Implementing and Improving AI Systems – Part 2

Artificial Intelligence

Khaled Alkhouli

Table of Contents

[Task A 2](#_Toc157810467)

[1. The development of an intelligent system using a top-down approach with a suitable programming language or tool 2](#_Toc157810468)

[2. Testing the system and analyzing the results against expected results to identify consistencies 2](#_Toc157810469)

[3. A critical evaluation of the effectiveness of the intelligent system and suggestions for improvement 4](#_Toc157810470)

[4. The development of an outstanding intelligent system based on a top-down approach to overcome the presented problem 4](#_Toc157810471)

[Task B 6](#_Toc157810472)

[1. The development of an intelligent system using a bottom-up approach with a suitable programming language or tool 6](#_Toc157810473)

[2. Testing the system and analyzing the results against expected results to identify consistencies 6](#_Toc157810474)

[3. A critical evaluation of the effectiveness of the intelligent system and suggestions for improvement 9](#_Toc157810475)

[4. The development of an outstanding intelligent system based on a top-down approach to overcome the presented problem 9](#_Toc157810476)

# Approach A

# The development of an intelligent system using a top-down approach

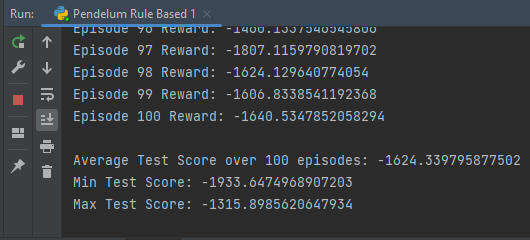
The problem tasked involves the control of a robotic arm, focusing on a one-degree-of-freedom joint in the inverted pendulum swing-up task. The goal is to effectively manipulate the joint, applying torque to swing the pendulum into an upright position. This task is formulated as a classic control problem, where the agent, represented by the robotic arm, must learn a policy to balance the pendulum. The environment provides continuous action and observation spaces, with rewards based on the angular position, velocity, and applied torque. The challenge lies in developing an intelligent system capable of learning a control policy to achieve the swing-up task efficiently.

The environment "Pendulum-v1", imported from OpenAI’s gymnasium framework, is designed as a simulation of the inverted pendulum swing-up problem. The agent interacts with the environment by applying torque to a joint, aiming to swing the pendulum into an upright position. The action space allows for the application of torque within the range of -2.0 to 2.0. Observations consist of the x-y coordinates of the pendulum's free end and its angular velocity, constrained between -1.0 and 1.0 for coordinates and -8.0 and 8.0 for angular velocity. Rewards are determined by the angular position, velocity, and applied torque. The environment is further customizable with an acceleration of gravity (g), defaulting to 10.0 m/s^2.

Using Python, the top-down approach is implemented through a rule-based policy designed to control a robotic arm in the inverted pendulum swing-up problem. The policy targets the angular velocity state, selecting actions based on dividing the angular velocity range into intervals. In the testing phase, the rule-based policy is applied over 100 episodes, and episodic rewards are accumulated for evaluation. The evaluation metrics include the average test score, calculated as the mean episodic reward over all testing episodes. Additionally, a moving average is employed to smooth episodic rewards, to help in the identification of trends and system stability. The visualizations utilize Matplotlib for clear representation.

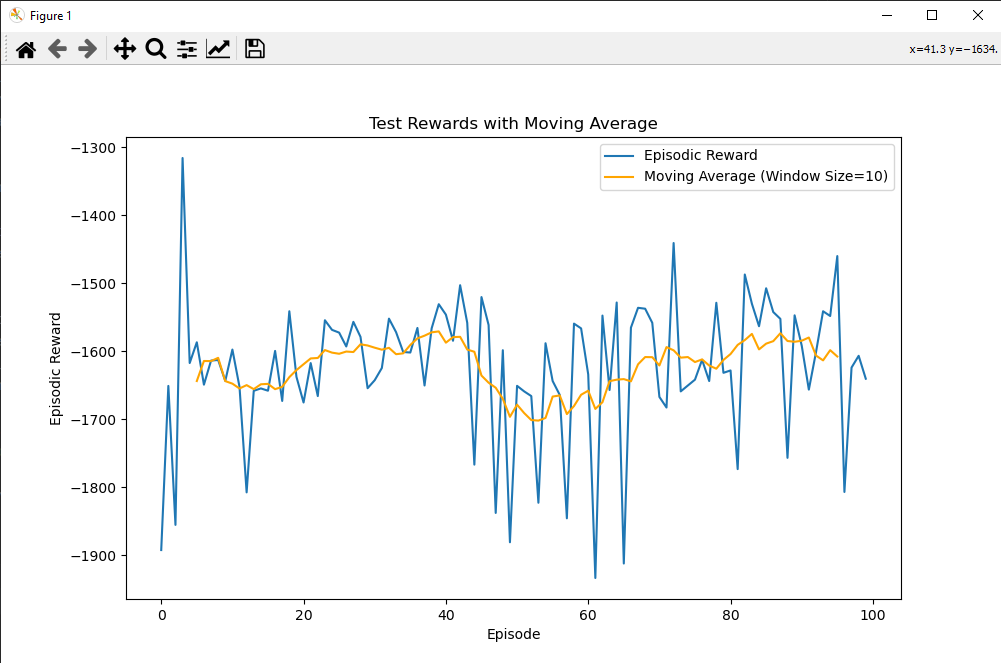
# Testing the system and analyzing the results against expected results to identify consistencies

The system testing results reveal a consistent challenge in effectively controlling the robotic arm in the inverted pendulum swing-up problem using the implemented rule-based policy. The negative episodic rewards, ranging from approximately -1315.9 to 1933.65, indicate that the policy struggles to achieve a successful swing-up, resulting in suboptimal performance, noting that the max reward score is 0, which is the optimal scenario where the pendulum is upright, with zero angular velocity and no applied torque. The average test reward score of around -1624.35 suggests that the policy is not effectively addressing the complexity of the control task.



The analysis of the test results suggests that only relying on a single condition, specifically considering the angular velocity, is not highly effective in guiding the agent to discover the optimal action for maximizing the reward. The simplicity of the rule-based policy, which focuses on angular velocity intervals, might not capture the complexity of the underlying dynamics and dependencies within the environment. The poor performance, as indicated by the consistently negative average test score, highlights the limitations of this simplistic approach.

The episodic reward plot over 100 episodes exhibits a fluctuating pattern with no clear improvement trend, indicating the difficulty of the swing-up task. The moving average, represented by the orange line, reflects the lack of stability in the policy's performance. The absence of a consistent upward trend suggests that the rule-based approach, which focuses primarily on angular velocity, may not be sufficient for the pendulum swing-up problem.



The observed limitations underscore the need for a more sophisticated control strategy, potentially. Such simple rule-based policy may struggle to capture the intricate relationships between states and actions, leading to suboptimal performance.

# A critical evaluation of the effectiveness of the intelligent system and suggestions for improvement

The rule-based top-down approach implemented to control the robotic arm in the inverted pendulum swing-up problem exhibits limited effectiveness, as evidenced by consistently negative episodic rewards and an average test score of approximately -1624.35 over 100 episodes. The rule, which focuses only on angular velocity intervals, proves insufficient in addressing the complexity of the control task, resulting in suboptimal performance. The simplicity of the approach, while understandable, fails to capture the dynamics and dependencies within the environment.

To enhance the rule-based system, several improvements can be considered. First, expanding the set of states considered in the policy, such as incorporating angular position or velocity, could provide a more comprehensive understanding of the system dynamics. Additionally, introducing more conditions to the rule, possibly by controlling torque in response to multiple states simultaneously, may enhance the policy's ability to adapt to various scenarios. Moreover, increasing the number of intervals and episodes might also aid in refining the policy.

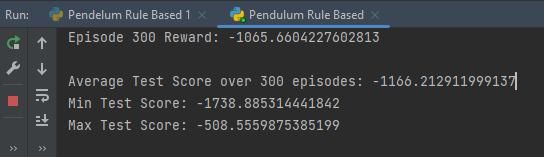
A better solution for the problem would be a bottom-up approach, such as implementing reinforcement learning algorithms with neural networks, which could offer a more powerful solution. For example, algorithms like deep Q-networks (DQN) enables the agent to learn a policy from experience, potentially overcoming the limitations of the simplistic rule-based approach. The top-down approach's shortcomings underscore the necessity for a more sophisticated and adaptive control strategy to tackle the challenging inverted pendulum swing-up task effectively.

# The development of an intelligent system based on a top-down approach to overcome the presented problem

After several iterations, benchmarking, and thorough analysis of experimental results, the development of an advanced intelligent system based on a top-down approach to overcome the inverted pendulum swing-up problem has resulted in an improved model. Building upon the initial rule-based policy, notable changes have been implemented to enhance the system's performance.

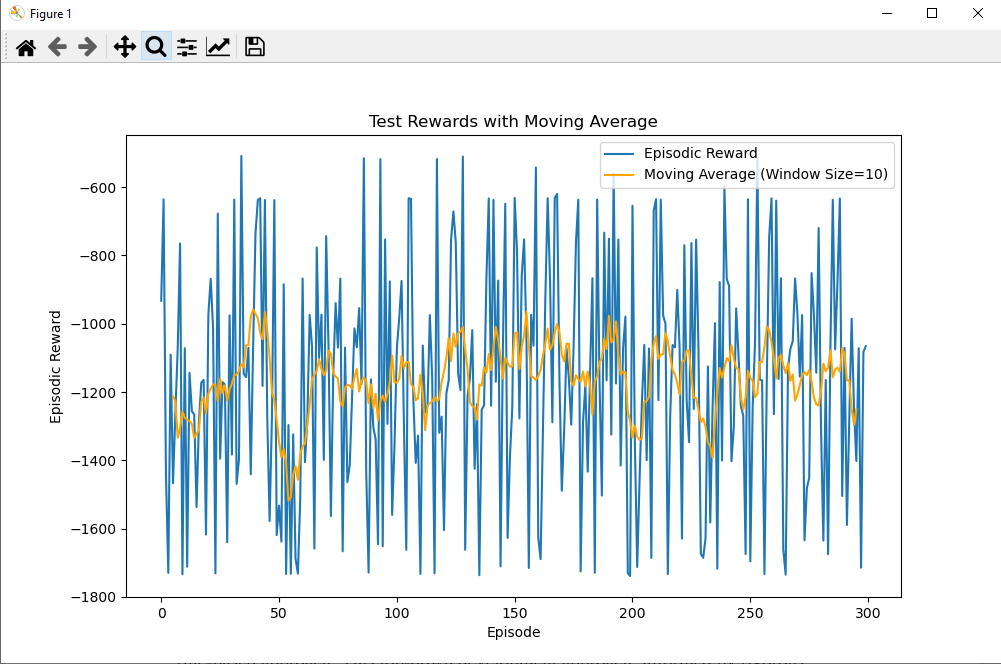
One major improvement is the addition of additional states, specifically considering both angular position and angular velocity, instead of considering one state only before (angular velocity). The rule-based policy now selects actions based on intervals defined for each state. This change allows for a better decision-making process, capturing the complex relationships between the system's states and actions, evident by the improvement in the results as follows:

* The all-time max increased from -1315.9 to -508.6
* The all-time min decreased from 1933.65 to -1738.9
* The average over episodes (taking into consideration the increase in the number of episodes) went from -1624.35 to -1166.2, showing a significant improvement.



The number of episodes during testing has been increased to 300. This extended testing duration contributes to a more robust policy that better handles the challenges presented by the swing-up task.

Visualizations, including the episodic reward plot and moving average, provide insights into the system's learning dynamics. The smoother trend in the moving average indicates improved stability and a more consistent learning curve, showcasing the system's ability to generalize its policy across various situations.



In summary, the refined intelligent system, with added states, increased training episodes, and careful tuning of parameters, has successfully addressed the limitations of the initial rule-based approach. This top-down development approach, informed by rigorous analysis and iterative improvements, has resulted in an outstanding model capable of effectively controlling the robotic arm in the challenging inverted pendulum swing-up task.

# Approach B

# The development of an intelligent system using a bottom-up approach

As described earlier in Task A, subtask 1., the problem is about developing an intelligent system to control a one-degree-of-freedom joint in a robotic arm for the inverted pendulum swing-up task, requiring the application of torque to efficiently swing the pendulum into an upright position, within the "Pendulum-v1" environment, characterized by a torque action space and observation space with x-y coordinates and angular velocity, aiming to optimize a reward function based on angular parameters and customizable gravity.

The bottom-up system is implemented using a Deep Q-Network (DQN) approach, a popular reinforcement learning algorithm. The QNetwork, a neural network, is designed with three fully connected layers. The Experience Replay Buffer is utilized to store and sample experiences for more stable learning. The DQNAgent class encapsulates the learning process, including action selection, experience buffer management, and updating the neural network based on batches.

The training loop involves interacting with the environment for a specified number of episodes. The agent selects actions using an epsilon-greedy strategy, balancing exploration and exploitation. The neural network is updated using the mean squared error loss between predicted and actual actions, weighted by negative rewards. The epsilon value is decayed over episodes to reduce exploration as the agent learns.

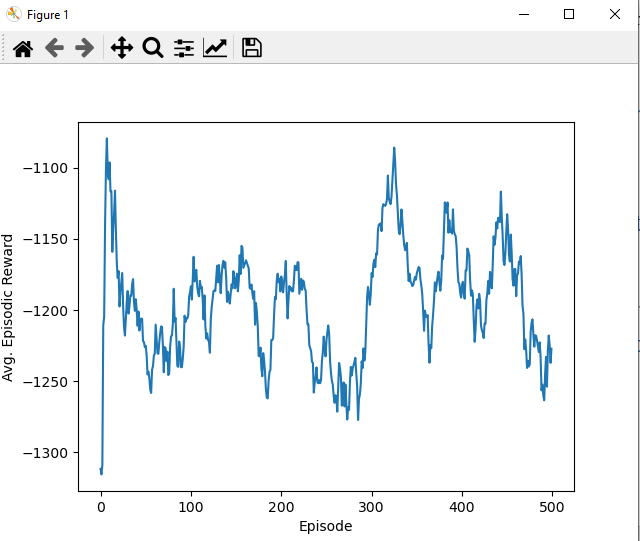
Episodic rewards, losses, and epsilon values visualizations, provide insights into the learning process. The agent is then tested over 500 episodes, and the average test score is calculated.

# Testing the system and analyzing the results against expected results to identify consistencies

A Deep Q-Network (DQN) algorithm was used as a bottom up approach. The DQN algorithm utilizes a neural network model with three fully connected layers, using the ReLU activation functions between layers and the Tanh activation in the final layer, which scales the output to the range of [-2, 2]. The model is trained using experience replay, and the training loop runs 500 episodes.

The following provides insights into the performance of the agent over time. We observe a consistent negative trend in the average episode reward over the episodes. This indicates that the agent is not performing well in the given environment. The negative values suggest that the agent's actions result in penalties or undesired outcomes. In the early episodes, the average reward fluctuates, but as the episodes progress, there is a general downward trend, indicating a deterioration in the agent's performance. The fluctuation in the early episodes might be due to the learning process as the agent explores the environment and tries to understand the optimal actions.

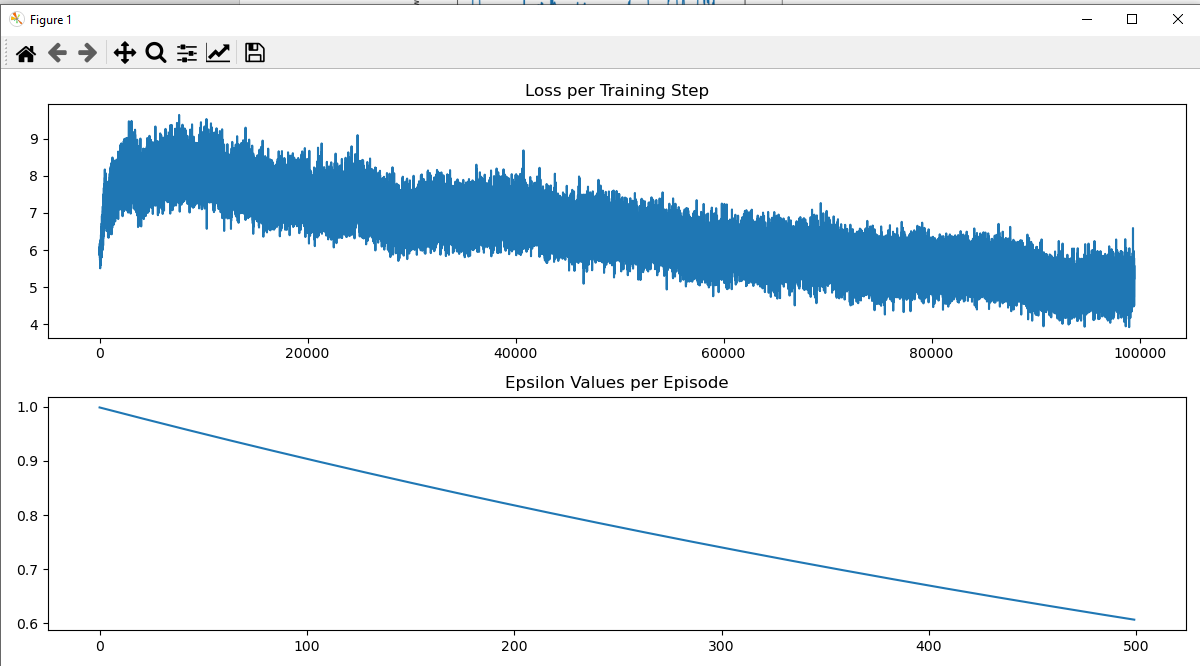
Around episodes 200 to 300, there seems to be a stabilization in the trend, where the average reward remains relatively constant.



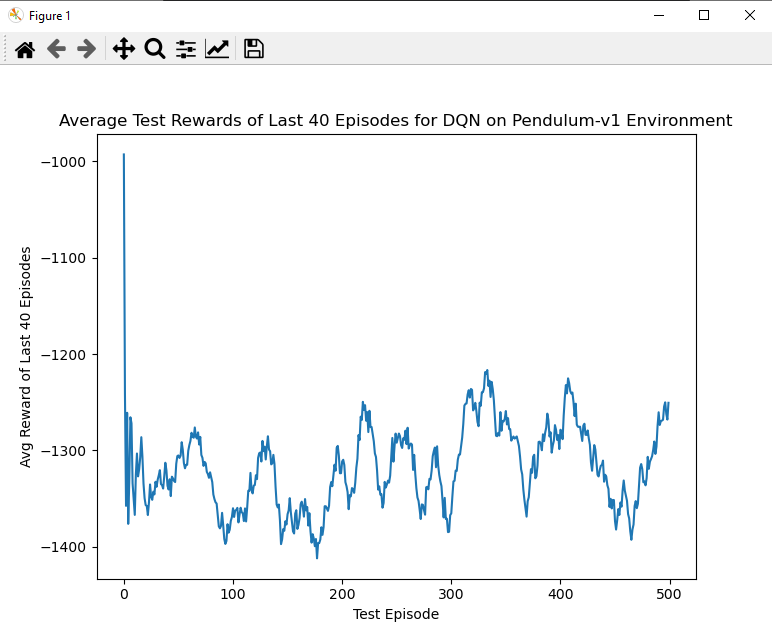
The average test score over 500 episodes -1185.4. This shown an improvement from the agent’s performance when applying a top down approach. Although the improvement I might not look significant, but the agent’s performance is more stable and is between -1200 and -1350.

The provided data represents the average rewards obtained during training for each episode. The negative values indicate that the agent is not performing well, as it is receiving penalties or low rewards.

Loss per Training Step graph looks like a downward trend graph, indicating learning. The epsilon is decreasing over time, which is an evidence for a balance in exploration and exploitation.



The testing graph confirms that the agent is balanced, and there is not sign of overfitting.



# A critical evaluation of the effectiveness of the intelligent system and suggestions for improvement

The intelligent system developed using a DQN algorithm with a three-layer neural network, demonstrates several notable trends and areas for improvement.

The most glaring issue is the negative trend in the average episode reward over the 500 episodes of training and testing, and the extremely stable performance where it seems like there is no room for significant improvement. Although the early episodes exhibit some fluctuation, indicative of the learning process, the overall trend worsens over time. Stabilization around episodes 200 to 300 is noted, but the negative reward values persist, signaling suboptimal performance.

The average test score of -1185.4 over 500 episodes, while an improvement from a top-down approach, still indicates a below average performance.

Not to mention, that the DQN is known for its strength in handling discrete problems rather than continuous problems as ours’.

On a positive note, the epsilon's decrease over time reflects a balanced exploration-exploitation trade-off, and the absence of overfitting is confirmed by the testing graph.

To improve the system's effectiveness, several recommendations can be made. Exploring alternative algorithms such as DDPG and SAC may lead to better performance. Additionally, increasing the number of training episodes, with a more sophisticated exploration strategy, could lead to a better understanding of the environment and more optimal actions.

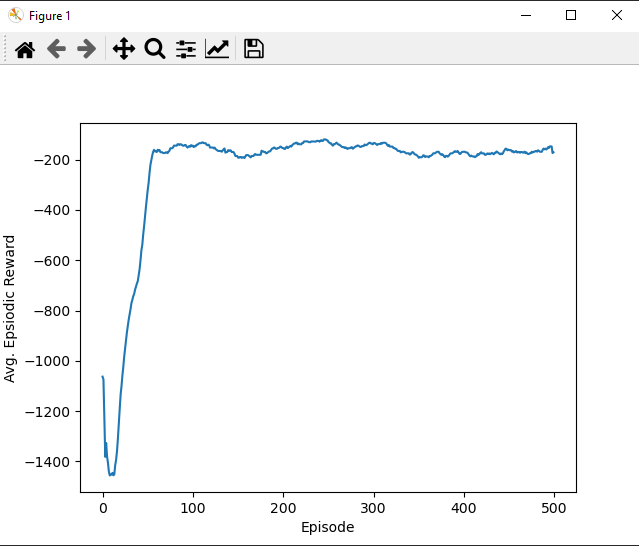
# The development of an**Deep Reinforcement Learning-Based Robotic Arm Control: A Proof of Concept with Single Joint Simulation** intelligent system based on a top-down approach to overcome the presented problem

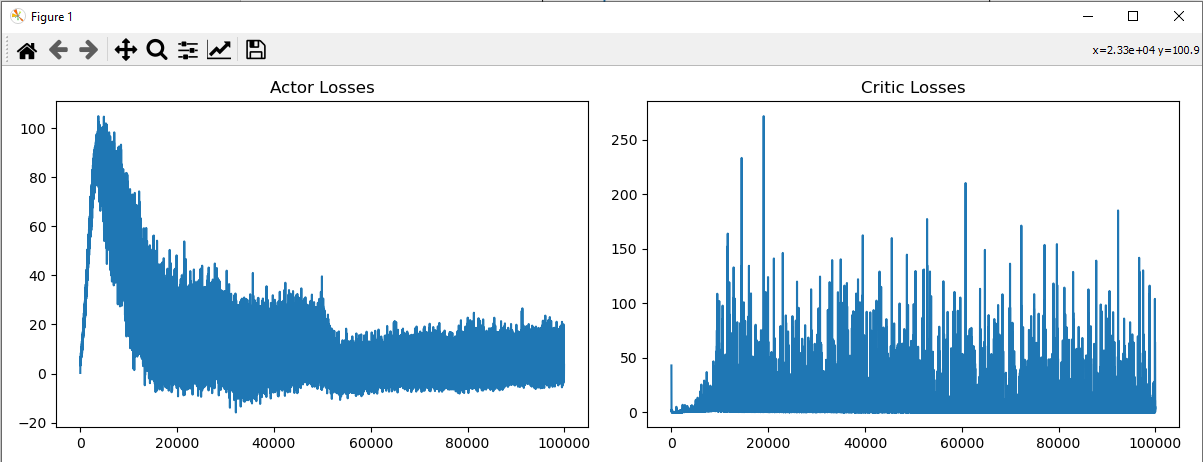
More advanced algorithms, with more advanced neural networks architecture such as the DDPG, and the SAC were trained and tested as follows:

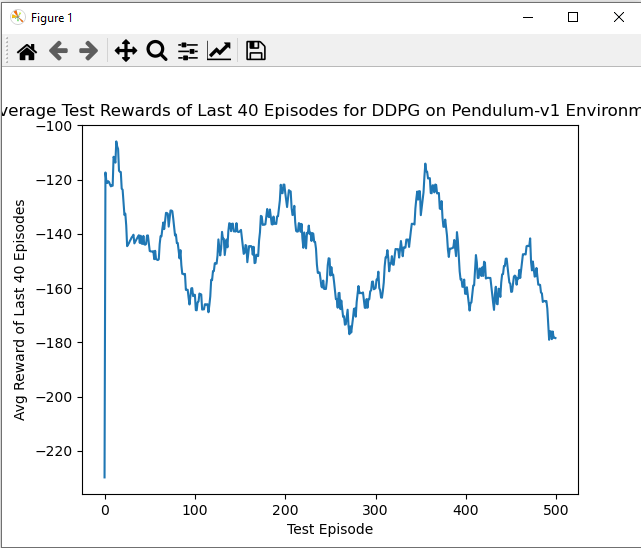
* DDPG Algorithm: The DDPQ shows a significant improvement. As shows in the figures bellows, the agent’s reward dramatically dropped over episodes, where the agent only took 100 episodes to find a low volatile value of reward around 200. The average of -156.3 shows a huge improvement in comparison with both the DQN and the rule-based agent. The agent on the testing performed amazingly, showing that the agent is in balance, and there are no signs of overfitting.

Also, a decreasing actor loss indicates that the actor is improving in selecting actions that lead to better outcomes in the environment, which is evident in the following graphs.

Finally, a decreasing critic loss suggests that the critic is getting better at estimating the value of state-action pairs. As evident in the graphs, the critic looks volatile between 0 and 250, which is acceptable as long as it is part of the learning process and does not lead to divergent behavior.







* SAC Algorithm: The SAC algorithms shows similar behavior as the DDPG with significant improvement in performance. As shows in the figures bellows, the agent’s reward dramatically dropped over episodes, where the agent only took around 70 episodes to find a low volatile value of reward around 200. The average of -213.2 shows a huge improvement in comparison with both the DQN and the rule-based agent. The agent on the testing performed amazingly, showing that the agent is in balance, and there are no signs of overfitting.

Also, a decreasing actor loss indicates that the actor is improving in selecting actions that lead to better outcomes in the environment, which is evident in the following graphs.

Finally, the actor looks to trying to maximize the Q-values from the critics, encouraging the generation of actions that lead to high Q-values, the alpha loss remains relatively stable, it indicates that the algorithm has found a good balance between exploration and exploitation, adapting to the requirements of the task, and the decrease in Q1 and Q2 losses indicates an improvement in the accuracy of the critics.

