**Research Aim**:

* Investigate how MLAs learn, with a focus on practical applications.

**Simulation Setup**:

* Built a 2D physics simulation using Python, Pygame, and NumPy.

**MLA Integration**:

* Used neat-python library; designed fitness functions; ran model for three days.

**Challenges Faced**:

* Disruptions from PC fans causing a lack of sleep and limited expert guidance within both the physics and machine learning applications.
* Persistent rightward movement of the cart due to simulation inaccuracies.
* Time and performance restrictions, partially due to how the language Python is built.

**Data Logging**:

* Monitored generations’ fitness scores and pendulum angles for adjustments.

**Primary Goal**:

* Understand MLA training and evaluate methods for stabilizing the pendulum.

**Pendulum Model**:

* Governed by the formula for motion dynamics.

**Training Process**:

* Five inputs (angle, angular velocity, position, speed, acceleration) provided to the algorithm which should provide a successful algorithm.

**Performance Monitoring**:

* Fitness score calculated by pendulum angle; higher scores indicate more consistent success.

**Results Overview**:

* Gradual average improvement noted over 780 generations, plateauing at 550 generations.
* Sudden performance increase around generation 370; stagnation by generation 690.
* Compare first attempt to second attempt at training an MLA
* Identify what problems were evident within the MLA, and analyze what caused these issues and potential solutions.

**Conclusion**:

* Simulation design flaws hindered progress; highlights need for refined environments and fitness functions in MLA training.

Machine Learning Algorithms (MLAs) are becoming a central focus of artificial intelligence research, offering powerful solutions for a variety of tasks. My recent interest in AI and MLAs inspired me to explore how these algorithms learn and operate. Specifically, I chose the task of balancing a pendulum on a moving cart in a 2D simulation to understand MLA training on a deeper level. Through this project, I aimed to grasp how MLAs function in practical scenarios and to develop the skills needed to train them for other simple simulations in future applications.

My research aimed to discover how a Machine Learning Algorithm (MLA) learns, with the broader goal of learning how to apply MLAs to general solutions. To achieve this, I built a 2D physics simulation from scratch in Python, using the Pygame library for rendering and the NumPy library for the math. I then integrated an MLA using the neat-python library, designed fitness functions, and ran the model for three days straight. Challenges included the disruptive noise from my PC’s fans and limited expert guidance beyond YouTube videos. A major challenge after training was the persistent movement of the cart to the right without stopping. This could be due to inaccuracies in the formula, environment variables like resistance or gravity, or lack of time to train. I was unable to resolve these issues fully without more extensive testing. I logged data across the simulation, tracking each generation’s fitness score and pendulum angle to assess performance and plan for adjustments. However, the rightward drift continued to limit the MLA’s effectiveness. I never fully resolved this issue, and I assume it was a part of the simulation being innacurate or applying a formula/function wrong instead of a limition of time or resources.

My primary goal was to gain a deeper understanding of how MLAs learn and adapt over time, specifically through applying them to solve a relatively simple task like balancing a pendulum. I had initially aimed to compare and evaluate different training methods to determine which would most effectively teach the MLA to stabilize the system, which I did not have time or resources to do proficiently. Balancing a pendulum was chosen because it is straightforward, fits within the project's time limits, and serves as a fundamental problem that could be expanded to more complex scenarios. Understanding how MLAs learn is vital, as it provides insight into broader machine learning applications, and learning how to implement them directly aligns with my interest in artificial intelligence research.

I chose the simulation of a cart-pendulum because of its simplicity and how commonly it is used in machine learning studies, offering a good balance between complexity and accessibility. Machine Learning Algorithms (MLAs) rely on fitness functions to assess performance; here, the goal was to minimize the pendulum’s deviation from a vertical position by adjusting the cart’s motion.

The core of the physics simulation relied on the formula for a pendulum attached to a moving cart to model the pendulum’s motion accurately, , where is the radius of the pendulum, is joint/air resistance, is the mass, is gravity, is the angle of the pendulum, and is the current x position of the cart. The notation is used to denote the change of over time, so denotes the speed of the cart in the x direction, and denotes the acceleration of the cart in the x direction. I had quickly relalised that I could simplify this formula, as the y-position of the cart never changes. This cancels out the second term, giving the equation . I had decided to implement this change, as calling an additional function every frame is inneficient due to how slow sin is to calcualte. This formula is crucial because it governs the simulation's mechanics, especially the pendulum's response to external forces. However, a persistent bug caused the pendulum to continue rotating unexpectedly, where it wouldn’t switch direction when flipping over vertically. Performance issues also limited the number of MLA models I could run due to Python’s lack of multithreading capabilities and slow execution due to being an interpreted language rather than compiled. The reinforcement learning MLA used trial and error to learn, while a fitness function rewarded successful balance attempts, driving its learning process. Training and testing were essential since this task formed the foundation of my research.

To implement the MLA, I began by defining the core problem: balancing a pendulum on a moving cart within a physics simulation. I used neat-python for its accessible documentation and ease of integration after trying pyneat, which proved too difficult due to a lack of resources. Pygame was selected as the rendering engine because of its well-documented and user-friendly approach to graphical rendering. The cart's movement was determined by an outcome formula, where the result was clamped between -1 and 1 to set the cart’s acceleration. I designed a fitness function based on the pendulum’s angle: , rewarding the MLA for balancing the pendulum as long as possible and deducting points when it failed. Training ran for over 400 generations, with 50 models per generation, each model running for a maximum of 10 seconds. The best-performing models evolved into the next generation. Due to performance issues caused by Python being an interpreted and not compuled language and its lack of multithreading, only one model could run at a time, making the entire process last almost 70 hours, with the simulation running non-stop for three days. This led to challenges such as loud fan noise and disrupted sleep.

The simulation passed five input parameters into the MLA: the pendulum’s angle, its angular velocity, the cart’s position, speed, and acceleration. These variables should have provided enough information for the algorithm to predict the necessary movement to keep the pendulum balanced. While I wasn’t entirely sure how the fitness function influenced the MLA’s evolution, the intention was that it would evolve based on higher scores achieved through better pendulum stabilization.

A black background with text

Description automatically generatedEventually, I had realised that the model was not advancing any further than it already was. This is called stagnation, where the current models have reached a local minimum (*figure 1*). Due to the amount of variables at play, it was very unlikely that I would find a global minimum with the time, resources and training data I had provided. Due to this I had decided to restart and retrain, in which the newly trained models performed much better and developed new skills for much longer.

Testing involved monitoring the fitness score, which increased as the pendulum stabilized, as well as tracking the pendulum’s angular position. A high fitness score and minimal angular deviation from the neutral position signified a successful simulation.

A major challenge was the persistent movement of the cart to the right without stopping. This could be due to inaccuracies in the formula or environment variables like resistance or gravity, which I was unable to resolve fully without more extensive testing. I logged data across the simulation, tracking each generation’s fitness score and pendulum angle to assess performance and plan for adjustments. However, the rightward drift continued to limit the MLA’s effectiveness.

*Figure 1 – Graph displaying simple example of local and global minimums and maximums*

Throughout the simulations, the pendulum exhibited an initial movement followed by a persistent rightward drift. While the MLA occasionally solved the task of balancing the pendulum, it failed when the cart hit the right wall, causing the pendulum to collapse. Retraining with adjusted variables resulted in no significant improvement over an additional 3-day training period.

The second MLA had showed gradual improvement over the 780 generations, but significant progress plateaued around generation 550, still far beyond the first. The algorithm’s fitness scores increased slowly after that, with only minimal gains toward the end. A sudden leap in performance occurred at generation 370, followed by stagnation around generation 690. The second, more successful MLA will be analyzed through the rest of this paper, while the first one will be mostly ignored.

The stagnation of the MLA's fitness scores around generation 550 and the fact it always lead right indicates inherent flaws in the simulation that prevent further improvement. As the algorithm reached this point, it became clear that without addressing the underlying issues within the simulation, progress would remain limited. The fitness function, while crucial for assessing the MLA's performance, may not have accurately reflected the pendulum's balance capabilities in this flawed environment.

Insights gained from this experiment include a deeper understanding of how MLA learning is influenced by both the design of the simulation and the specific parameters used. The rightward drift of the pendulum suggests that the simplified formula might not have captured all necessary dynamics, further impacting the algorithm's effectiveness. Moving forward, adjusting the simulation's mechanics and experimenting with different fitness functions could lead to more favorable outcomes in future projects.

In conclusion, my research highlights the critical role of simulation design in training Machine Learning Algorithms (MLAs). While the MLA demonstrated initial promise in balancing the pendulum, flaws in the simulation limited its potential for improvement. This study underscores the necessity of refining both the simulation environment and fitness functions to enhance learning outcomes.

The insights gained here provide a practical foundation for future research, suggesting that a focus on sophisticated simulations will be essential for effectively training MLAs across various tasks. Moving forward, I plan to address the identified limitations and explore more complex MLAs to broaden their applicability. Ultimately, this work contributes to a deeper understanding of how MLAs learn and the factors influencing their performance, paving the way for future advancements in the field.