Techno-Science Research Journal V13 (2) (2025) P 41-49

Content list available at ITC

Techno-Science Research Journal

Journal Homepage: http://techno-srj.itc.edu.kh/

Techno-Science Research Journal

Tuning Hyperparameters Learning Rate and Gamma in Gym Environment Inverted Pendulum

Nita Chek* , Rothna Pec , Sokchenda Sreng

Department of Telecommunication and Network Engineering, Institute of Technology of Cambodia, Russian Federation Blvd., P.O. Box 86, Phnom Penh, Cambodia

Received: 28 August 2024; Revised: 11 September 2024; Accepted: 19 September 2024; Available online: 30 August 2025

Abstract: *In this paper, reinforcement learning, as a subfield of artificial intelligence and machine learning, is examined based on its capacity for interdependent statistics, optimization, and mathematical concepts and how agents can use trial-error learning to solve tasks. Furthermore, fine-tuning of parameters in machine learning is the ability of scientists to directly influence the architecture, usability, and effectiveness of models in finding the right solution for a particular problem. This paper investigates the impact of learning rate and gamma on the reinforcement algorithm of deep reinforcement learning in a gym environment using inverted pendulum version 4 (inverted pendulumv4) that will acted as agent on Google Collab. As a result, the model's hyperparameter choice significantly impacts the agent's performance in terms of sample efficiency, learning stability, and cumulative rewards, determining how quickly and reliably the agent can learn from interactions, maintain consistent performance across training sessions, and achieve optimal outcomes over time. The careful tuning of these hyperparameters is therefore crucial to maximizing the agent's effectiveness in various environments, as even small adjustments can lead to substantial differences in performance metrics. From experiments, the optimal results were observed with a learning rate of 0.0001 and a Gamma value of 0.99. These settings yielded the highest cumulative rewards and demonstrated effective learning stability comparing to others value.*

Keywords: Reinforcement Learning, Hyperparameters, Reinforce Method, Policy, Agent

1. INTRODUCTION

Adjusting hyperparameters in machine learning allows to fine-tune model performance for optimal results by directly controlling model structure, function, and effectiveness. To ensure the model performance and training efficiency, tuning hyperparameters is a must because it helps optimizing machine learning models , ensuring they perform well, generalize effectively and are computationally efficient. Since machine learning plays a key role in processing complex movement data, a virtual platform are perfect to use and practice all those dataset. Furthermore, the interplay between different hyperparameters can introduce complex dynamics, where the optimal configuration may vary depending on the specific task, data distribution, and computational resources available. Consequently, the process of hyperparameter optimization becomes an integral part of the model development cycle, often requiring a combination of empirical experimentation and automated search techniques to navigate the high-dimensional space of potential settings. In Gym, there are many type of environment that would fit with the reinforcement learning but we will use classic control Mujoco environment that is continuous actions as agent which is called Inverted Pendulum with action space of -3 to 3 and 11 of observation spaces [1]. The environment contain a cart that can moved linearly with left and right attached by a fixed pole at one end and having another end free. Many studies that contain various techniques such as DQN , Policy Gradient , Actor -Critic have been studied to provide the best approach for this environment [2] [3]. Rather than using the existing optimal approach techniques like Bayesian optimization [4] we have used manual search technique to selects and adjusts the hyperparameters. To train the model, the reinforce algorithm has been used and approach with

* Corresponding author: Chek Nita

E-mail: chek.nita@itc.edu.kh; Tel: +855-11 220 112

difference hyperparamters of learning rate and gamma to see how it is optimize its performance and adaptabitity using Python and Pytorch libraries. The findings show that the model's hyperparameter selection has a considerable influence on the agent's performance in terms of sampling efficiency, learning stability, and cumulative rewards. In this research, we propose to apply deep reinforcement learning techniques that enabling agents to solve tasks through trial and error to develop a more effective and efficient approach for analyzing motions by using virtual environment called Gym which is a standard API for reinforcement learning and and a diverse collection of reference environments [5] [6].

The remaining of the is paper is organized as follows. Section 2 outlines the methodology by describing the mathematical modeling and block diagram. Section 3 provides a deep understanding about the system overview by digging into the small elements that going to use in this research. Section 4 concludes the overall of the paper and the result by comparing each hyperparamters depending on their behavior and learning graph curve.

2. METHODOLOGY

In this part, the model and alrogith are going to introduce with some explanation about the process of the implementation including equation. The first part and second part will introduce the algorithm including some explanation about the process and the third and forth part will cover hyperparameter with the environment that are going to use.

2.1 Reinforcement Learning

Reinforcement learning is part of a decades-long trend within artificial intelligence and machine learning toward greater integration with statistics, optimization which it learning what to do, how to map situations to actions so as to maximize a numerical reward signal [7] To put it as simple, Reinforcement learning a framework for solving control tasks by building agents that learn from the environment by interacting with it and received rewards either positive and negative reward. The process of reinforcement learning involves these simple steps: observation of the environment, deciding a way to act the usage of a few strategy, acting accordingly, obtain a reward or penalty, learning form the experiences and refining the strategy and iterate until an optimal strategy is found [8].

As mention, Fig.1 represents the Reinforcement Learning loop outputs a sequence of State (S_t) , Action (A_t) , Reward (R_t) and Next State (S_{t+1}). The agent's purpose is to maximize its cumulative reward, called the expected return. To have the best behavior, the agent aims to learn to take actions that maximize this expected cumulative reward.

Observations/States are the information our agent gets from the environment. There is a differentiation to make between observation and state. State (S_t) is complete description of the state of the world in a fully observed environment with no hidden information. While observation (O) is a partial description of the state.

Action Space is the set of all possible actions in an environment. The action can come from discrete space where the number of possible actions is finite and continuous space where the number of possible actions is infinite.

Reward is fundamental in RL because it is the only feedback for the agent to know if the action taken was good or bad. The cumulative reward at each time step t can be written as:

$$R(\tau) = r_{t+1} + r_{t+2} + r_{t+3} \dots \quad \text{(Eq. 1)}$$

Where τ represent the sequence of states and actions, The cumulative reward equals to the sum of all reward of the sequence which is equivalent to:

$$R(\tau) = \sum_{k=0}^{\infty} r_{t+k+1} \quad \text{(Eq. 2)}$$

In reinforcement learning , tasks are categorized as either episodic or continuing. Episodic tasks have a clear beginning and an end (a terminal state), while continuing tasks proceed indefinitely without a terminal state.Two key strategies in solving RL problems are explorationand exploitation. Exploration involves trying random actions to gather more information about the environment, whereas exploitation leverages existing knowledge to maximize rewards.The policy (π) acts as the agent's decision-making function, determining actions based on the current state. The primary goal in RL is to find the optimal policy (π), which maximizes the expected return for the agent. This optimal policy is typically discovered through training, using various approaches.

$$\pi(a|s) = P[A|s] \quad \text{(Eq. 3)}$$

Value-Based methods: the agent learn a value function that maps a state to the expected value of being at that state. The value of the state is the expected discounted return the agent can get if it starts in that state, and then act according to the policy.

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + .. | S_t = s] \quad \text{(Eq. 4)}$$

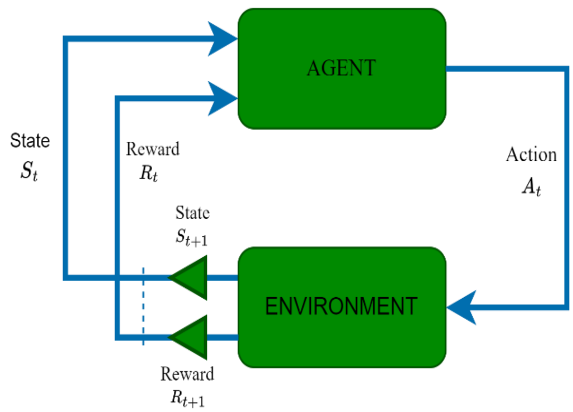

Fig. 1. Reinforcement Learning Process

2.2 Reinforce

Reinforce algorithm belongs to a special class of Reinforcement Learning algorithms called Policy Gradient algorithms which is a simple implementation that would involve creating a policy that a model takes a state as input and generates the probability of taking an action as output. A policy is simply a cheat-sheet or guide for the agent that tells it what action to perform at each stage [9].

The policy is iteratively refined and adjusted at each stage of development, with minor modifications made to improve its effectiveness. This process continues until a policy is crafted that effectively addresses the problem at hand. Typically, the policy is represented by a Neural Network, which is a sophisticated model that processes the current state of the environment as input and generates a probability distribution over possible actions as output. This probability distribution guides the decision-making process, determining which actions are more likely to be chosen in a given state [10]. The ultimate objective of this policy is to maximize the "Expected Reward," which is a measure of the long-term benefits gained from following the policy. This expectation is calculated by considering the rewards accumulated over time, taking into account the probabilities of various outcomes and the desirability of those outcomes according to the problem's goals [11].By continuously refining the policy, the model becomes more adept at making decisions that lead to higher rewards, ultimately solving the problem more efficiently and effectively. The G(θ) represent the expected reward and have the function as follow:

$$G(\theta) = \sum_{i=0}^{n} Probability\ of\ action_k\ at\ state_i * discounted\ reward_{(k,i)}$$

Where, θ = Policy, n = Number of steps in the episode
k = index of action

To implementing the REINFORCE, policy gradient method is a must to build the algorithm by initialize a random policy typically a neural network with random weights, which will determine the actions that the agent takes based on the current state. At this point, the policy is not optimized and will produce actions randomly. Use policy to sample N steps : Run the policy in the environment for N steps. During this process, record the action probabilities generated by the policy, the actions taken, and the rewards received from the environment. This data captures the agent's interaction with the environment. Calculate the discounted reward for each Step: Compute the discounted reward for each step by summing rewards from the current step onwards, adjusting future rewards by a discount factor (γ, where $0 \leq \gamma \leq 1$) [10] [12]. This factor reflects the principle that immediate rewards are more valuable than future ones. Calculate the expected reward G(θ) : Determine the expected reward G(θ) for the entire sequence, which represents the cumulative outcome the agent aims to maximize. This expectation is based on observed rewards and the policy's probabilities over all possible action sequences. Adjust weights of policy to increase G(θ) : Update the policy's weights to maximize the expected reward G(θ). This is done using gradient ascent, where the policy's parameters are adjusted in the direction that increases G(θ) [13]. This iterative adjustment improves the policy's performance by enhancing its ability to select actions that lead to higher rewards.

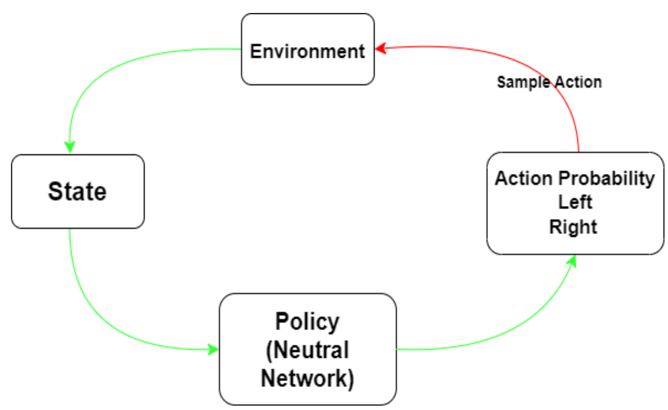

Fig. 2. Reinforce Process

2.3 Hyperparameters

Hyperparameters play a crucial role in machine learning algorithms, serving as essential components for fine-tuning models to achieve optimal performance. These parameters are set prior to the training process and significantly influence the behavior and efficiency of the learning algorithm. Properly selecting and adjusting hyperparameters can make the difference between a model that performs adequately and one that excels at its task [14].The learning rate is a crucial hyperparameter in neural networks, determining the size of the steps the model takes when updating its parameters in response to training errors. It controls the speed of learning by adjusting parameters based on the gradient of the loss function. A high learning rate can cause the model to overshoot the optimal solution, leading to oscillations or divergence, while a low learning rate can result in slow learning and potentially getting stuck in local minima. Finding the right balance is essential for efficient and accurate convergence. Another key hyperparameter, particularly in reinforcement learning, is the discount factor (gamma, γ), which ranges from 0 to 1. It dictates how much future rewards are valued compared to immediate rewards. A gamma close to 0 emphasizes immediate rewards, potentially ignoring long-term gains, while a gamma close to 1 values future rewards almost as much as immediate ones, encouraging long-term strategies. However, a very high gamma can make the learning process difficult or cause the agent to overvalue distant rewards [15].Proper tuning of both the learning rate and discount factor is crucial for effective learning and decision-making in machine learning models.

2.4 Inverted Pendulum

The Inverted Pendulum is a more challenging version of the CartPole problem, commonly used in reinforcement learning and control systems. Both tasks share the goal of balancing a pole on top of a cart by applying forces to the cart, which moves linearly along a track. The objective is to keep the pole upright as the cart moves to counteract destabilizing forces [16].The Inverted Pendulum is more complex due to its continuous action space, allowing the agent to apply forces within a range (typically -3 to 3), requiring precise control over the cart's movement. This contrasts with CartPole's discrete action space, adding complexity to the learning process. As shown in table 1, the observation space consists of four variables: cart position, cart velocity, pole angle, and pole angular velocity where the cart position indicates where the cart is located on the track, cart velocity measures how fast the cart is moving, pole angle represents the tilt of the pole relative to the vertical axis, pole angular velocity captures the rate at which the pole's angle is changing. These describe the system's state and guide the agent's decisions to maintain balance.The reward structure encourages the agent to keep the pole upright, with a maximum reward often capped at 1000. Reaching this reward indicates effective control of the cart and pole.

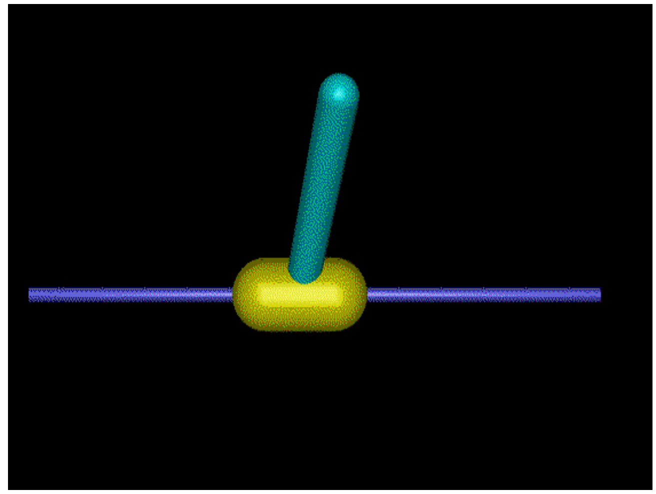

Fig. 3. Inverted Pendulum Agent

This environment is a part of Mujoco environment which contains information as shown in Table 1 below:

Table 1. Inverted Pendulum Agent Elements

Action Space	Continuous [-3,3]
Observation Space	Cart Position, Cart Velocity, Pole Angle, Pole Angular Velocity
Rewards	Max 1000

3. IMPLEMENTATION

3.1 System Overview

As hyperparameter controls the learning process and therefore their values directly impact other parameters of the model such as weights and biases which consequently impacts how well the model performs. The accuracy of any machine learning model is most often improved by fine-tuning these hyperparameters.

The goal for our experiment is not to find the best hyperparameter but to gather insights into how hyperparameters affect our RL algorithm and how we can optimize them effectively. In order to do that we choose to test our reinforce algorithm which is one of the earliest policy gradient methods that optimizes the policy directly or in other words is to trained to maximize the probability of Monte-Carlo returns.

The process are shown in Fig.4, the first step is importing the inverted pendulum data from gym by using computing resource, google colab notebook.

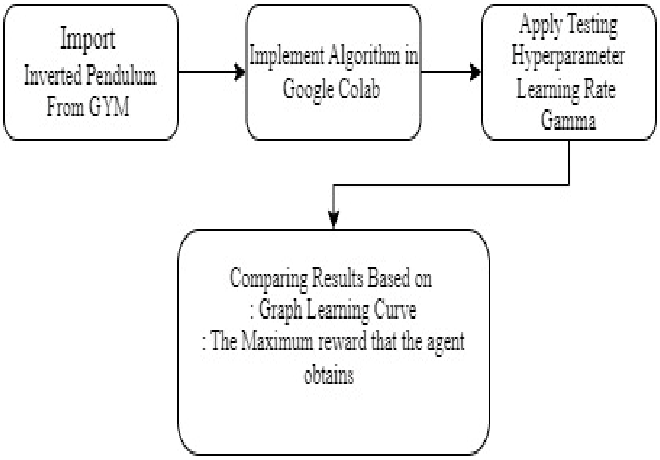

Fig. 4. Implementaition Process

3.2 Policy Network

We built our policy so that agent will learning using reinforce. As shown in Fig.5, a policy is a mapping from the current state of the environment to a probability distribution of actions to be implemented. A neural network is used to parameterize the policy. It is made up of two linear layers that are shared by the expected mean and standard deviation. Furthermore, the mean and standard deviation are estimated using single individual linear layers where it gave output as probability.

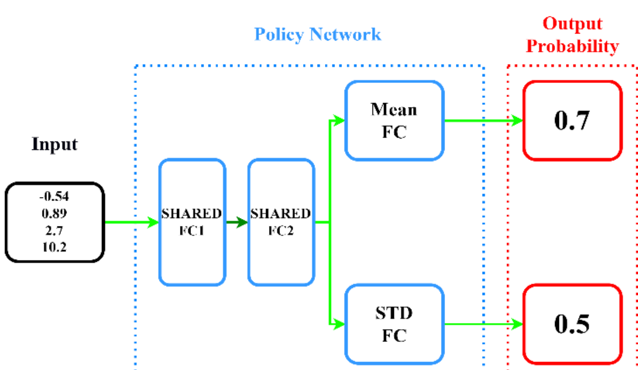

Fig. 5. Policy Network Process

3.3 Agent

By tuning hyperparameters which is include learning rate, gamma, we are going to see the difference how each hyperparameters impact our model. To build our agent, we kind of follow the rule of REINFORCE which is describe as below as in Fig. 6

function REINFORCE
  Initialize θ *arbitratily*
  For each episode $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi\theta$ *do*
    For t = 1 to T – 1 do
      $\theta \leftarrow \theta + \alpha\nabla\theta \log_{10} \pi\theta(s_t, a_t)v_t$
    End for
  End for
  Return θ
End function

Fig. 6. Peudo Code of REINFORCE Implementation

3.4 Hyperparameters for testing

Table 2. Testing Value of Hyperparameters

Hyperparameter	Value
Learning Rate	0.01
	0.001
	0.0001
Gamma	0.70
	0.80
	0.99

The testing phase was conducted across a uniform set of episodes, totaling 5000 episodes, ensuring consistency in evaluating the agent's performance. The specific hyperparameters under examination are detailed in Table 2 above. These hyperparameters were carefully selected to assess their influence on the agent's behavior when utilizing our algorithm. The hyperparameters tested include the learning rate and the gamma (discount factor). The learning rate which governs the speed at which the model updates its parameters based on the error observed during training was varied across three levels: 0.01, 0.001, and 0.0001. Each of these values was chosen to explore how quickly or slowly the agent should adjust its internal parameters to improve its decision-making process. The gamma (γ), or discount factor, was tested at three different levels: 0.70, 0.80, and 0.99. This parameter determines the weight given to future rewards compared to immediate ones. A lower gamma value, such as 0.70, places more emphasis on immediate rewards, while a higher gamma value, like 0.99, indicates a stronger focus on long-term rewards. By systematically testing these combinations of learning rates and gamma values, we aimed to observe how each set of hyperparameters influenced the agent's performance, allowing us to identify the most effective configuration for optimizing the algorithm's results. This thorough approach ensures that the agent can operate efficiently under a variety of conditions, ultimately improving the overall robustness and reliability of the model.

4. RESULT AND DISSCUSSION

The experiment results are shown in a graph where the x-axis represents the number of episodes and the y-axis shows the cumulative reward. The maximum reward possible is 1000, indicating the highest performance level, while the training includes 5000 episodes to ensure thorough learning. The graph features two lines: the Light Blue Graph shows the peak cumulative reward achieved in any episode, reflecting the agent's highest performance, and the Dark Blue Graph represents the average cumulative reward over all episodes, highlighting the agent's overall performance and consistency. Together, these lines provide insights into both the agent's best performance and its average efficiency throughout the training process.

4.1 Learning Rate of 0.01

As depicted in **Fig.7.** , the results indicate that the agent’s performance during training was sub-optimal. The dark blue line, which represents the average cumulative reward, shows that the learning curve reached a maximum of nearly 60 points. However, instead of continuing to improve or stabilize at a higher reward level, the curve dropped back down, indicating a decline in the agent’s performance. This pattern suggests that the agent was not effectively learning from the policy being applied. Ideally, a successful learning curve would show a steady increase in cumulative reward as the agent improves its strategy and better understands how to maximize rewards. However, the drop in the curve indicates that the agent either failed to learn or was unable to retain and apply the knowledge it gained during earlier episodes. This outcome reflects that the training process or the policy itself may need adjustment, as the agent did not demonstrate consistent or meaningful improvement in its ability to perform the task at hand.

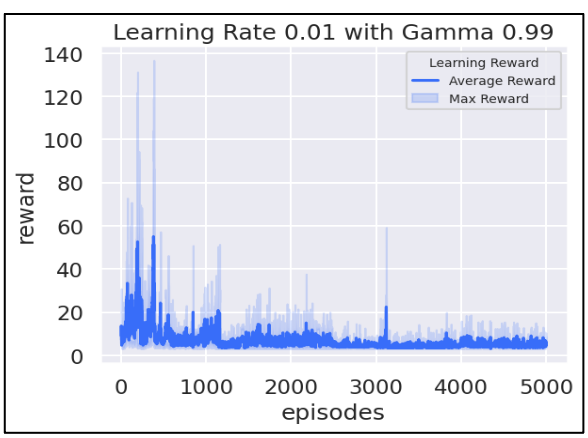

Fig. 7. Agent performance of learning rate 0.01

4.2 Learning Rate of 0.001

Fig. 8 illustrates the agent's performance at various stages of training. During the initial learning phase (first 1000 episodes), the agent shows a significant upward trend in cumulative rewards, indicating effective learning and adaptation. By around 1500 episodes, the agent reaches peak performance, achieving near-optimal outcomes. However, after this peak, the graph shows a sudden decline and instability in the learning curve, suggesting a deterioration in the agent's performance. This instability indicates that while the agent initially learned well, it struggled to maintain and sustain its progress over time.

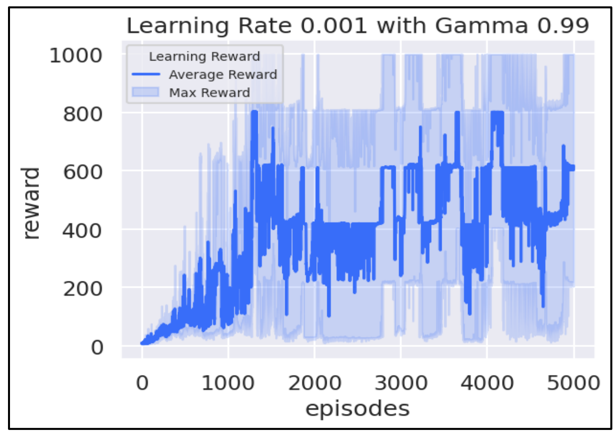

Fig. 8. Agent performance of learning rate 0.001

4.3 Learning Rate of 0.0001

Fig.9 shows the learning rate curve for the inverted-pendulum agent over 5000 episodes. The graph highlights two key metrics: a maximum cumulative reward of 1000 and the total number of episodes used for training. Initially, from episodes 0 to 2000, the learning rate increases slowly, reflecting the agent's early stage of learning. This gradual rise indicates that the agent is cautiously building its understanding. Between episodes 2000 and 3000, the learning rate accelerates, suggesting that the agent's learning has gained momentum and is now leveraging acquired knowledge more effectively. By episode 4000, the curve reaches its peak, representing the highest cumulative reward achieved. This peak marks a significant milestone in the agent’s learning process.Post-episode 4000, the learning rate shows a steady increase, indicating stability and continued improvement in the agent’s learning capabilities. This trend demonstrates the agent's ability to maintain and enhance its performance over time. In summary, Fig 9 illustrates the agent’s development through a gradual increase, followed by rapid growth and sustained improvement, highlighting the effectiveness and stability of its learning process.

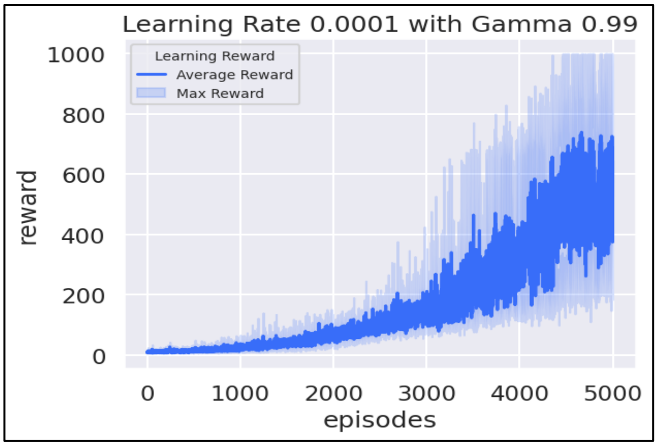

Fig. 9. Agent performance of learning rate 0.0001

4.4 Gamma of 0.70

The discount factor is a crucial hyperparameter in reinforcement learning, influencing how future rewards are valued relative to immediate ones. In the tests, an optimal learning rate of 0.0001 was used, previously identified as the best for the agent. Fig. 10 illustrates how varying the Gamma value impacts the agent's performance over 5000 episodes, showing a gradual increase in cumulative rewards. The agent's performance improves steadily, though at a slow pace, with a maximum reward of around 250 and an average reward near 150. These results indicate that while the agent is learning and improving, the progress is not as rapid or optimal as desired. The findings suggest that further tuning of hyperparameters, especially Gamma, may be necessary to enhance performance and achieve better results.

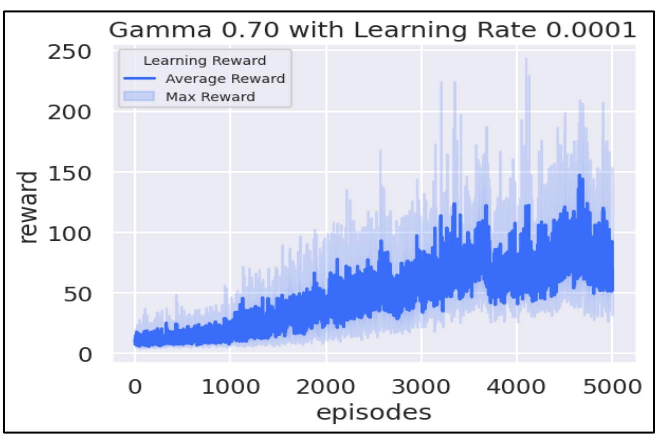

Fig. 10. Agent performance of gamma 0.70

4.5 Gamma of 0.80

As shown in Fig. 11, the agent's performance with a Gamma value of 0.80 demonstrates a gradual increase in cumulative rewards from episode 0 to 5000. The learning curve initially rises slowly, with the agent reaching its peak reward of approximately 250 by episode 4000. Although this peak is a significant milestone, it remains below the potential maximum reward, indicating limitations in the agent's learning. The average reward stabilizes around 150 by episode 4700, reinforcing that while the agent is consistently earning rewards, overall performance is constrained. In summary, the agent's reward performance under a Gamma value of 0.70 reaches only 0.25% of the maximum possible reward, highlighting suboptimal learning progress despite steady improvements.

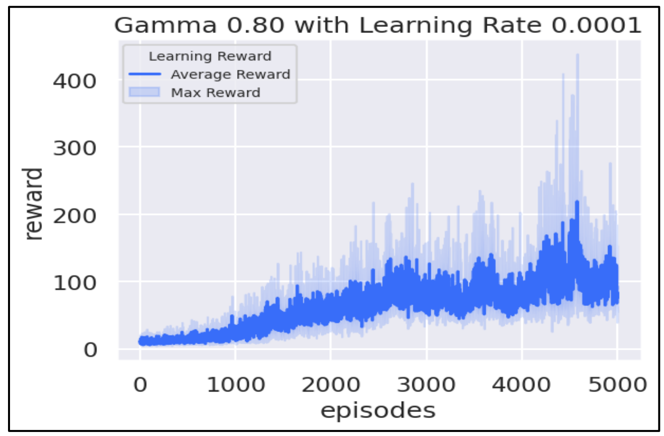

Fig. 11. Agent performance of gamma 0.80

4.6 Gamma of 0.99

In Fig.12, the agent's performance with a Gamma value of 0.99 reveals a significantly more favorable outcome compared to other gamma settings. From the graph, it is evident that the agent demonstrates effective learning, particularly evident between episodes 2000 and 3000. During this period, the learning curve shows a notable improvement, indicating that the agent begins to refine its strategy and optimize its performance. By episode 3000, the agent's performance reaches a crucial turning point. The cumulative reward achieves its maximum value of 1000, representing the highest level of reward that the agent can earn in this environment. This peak demonstrates that the agent has successfully learned to navigate the task efficiently and maximize the rewards available. Moreover, the average reward stabilizes around 700, further underscoring the agent's improvement. This stable average reward, coupled with the rising trend of the cumulative reward, illustrates that the agent not only reaches high rewards but also maintains a consistent performance over time. The learning curve remains steady and continues to rise, reflecting that the agent's learning process is both effective and stable. The stable and increasing trend of the plot highlights that with a Gamma value of 0.99, the agent achieves a high level of learning stability and effectiveness. This setting allows the agent to balance the importance of future rewards more effectively, leading to sustained improvement and a higher overall reward. The results suggest that Gamma of 0.99 is highly conducive to the agent's learning, enabling it to achieve optimal performance and consistently reach maximum rewards.

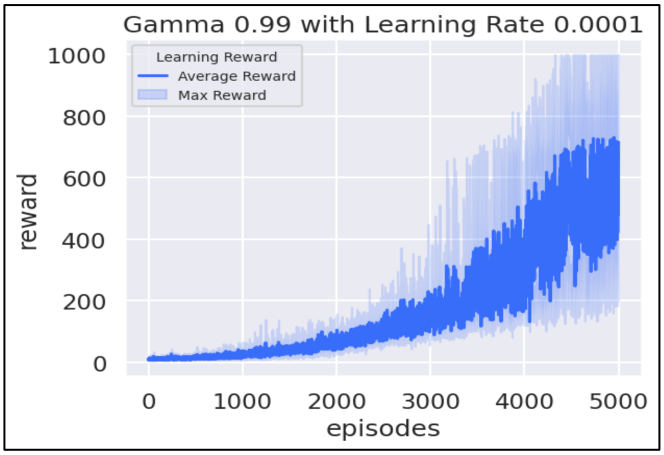

Fig. 12. Agent performance of gamma 0.99

5. CONCLUSIONS

In summary, our comprehensive implementation of the REINFORCE algorithm utilized the free computing resources available through Google Colab, leveraging Python with the PyTorch library for executing and tuning our experiments. We focused on adjusting the hyperparameters, specifically the learning rate and Gamma (discount factor), within the Gym environment's Inverted Pendulum task. Our findings underscore the impact of different learning rates and Gamma values on the model's performance, particularly on learning stability and the total cumulative rewards achieved by the agent. From our experiments, the optimal results were observed with a learning rate of 0.0001 and a Gamma value of 0.99. These settings yielded the highest cumulative rewards and demonstrated effective learning stability. However, it is important to note that these hyperparameters are tailored to the REINFORCE algorithm and may not necessarily be optimal for other models or algorithms. Our current focus was specifically on fine-tuning REINFORCE, and we did not explore other reinforcement learning algorithms within this scope. For future work, we aim to further investigate the effects of varying the learning rate and Gamma across a broader range and to examine additional hyperparameters that could influence model performance. Additionally, we plan to extend our experiments to include other reinforcement learning algorithms to gain a more comprehensive understanding of their behavior and effectiveness in different environments. This expanded exploration will help refine our approach and potentially lead to more robust and generalized results.

ACKNOWLEDGMENTS

This work is a part of research projects in the final year of Artificial Intelligence and Software Engineering (2IA) at IMT Mine Ales , France.

REFERENCES

[1] "Gym Documentation," Gym. [Online]. Available:https://www.gymlibrary.dev/environments/mujoco/inverted_pendulum/. [Accessed: May. 25, 2022].

[2] S.u. Mahmud, "Medium," Medium. [Online]. Available: https://medium.com/dabbler-in-de-stress/the-inverted-pendulum-problem-with-deep-reinforcement-learning-9f149b68c018. [Accessed: May. 28, 2022].

[3] I.Sardor,L.Fu,J.S.,F.Fusco,G.Alibert,C.Raufaste,M. Argentina. "Reinforcement learning approach to control an inverted pendulum: A general framework for educational purposes.," PLOS ONE, Feb 2023.

[4] R. S. Sutton, D. McAllester, S. Singh, Y. Mansour, "Policy Gradient Methods for Reinforcement Learning with Function Approximation," Advances in Neural Information Processing Systems 12. pp. 1057–1063, MIT Press, 2000

[5] Barto., R.S. Sutton and G. Andrew, Reinforcement learning: an introduction, CMR by Westchester Publishing Services, 2018.

[6] B., G., C., V. , "Openai gym," Openai gym. [Online]. Available: https://openai.com/ [Accessed: May. 30, 2022].

[7] J.Wang,Z. Kurth-Nelson ,D.Tirumala ,H. Soyer ,J.Z. Leibo, R.Munos,C.Blundell,D.Kumaran,M.Botvinick, "Learning to reinforcement learn," CoRR, vol. abs/1611.05763, 2016.

[8] A. Plaat, Deep Reinforcement Learning, Springer Singapore, 2022.

[9] K.Arulkumaran, M.P.Deisenroth,M.Brundage, A.A Bharath, "A Brief Survey of Deep Reinforcement Learning," CoRR, vol. abs/1708.05866, 2017.
[10] S. K. Singh, "Hybrid Machine Learning Approach for Predictive Modeling of Complex Systems," 2019.
[11] V. Efimov, "Medium," Medium. [Online]. Available: https://towardsdatascience.com/reinforcement-learning-introduction-and-main-concepts-48ea997c850c.[Accessed: Jan. 25, 2023].
[12] C. Szeoesvari, Algorithms for Reinforcement Learning, Morgan & Claypool, pp.100, 2009.
[13] runai, "runai," [Online]. Available: https://www.run.ai/guides/hyperparameter-tuning. [Accessed: Feb. 11, 2023].
[14] H. Z. Tong Yu, "Hyper-Parameter Optimization: A Review of Algorithms and Applications," 2020.
[15] M. Lapan, Deep Reinforcement Learning Hands-On, Pacjt Publising Ltd., ,pp.52, 2020.
[16] I.G.,Y. B.,A.C., Deep Learning,pp.200-201, MIT Press, 2016.