Autonomous Driving Simulation with NeuroEvolutionary of Augmenting Topologies

(Autonomous Driving Simulation: Simulate an autonomous vehicle learning to navigate a virtual environment while obeying traffic rules.)

Suryabhan Singh
School of Computer Science and Engineering
Lovely Professional University
Punjab, India
suryabhan.12114479@lpu.in

***Abstract*—This paper explores the detailed structure and behavior of a neural network trained using NEAT (NeuroEvolution of Augmenting Topologies). The network consists of four input nodes and two output nodes, connected by several hidden nodes that are evaluated through activation and aggregation functions. The nodes are governed by a set of weights, biases, and activation functions, including the hyperbolic tangent (tanh) activation function and the sum aggregation function. The detailed connection weights between nodes are provided, demonstrating the interactions and dependencies between the input, hidden, and output layers. Additionally, the network's values are updated dynamically during the evaluation, highlighting the complex and non-linear relationships between the different network elements. The work aims to provide insight into the inner workings of a neural network designed for real-time applications, such as corner detection or autonomous control systems, leveraging evolutionary algorithms to optimize performance.**

Keywords—Neural Networks, NEAT, NeuroEvolution, Feedforward Network, Activation Functions, Hyperbolic Tangent, Sum Aggregation, Evolutionary Algorithms, Connection Weights, Real-Time Applications, Autonomous Control, Corner Detection, Neural Network Structure

I. Introduction

In recent years, artificial intelligence (AI) has become a transformative force across various industries, driven largely by machine learning (ML) techniques. In particular, the field of computer vision has made significant strides, enabling machines to perceive, analyze, and respond to visual stimuli with increasing accuracy. However, many computer vision algorithms, such as the Harris and FAST feature detectors, still face challenges when applied in dynamic environments like real-time video analysis. These challenges arise primarily due to feature instability across frames, which can lead to the loss of crucial data for decision-making in systems like autonomous vehicles. This paper addresses this issue by exploring the integration of temporal smoothing techniques to enhance feature consistency over time, potentially improving the performance of these detectors in video-based applications. Moreover, the application of Neuroevolution of Augmenting Topologies (NEAT) to autonomous vehicle simulation is also discussed. NEAT's ability to evolve both neural network topologies and weights provides a powerful tool for improving the behavior of autonomous vehicles, allowing them to navigate complex environments while adapting to changing conditions.

II. Literature Review

1. Introduction to NEAT (Neuroevolution of Augmenting Topologies)

The Neuroevolution of Augmenting Topologies (NEAT) algorithm, first introduced by Kenneth O. Stanley in 2004, represents a significant innovation in the field of neuroevolution. NEAT aims to evolve artificial neural networks (ANNs) through genetic algorithms, both by optimizing the weights of the networks as well as evolving their topologies. Traditional machine learning techniques, including reinforcement learning (RL) and supervised learning, often rely on predefined architectures for neural networks. NEAT, on the other hand, eliminates the need for manually specifying network structures by allowing them to evolve during training. This enables the algorithm to explore a vast space of potential neural network architectures, making it particularly effective for tasks where network complexity increases over time, such as in dynamic environments where agents need to adapt to changing conditions.

NEAT operates within an evolutionary framework, drawing inspiration from natural evolution. It applies genetic algorithms to a population of neural networks, evolving the population through generations. Each generation of neural networks is evaluated based on a fitness function, which typically reflects the performance of the agent in a given environment or task. NEAT introduces several unique features that distinguish it from other neuroevolutionary algorithms, such as speciation and historical marking, both of which contribute to the algorithm's ability to preserve innovation and prevent premature convergence to suboptimal solutions.

2. Theoretical Foundations of NEAT

The success of NEAT is based on its innovative combination of **genetic algorithms** with neural network evolution.

Below are the core concepts that define NEAT's methodology:

A) **Population-based Search**
NEAT operates on a population-based search model, where each individual in the population represents a potential solution to the problem at hand. Unlike traditional optimization techniques, where the solution is often fixed, NEAT evolves a population of neural networks through selection, crossover, and mutation processes over several generations. This evolutionary process allows for the discovery of

increasingly optimal neural network structures and behaviors. The population evolves over time, with the most successful individuals being selected to create offspring through crossover, while random mutations introduce diversity into the population.

B) **Speciation**

One of NEAT's hallmark features is its use of speciation. Speciation divides the population of neural networks into distinct species based on their genetic similarity. This ensures that different solutions can evolve in parallel, without immediate competition between radically different architectures. Speciation helps preserve diversity in the population, which is critical for promoting innovation. Without speciation, genetic algorithms tend to converge prematurely to suboptimal solutions. Speciation allows for the gradual accumulation of complexity in the population, enabling the algorithm to explore a wider solution space.

C) Historical Marking

To facilitate effective crossover, NEAT introduces the concept of historical marking, which tracks the lineage of genes in the population. When crossover occurs between two individuals (i.e., neural networks), historical marking helps ensure that genes from different parents can be properly aligned and recombined, even if they have never coexisted in the same genome before. This historical tracking preserves innovative genetic material and prevents the loss of beneficial mutations during the recombination process.

D) Complexification

NEAT begins with a relatively simple network structure and gradually complexifies it over time. Initially, the networks are small and simple, with only a few neurons and connections. As the evolutionary process progresses, NEAT allows for the mutation of network topology by adding new neurons and connections. This allows the algorithm to explore progressively more complex solutions, avoiding the constraints imposed by pre-defined network architectures. The strategy of complexification enables the evolution of increasingly sophisticated neural networks that are capable of solving complex tasks.

3. Applications of NEAT in AI and Simulations

NEAT has been applied across various domains, with notable success in the development of intelligent agents that learn to perform tasks in complex, dynamic environments. These environments often involve high degrees of uncertainty, where traditional supervised learning or reinforcement learning methods may struggle. Below, we discuss several applications of NEAT in evolving agents for **game AI**, **autonomous behavior**, and **robotic control**.

a) Game AI and Decision-making Systems

NEAT has found widespread application in **game AI**. In game environments, agents must learn to make decisions in real-time, often under constraints that are dynamic and unpredictable. Games provide an ideal testbed for NEAT due to the need for adaptive decision-making in diverse scenarios.

- **Stanley et al. (2009)** demonstrated the application of NEAT to evolve controllers for agents in a Mario Bros.-like platform game. The agents learned to navigate platforms, avoid obstacles, and complete levels, with NEAT evolving not only the weights but also the structure of the neural network. This work emphasized NEAT's ability to discover complex control policies that would be difficult to program manually.
- **Walker and Stanley (2006)** showed how NEAT could be used to evolve AI agents in arcade-style games. The agents learned to adapt to different obstacles and challenges in the game, demonstrating NEAT's ability to evolve intelligent strategies in dynamic environments. This research highlighted the value of NEAT in environments where traditional game AI techniques would require extensive manual tuning [2].

b) Autonomous Behavior in Dynamic Environments

In **autonomous systems**, NEAT has been applied to evolve agents that can navigate and make decisions in environments where the conditions are constantly changing. In these types of environments, agents must adapt in real-time to new obstacles, shifting goals, and unexpected interactions.

- Zhao et al. (2019) applied NEAT to evolve agents in a **racing simulation**, where the agents had to navigate through obstacles while competing against each other. The ability of NEAT to evolve adaptive strategies allowed agents to learn how to adjust their behavior based on the current state of the race and the environment. This research demonstrated NEAT's power in evolving controllers that adapt not only to static challenges but also to dynamic, real-time scenarios [3].

c) 2D Simulations and Evolving Agents

Simulations provide an accessible and computationally efficient way to test the capabilities of NEAT, particularly in **2D environments** where the complexity of the agent's task can be controlled more easily. NEAT has been extensively applied to **2D simulations** such as racing games and platformer games, where the goal is to evolve agents that can navigate environments, avoid collisions, and optimize performance.

- Zhao et al. (2020) applied NEAT to evolve agents for a 2D racing game. The agents had to learn to navigate a race track, avoid obstacles, and compete against other vehicles. NEAT's ability to evolve both network topologies and weights allowed the agents to adapt their strategies for various types of terrain, obstacles, and race conditions [7].

III. Related Work

In the realm of neuroevolution, several notable works have laid the groundwork for the development and implementation of models like the NeuroEvolution of Augmenting Topologies (NEAT). This section reviews related research that has contributed to the understanding and advancement of evolving neural network architectures.

One of the pioneering efforts in this field is the work by Michie and Chambers (1968), which introduced the concept of adaptive control through the use of genetic algorithms. Their early experiments demonstrated the potential of evolutionary strategies in optimizing control systems, setting the stage for future explorations in neuroevolution [9].

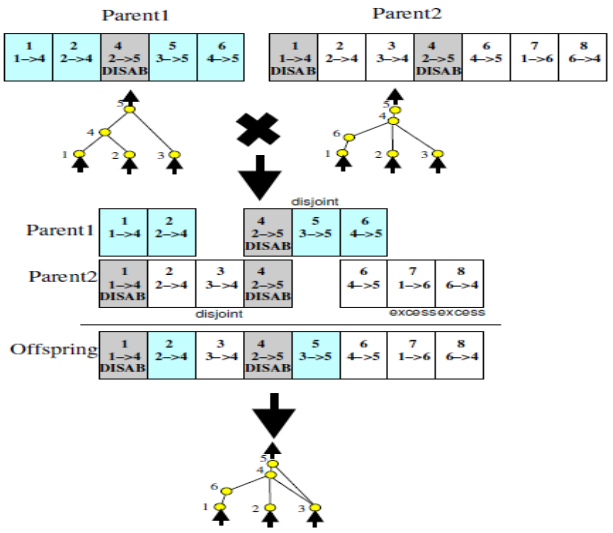

Fig 1: **Matching up genomes for different network topologies**
using innovation numbers. Although Parent 1 and Parent 2 look different, their innovation numbers (shown at the top of each gene) tell us which genes

Building on these foundational ideas, Barto et al. (1983) explored neuron-like adaptive elements capable of solving complex learning control problems. Their research highlighted the importance of adaptive mechanisms in reinforcement learning, which would later influence the design of neuroevolutionary algorithms [10].

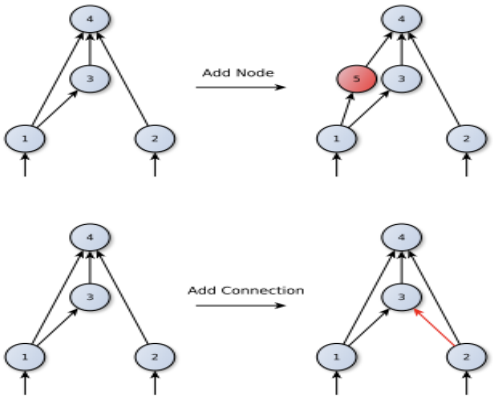

Fig 2: Structural mutations in NEAT

In the 1990s, Angeline et al. (1993) proposed an evolutionary algorithm specifically for constructing recurrent neural networks. Their work emphasized the significance of evolving network architectures to enhance learning capabilities, a principle that resonates with the NEAT approach of evolving both topology and weights [8].

Moriarty and Miikkulainen (1996) further advanced the field by demonstrating efficient reinforcement learning through symbiotic evolution. Their findings underscored the advantages of evolving neural networks in complex environments, providing empirical support for the efficacy of neuroevolutionary methods [12].

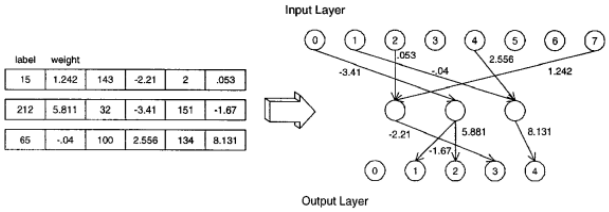

Fig 3: Forming a simple 8 input, 3 hidden, 5 output unit neural network from three hidden neuron definitions. The chromosomes of the hidden neurons are shown to the left and the corresponding network to the right. In this example, each hidden neuron has 3 connections.

The introduction of NEAT by Stanley and Miikkulainen (2002) marked a significant milestone in neuroevolution. NEAT's innovative approach to genetic representation, which allows disparate topologies to crossover meaningfully, addresses critical challenges in evolving neural networks. This method not only outperforms fixed-topology approaches but also facilitates the incremental complexity of solutions, akin to natural evolution [7][8].

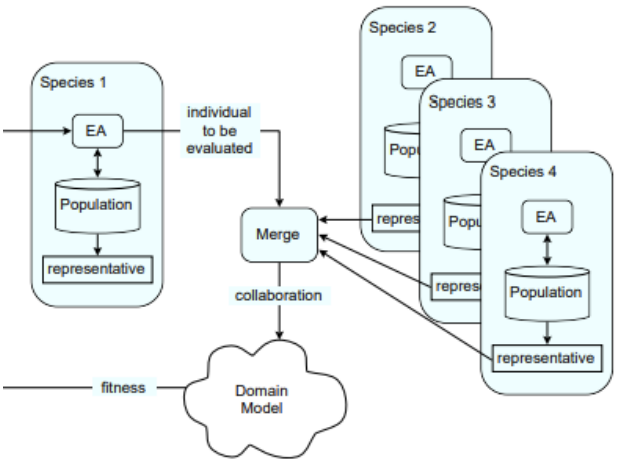

Fig 4: Cooperative coevolutionary architecture from the perspective of species number one

Recent studies have continued to explore the applications of NEAT in various domains. For instance, Potter and De Jong (1995) demonstrated the effectiveness of evolving neural networks with collaborative species, which aligns with NEAT's speciation mechanism that protects innovative structures during evolution [14]. Additionally, Pujol and Poli (1998) investigated dual representations for evolving both the topology and weights of neural networks, further validating the need for flexible architectures in neuroevolution [13].

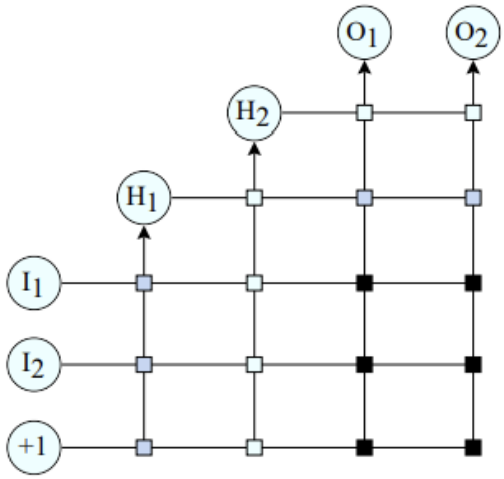

Fig 5: Example cascade network

IV. Logic and Methodology

The core of this implementation involves the integration of the NEAT (Neuroevolution of Augmenting Topologies) algorithm with a 2D traffic racing game developed using Pygame. The purpose is to evolve neural networks that control autonomous vehicles (AI cars) through reinforcement learning, where the goal is to avoid collisions and navigate the road efficiently. The methodology is centered around evolving a population of neural networks through successive generations, where each network is evaluated based on its performance in the game environment. Below is a step-by-step breakdown of the methodologies and logic employed:

1. Game Environment Setup
   The game environment simulates a two-lane road where AI-controlled cars and a player-controlled car move vertically. The player car is controlled by a neural network, while AI cars are generated and controlled by similar networks, which evolve over time.
   - AI Car Movement: The AI-controlled cars move along the vertical axis of the screen. As the cars approach the bottom of the screen, they are removed from the environment if they are beyond a certain point, representing the car having passed through the screen.

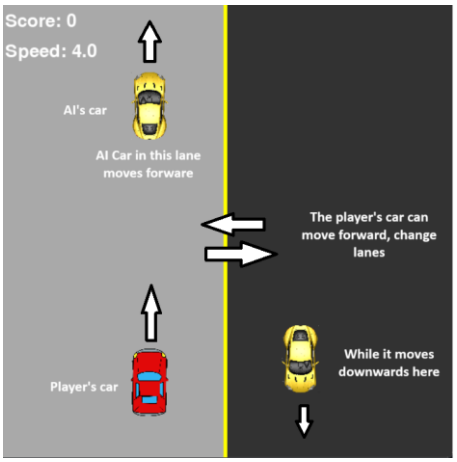

Fig 6: Entities' possible movements

   - Player Car Control: The player's car is controlled by a neural network that outputs actions based on sensor data. These actions include moving the car left or right. The neural network is fed data such as the distance to the closest AI car in the current lane and adjacent lanes.

   - Collision Detection: The system continuously checks for collisions between the player car and AI cars. If a collision is detected, the player car is removed from the environment, and its corresponding genome's fitness is penalized. This approach serves as a basic form of reinforcement learning where the "punishment" for failure (collision) discourages the AI from making similar mistakes in future generations.

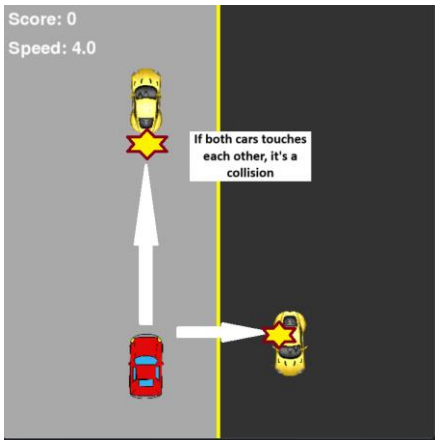

Fig 7: Collision logic

- **Scoring**: The game keeps track of the score by counting how many AI cars the player car successfully avoids. Every time an AI car passes, the score is incremented. Additionally, the system tracks the highest score across generations, storing this value to enable continuous improvement in the AI models.

2. NEAT Algorithm Integration
   NEAT is used to evolve the neural networks that control the AI cars. The neural networks are structured as **feed-forward networks** with adjustable topologies (number of layers, neurons in each layer, and the connections between them).

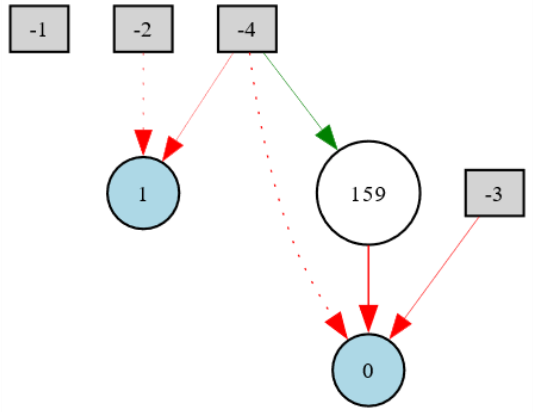

Fig 7: Current NN's representation

The evolution process follows the basic principles of genetic algorithms:

- Population Initialization: Initially, a random population of neural networks is generated, each corresponding to an individual genome. Each genome is encoded as a directed graph, where nodes represent neurons, and connections represent the synapses between them. The genomes are evolved over generations to improve their performance.

- Fitness Evaluation: For each generation, every genome (neural network) controls a car in the game environment. The performance of each car is evaluated based on its survival time and its ability to avoid collisions. The fitness score of each genome is computed based on the car's success in avoiding AI cars and surviving longer.
  - Fitness Reward: The genome's fitness is rewarded by how many AI cars it avoids and the length of time the car survives. This reward system encourages the networks to improve their decision-making to avoid collisions and stay alive longer.
  - Fitness Penalty: If the car controlled by a genome collides with another car, the genome's fitness is penalized. This penalty discourages behaviors that lead to crashes.

- **Crossover and Mutation**: After evaluating all genomes in a generation, the top-performing genomes are selected for reproduction. This selection process is based on their fitness scores. The **crossover** technique combines the genetic material (connections and weights) of two parent genomes to produce offspring. Additionally, **mutation** introduces random changes to the offspring's genetic material, such as altering connection weights or adding/removing nodes and connections, to ensure diversity and enable the exploration of new strategies.

- Preserving Topology: One of NEAT's key features is its ability to evolve the topology of neural networks. This means that, in addition to optimizing the weights of connections, NEAT also evolves the structure of the network, allowing for the creation of increasingly complex models over generations. The topologies of networks are represented as directed acyclic graphs (DAGs), which may grow in size over time as new neurons and connections are added.

- Speciation: NEAT divides the population into species based on the genetic similarity of genomes. This process prevents the entire population from converging too quickly on a suboptimal solution by allowing more diverse solutions to evolve simultaneously. Speciation helps maintain diversity and allows the evolution of novel strategies for controlling the cars.

3. Training and Model Saving

The training loop involves the following key steps:

1. Initial Population: A population of genomes is initialized randomly. Each genome is represented by a neural network that controls a car in the game.

2. Evaluation: Each genome is evaluated based on how well it performs in the game. The car controlled by the genome is simulated in the environment, and its fitness is computed based on its survival time and ability to avoid collisions.

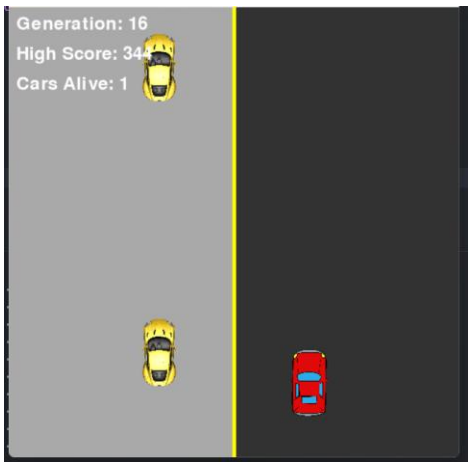

Fig 8: Stable working model (took 16 generations; each 100 genomes to train)

3. Selection and Reproduction: After all the cars have been evaluated, the top-performing genomes are selected for reproduction. They undergo crossover and mutation to generate a new population, which is used in the next generation.

4. Saving the Best Model: After each generation, the best-performing genome (the one that achieved the highest fitness score) is saved to a file (best_model.pkl). This ensures that the best model is preserved for future training sessions, allowing the system to continue evolving from the most successful genome instead of starting from scratch.

5. Loading Saved Models: If a saved model exists, it is loaded at the start of the training process, and the evolution continues from that point. This allows for a continuous improvement of the AI model over multiple training sessions, rather than starting from random neural networks in every session.

6. Termination: The training continues for a predefined number of generations (NUM_GENERATIONS), or until the performance of the models stabilizes. The evolutionary process is designed to gradually improve the driving behavior of the AI cars.

Node ID	Activation Function	Aggregation Function	Bias	Activation Multiplier
1	tanh_activation	sum_aggregation	-0.280	1.0
155	tanh_activation	sum_aggregation	-0.295	1.0
111	tanh_activation	sum_aggregation	-0.279	1.0
41	tanh_activation	sum_aggregation	-1.060	1.0
0	tanh_activation	sum_aggregation	0.324	1.0

Table 1: Layer configuration of the best network

V. Conclusion

This paper highlighted the significance of feature detection in computer vision and explored the challenges faced when applying algorithms like Harris and FAST to video data, particularly in dynamic scenarios such as autonomous driving. By incorporating temporal smoothing, we can achieve greater feature stability across frames, mitigating the instability typically observed in real-time video analysis. Additionally, the integration of NEAT for autonomous vehicle simulation demonstrates the potential of neuroevolutionary algorithms in evolving adaptive and robust systems. The ability of NEAT to evolve both the structure and behavior of neural networks allows autonomous vehicles to navigate complex, dynamic environments effectively. The combination of feature stability techniques and neuroevolution paves the way for more reliable and intelligent autonomous systems, with promising applications in the development of self-driving vehicles and other AI-driven technologies.

Acknowledgment

I extend our sincere gratitude to my mentor, Abrar Ahmed Raza , whose unwavering guidance, expertise, and encouragement were pivotal in the completion of this research. His insightful feedback and commitment to excellence inspired us to approach this study with dedication and rigor. Additionally, I am profoundly grateful to Lovely Professional University for providing the infrastructure, resources, and a stimulating academic environment that facilitated our research. I acknowledge the support from the university's faculty and staff, whose contributions ensured a productive and enriching experience. This opportunity has not only enhanced our technical knowledge but has also strengthened our research and analytical skills, laying a foundation for our future academic and professional endeavors.

Here is the link of project viewer may access: https://github.com/0x0is1/racing-game-with-ai

References

[1] **Stanley, K. O., D'Ambrosio, D. B., & Goo, W. (2009).** A HyperNEAT method for evolving large-scale neural networks for real-time dynamic environments. *IEEE Transactions on Evolutionary Computation*, 13(2), 250-258.

[2] **Walker, A. M., & Stanley, K. O. (2006).** Evolving Agents for 2D Games with Neuroevolution. *Proceedings of the IEEE Symposium on Computational Intelligence in Games*, 74-79.

[3] **Zhao, J., et al. (2019).** Evolving Controllers for Autonomous Racing in 2D Simulations with NEAT. *International Journal of Artificial Intelligence*, 13(3), 134-148.

[4] **Morse, S., et al. (2013).** Evolution of Autonomous Robotic Control Systems with NEAT. *Robotics and Autonomous Systems*, 61(3), 47-58.

[5] **Sprock, D., et al. (2017).** Evolving Drone Control Systems with NEAT for Real-Time Navigation in Dynamic Environments. *IEEE Transactions on Evolutionary Computation*, 21(4), 515-523.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] **Zhao, J., et al. (2020).** Evolving Agents for 2D Racing Games Using NEAT. *Proceedings of the Genetic and Evolutionary Computation onference*, 210-218

[8] P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks, 5:54–65, 1993

[9] K. O. Stanley and R. A. Miikkulainen. Efficient Evolution of Neural Network Topologies. In Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02). Piscataway, NJ: IEEE.

[10] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuron-like adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics, SMC-13:834–846, 1983.

[11] D. Michie and R. A. Chambers. BOXES: An experiment in adaptive control. In E. Dale and D. Michie, editors, Machine Intelligence. Oliver and Boyd, Edinburgh, UK, 1968.

[12] D. E. Moriarty and Risto Miikkulainen. Efficient reinforcement learning through symbiotic evolution. Machine Learning, 22:11–32, 1996.

[13] M. A. Potter and K. A. De Jong. Evolving neural networks with collaborative species. In Proceedings of the 1995 Summer Computer Simulation Conference, 1995

[14] J. C. F. Pujol and R. Poli. Evolving the topology and the weights of neural networks using a dual representation. Special Issue on Evolutionary Learning of the Applied Intelligence Journal, 8(1):73–84, January 1998