Flappy Bird AI

Abenezer Wallelign (TG3042)

Bereket Tilahun (LT3105)

Finhas Gustavo (ZD0728)

Naod Mesfin (HP6952)

HiLCoE School of Computer Science & Technology

CS488 - Artificial Intelligence

To Instructor - Seyoum Abebe

Wednesday- January 8, 2025

Table of Contents

Introduction

This report presents the development and implementation of an AI system designed to play the popular game Flappy Bird autonomously. By leveraging the NeuroEvolution of Augmenting Topologies (NEAT) algorithm, the project explores evolutionary strategies to create adaptive and intelligent behavior in virtual agents. The AI evolves through successive generations, optimizing neural network structures to navigate complex obstacles effectively. Beyond its entertainment value, this project demonstrates the broader applicability of evolutionary algorithms in autonomous systems, robotics, and artificial intelligence research.

How NEAT Works

NeuroEvolution of Augmenting Topologies (NEAT) is a fascinating algorithm designed to teach machines how to evolve their own programs. Inspired by natural evolution, NEAT was introduced in 2002 by Kenneth Stanley and Risto Miikkulainen. It takes a unique approach by evolving not only the “brains” of artificial neural networks, their connection weights but also their structure, or topology..

NEATS Process:

- **Starting Simple:** NEAT begins with basic neural networks that only have input and output neurons. From this modest starting point, these networks grow in complexity, adding new neurons and connections as needed.
- **Mutations:** Just like in biology, NEAT relies on mutations to drive innovation:
    - **Connection Mutation:** This adds new links between neurons, creating fresh pathways for information flow.
    - **Node Mutation:** By splitting an existing connection and inserting a new neuron, this mutation allows the network to develop more sophisticated processing abilities.
- **Combining Ideas:** NEAT uses a process called crossover to combine the "genetic material" of two parent networks into a new offspring network. To ensure this process works smoothly, NEAT keeps track of each gene's history with innovation numbers, which help align genes properly during the recombination process.
- **Encouraging Diversity:** To foster innovation and avoid everyone copying the same strategy, NEAT groups similar networks into species. This system allows new ideas to evolve and grow without competing directly with well-established designs, ensuring a richer variety of solutions.
- **Evaluating Fitness:** Each network’s performance is measured using a fitness function, which varies depending on the task. For instance, in a video game, the fitness score might depend on how far the character advances, how many obstacles it avoids, or how quickly it completes a level. Networks with higher scores are more likely to be chosen as parents for the next generation, driving the evolution of smarter networks.

How Our Game Reached Its Current Capabilities

Our implementation focuses on combining a **customized game environment** with NEAT's evolutionary process to produce an AI that plays Flappy Bird autonomously. We keep the NEAT mechanics (genome, species, mutation) mostly under the hood, and instead concentrate on **how** we shaped the inputs, outputs, and rewards to help the AI learn:

Game Environment & Fitness:

- We give each bird (controlled by a neural network) six input signals, as specified in the config (i.e., num_inputs = 6). These inputs include the bird's yy-position, distances to the top and bottom of the upcoming wall, whether wind is incoming, and a flag for active wind.
- The network has **two outputs** (num_outputs = 2): one determines a normal jump, the other triggers a "high jump." A high jump is more powerful or cancels the wind's downward pull.
- Whenever the bird **survives** (stays in the air) we increment the genome's fitness slightly. Passing a wall yields a more significant reward. Collisions with walls or the floor subtract from the fitness, as do misuses of the high jump (e.g., inside a wall gap).
- A **wind effect** randomly kicks in to push the bird downward, forcing the AI to consider whether a normal jump is sufficient or if it should execute a high jump to counteract wind.

NEAT Config Highlights:

- We set a **population size** of 200 (pop_size = 200) to maintain a fair amount of diversity.
- The network starts with num_hidden = 1 hidden layer, but can evolve more complexity over generations through node_add_prob = 0.2 and conn_add_prob = 0.5.
- We rely on the **tanh** activation function (activation_default = tanh), letting outputs range smoothly from -1 to 1.
- We allow weights to mutate aggressively (weight_mutate_rate = 0.8) and occasionally replace them entirely (weight_replace_rate = 0.1), promoting exploration.
- The **fitness threshold** is set to 10000, so the run stops if any genome surpasses that fitness score—meaning the AI has become highly proficient at the game.

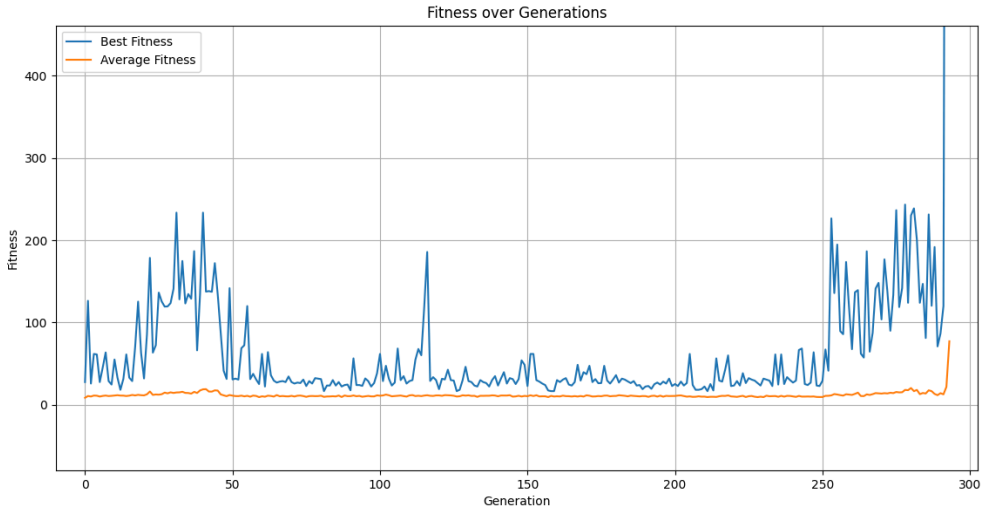
Running the Code & Collecting Data:

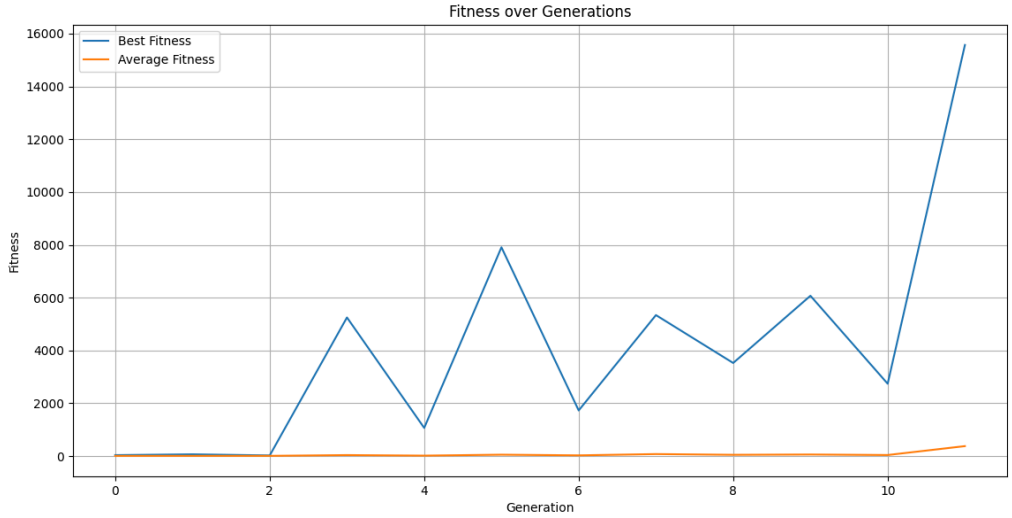
- We call population.run(main, 300), which executes our main() function for up to 300 generations. In each generation, every genome is turned into a neural network that plays Flappy Bird, logging fitness as it goes.

- As the generations progress, NEAT's StatisticsReporter records both the **best** and **average** fitness. After the run, we retrieve these statistics and plot them with **matplotlib**. The code iterates over the stats.most_fit_genomes list to get each generation's top genome, and similarly calls stats.get_fitness_mean() for average values. We then generate a line graph with **Generation** on the x-axis and **Fitness** on the y-axis to track performance.

What is the average training time?

- **Random Initialization:** Each run begins with a fresh population of neural networks whose weights and structures are assigned randomly. This means the AI effectively "starts from scratch" on every new training session.
- **Mutation and Crossover Variability:** Once training starts, the way genomes mutate or cross over also involves chance. Small changes can lead to radically different performance outcomes.
- **Early Breakthroughs vs. Slow Progress:** Sometimes, a well-timed mutation appears in an early generation, causing a sudden fitness spike. In other runs, no such lucky mutation emerges until much later, so the AI improves more slowly.
- **Independent Events:** Each training run is essentially isolated from every other run; one session discovering a great solution early doesn't influence another session. As a result, two graphs can show dramatically different fitness trajectories even with the same code and configuration. Here are the graphs below to show these effects.





By using a combination of **relevant game inputs**, **meaningful rewards/penalties**, and a **customizable NEAT configuration**, our approach successfully trains an AI bird to dodge walls, handle wind, and intelligently choose between normal and high jumps. Over multiple generations, the system refines its strategies until it reaches a level of mastery demonstrated by significantly increasing fitness scores in our matplotlib plots.

Conclusion

The Flappy Bird AI project highlights the effectiveness of the NEAT algorithm in training neural networks to solve dynamic and complex tasks. Through adaptive mutations, fitness evaluation, and iterative learning, the AI developed strategies to handle obstacles and environmental challenges, showcasing its ability to evolve over time. The insights gained from this project emphasize the potential of neuroevolutionary approaches in advancing AI applications.

Recommendation

This project is a great display of how **NEAT (NeuroEvolution of Augmenting Topologies)** builds complex and adaptive systems by utilizing the strengths of both neural networks and Genetics algorithms to optimize decisions with no supervision from the outside. It excels at tasks that demand flexibility and resilience which is exactly what is needed for a project like ours.

What makes NEAT exciting?

- **Adaptability:** Its remarkable ability to handle dynamic challenges like wind interference.
- **Learning through Mutation:** Its efficient methods to improve decision-making step-by-step with iterative changes.
- **Broad Applicability:** its suitability for robotics, autonomous navigation, and optimization tasks.

Looking Ahead:

- **Scalability:** How effectively can NEAT adapt to larger, more intricate challenges?
- **Reinforcement Learning Integration:** Its ability to Merge techniques to achieve superior adaptability and efficiency will be closely inspected.

References

Welcome to NEAT-Python's documentation! — NEAT-Python 0.92 documentation, https://neat-python.readthedocs.io/en/latest/. Accessed 10 December 2024.

Danny, Zhu. "How I Built an AI to Play Flappy Bird." *medium.com*, 26 May 2020, https://medium.com/analytics-vidhya/how-i-built-an-ai-to-play-flappy-bird-81b672b66521. Accessed 10 December 2024.

Omelianenko, Iaroslav. *Hands-On Neuroevolution with Python: Build High-Performing Artificial Neural Network Architectures Using Neuroevolution-based Algorithms*. Packt Publishing, Limited, 2019. Accessed 2 December 2024.