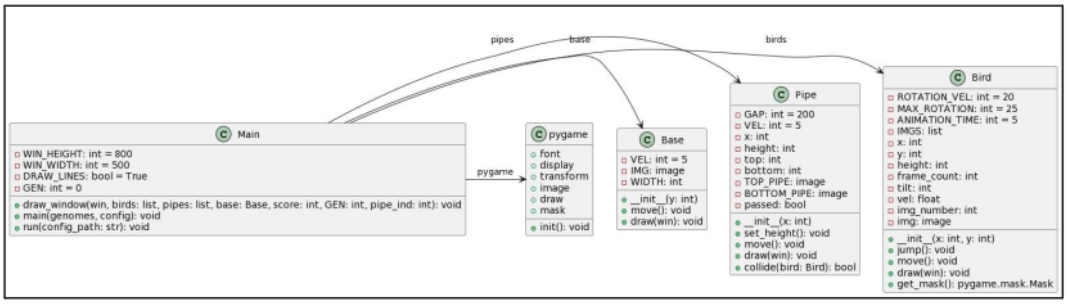

Figure6: UML diagram

4. Methodology

4.1. Neat Architecture

- Populations and Genomes:
  - NEAT starts with a population of individuals, each representing a candidate neural network.
  - Each individual's structure and weights are encoded in a "genome." This genome doesn't directly represent the network architecture but serves as a blueprint for generating it.
- Fitness Function:
  - Each individual is evaluated based on a fitness function specific to the problem. In the Flappy Bird game, the fitness function might consider how long a bird controlled by the network survives.
- Selection and Reproduction:
  - After evaluation, NEAT selects individuals with higher fitness for reproduction. Techniques like crossover and mutation are used to create new offspring genomes.
  - **Crossover:** Combines genetic material from two parent genomes to create a child. This allows the offspring to inherit beneficial traits from both parents.

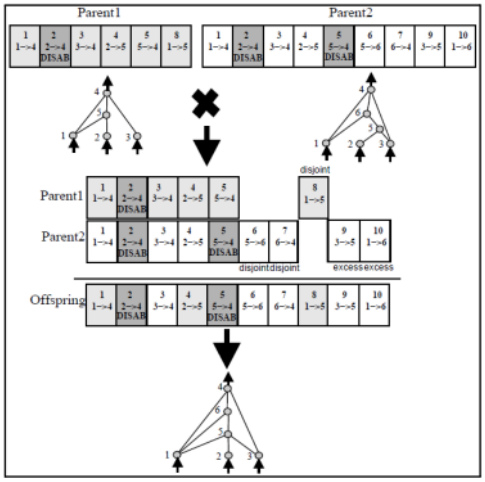

Figure7: This figure shows how different network designs (parents) can be combined to create new ones (offspring). Even though the parents look distinct, numbers above the genes (innovation numbers) reveal which genes correspond to each other. This allows for creating new offspring that inherit traits from both parents, along with entirely new features.

Mutation: Introduces random changes to the offspring's genome. This helps explore new network configurations and prevents getting stuck in local optima.

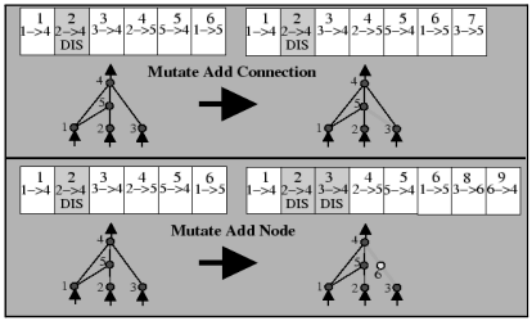

Figure8: The two types of structural mutation in NEAT. Both types, adding a connection and adding node, are illustrated with the genes above their phenotypes. The top number in each genome is the innovation number of that gene. The bottom two numbers denote the two nodes connected by that gene. The weight of the connection, also encoded in the gene, is not shown. The symbol DIS means that the gene is disabled, and therefore not expressed in the network. The figure shows how connection genes are appended to the genome when a new connection and new node is added to the network. Assuming the depicted mutations occurred one after the other, the genes would be assigned increasing innovation numbers as the figure illustrates, thereby allowing NEAT to keep an implicit history of the origin of every gene in the population.

Innovation and Speciation:

- NEAT's key strength lies in its ability to handle evolving network topologies (structures). During mutation, new connections or nodes can be added to the network.
- NEAT keeps track of these "innovations" to ensure that similar structural changes in different genomes are treated as equivalent during crossover. This promotes the exploration of diverse network architectures.
- Optionally, NEAT can also use speciation. This divides the population into sub-populations (species) based on their network similarities. Evolution can then occur within each species, allowing for exploration of niches within the solution space.

- Evolution Loop:
  - The process of selection, reproduction, and innovation is repeated for multiple generations.
  - Over time, the population should evolve towards networks that perform well on the fitness function. In the Flappy Bird game, this means evolving networks that can control the bird to avoid obstacles and survive for longer.

4.2. Proposed Method

1. Game Environment Setup:
   a. Component Development: The game environment comprises bird, base, pipes, and background elements to create a visually engaging and challenging setting for the neural network to navigate.

2. NEAT Configuration and Neural Network Design:
   a. Hyperparameter Loading: A comprehensive NEAT config file is loaded, fine-tuning the training parameters for optimal neural network evolution.
   b. Neural Network Architecture: The neural network architecture is intricately designed with 4 inputs (bird's y position, top pipe position, bottom pipe position, and memory), a hidden layer with 5 nodes, and 1 output node for decision-making (jump or not). The activation function employed is the hyperbolic tangent (tanh), enhancing the network's ability to learn complex patterns

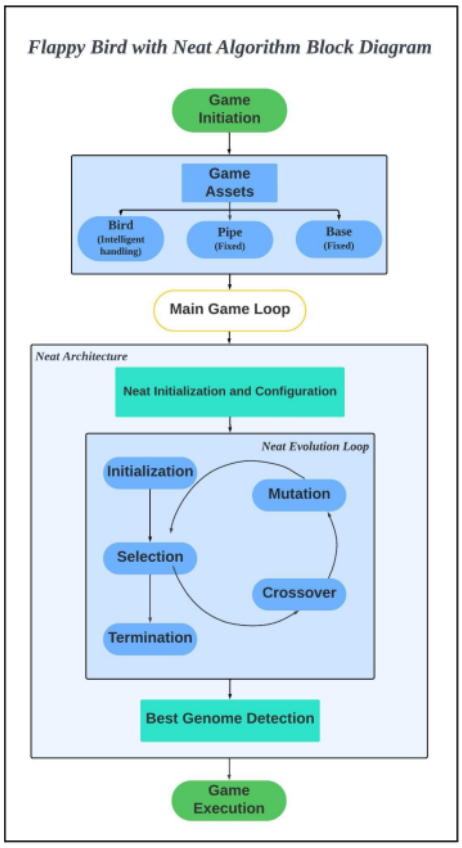

Figure9:Block Diagram

3. Performance Evaluation:

a. Fitness parameters: A reward based system is implemented to assess the bird's performance. Successful passage through a pipe yields a substantial reward of +5, while collisions incur a penalty of -1. Continuous movement without crashing is incentivized with a reward of +0.1, promoting strategic decision-making.

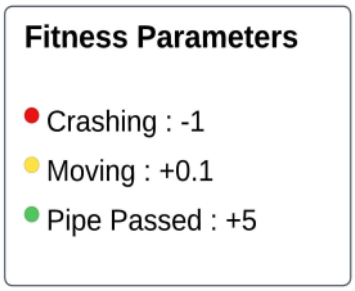

Figure10: Fitness Parameters

4. Collision Detection and Population Management:

a. Collision Mechanism: Advanced collision detection algorithms utilizing masks for precise interaction detection between the bird and pipes are implemented. Upon collision detection, i.e. if the masks of the bird and the pipes get overlapped, the affected bird is removed from the population or if the bird moves out of the frame it is removed dynamically adjusting the alive population count.

b. Population Evolution: Evolutionary strategies are employed as generations progress. The top 2 genomes of each generation undergo crossover to seed the next generation, ensuring the propagation of successful traits. To introduce diversity, mutation is applied to the remaining population, fostering exploration and innovation.

5. Decision-Making and Adaptive Learning:

a. Distance Calculation:

The optimal path for the bird to navigate through gaps between pipes is determined by considering the factors presented in the diagram and making

informed decisions about when to make the bird flap its wings and change its trajectory.

The bird's vertical position (Bird Y position) and the vertical positions of the top and bottom pipes (Top Pipe Y position and Bottom Pipe Y position) provide information about the spatial relationship between the bird and the pipes. By comparing the bird's position to the positions of the pipes, the model can determine whether the bird is approaching a gap or a pipe and adjust its trajectory accordingly.

The horizontal distance between the bird and the nearest pipe (Distance from pipe) is another crucial factor in determining the optimal path. By monitoring this distance, the model can anticipate when the bird needs to change its trajectory to navigate through the gap between the pipes.

The bird's velocity (Bird Velocity) is also an important consideration for finding the optimal path. By understanding the bird's speed, the model can predict how quickly the bird will approach the pipes and adjust its trajectory in advance to avoid collisions. By considering these factors in combination, the model can make informed decisions about when to make the bird flap its wings and change its trajectory to navigate through gaps between pipes successfully. The optimal path is determined by finding the right balance between the bird's position, velocity, and the positions of the pipes, allowing the bird to avoid collisions and maximize its score.

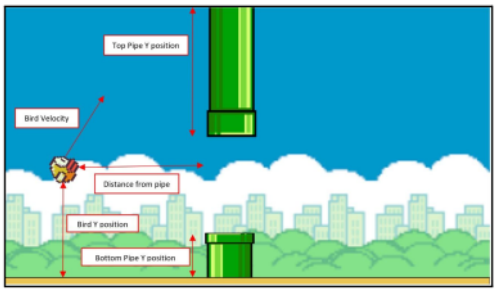

Figure11: Game Calculations

Birds Motion: The bird's vertical motion can be described using equations of motion under gravity. Assuming no air resistance, we can use the following equation:

y is the current vertical position of the bird.
y_0 is the initial vertical position of the bird.

$$y=y_0+v_0t+\frac{1}{2}gt^2$$

v_0 is the initial vertical velocity of the bird.
t is the time elapsed.
g is the acceleration due to gravity.

Pipe Motion: The pipes move horizontally across the screen with a constant velocity. Therefore, the position of the pipes can be described as:

x is the current horizontal position of the pipe.
x_0 is the initial horizontal position of the pipe.
v is the constant velocity of the pipe.
t is the time elapsed.

$$x = x_0 + vt$$

Fitness: To compute fitness of the agent we compute three factors:

- **Distance covered**: A counter that increments with each interaction of the agent with the environment.
- **Score**: The count of pipe pairs already surpassed
- **Stopping Factor (ΔSF)**: The metric derived when the agent encounters failure within the scenario. It is determined by the disparity between the agent's y-coordinate and the y-coordinate of the midpoint of the gap between consecutive pipes.

$$\Delta SF = SF_{bird} - SF_{passage}$$

The three individual fitness factors have been consolidated into a unified expression termed as fitness function (FF):

$$FF = \alpha \times DC_s + \beta \times Score_s - \gamma \times \Delta SF_s$$

This equation illustrates that the overall scenario fitness of an agent is determined through a linear combination of the individual SFCs, with the inclusion of three constant weights, namely α, β, and γ. These weights govern the significance of each factor within the fitness assessment. Consequently, the agent's objective is to maximize its travel distance, achieve the highest score, and maintain proximity to the passages. The subscript 's' denotes the standardized version of the fitness factor:

DC_s = (DC/DC_{max}) represents the Distance Covered, calculated as the ratio of the Distance covered (DC) to the maximum Distance covered (DC_{max}) achievable when transposing the maximum number of pairs of pipes in a scenario.

$Score_s$ = ($Score/Score_{max}$) normalized score attained by the agent, calculated by dividing the current score (Score) by the maximum achievable score ($Score_{max}$) within a scenario.

ΔSF_s = ($\Delta SF/H_w$) the normalized SF factor (SF), which is determined by dividing the Score (ΔSF) by the height of the game window, denoted as H_w. Here, H_w represents the maximum y-coordinate value within the game window.

The above fitness functions (FF) from the game are combined for each scenario to calculate the agent fitness function FF_{Agent} which calculates how fit an agent (bird in our case) is:

$$FF_{Agent}=\frac{\sum_{i=1}^{T_w}(w_i \times FF_i)}{\sum_{i=1}^{T_w} w_i}$$

$\mathbf{w}_i$ =Weight in each scenario
$\mathbf{T}_w$ =Total number of scenarios

Calculating the crossovers:

[1,2,3]
X[3,2,1]
Crossover[1,2,1] [3,2,3]

b. Adaptive Decision-Making: The neural network's decision to jump or not is intelligently determined based on the XOR parameter, enabling the network to adapt its actions dynamically to the evolving game environment.

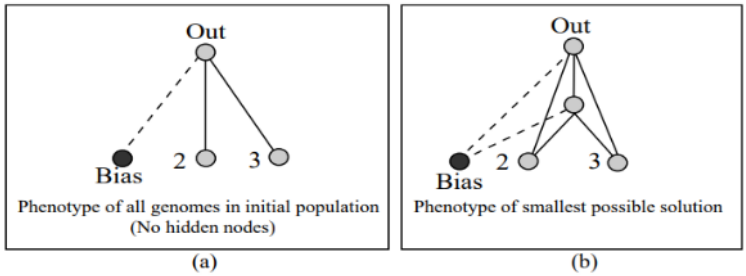

Figure12: Initial phenotype and optimal XOR. Figure (a) shows the phenotype given to the entire initial population. Notice that there are no hidden nodes. In NEAT, a bias is a node that can connect to any node other than inputs. Figure (b) shows an optimal solution with only 1 hidden node. (A network without hidden nodes cannot compute XOR.) The bias connections are not always needed depending on the solution; All other connections are necessary.

The XOR operation is a fundamental logic gate that returns true if and only if precisely one of its inputs is true. This represents a nonlinear problem that challenges the network's ability to learn complex relationships between inputs and outputs. In our Flappy Bird training scenario, we have integrated the XOR operation to create a more intricate and challenging environment for the AI agent. By doing so, we aim to enhance the agent's learning capabilities and problem-solving skills, enabling it to adapt to the game's dynamic and unpredictable nature. By combining the NEAT algorithm's powerful neuroevolution capabilities with the XOR operation's complexity, we have created a unique and engaging training environment for the AI agent. This approach not only showcases the adaptability of NEAT but also highlights the potential of XOR in training AI agents for complex tasks.

- Toggling: XOR can be used for toggling between two values. The combined value x ^ y 'remembers' both states, allowing easy switching between them. Here x and y represents to move upwards or downwards
- Reduced Space Complexity: XOR can be used to save memory allowing for efficient storage and retrieval of information.

Parameter Representation

1. NEAT Configuration:
    a. Fitness Criteria: Maximizing fitness values.
    b. Fitness Threshold: The fitness level to achieve before stopping.
    c. Population Size: Number of individuals in each generation.
    d. Reset on Extinction: False - Population is not reset if gone extinct.

2. Default Genome Settings:
    a. Activation Options: Using the tanh activation function
    b. Aggregation Options: Summing node inputs.
    c. Bias Options: Bias values specified within mutation range.

3. Genome Compatibility Options:
    a. Disjoint Coefficient: Coefficient for disjoint genes.
    b. Weight Coefficient: Coefficient for weight of each gene.

4. Connection Rates:
    a. Add/Delete Probabilities: Defines the connection rates

5. Connection Enable Options:
    a. Default Enabled: Initial connection status.
    b. Mutation Rate: Rate of connection enables mutation.

6. Network Parameters:
    a. Hidden/Input/Output Nodes: 5 hidden, 4 input, and 1 output nodes.

7. Default Species Set:
    a. compatibility Threshold: Threshold for species compatibility.

8. Default Stagnation Settings:
    a. Species Fitness Function: Fitness function for species.
    b. Max Stagnation: Maximum generations without improvement.
    c. Species Elitism: Number of elite species preserved.

9. Default Reproduction Settings:
    a. Elitism: Number of elite genomes preserved.
    b. Survival Threshold: Threshold for survival selection.

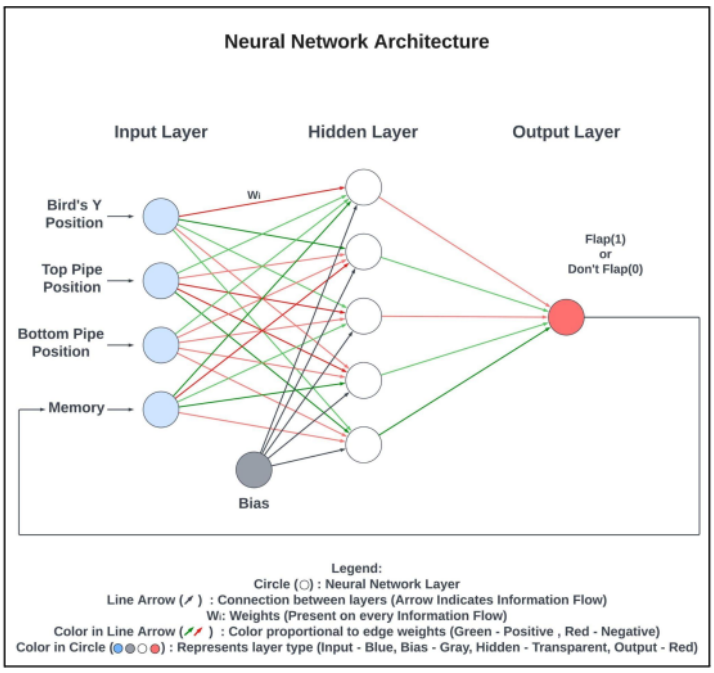

Figure13: NN Architecture

The Source Code of our project is available at Flappy-Bird-With-NEAT-Algorithm.

4.3. Environment

4.3.1. Training Environment

In the training environment for playing Flappy Bird with NEAT, the process begins with an initial population of 100 birds, each represented by a genome. These birds are tasked with navigating through a series of obstacles in the form of randomly generated pipes that appear at different positions over frames. The neural network guiding their decisions is structured with 4 inputs: the y position of the bird, the position of the top pipe, the position of the bottom pipe, and the feedback received from the network's output with 5 nodes at hidden layers and 1 output i.e. to jump or not.

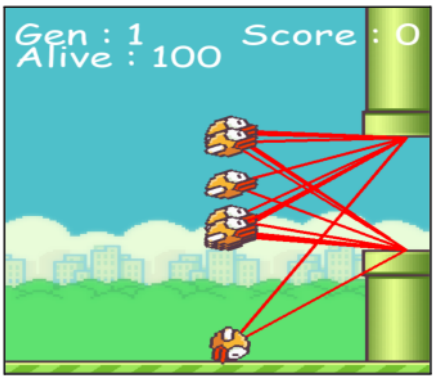

Figure14: Training Environment

Fitness parameters:

- Crashing incurs a penalty of -1, discouraging collisions with obstacles.
- Merely moving forward is rewarded with a small positive value of 0.1 to encourage continuous progress.
- Successfully passing through a pipe is highly rewarded with a significant value of +5, reinforcing the behavior of navigating through the gaps effectively.

The training process involves multiple generations of birds evolving and learning through finding the optimal point to cross the gap between the pipes. Each generation aims to improve upon the performance of the previous one, with natural selection and genetic algorithms driving the evolution towards higher fitness levels. The training continues until a predefined maximum fitness threshold value is reached, indicating that the birds have effectively learned to navigate the Flappy Bird environment.

4.3.2. Testing Environment with the Best Genome

In the testing environment, the best-performing genome from the training phase is saved as 'best.pickle'. This saved genome represents the most successful bird that has demonstrated the ability to play the game optimally by effectively navigating through the obstacles. The 'best.pickle' file contains the genetic information and neural network configuration of this top-performing bird.

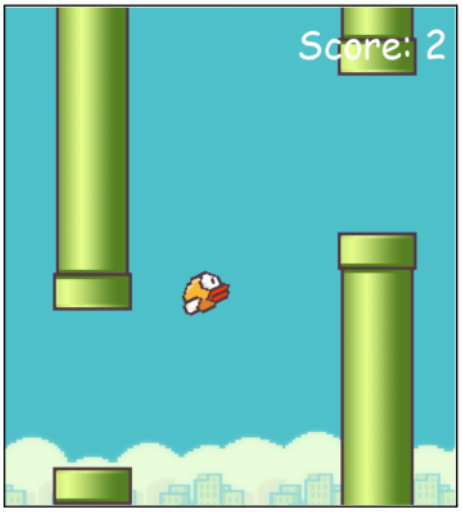

Figure15: Test Environment Screenshot

During testing, this saved genome is loaded to create a single bird that embodies the learned behaviors and strategies acquired during the training process. This bird is then put through the Flappy Bird game environment to assess its performance and validate the effectiveness of the training. By focusing on the best-performing genome, the testing environment provides a targeted evaluation of the learned behaviors and the neural network's ability to play Flappy Bird with precision and skill.

5. Result Analysis

Our experiment utilizing NEAT (Neuro-Evolution of Augmenting Topologies) to train an AI for the classic game Flappy Bird yielded highly encouraging results. Notably, the NEAT algorithm demonstrated exceptional efficiency in achieving optimal performance. Within a mere 3 generations of the AI population, the model surpassed a score of 20,000 points, all while remaining still active in the game. This rapid improvement highlights the effectiveness of NEAT's core principles, particularly its ability to evolve neural networks with minimal training data.

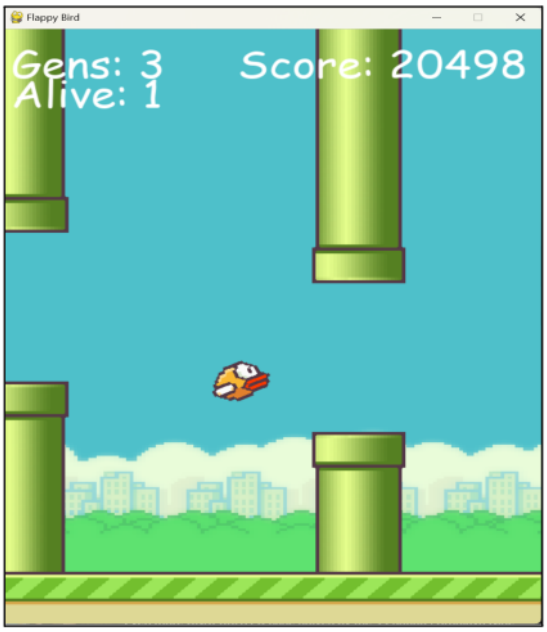

Figure16: Sample Result

This efficiency translates to a significant advantage in terms of computational resources. Compared to other training methods that may require extensive data and processing power, NEAT's ability to achieve such high performance within a limited timeframe positions it as a more practical and cost-effective solution. Further exploration is necessary to determine NEAT's scalability to more complex tasks, but these initial results suggest its potential for various applications requiring efficient AI training.

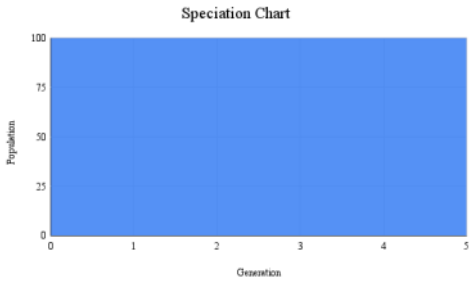

Figure 17 shows the Speciation Chart, in which the x-axis informs the generations and the y-axis informs the size of the species that is 100 which is the constant population size..

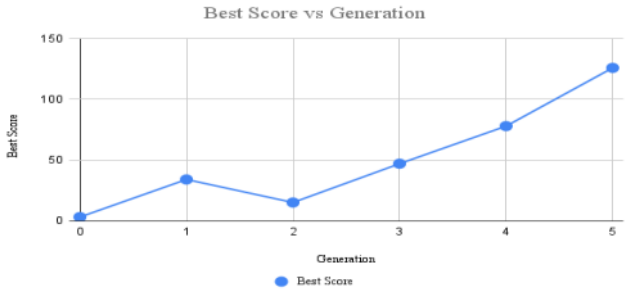

Figure18 is quite similar to the Fitness Chart but instead of Fitness it showcases the actual Maximum Score achieved by a single genome (bird) at each generation.

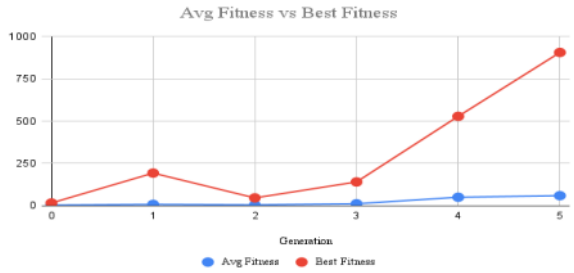

Figure 19 represents the Fitness results. The x-axis of the chart corresponds to generations, from the beginning going up to only 5, while the y-axis corresponds to the average fitness (blue line) and the best fitness (red line) on every generation.

From both the graphs we can infer that almost all scores and fitness increases with each generation.

6. Challenges and Considerations

While NEAT offers a powerful approach for training AI models, there are challenges associated with scaling it for real-world applications:

- **Data Collection and Analysis:** For industrial applications, the game would need to collect data on how the AI model is performing and analyze it to identify areas for improvement. This data analysis would be crucial for further refinement of the NEAT algorithm and the agent itself.
- **Scalability and Complexity:** Real-world applications often involve more complex environments than the basic Flappy Bird game. The NEAT algorithm and the game environment would need to be scaled to handle this increased complexity, potentially requiring more sophisticated obstacle patterns and a larger number of input parameters for the neural networks.
- **Safety and Security Considerations:** If your platform is used to train AI models for critical tasks like autonomous vehicles, safety and security of the training data and the resulting AI model would be paramount. Robust security measures would be necessary to prevent manipulation of the training data or the AI model itself.

18

7. Conclusion and Future Scope

7.1. Conclusion

This project demonstrated the potential of NEAT in training AI agents to play Flappy Bird. The NEAT algorithm successfully evolved neural networks that could navigate the game environment with increasing skill. The project also explores the potential applications of NEAT-based training for various real-world scenarios. By addressing the challenges of data collection, scalability, and safety, NEAT showcases to be a valuable tool for developing intelligent and adaptable AI systems.

7.2. Future Scope

- **AI Reinforcement Learning**: The core gameplay loop in Flappy Bird involves decision-making based on image recognition (obstacles) and precise action execution (tapping to fly). This is very similar to the challenges faced by reinforcement learning algorithms in various industries.
- **Training and Evaluating AI Models**: The Flappy Bird game could be adapted into a platform to train and evaluate AI models used in autonomous vehicles, navigation systems for robots, or stock trading algorithms. By tuning the difficulty and obstacles to mimic real-world scenarios, you could test the decision-making capabilities of the AI model in a safe and controlled environment.

  While the Flappy Bird game offers a valuable training ground for AI algorithms, its simplicity currently limits its applicability. The bird's movement is primarily optimized along the vertical axis (y-axis) to avoid obstacles. Integrating movement along the horizontal axis (x-axis) would introduce a new layer of complexity, mimicking the need for autonomous vehicles to navigate not just vertically (avoiding collisions) but also horizontally (maintaining lane position and overtaking).

  Furthermore, the current obstacle design with a single pair of pipes offers limited training for the diverse scenarios encountered in real-world driving. Introducing multiple obstacles, varying their positions on both axes, and potentially including dynamic obstacles (moving cars, pedestrians) would create a richer training environment for autonomous vehicle control.

  Finally, extending the training space to a third dimension (z-axis) could unlock the potential for training algorithms for autonomous flight. By incorporating vertical movement and obstacles in a simulated airspace, the training could encompass scenarios relevant to drone navigation or robot navigation.
  The current Flappy Bird adaptation lays the groundwork for AI training, expanding its complexity in terms of movement and obstacle design holds the key to unlocking its true potential for training AI in various domains like autonomous driving and even autonomous flight.

References

- Kevin Chen, Deep Reinforcement Learning in Stanford CS229 Machine Learning 2019.
- Atakan Sahin, Efehan Atici, Tufan Kumbasar IEEE: Type-2 fuzzified flappy bird control system in 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
- Tai Vu, Leon Tran Arxiv: [2003.09579] FlapAI Bird: Training an Agent to Play Flappy Bird Using Reinforcement Learning Techniques in 2020.
- Iveta Dirgová Luptákovág, Martin Kubovtik and Jiti Pospichal, Playing Flappy Bird Based on Motion Recognition Using a Transformer Model and LIDAR Sensor in Special Issue: Robust Motion Recognition Based on Sensor Technology 2024
- Matthew Piper, Pranav Bhounsule, Krystel K. Castillo-Villar How to Beat Flappy Bird: A Mixed-Integer Model Predictive Control Approach in Dynamic Systems and Control Conference 2017.
- Heman Mohabeer and K. M. S. Soyjaudah, Improving the Performance of NEAT Related Algorithm via Complexity Reduction in Search Space | SpringerLink in Distributed Computing and Artificial Intelligence 2017
- V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. In Deep Learning, Neural Information Processing Systems Workshop, 2013.
- T. Schaul, J. Quan, I. Antonoglou, D. Silver. Prioritized Experience Replay. arXiv: http://arxiv.org/abs/1511.05952.
- Kenneth O. Stanley, Bobby D. Bryant and Risto Miikkulainen, Real-Time Neuroevolution in the NERO Video Game in IEEE Transactions on Evolutionary Computation (Special Issue on Evolutionary Computation and Games) December 2005.
- Cristian-Bogdan Pätrasu, David-Traian Iancu Neat algorithm for simple games in 2023 15th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)
- Overview of the basic XOR example (evolve-feedforward.py) — NEAT-Python 0.92 documentation
- Configuration file description — NEAT-Python 0.92 documentation
- Overview: module code — NEAT-Python 0.92 documentation
- The XOR Operation | Baeldung on Computer Science
- How Neural Networks Solve the XOR Problem | by Aniruddha Karajgi | Towards Data Science

Individual Contribution

Sayan Banerjee (21051087)

Individual Contribution and Findings:

I spearheaded the exploration of the NEAT algorithm and its potential applications. This involved in-depth research to understand NEAT's capabilities and use cases. Building upon this knowledge, I designed and developed the XOR mechanism specifically tailored for this project. Furthermore, I crafted the neural network architecture that served as the foundation for our experimentation. To optimize performance, I conducted a series of trials, adjusting various training hyperparameters.

Individual Contribution to Project Report Preparation:

I drafted the Methodology section of the project report. This section serves as a roadmap, detailing the research and development process we followed. It outlines the steps taken, from the initial research on NEAT to the design and implementation of the XOR mechanism and neural network architecture. Most importantly, it highlights the methodologies employed during the experimentation phase.

Individual Contribution for Project Presentation and Demonstration:

I crafted the initial portions of the project presentation, focusing on the Introduction and Defining the Problem Statement. In the Introduction, I explained the functionalities of Self Learning Agents, NEAT (Neuro-Evolution of Augmenting Topologies) why have we selected Flappy Bird as our experiment environment. Following the Introduction, I transitioned to clearly Define the Problem Statement. This section outlined the specific challenge we aimed to address through our experimentation with NEAT and the optimization of its training hyperparameters. Essentially, it framed the question our project sought to answer by leveraging NEAT's capabilities.

Full Signature of Supervisor:
…….

Full signature of the student:
……..

Shubham Patel (21051094)

Individual Contribution and Findings:

My journey with NEAT in this project started as something entirely new to me. Intrigued by its potential, I embarked on a journey of thorough research to understand its workings and implications. Through this process, I gained a deep understanding of NEAT's functionalities and parameters, uncovering its relevance to our project's objectives. Taking on a proactive role, I led the design phase by meticulously crafting the base model. This involved addressing fundamental aspects such as bird movement, pipe generation, and collision detection, which served as the cornerstone for our exploration into optimizing the Flappy Bird algorithm.

Individual Contribution to Project Report Preparation:

In preparing our project report, I played a significant role in ensuring its accessibility and clarity. Leveraging my understanding of the project intricacies, I created visual aids, including detailed block diagrams illustrating the integration of NEAT within our system architecture. These diagrams were complemented by explanatory notes, aimed at demystifying complex concepts and making them understandable to a broader audience. By combining technical accuracy with simplicity, I contributed to the creation of a report that effectively communicated our project's methodologies and findings.

Individual Contribution for Project Presentation and Demonstration:

I actively participated in creating slides outlining the implementation aspect of our project. During the demonstration, I effectively communicated insights gained and showcased the functionality and future scope of our project. Through clear communication, I played a vital role in captivating our audience's attention and highlighting our project's significance.

Full Signature of Supervisor:
…….

Full signature of the student:
……..

Anjali Raj (21051032)

Individual Contribution and Findings:

My role primarily involved delving into existing scholarly articles on artificial intelligence, evolutionary algorithms, and gaming to understand the theoretical foundations and practical applications of the NEAT algorithm.

Individual Contribution to Project Report Preparation:

In the project report, I focused on drafting the Software Requirements Specification (SRS) section, outlining the goals, functionalities, and technical specifications of our Flappy Bird AI system. Additionally, I conducted a comprehensive literature review of scholarly articles, research papers, and academic discourse to enrich our understanding of NEAT's capabilities and applications. I defined the problem statement and contributed to formulating the mathematical equations underlying our project.

Individual Contribution for Project Presentation and Demonstration:

For the project presentation, I crafted the Methodology section, providing a roadmap of our research and development process from conceptualization to execution. This section outlines the methodologies employed in our experimentation, including the selection and refinement of training hyperparameters, offering a transparent overview of our approach.

Full Signature of Supervisor:
……..

Full signature of the student:
……..

Anil Padhan (21051031)

Individual Contribution and Findings:

I developed the design and development of the game components and structure for our project. This involved defining the core gameplay mechanics, user interactions, and level design. I ensured these elements aligned seamlessly with the overall project goals and target audience.

Individual Contribution to Project Report Preparation:

I drafted the initial sections of the project report, including the Introduction, Data Flow Diagram (DFD), and Unified Modeling Language (UML) diagrams. The Introduction provides context and background information about the project. The DFD visually represents the flow of data within the game system. The UML diagrams further illustrate the system's architecture and functionality.

Individual Contribution for Project Presentation and Demonstration:

Result Analysis, Conclusion, and Future Scope

I prepared the concluding sections of the project presentation, focusing on Result Analysis, Conclusion, and Future Scope.

The Result Analysis section interprets the data gathered during testing and highlights key findings. The Conclusion summarizes the project's achievements and their significance.

The Future Scope section outlines potential areas for further development and exploration.

Full Signature of Supervisor:
……..

Full signature of the student:
……..

Optimization of Flappy Bird using NEAT

ORIGINALITY REPORT

16%	15%	10%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	pdfs.semanticscholar.org Internet Source	4%
2	www.sbgames.org Internet Source	2%
3	www.ijariit.com Internet Source	2%
4	Submitted to Banaras Hindu University Student Paper	1%
5	www.coursehero.com Internet Source	1%
6	Submitted to Liberty University Student Paper	1%
7	www.arxiv-vanity.com Internet Source	1%
8	Submitted to KIIT University Student Paper	<1%
9	ijtre.com Internet Source	<1%

10	www.iieta.org Internet Source	<1 %
11	www.silvergames.com Internet Source	<1 %
12	Submitted to University of Hertfordshire Student Paper	<1 %
13	Eman Ashraf, A.E. Kabeel, Yehia Elmashad, Sayed A. Ward, Warda M. Shaban. "Predicting solar distiller productivity using an AI Approach: Modified genetic algorithm with Multi-Layer Perceptron", Solar Energy, 2023 Publication	<1 %
14	Submitted to University of Illinois at Urbana-Champaign Student Paper	<1 %
15	Submitted to Terna Engineering College Student Paper	<1 %
16	linknovate.com Internet Source	<1 %
17	Submitted to RMIT University Student Paper	<1 %
18	docshare.tips Internet Source	<1 %
19	intkonf.org Internet Source	<1 %

20	Kenneth O. Stanley. "Generative encoding for multiagent learning", Proceedings of the 10th annual conference on Genetic and evolutionary computation - GECCO 08 GECCO 08, 2008 Publication	<1 %
21	Submitted to Oxford Brookes University Student Paper	<1 %
22	Submitted to ITESM: Instituto Tecnologico y de Estudios Superiores de Monterrey Student Paper	<1 %
23	docplayer.net Internet Source	<1 %

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		