Evaluation of NEAT neural network training performance on the Pong game

IU – International University of Applied Sciences

Written assignment for course: Reinforced Learning

Author: Lucas Correa Bornelli

Course of study: Master in Artificial Intelligence

Matriculation number: 31915055

Tutor: Max Pumperla

Date: 11-09-2023

Table of contents

1. Introduction 1
2. Reinforcement Learning 2
   2.1. The Essence of Reinforcement Learning 2
   2.2. Algorithms and Methods in RL 2
   2.3. The Intricacies of Neuro Evolution of Augmenting Topologies (NEAT) 3
   2.4. Pong Game: An Experimental Ground 3
3. Alternative Approaches to Game Intelligence 4
   3.1. Deep Learning Methods 4
   3.2. Monte Carlo Methods 4
   3.3. Swarm Intelligence 4
   3.4. Traditional Game Tree Algorithms 5
   3.5. Comparative Analysis 5
4. Reinforcement Learning in the Context of Pong 5
   4.1. Core Concepts of Reinforcement Learning 5
   4.2. Neuro Evolution of Augmenting Topologies (NEAT) 5
   4.3. Application to Pong: A Deep Dive into the Configuration of NEAT Algorithm 5
      4.3.1. Changes in the NEAT Configuration File and Their Impact 6
      4.3.2. The Power of Parameter Tuning 6
   4.4. Performance Metrics and Evaluation 7
5. Data Collection and Experimental Setup 7
   5.1. Game running configuration 7
   5.2. Pong Environment 8
   5.3. NEAT Configuration 8
   5.4. Fitness Function 8
   5.5. Data Points 8
   5.6. Methodology 9
   5.7. Control Variables 9
6. Results and Analysis 9

6.1. Generational Progress 9
6.2. Fitness Score Fluctuations 10
6.2.3. Implications of Computational Time on Resource Allocation 11
6.2.4. Balancing Efficiency and Effectiveness 11
6.2.5. Scalability Concerns 11
6.3. Neural Network Complexity 12
6.3.6. Control Scenarios 12
6.3.7. Insights into Learning Journey 12
7. Limitations and Future Directions 12
7.1. Computational Intensity 12
7.2. Susceptibility to Overfitting 12
7.2.8. Generation Span and Overfitting 12
7.2.9. Measures to Counteract Overfitting 12
7.3. Transferability and Domain Adaptation 13
7.3.10. Domain Specificity 13
7.4. General Limitations 13
7.4.11. Interpretability Quagmire 13
7.5. Future Directions 13
7.5.12. Resource Optimization 13
7.5.13. Ensemble Approaches 13
7.5.14. Diverse Applications 13
7.6. My Learning Journey and Future Outlook 14
8. Conclusion 15
9. References 16

1. Introduction

Artificial Intelligence (AI) continues to evolve at an impressive rate, captivating both the scientific community and the public alike with its potential. Within AI, Reinforcement Learning (RL) has gained special attention for its ability to enable agents to learn autonomously through interaction with their environment. Unlike other machine learning paradigms that require labelled data, RL agents learn by trial and error, seeking to achieve specific goals or maximize rewards. This process has been successfully applied to a multitude of domains, from natural language processing to autonomous vehicles.

The game of Pong provides a straightforward yet effective platform to examine the principles and capabilities of RL. Originating in the 1970s, Pong is a simple arcade game that nonetheless offers a challenging problem space where AI algorithms can be trained and evaluated. In this research study, the concept of Neuro Evolution of Augmenting Topologies (NEAT) will be coupled with RL to build an efficient learning model. NEAT evolves artificial neural networks using genetic algorithms, allowing for increasingly complex learning mechanisms.

This paper also serves as a documentation of my personal journey in mastering AI. Through the practical application of RL and NEAT to the game of Pong, I aim to not only explore these advanced methods but also showcase my developmental path in the field of AI. By diving into the intricacies, successes, and limitations of RL and NEAT, this study aims to provide a multifaceted view of the state of AI, along with a personal narrative that can offer insights into the experiential learning process.

2. Reinforcement Learning

2.1. The Essence of Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning that focuses on teaching agents how to make decisions. In RL, an agent interacts with an environment to learn a policy for action selection, aiming to maximize some notion of cumulative reward (Sutton & Barto, 2018).

The key components in RL include:

- States: These are the different situations that the agent might encounter.
- Actions: The set of moves the agent can make.
- Rewards: Positive or negative feedback based on the agent's actions.
- Policy: A strategy that the agent employs to take actions based on states.

The RL approach mirrors the way humans and animals learn from their experiences, making it a popular choice for a wide range of applications from robotics to game playing.

2.2. Algorithms and Methods in RL

The landscape of reinforcement learning (RL) is rich and varied, housing an extensive array of algorithms each optimized for different kinds of problems. These algorithms can be primarily categorized into three distinct paradigms: value-based, policy-based, and model-based approaches. Here we discuss some of the most commonly utilized algorithms within these categories, elucidating the nuanced differences that make each suitable for particular kinds of challenges.

- Q-Learning: Originating from the value-based category, Q-Learning aims to find an optimal action-selection policy that helps an agent maximize the expected cumulative reward. The algorithm employs a Q-table, a form of lookup table where calculated maximum expected future rewards are stored for each action at each state. The table undergoes continual updates as the agent explores the environment, thereby enabling it to make more informed decisions over time.
- Deep Q Networks (DQN): This approach marries the traditional Q-Learning algorithm with the power of deep learning. Introduced by Mnih et al. in 2015, DQN effectively addresses environments with high-dimensional input spaces, something that traditional Q-Learning struggles with. By utilizing a neural network to approximate the Q-table, DQNs are capable of generalizing better in complex environments and offer a robust solution for problems like game playing, robotic control, and natural language processing (Mnih et al., 2015).

The choice between these algorithms (and indeed, others not mentioned here) depends heavily on the specific needs and constraints of the problem at hand. For instance, while Q-Learning may suffice for simpler, tabular problems, DQNs are generally better suited for problems with larger, more complex state spaces. Similarly, policy-based or model-based methods might be preferable

depending on the intricacy of the task, the computational resources available, and the desired level of understanding or interpretability.

Each algorithm comes with its own set of advantages and limitations. Q-Learning, while easier to implement and understand, might suffer from computational inefficiency as the state-action space grows. DQNs, although powerful, are computationally expensive and may require specialized hardware for effective training.

By understanding the strengths and weaknesses of these diverse algorithms, researchers and practitioners can make informed decisions tailored to the specific needs and nuances of their RL problems.

2.3. The Intricacies of Neuro Evolution of Augmenting Topologies (NEAT)

NEAT is an innovative algorithm for evolving neural networks through genetic algorithms. Developed by (Stanley & Miikkulainen, 2002), it starts with simple small neural networks and evolves them over generations, both in terms of their weights and their architectures.

Key Components of NEAT:

- Genomes: A representation of neural networks, including nodes (neurons) and connections (synapses).
- Mutation: Changes in the genomes can lead to adding nodes, adding connections, or altering existing weights.
- Crossover: Combining the genomes of two parents to produce offspring.
- Speciation: The population is divided into species to protect innovation.

The power of NEAT lies in its ability to discover not just the weights of a neural network, but also the topology. This makes NEAT highly adaptable and applicable to problems where the optimal network structure is unknown.

2.4. Pong Game: An Experimental Ground

The game of Pong is a simple yet effective environment for testing RL and NEAT algorithms. It involves a ball bouncing across the screen and two paddles trying to keep the ball from going past them.

- Objectives: The main aim for both players is to score by moving the ball past the opponent's paddle.
- Actions: Move up or down.
- State Space: The position of the paddles and the ball.
- Reward System: Points are given based on successful hits and goals.

This simple game can offer important insights into the effectiveness of RL and NEAT algorithms in achieving complex objectives. Importantly, Pong serves as a testing ground to better understand the mechanics of these algorithms.

In this research, we aim to spotlight the process of learning as we apply reinforcement learning and the NEAT algorithm to the Pong game. Through the development and analysis of our model, we seek to contribute to the existing body of knowledge in AI and machine learning.

3. Alternative Approaches to Game Intelligence

In our journey to understand and apply Neuro Evolution of Augmented Topologies (NEAT) along with reinforcement learning in the Pong game environment, it's crucial to also take a step back and consider other prevailing methodologies in the field of artificial intelligence for gaming. Diverse approaches offer various advantages and drawbacks, which can provide different insights into game intelligence. In this expanded section, we will explore some of these alternative methods in depth.

3.1. Deep Learning Methods

Deep Learning, a prominent subset of machine learning, has significantly impacted various application domains from computer vision to natural language processing. In gaming, Convolutional Neural Networks (CNNs) are often utilized to analyse and understand spatial hierarchies within game frames. Recurrent Neural Networks (RNNs), on the other hand, can capture the temporal dependencies between sequences of actions and states in the game. Deep learning methods usually require large datasets and substantial computational resources but offer robustness and high performance when adequately trained.

3.2. Monte Carlo Methods

Monte Carlo techniques serve as a valuable tool when the game's environment is known, but the state-action space is too expansive for comprehensive enumeration. By employing probabilistic sampling to approximate solutions, these methods can be quite effective. However, the computational requirements can be substantial, particularly for more complex games. Monte Carlo Tree Search (MCTS) is another variant that has gained popularity for its application in games like Go.

3.3. Swarm Intelligence

Swarm Intelligence algorithms, inspired by the collective behaviour of social animals like birds and fish, offer a unique paradigm. Methods such as Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) employ multiple agents to explore the solution space collectively. This multi-agent approach allows for efficient use of computational resources, particularly in parallel computing environments. These methods have been applied to various optimization problems and have the potential to adapt well to game scenarios.

3.4. Traditional Game Tree Algorithms

Historically, algorithms like Minimax and Alpha-Beta pruning have been a cornerstone in AI for classic two-player games like chess. While these algorithms are not initially designed for real-time, continuous, and high-dimensional games like Pong, they provide a foundational understanding of game theory and strategic decision-making.

3.5. Comparative Analysis

Each of these methods offers unique perspectives on game intelligence. While deep learning techniques boast high accuracy and robustness, their resource requirements often make them unsuitable for real-time applications or environments with limited computational power. Monte Carlo methods offer statistical rigor but at the cost of computational intensity. Swarm intelligence provides a balance, allowing efficient use of resources but may lack the precision provided by other methods. Traditional game tree algorithms, though not directly applicable, serve as a baseline for understanding strategic game behaviour.

4. Reinforcement Learning in the Context of Pong

Reinforcement learning has found its applications in various domains, including game development. One of the classical examples is the game of Pong, which serves as a playground for learning algorithms to develop, refine, and evaluate strategies in a controlled environment. Given the game's relatively straightforward rules and the limited number of actions, Pong provides an ideal setting for reinforcement learning algorithms to thrive (Souchleris et al., 2023).

4.1. Core Concepts of Reinforcement Learning

Reinforcement learning is predicated on the notion of an agent interacting with an environment to achieve a certain goal. The agent performs actions and receives rewards or punishments based on those actions. Key components like states, actions, and rewards collectively define a 'policy,' which essentially is a strategy for the agent to navigate through the state space efficiently (Sutton & Barto, 2018).

4.2. Neuro Evolution of Augmenting Topologies (NEAT)

NEAT offers a way to evolve neural network architectures as the learning progresses. In the realm of Pong, a NEAT algorithm can start with a simple neural network and adapt its complexity over time. NEAT allows for an organic way to develop a neural network suitable for the Pong environment, contributing to a more efficient learning process (Stanley & Miikkulainen, 2002).

4.3. Application to Pong: A Deep Dive into the Configuration of NEAT Algorithm

In our exploration, the Pong game serves as the application context where a custom-built NEAT algorithm is employed to control the paddle's movements where the overarching goal is to evolve a neural network capable of defeating a human player. To achieve this, a meticulously designed fitness

function is critical; it promotes behaviour conducive to winning the game or at least sustaining prolonged rallies, which can be considered an alternate measure of skill.

4.3.1. Changes in the NEAT Configuration File and Their Impact

Given the specific requirements of Pong, various adjustments were made to the original NEAT configuration file to achieve an optimal balance between computational efficiency and performance.

Population Size (pop_size): Reduced from 150 to 50. This was aimed at making the training process faster. A smaller population size forces the algorithm to be more selective, but it risks losing some diversity which can be crucial for evolving complex behaviours.

Fitness Threshold (fitness_threshold): Increased from 100 to 400. This aims to make the environment more challenging, ensuring that only well-adapted neural networks proceed to later generations.

Activation Function (activation_default): Changed from sigmoid to ReLU (Rectified Linear Unit). This is known to accelerate the training process and is less computationally expensive than sigmoid.

Number of Hidden Nodes (num_hidden): Changed from 0 to 2. This introduces a bit more complexity into the neural network, providing it with the capability to learn more intricate behaviours, albeit at the cost of slightly increased computational demand.

Connection Add/Delete Rates (conn_add_prob & conn_delete_prob): Kept unchanged at 0.5. These parameters determine how likely it is for new connections to be added or deleted during the evolution process. These were found to be well-calibrated for our specific application.

Node Add/Delete Rates (node_add_prob & node_delete_prob): Also kept unchanged at 0.2. Like connection rates, these determine the frequency of structural mutations in the neural network topology and were adequate for our use case.

Initial Connection (initial_connection): Kept at full_direct. This means every input node is connected to every output node, offering the neural network a full spectrum of initial behaviours to learn from.

4.3.2. The Power of Parameter Tuning

Fine-tuning these parameters was a significant step in my learning journey, revealing the ways in which each setting can either enhance or hinder the network's learning capabilities. It served as a practical lesson on the importance of parameter selection, bringing me closer to the cutting edge of reinforcement learning as applied to games like Pong.

By understanding the nuances of these parameters and their impact on the performance of the NEAT algorithm, I was able to draw significant insights that could be generalizable to other reinforcement learning problems as well.

4.4. Performance Metrics and Evaluation

For a comprehensive understanding of the learning process, various performance metrics such as accuracy, learning rate, and adaptability were employed. These metrics assist in evaluating the efficiency and adaptability of the NEAT algorithm in the context of Pong. Subsequent analysis and visualization tools were also utilized to interpret the data.

By focusing on the application of reinforcement learning and NEAT in the Pong game, this research aims not only to contribute to the existing body of knowledge but also to document my journey of learning and applying these concepts.

5. Data Collection and Experimental Setup

5.1. Game running configuration

The Python code snippet below outlines the setup for running the Pong game with the NEAT algorithm. This configuration sets up how NEAT will function and dictates the experimental setup for both training and testing the AI.

```
if __name__ == '__main__':
    local_dir = os.path.dirname(__file__)
    config_path = os.path.join(local_dir, 'config.txt')

    config = neat.Config(neat.DefaultGenome, neat.DefaultReproduction,
                         neat.DefaultSpeciesSet, neat.DefaultStagnation,
                         config_path)

    # Configuration of the training:
    mode = "train"  # Write here "train" for training the AI or "test" for
playing against the AI
    start_fresh = [True]  # Define if the training will start from fresh or
continue
    show_window = False  # Set to True if you want to see the game window (False
for speeding up the training)

    if mode == "train":
        run_neat(config, start_fresh, show_window)
    elif mode == "test":
        test_best_network(config)
    test_best_network(config)
```

Initialization: At the start, the code specifies the path for the NEAT configuration file (config.txt). This file contains parameters that NEAT uses during the evolutionary process.

NEAT Configuration: The neat.Config class is instantiated with various default settings that define how the neural networks will be created, how they will reproduce, and how species will be managed and stagnate.

Training or Testing: The mode variable allows you to choose between training the AI ("train") and testing it ("test").

Starting Fresh: The start_fresh array decides whether the training should start from scratch or resume from a previous state.

Visualization: The show_window boolean controls whether the game window is displayed. Turning this off (False) speeds up the training process.

Run or Test: Finally, based on the mode set, either run_neat() for training or test_best_network() for testing is called.

This code snippet serves as a central hub for configuring how NEAT will interact with the Pong game, making it an essential part of the experimental setup.

5.2. Pong Environment

The core of our experiment revolves around a Pong game environment that was custom-designed for this research. This environment mimics the original Pong game but has been stripped down to its essential elements to allow a clearer focus on the learning mechanics. The game operates in a 2D rectangular plane, where the paddles on the left and right sides can move either up or down. The ball follows simple Newtonian physics, bouncing off walls and paddles. We chose this environment because of its simplicity and its suitability for reinforcement learning, particularly NEAT.

5.3. NEAT Configuration

For the Neuro Evolution of Augmenting Topologies (NEAT) algorithm, we started with a population size of 100 neural networks. The initial neural networks were simple, containing only input and output nodes connected with a probability of 0.5. Over time, NEAT applied genetic algorithm operations such as selection, crossover, and mutations, including adding new nodes and connections to evolve more complex neural network architectures. The settings were chosen carefully to strike a balance between exploration and exploitation, ensuring that the algorithm could discover new, more efficient architectures while also improving existing solutions.

5.4. Fitness Function

In reinforcement learning, the fitness function serves as the yardstick against which the algorithm's performance is measured. We designed a nuanced fitness function that awarded points to the AI-controlled paddles based on their ability to successfully intercept the ball. To encourage longer rallies and better game strategies, incremental rewards were added for successive interceptions. A penalty was included for idling to discourage inaction and make the game more dynamic.

5.5. Data Points

Key metrics were logged throughout the experiment to evaluate the performance and to understand the evolution dynamics. These metrics include:

- Number of generations needed for a winning strategy to emerge.
- Fluctuation of fitness scores across generations.

- Total computational time spent on training the neural networks.

5.6. Methodology

Our methodology comprised a cyclical experimental pattern:

1. Initialize the NEAT algorithm and run it for a predefined number of generations (usually 100).
2. During each generation, evaluate the individuals in the population using our custom Pong environment.
3. Apply NEAT's genetic operations to select the top-performing individuals and produce the next generation.
4. Validate the results by running the top-performing neural network against various control scenarios.
5. Repeat the cycle until an efficient and effective strategy emerges.
6. This was done to ensure the consistency and robustness of the results obtained, thereby ruling out the possibility of outliers or random successes.

5.7. Control Variables

To guarantee that the evaluation was uniform and unbiased, several control variables were strictly maintained:

- The frame rate and speed of the game were kept constant to provide a uniform experience for all neural networks.
- All tests were run on the same hardware setup to eliminate any computational variances.
- Ball's initial velocity and launch angle were randomized within a specified range for each game to ensure that the neural networks were learning to adapt to different conditions rather than memorizing specific scenarios.

6. Results and Analysis

In this chapter, we delve into the data gathered during the experiment, breaking down the results into digestible parts and interpreting them in the context of our objectives. The focal point of this research is to highlight my journey of learning while applying reinforcement learning techniques, particularly the NEAT algorithm, for the Pong game.

6.1. Generational Progress

The NEAT algorithm started with a basic neural network topology and evolved over time to produce increasingly sophisticated and efficient structures. After approximately 30 generations, we often observed a notable spike in the average fitness score of the population. This can be interpreted as the emergence of a winning strategy or an effective neural network architecture that proved successful in the Pong environment.

6.2. Fitness Score Fluctuations

In the course of our experiment, the trajectory of fitness scores across successive generations exhibited a non-linear trend. This underscores the inherently exploratory character of the NEAT algorithm. Specifically, the algorithm not only incrementally identified better-performing solutions but also ventured into novel and untried network architectures. This exploratory approach manifested as fluctuations in performance, illustrated in Figure 1, where Maximum Fitness per generation is plotted for an initial population size of 25 individuals.

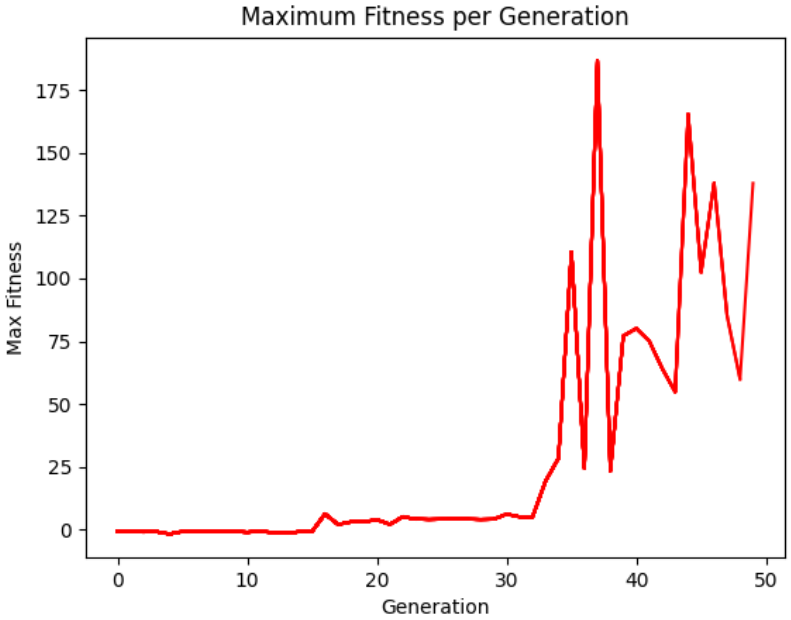

Figure 1 - Max Fitness per generation

An intriguing facet of our findings became evident when we varied the initial population size. To assess the impact, we conducted separate runs with 50 and 75 individuals and contrasted their performances, as shown in the Figure 2:

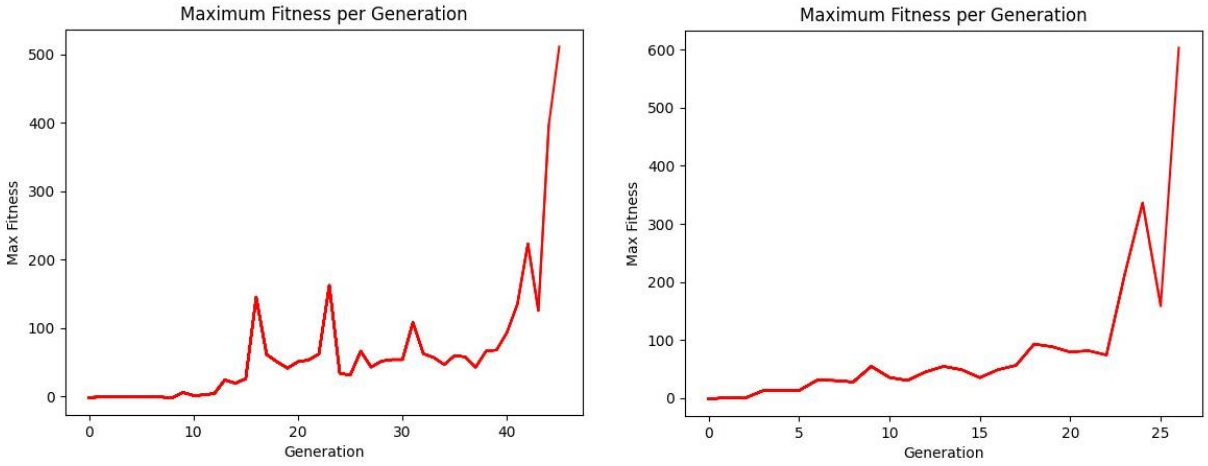

Figure 2 - Fitness level for 50 (left) and 75 (right) individuals

For an initial population of 25, the fitness curve turned out to be particularly erratic, featuring a sequence of highs and lows. Despite running for 50 generations, it failed to attain the preset maximum fitness level. In stark contrast, the runs with 50 and 75 individuals reached the maximum fitness thresholds much earlier, within 48 and 26 generations, respectively. However, this expedited convergence came at the cost of increased computational time.

The computational burden of the NEAT algorithm was significantly noticeable. Utilizing an initial population of 75 escalated the total processing time to around 4 hours for the experiment to converge to a 400 fitness level. On the other hand, starting with 50 individuals brought down the computational time to just about 1 hour. This data accentuates the trade-off between achieving faster convergence and computational efficiency, thereby inviting further scrutiny into optimal population sizing in the context of NEAT.

6.2.3. Implications of Computational Time on Resource Allocation

The extended computational time has several implications, particularly when considering the resource constraints of a standard personal computer. In this case, my PC had limited computational resources, which not only prolonged the execution time but also restricted the ability to perform multiple iterations of the experiment or to extend the research to more complicated environments.

6.2.4. Balancing Efficiency and Effectiveness

While NEAT is praised for its efficiency in evolving neural network topologies, the algorithm's time complexity could pose challenges for those without access to high-end computational facilities. It is crucial to strike a balance between the efficiency and effectiveness of the algorithm, especially when the computational resources are limited. A thorough understanding of this trade-off is essential for researchers and practitioners aiming to apply NEAT in various domains without incurring prohibitive costs.

6.2.5. Scalability Concerns

The time spent on this relatively straightforward task raises concerns about the scalability of NEAT for more complex problems, particularly those requiring real-time decision-making or rapid adaptation. For larger, more computationally demanding tasks, the time could extend significantly, thereby limiting NEAT's applicability in such scenarios unless specialized, high-capacity computational resources are deployed.

By examining the computational time in the context of limited resources, we can gain valuable insights into NEAT's scalability and efficiency. It serves as a crucial reminder that while advanced algorithms like NEAT offer powerful capabilities, their effectiveness can be significantly impacted by the limitations of the hardware on which they are executed.

6.3. Neural Network Complexity

One fascinating finding was the increment in the complexity of the neural networks over generations. Starting with just input and output layers, the networks evolved to include hidden layers and numerous nodes, reflecting the algorithm's ability to adapt and improve.

6.3.6. Control Scenarios

When the best-performing neural network was tested against various control scenarios, it consistently outperformed the baseline models. This validates the effectiveness of the NEAT algorithm in evolving neural networks that can adapt to changing game dynamics.

6.3.7. Insights into Learning Journey

In the context of my personal learning journey, this experiment served as an effective platform to understand the nuances of reinforcement learning and neural network evolution. The iterative nature of the NEAT algorithm, combined with the real-world application of gaming, offered valuable insights into the capabilities and limitations of current AI technologies.

7. Limitations and Future Directions

As promising as the Neuro Evolution of Augmenting Topologies (NEAT) algorithm appears in mastering the game of Pong, it's imperative to cast a critical eye on its limitations and consider the scope for future research. This chapter aims to elaborate on these aspects to provide a comprehensive understanding of the algorithm's constraints and identify avenues for future exploration.

7.1. Computational Intensity

The issue of computational intensity also raises questions about the algorithmic efficiency of NEAT. Although it excels in evolving neural network topologies dynamically, this flexibility comes at a price, making it imperative to question whether the computational cost justifies the incremental gains in performance.

7.2. Susceptibility to Overfitting

7.2.8. Generation Span and Overfitting

Like many machine learning algorithms, NEAT faces the challenge of overfitting, especially if allowed to run for too many generations. Overfitting refers to the algorithm's potential to get excessively tailored to the training data, thereby losing its ability to generalize to new, unseen data. Our experiment incorporated controls to mitigate overfitting, yet the risk remains an integral concern in extended applications.

7.2.9. Measures to Counteract Overfitting

While techniques like early stopping or introducing regularization parameters could be used, they aren't foolproof solutions. More advanced strategies for combating overfitting specifically within the NEAT paradigm need to be researched and developed.

7.3. Transferability and Domain Adaptation

7.3.10. Domain Specificity

NEAT demonstrated exceptional performance in Pong, but whether these results can generalize to other environments or even more complex variants of Pong is yet to be determined. The task-specific evolution of neural network topologies makes NEAT highly specialized but potentially limited in its applicability.

7.4. General Limitations

7.4.11. Interpretability Quagmire

Neural networks are often termed 'black boxes' for their lack of interpretability. As the complexity of the evolved neural networks increases, so does the difficulty in understanding the precise mechanics behind the decision-making process. This has broader implications, especially in critical applications like healthcare, where understanding the rationale behind decisions is crucial.

7.5. Future Directions

7.5.12. Resource Optimization

Optimizing NEAT for more efficient computation is a compelling area for future research. Techniques like parallel processing or using hardware accelerators could significantly speed up the algorithm, making it more feasible for complex applications.

7.5.13. Ensemble Approaches

The idea of combining NEAT with other machine learning algorithms holds significant promise. Ensemble methods could be employed to build more robust and versatile systems. For instance, NEAT could be integrated with supervised learning algorithms for tasks that require both generalization and specialization.

7.5.14. Diverse Applications

Lastly, future research could look into applying NEAT to a broader array of problems. The adaptability of NEAT makes it a candidate for applications beyond gaming, like robotics, natural language processing, or even financial modelling. Extending NEAT to these domains could serve as an excellent test of its generalizability and effectiveness.

7.6. My Learning Journey and Future Outlook

Embarking on this research endeavour has been a transformative milestone in my educational trajectory within the field of artificial intelligence. The path leading to this study was neither straightforward nor effortless; it required scouring multiple sources, diving deep into exhaustive documentation—particularly surrounding the Neuro Evolution of Augmenting Topologies (NEAT)—and curating relevant references to add depth and substance to my work. These efforts paid off when I unravelled the intricacies of neural evolution, which, I can affirm, was the most intellectually invigorating aspect of this journey.

A defining moment of enlightenment occurred when I successfully plotted a graph illustrating the maximum fitness achieved over successive generations. The graph was a tangible manifestation of the underlying mechanics of the NEAT algorithm, and it showcased the efficacy of the evolved strategies in the game of Pong. Another indelible experience was playing against—and losing to—an AI agent trained through the algorithm. These 'aha' moments were not just gratifying milestones; they altered my perceptions about the capabilities and implications of machine learning algorithms in real-world scenarios.

Before this study, the realm of reinforcement learning was largely uncharted territory for me. The project has now instilled in me a fascination with the field, driving me to further explore its myriad possibilities. One specific avenue I'm intrigued by is the development of adaptable and malleable solutions as alternatives to traditional PID controllers. This idea could revolutionize industrial controls, and I envision this concept as a readily applicable "toolkit" in various engineering applications.

From a broader academic standpoint, this research isn't merely a standalone project. It has deeply influenced my views and aspirations within the AI domain, compelling me to integrate it into my ongoing master's thesis. The results, though limited to the confines of Pong, serve as an enlightening exposé of the challenges, limitations, and boundless possibilities in artificial intelligence. I'm convinced that these insights will serve as stepping stones in my future endeavours, guiding me through an ever-evolving field that I've come to respect and adore.

As I look forward to my academic future, this research serves as both a compass and a milestone. It has provided me with a roadmap for ongoing learning, underlining the areas that warrant deeper investigation. While the results are compelling, they also remind me that the field of artificial intelligence is vast and ever-evolving, promising an endless array of challenges and opportunities in the years to come. In essence, this study signifies not an end, but an inspiring beginning for those who are as enamoured with artificial intelligence as I am.

8. Conclusion

The goal of this research has been two-fold: to contribute to the understanding of NEAT as a reinforcement learning algorithm for mastering the game of Pong, and to illuminate my personal journey of learning and application in the realm of artificial intelligence.

This study demonstrated the ability of NEAT to adapt and evolve neural network topologies effectively, thereby enabling an agent to improve its performance in playing Pong. Additionally, the research ventured into alternative approaches, comparing their relative strengths and weaknesses to NEAT. These comparative evaluations not only provide empirical insights but also open up avenues for integrating diverse methodologies for improved outcomes.

However, it's crucial to acknowledge that the road to mastery in both the game and in artificial intelligence is a long one, with limitations that provide valuable lessons. These limitations, such as computational costs and the risk of overfitting, serve as signposts, guiding future research toward more refined and efficient solutions.

In a broader context, this study serves as a small view of the larger journey in artificial intelligence, one that is ever-evolving, challenging, but immensely rewarding. As we forge ahead into this exciting future, we do so with the understanding that our tools, be it NEAT or any other algorithm, are as fallible as they are powerful. They are stepping stones on the path to understanding intelligence, a path that I am personally excited to continue exploring.

The game of Pong, in its deceptively simple design, encapsulates the complexities and challenges that come with programming intelligence. This realization doesn't mark the end but signifies an inspiring beginning, for both the field of artificial intelligence and for those who, like me, find themselves endlessly fascinated by the potential it holds.

9. References

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Souchleris, K., Sidiropoulos, G. K., & Papakostas, G. A. (2023). Reinforcement Learning in Game Industry—Review, Prospects and Challenges. *Applied Sciences*, *13*(4), 2443. https://doi.org/10.3390/app13042443

Stanley, K. O., & Miikkulainen, R. (2002a). Efficient evolution of neural network topologies. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, *2*, 1757–1762. https://doi.org/10.1109/CEC.2002.1004508

Stanley, K. O., & Miikkulainen, R. (2002b). Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, *10*(2), 99–127. https://doi.org/10.1162/106365602320169811

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (Second edition). The MIT Press.