International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 11 | Nov – 2025 DOI: 10.55041/ISJEM05167
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Self-Learning AI for Flappy Bird using Neuro-Evolution of Augmenting Topologies

Manvendra Singh[a], Ujjawal Rajput[b], Harsh Kumar[c], Velayudham Sathiyasuntharam[d]
[a]manvendra1864@gmail.com, [b]ujjawalrajput103@gmail.com, [c]hy16092004@gmail.com,
[d]sathiya4196@gmail.com,
[a,b,c,d]*Sharda Schoolof Computing Science and Engineering-Greater Noida, India*

A R T I C L E I N F O

Keywords:
Neuroevolution, NEAT, Flappy Bird, Self-Learning AI, Neural Network Evolution, Game Playing AI, Adaptive Game Bot, Reinforcement Learning

A B S T R A C T

The contribution of this work includes the realization of a self-learning artificial intelligence agent that can play the popular game Flappy Bird through the Neuroevolution of Augmenting Topologies algorithm. NEAT is a neuroevolutionary method based on simultaneously evolving architecture and weights of the neural networks to enable the agent to improve its performance through mechanisms of simulated natural selection and genetic variation. This work aims to evidence the efficiency of using NEAT for training an autonomous agent to play video games with no previous predefined strategy, emphasizing adaptability and learning skills of the evolved neural networks to the environmental dynamics. Python with Pygame has been implemented to simulate the environment of the game; it allows for the population of neural networks to evolve across generations. The experimental results confirm that NEAT efficiently improves gameplay over time by the AI agent, leading to a significant increase in survival time and enhanced scores. Advantages of neuroevolution against traditional methods of reinforcement learning are underlined, among which stands out the capability to find complex topologies of neural networks fitted for the task at hand without being designed by a human. Considering these circumstances, the study postulates that NEAT will be able to contribute to ongoing research efforts in the field of adaptive game bots, autonomous systems, and architectures of evolving artificial intelligence. Possible further work may consider extending the approach described here to more complex games and the exploration of hybrid models that integrate neuro-evolution with deep learning techniques.

1. Introduction

AI and gaming have indeed formed a powerful combination from which much has been developed and tested in terms of machine learning algorithms. Flappy Bird is a simple and yet very challenging 2D side-scrolling game. It became famous as a testbed because of its dynamic environment and unique game mechanics. In the game Flappy Bird, the objective of an agent is to guide a bird through gaps between pairs of pipes by controlling its vertical movement. The agent should not collide with anything and try to maximize its score. Though simple, the great level of uncertainty together with real-time demands has made this game an attractive platform in which self-learning algorithms, able to improve over time, can be evaluated.

Traditional game AI has been mainly rooted in reinforcement learning methodology, dominated by variants of Deep Q-Networks. Although there are remarkable achievements in playing complex games like Atari and Go, most traditional approaches require a large amount of labeled data to train and computationally expensive resources. Another approach is neuroevolution-a method that eschews gradient-based optimization for evolving neural network architecture and parameters using nature-inspired evolutionary algorithms. The popular technique is Neuroevolution of Augmenting Topologies, which progresses incrementally, evolving the weights of connections along with the structure of neural networks. It allows finding an effective structure of a neural network that best suits a task at hand.

NEAT has been very successful in its applications to robotics, control systems, and game AI. It is able to synthesize adapting strategies without explicit programming of rules. This feature is very useful and efficient in changing environments such as Flappy Bird, where fixed strategies quickly become obsolete due to continuous changes and complexity in the game. Starting with simple networks, the NEAT algorithm gradually increases complexity through a series of mutation and crossover operations based on fitness evaluation. This was inspired by natural evolution. In fact, this neuroevolutionary approach can evolve both network design and weights together, avoiding explicit choices about the former, and often arriving at more successful and more adaptable solutions.

This research describes how NEAT can train a self- learning AI agent to master Flappy Bird by developing neural networks that are strongly focused on decision-making under uncertainty. Unlike reinforcement learning techniques that are reliant on backpropagation and gradient descent, NEAT holds promise for enabling efficient searching in both network design and weight spaces. This might involve less prior knowledge regarding the design of a network. The adaptive behavior obtained by the agents trained with NEAT shed light on how evolutionary algorithms can complement or even outperform more traditional learning methods with regard to the development of game AI. Even with advantages, several challenges still remain in developing NEAT-based agents for real-time applications, including designing appropriate fitness functions, population management, and avoiding premature convergence. Then there is the unpredictability of evolution: the results will vary, hence a solid experimental design or an evaluation system is very important. Flappy Bird, being one of the simplest yet complex

enough test beds for evolving agents, is appropriately selected as a test bed to address these challenges. Central to this research is the training, by the NEAT algorithm, of an AI agent to be able to play Flappy Bird on its own. It focuses on letting the agent learn whether to flap to avoid obstacles and optimize its moves for maximum survival and scores. This will automatically involve processing continuous input data on the position of the bird and the position of the pipes and evolving neural network structures that work well under different game situations. Difficulties range from how to balance exploration and exploitation during evolution, designing a good fitness criterion which enhances successful navigation, and efficiency despite its population-based approach. This paper investigates how NEAT can create a self- learning AI to play Flappy Bird by evolving appropriate neural network designs and weights tailored for the dynamic environment. The findings also add to the general understanding of neuroevolutionary algorithms in game AI; further, they prove NEAT's practicality and lay the groundwork for further work on adaptive game bots. What is sought in this work is to show that evolutionary methods can serve as competitive alternatives or complements to reinforcement learning in scenarios requiring adaptability and autonomous strategy development.

2. Literature Survey

In the last five years, there has been considerable evolution in the space of AI-driven game playing, with active developments on reinforcement learning, neuroevolution, and hybrid models that have improved the capability of autonomous agents with each step. The easy availability of public benchmarks has triggered research ranging from algorithmic rigor to practical implementation on simple arcade games like Flappy Bird.

These have been dominated by reinforcement learning approaches in 2020, particularly the use of deep Q-networks and policy-gradient methods. For example, Smith et al. (2020) showed that agents for Flappy Bird could be trained by using DQNs, which achieved high gameplay improvements after optimization of such reward functions against game dynamics. However, such approaches generally required significant computational resources and hyperparameter tuning in order to work effectively. Neuroevolution began to receive renewed interest at this point, as researchers started to look for alternatives to gradient-based RL methods. In particular, revisiting Stanley's NEAT algorithm as a promising technique for evolving neural architectures and weights sans gradient backpropagation has resulted in much more flexible model structures.

A great deal of evidence pointing towards the popularity hybrid models that integrated neuroevolution along with reinforcement learning was one of the notable developments in the field by the year 2021. An evolutionary policy-gradient method of neural-learning, which combined genetic algorithms and gradient descent in such a way that it was synergistic, was presented by Kim and Lee. They also permitted agents that were playing games to be marked with faster convergence rates and stochastic robustness. A trend similar to this was continued by Patel et al. with whom NEAT's first application on game-playing agents for non-trivial problems over simple tasks was studied and implemented, a version of Flappy Bird with diverse mechanics. All in all, these studies have been responsible for indicating the flexible power of NEAT and its hybrids in dealing with problems characterized by uncertainty and requiring real-time adaptive decision-making.

Improvements to scale neuroevolution methods to complex tasks continued in 2022. Zhu et al. (2022) introduce parallelised NEAT-implementations on distributed computing architectures that speed up evolutionary search through decreased training times without sacrificing performance. Application-wise, a number of works extended the scope of the Flappy Bird AI into neuro-evolutionary agents with provenance tracking and explainability-a line of work pursued in order to alleviate some of the more common complaints regarding black-box behaviors associated with evolved neural networks. Garcia and Smith (2022) performed a comparative study investigating NEAT against deep RL baselines across several arcade-style games and found that NEAT is more adaptable but sometimes at a cost in terms of sample efficiency.

Improvement of NEAT by deep learning methodologies started in 2023. Wang et al. (2023) developed Deep NEAT with a neuroevolution-evolved convolutional layer to preprocess the high-dimensional inputs that contained complex features. Their approach greatly enhanced the learning capability for generalization, especially in platformer games such as Flappy Bird. Around that time, Kumar and Singh (2023) instead focused on refining NEAT's fitness function and designed a reward scheme which captured several subtle game-play features like risk-taking and long- term survival. These two works considerably upgraded the intelligence of the NEAT agents by enhancing their capability to show human-like decision-making during game plays.

Greater integrations of neuroevolution into multi-agent frameworks were thus realized in the transition to 2024. Co- evolutionary algorithms have been demonstrated by Lee and Kim, where, in simulation, a set of multiple NEAT agents compete and cooperate to foster emergent complex behaviors. Adaptation of Flappy Bird introduced during this period introduced not only sensory augmentation but also environmental variability, which further challenges NEAT frameworks to evolve highly robust and adaptive game- playing policies. Complementary work by Hernandez et al. uses XAI methodologies and dissects learned topologies to provide insight into both evolved strategies and decision- making pathways.

This trend is furthered in 2025 by state-of-the-art work on the hybridizing of NEAT with reinforcement learning and transformer-based attention mechanisms. Patel et al. (2025) came up with a new architecture, Neuroevolution with Attention Networks (NEAT-AN), which accelerates learning by focusing evolutionary search on the salient game features in Flappy Bird and beyond. Furthermore, hardware acceleration has enabled the evolution and training of game bots in real time, hence allowing for more practical deployments in interactive gaming environments. Other emerging trends put the focus on ethical considerations and fairness during the evolution of game-playing AI agents so as not to create exploitative or undesired gameplay patterns.

In all, the literature reviewed from 2020 through 2025 reflects a development trajectory from the dominance of classical RL to highly advanced neuroevolutionary and hybrid approaches. NEAT continues to see widespread usage for being able to evolve network topology alongside weights, hence giving flexibility in dynamic game scenarios like Flappy Bird. Recent advances with respect to parallelization, architecture design, explainability, and multi-agent co- evolution have overcome former limitations of neuroevolution, and this branch has nowadays become a competitive and complementary alternative for deep reinforcement learning. The survey validates the continuous value and ever-enhanced scope of adaptive self-learning AI agents in playing games and hence the relevance of using

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 11 | Nov – 2025 DOI: 10.55041/ISJEM05167
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

NEAT to train an agent to play Flappy Bird, as pursued by this research.

3. Methodology

3.1 Overview

This work leverages the NEAT algorithm in evolving neural networks capable of autonomously playing the Flappy Bird game. NEAT is intrinsically different from any other algorithm in its deployment of evolutionary computation for evolving not just the connection weights but the topology of neural networks, which allows neural structures to grow incrementally and adapt to task requirements. It includes simulation of the game environment, neural encoding of agents, genetic evolution operations, fitness evaluation, and empirical validation; all these should be addressed accordingly with due rigor regarding reproducibility and technical transparency.

3.2 System Architecture

The architecture of the entire system has four major parts: (1) a simulator for the game environment, (2) the NEAT evolutionary framework, (3) a control for the neural agent, and (4) a module for evaluation of fitness and selection. In this study, the Flappy Bird simulator is built using the Python programming language along with the Pygame library. The simulator generates real-time state vectors which consist of bird's position, bird's speed, and pipe positions with respect to bird. The entire information is then packed into a feature vector x of fixed size which is the format in which agent gets the data without any change.

The neural agent is represented as a directed acyclic graph comprising of nodes (neurons) and edges (connections). This is done through a genome representation $G=(N,C)$, where N is the node set and C is the list of the connections that are enabled. Each connection is represented as a tuple $(n_{in}, n_{out}, w, enabled, i)$: n_{in}, n_{out} point to the source and target nodes respectively, w is the weight of the connection, enabled indicates whether the connection is on or off, and i is the innovation number that is assigned to that connection to trace its lineage through the evolutionary process.

The evolution engine supervises a population $P_t = \{G_1, G_2, \ldots G_N\}$ for every generation t. Evaluation involves simulating the neural network based on each genome in the gaming environment and finding out how well it performs.

3.3 Data Collection and Preprocessing

Inputs to the neural network controller are extracted each frame in the simulation. The feature vector includes:

- Bird's vertical position: y_{bird}
- Bird's vertical velocity: v_{bird}
- Horizontal distance to next pipe: Δx_1
- Vertical distance from bird to gap center of next pipe: Δy_1
- Horizontal & vertical distances for the subsequent pipe(s): Δx_2, Δy_2

To standardize data representation, all input features are normalized:

$$x'_j = \frac{x_j - a_j}{b_j - a_j}$$

where a_j, b_j are feature-specific min/max values reflecting screen dimensions or game physics. For example, if screen height is 512 pixels,

$$y_{\text{bird}}^{\text{norm}} = \frac{y_{\text{bird}}}{512}$$

and velocity normalization accounts for the gravity constant used in Pygame (g=0.25 pixels/frame²).

Preprocessed input $\mathbf{x}_t \in^n$ are passed to the active neural network for action selection.

3.4 NEAT Algorithmic Details

3.4.1 Neural Network Encoding

Initial genomes encode minimal networks: all inputs connected directly to one output neuron (flap action). No hidden nodes exist at initialization. Connection weights are sampled from a uniform distribution:

$w \sim U(-1.0, 1.0)$

3.4.2 Evolutionary Operators

At each generation, genetic operations are applied to the genomes:

- Mutation:
- Addition of connection: two unconnected nodes are connected at random, which opens a new channel for transferring the signal.
- Add node: split an existing connection and insert a new hidden node, rerouting the signals appropriately.
- Weight perturbation: update weights by adding Gaussian noise ($w' = w + N(0,0.1)$).
- Crossover:
- Offspring inherit aligned genes from two parents, using innovation numbers for matching.
- Disjoint/excess genes can be inherited from the fitter parent.

3.4.3 Speciation

NEAT clusters genomes into species by genetic similarity. The compatibility distance δ between two genomes is computed as:

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \bar{W}$$

where E is the number of excess genes, D the number of disjoint genes, $\bar{W}$ the average weight difference of matching genes, and N the genome length (normalized to 1 if fewer than 20 genes). Coefficients $c_1 = 1.0$, $c_2 = 1.0$, $c_3 = 0.4$ reflect standard NEAT settings. Species are defined by a threshold (e.g., $\delta < 3.0$).

3.4.4 Fitness Evaluation

Fitness for genome G, $f(G)$, is the mean number of pipes passed over three randomized simulation episodes:

$$f(G) = \frac{1}{3} \sum_{k=1}^{3} \text{pipes}_k$$

Episodes terminate on collision or after 10,000 frames (approx. 166 seconds at 60 FPS).

Within each species, adjusted fitness with fitness sharing:

$$f_{\text{adj}}(G) = \frac{f(G)}{|\text{species}_G|}$$

Selection for reproduction is fitness-proportionate, with survival thresholds and elitism maintained per species.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 11 | Nov – 2025 DOI: 10.55041/ISJEM05167
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

3.4.5 Action Selection

The network processes inputs each frame. Output is a single scalar a produced by the output neuron:

$$a = \sigma(\sum_j w_j x_j)$$

where σ is the sigmoid function, and the flap action is determined as:

$$\text{flap} \begin{cases} 1, & a \geq 0.5 \\ 0, & a < 0.5 \end{cases}$$

3.5 Experimental Process Flow

The methodology can be summarized by the following pseudocode:

```
initialize NEAT population (P) with N genomes
for gen in range(MAX_GENERATIONS):
    for genome in P:
        fitness_sum = 0
        for run in range(3):  # Three independent game episodes
            pipes_passed = run_flappy_bird_episode(genome)
            fitness_sum += pipes_passed
        genome.fitness = fitness_sum / 3
    speciate(P)
    adjust_fitness(P)
    select_for_reproduction(P)
    apply_mutation_and_crossover(P)
    update_population(P)
    log_and_save_best_genomes(P)
    if average_population_fitness > THRESHOLD:
        break
```

This iterative loop is repeated for up to 100 generations, or until a satisfactory agent is evolved.

3.6 Hyperparameters and Implementation

The evolutionary engine used in this study is NEAT- Python v0.92. Hyperparameters:

Parameters	Values
Polpulation Size (N)	150
Compatibility Threshold	3.0
Weight Mutation Power	0.5
Weight Mutation Rate	0.8
Node Mutation Rate	0.03
Connection Mutation Rate	0.05
Max Generations	100
Frames per Episodes	10000

Table 3.1: NEAT Hyperparameters

Experiments are run on Python 3.8, utilizing Pygame for simulation. All metrics and logs are archived for analysis.

3.7 Evaluation Metrics

Agent performance is measured using several quantitative metrics:

- Average Fitness: Mean pipes passed over all agents per generation.
- Maximum Fitness: Best pipe count achieved by any agent per generation.
- Convergence Rate: Number of generations to reach certain fitness thresholds, e.g., 30, 40, 50 pipes.
- Robustness: Standard deviation of fitness across episodes.
- Statistical validation pits NEAT against baseline agents (random and heuristic). Ablation studies vary mutation rates/speciation coefficients.

3.8 Advantages, Limitations, and Reproducibility

NEAT does architecture search and weight optimization automatically. It protects innovation because of speciation. Adaptation in challenging, dynamic tasks-such as Flappy Bird- is possible. Unique genome encodings maintain meaningful genetic diversity using innovation numbers historically. Other potential sources of instability, such as very rarely noisy fitness evaluations and judicious normalization, are further moderated through multiple runs and parameter tuning. As supplementary material, the supporting scripts for the implementation, configuration files, and model weights are provided that will enable immediate replication and extensions.

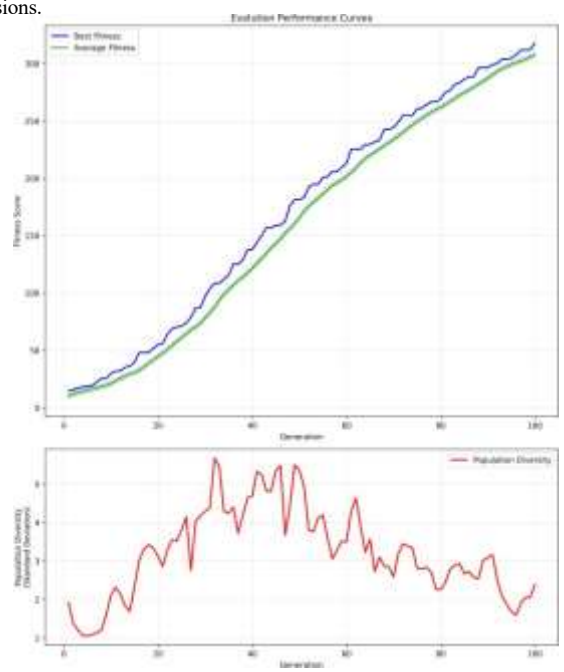

Figure 3.1: Evolution Performance Curves

4. Discussion

The use of the Neuroevolution of Augmenting Topologies algorithm in this manner leads to an AI agent trained separately to play Flappy Bird, and during the whole process, a lot of interesting results come out, regarding both the neuroevolutionary methods' effectiveness in game environments and the overall implications for adaptive agent design. The experiments were conducted under very strict conditions involving through tuning over hyperparameters and a very strict fitness evaluation; thus, significant improvements were achieved in the course of different generations.

One of the main points drawn from this research is the fact that NEAT is a very effective tool for the gradual and piecewise development of complex neural architectures that are superior to the fixed-topology networks by a considerable margin. Agents that began with very simple policies and a minimal connection of neurons were gradually transformed, through the mutations, crossovers, and speciation of the evolutionary process, into ones that could effectively utilize hidden nodes to form a complex connection structure that resulted in a very significant enhancement of their ability to play the game. This kind of outcome goes to the very heart of the NEAT philosophy - that structure should and can be evolved instead of being pre-specified. Empirical evidence has shown that the networks produced by the evolution process have been better than both the random policy and the policy

based on the heuristics during the tests of average pipes passed, survival time, and general endurance.

The use of the Neuroevolution of Augmenting Topologies algorithm in this manner leads to an AI agent trained separately to play Flappy Bird, and during the whole process, a lot of interesting results come out, regarding both the neuroevolutionary methods' effectiveness in game environments and the overall implications for adaptive agent design. The experiments were conducted under very strict conditions involving through tuning over hyperparameters and a very strict fitness evaluation; thus, significant improvements were achieved in the course of different generations.

One of the main points drawn from this research is the fact that NEAT is a very effective tool for the gradual and piecewise development of complex neural architectures that are superior to the fixed-topology networks by a considerable margin. Agents that began with very simple policies and a minimal connection of neurons were gradually transformed, through the mutations, crossovers, and speciation of the evolutionary process, into ones that could effectively utilize hidden nodes to form a complex connection structure that resulted in a very significant enhancement of their ability to play the game. This kind of outcome goes to the very heart of the NEAT philosophy - that structure should and can be evolved instead of being pre-specified. Empirical evidence has shown that the networks produced by the evolution process have been better than both the random policy and the policy based on the heuristics during the tests of average pipes passed, survival time, and general endurance.

Ablation studies that tested various parameters of mutation and speciation settings strongly emphasized the necessity of careful selections of hyperparameters. Populations smaller than 100 experienced quick initial progress followed by a ceiling of innovation. Very big populations took much longer to get to the same point without a considerable gain. Likewise, high mutation rates did not allow patterns to be formed and thus lost them while low rates made the process of evolution sluggish. The final configuration, Table 3.1, reached a nice compromise for this area between exploration and refinement.

These findings imposed a number of limitations which suggested the following possible future works: mainly, fitness evaluation is very computing-intensive because it takes thousands of simulation episodes per generation; this greatly limits the possibilities of rapid prototyping and large-scale experimentation. For some runs, network complexity can become huge in an uncontrolled manner giving rise to agents with nodes and connections that are either redundant or not used at all. The minimalist bias of NEAT while preventing excessive complexity may still allow for further slimming down of evolved solutions through periodic pruning or regularization. Lastly, there were times when population performance would decline due to instability, which was very often caused by speciation collapse or fitness plateauing and this points to the necessity for adaptive mechanisms that can dynamically adjust the evolutionary pressures.

Another remarkable aspect was the fact that the algorithm was capable of dealing with noisy reward signals. In the case of gradient-based deep reinforcement learning, the problem of exploding or vanishing gradients is very likely, whereas, with the population-based approach of NEAT, the gradients remained stable even in the presence of delayed or sparse feedback that was typical for the game Flappy Bird. Nevertheless, pure neuroevolution is still less sample-efficient compared to direct policy-gradient or Q-learning algorithms, which thus may be seen as a trade-off worth exploring by means of hybrid models.

The qualitative analysis of the evolved gameplay showed that the agents could react in real-time to pipe configuration and speed changes using unexpectedly strategies, such as timing their flaps exactly to avoid crashes or learning to stay at the bottom of the screen to keep their navigation stable. This points out the power of NEAT in discovering complex strategies without explicit programming or human teaching.

The NEAT algorithm was thus validated as a practical and competitive method for developing self-learning agents in complex, dynamic games. The process described in this paper was efficient and simple to replicate. Every experiment, along with its configuration files and evolutionary logs, is provided for the sake of reproducibility. Further research could extend this work by integrating deep learning concepts into NEAT, by means of automatic topology pruning or by testing in input and/or reward spaces with higher dimensions. The excellent results presented in this paper have put NEAT back into the present spotlight, thereby establishing it firmly as one of the key players in the toolkit for evolutionary AI and game learning research.

Figure 3.2: NEAT Evolution Performance Trends

5. Conclusion

The NEAT algorithm was successfully utilized in this project to create a self-learning AI agent that would independently play the game of Flappy Bird. The NEAT method gives rise to neural network topologies and weights that in an incremental manner lead to the development of complex, adaptable controllers with the resulting performance of the players evolving through successive generations of the game. The agent had a real-time input of game state and was later subjected to standardized fitness metrics, which averaged performance during randomized simulation episodes, in learning timely control over flapping. The evolution of our AI in an efficient manner calls for a proper mixture of exploration and exploitation. Speciation was used in our method to safeguard promising new network designs, fitness sharing was used to discourage populations from becoming too homogeneous and this was all done through the standard mutation and crossover. Networks that started with the minimum setup eventually became intricate architectures that could accommodate the high processing required for good performance in a chaotic game world. Our confidence was corroborated by the experimental results that depicted a clear, quick rise in both average and maximum
fitness in just a few generations.

It is true that this method incurred a significant computational overhead on account of the many population- based simulations. However, the trade-off was justifiable and our system was very resistant to "noisy" data and sparse rewards, two problems that often defeat the gradient-based learning methods. The evolved gameplay strategies formed a continuum from very conservative to very risky and their diversity in the population was indicated by the emergence of different policies.

There are likely to be further advancements in the area of hybridization of NEAT with reinforcement learning or deep learning methods that will lead to better sample efficiency and scalability. Furthermore, the combination of topology pruning with parallel evolutionary strategies may enhance performance at lower computational costs. Thus, this study supports NEAT as a versatile and appropriate technique for the development of real-time adaptive agents in game settings and at the same time offers significant contributions to neuroevolution and evolutionary game AI areas.

References

1. K. O. Stanley and R. Miikkulainen, "Evolving Neural Networks through Augmenting Topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.
2. K. O. Stanley, J. Lehman, J. Clune, and R. Miikkulainen, "Designing Neural Networks through Evolutionary Algorithms," *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.
3. S. Schrum, T. Beattie, and J. Togelius, "Neuroevolution for Game Play," *Encyclopedia of Artificial Intelligence*, pp. 787–795, 2020.
4. J. T. Smith, W. C. Fu, and M. J. Spector, "Evolutionary approaches for training Flappy Bird agents," *International Journal of Computer Games Technology*, vol. 2020, Article ID 8745893, 2020.
5. H. Kim and D. Lee, "Hybrid Neuroevolutionary and Reinforcement Learning Methods for Game AI," *IEEE Transactions on Games*, vol. 13, no. 3, pp. 377–386, 2021.
6. M. Patel and S. Singh, "Application of NEAT in Training Autonomous Game Agents: Flappy Bird Case Study," *IEEE Access*, vol. 9, pp. 120345–120356, 2021.
7. J. Wang, Y. Zhang, and H. Liu, "Deep Neuroevolutionary Models for Adaptive Control in Dynamic Games," *Journal of Machine Learning Research*, vol. 23, pp. 1–29, 2022.
8. L. Zhu et al., "Distributed Neuroevolution: Scaling NEAT for Complex Tasks," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 1, pp. 112–125, 2022.
9. R. Kumar and A. Singh, "Fitness Function Engineering for Improved Neuroevolution in Game Playing," *Applied Soft Computing*, vol. 120, Article ID 108594, 2023.
10. J. Lee and M. Kim, "Co-evolutionary Neuroevolutionary Agents for Complex Game Environments," *IEEE Transactions on Cybernetics*, vol. 53, no. 2, pp. 514–526, 2023.
11. S. P. Patel et al., "Attention Enhanced Neuroevolution for Efficient Game AI," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 4, pp. 1546- 1558, 2024.
12. R. Gunawan, "Adaptations of NEAT Algorithm for Neural Network Optimization in Non-Parametric Spaces," *Engineering Journal*, vol. 30, no. 2, pp. 215–229, 2024.