A study on handling intrinsic motivation for devising sample efficient actor-critic agents

André R. Quadros
Federal University of Pará - UFPA
Belém, Brazil
andre.rosario31@gmail.com

Roberto Xavier Junior
Federal University of Pará - UFPA
Belém, Brazil
rbxjunior@gmail.com

Kleber P. Souza
Federal University of Pará - UFPA
Belém, Brazil
kleber.padovani@gmail.com.br

Bruno D. Gomes
Federal University of Pará - UFPA
Belém, Brazil
brunodgomes@yahoo.com.br

Filipe O. Saraiva
Federal University of Pará - UFPA
Belém, Brazil
saraiva@ufpa.br

Ronnie O. Alves
Vale Institute of Technology - ITV
Belém, Brazil
ronnie.alves@itv.org

***Abstract*—Reinforcement learning has evolved in recent years, and overcoming challenges found in this field. This area, unlike conventional machine learning, does not learn through a set of observational instances, but through interaction with an environment. The sampling efficiency of a reinforcement learning agent is a challenge. That is, how to make an agent learn within an environment with as little interaction as possible. In this work we perform an experimental study on the difficulties to integrate a strategy of intrinsic motivation to an actor-critic agent to improve the sampling efficiency. We found results that point to the effectiveness of the intrinsic motivation as a approach to improve the agent's sampling efficiency, as well as its performance. We share practical guidelines to assist in the implementation of actor-critic agents to deal with sparse reward environments while making use of intrinsic motivation feedback.**

***Index Terms*—Reinforcement Learning, Intrinsic Motivation, Variational AutoEncoder**

I. Introduction

The reinforcement learning(RL) research area has faced several challenges, among them we can mention the difficulty in guiding the agent's learning, as there are some objectives that are difficult to translate into reward signals [1]. When we find environments with a not so sophisticated reward system or whose reward signals are not well defined, it is remarkable the increase in the learning difficulty of an RL agent. The reward system is therefore very important to the agent. Depending on the reward system, the agent can learn to reach their goal faster or not. The reward that the environment returns to the agent when the agent performs an action can be called extrinsic reward [2].

Bearing in mind that the reward system of an environment will not always be ideal, when applying an RL agent it is possible to find environments that have extremely sparse rewards. Sparse rewards can be defined as a series of rewards produced through the interaction of the RL agent with the environment in which most rewards received are not positive [3]. The absence or lack of rewards that would have the purpose of guiding the actions performed by the agent causes a considerable increase in the agent's learning difficulty about the environment. This difficulty is reflected in the agent's ability to achieve their goals or not even manage to achieve such goal.

When using the intrinsic reward in reinforcement learning to help guide an agent, we face some questions about the implementation of this method, one of them is the need for a parameter that regulates the influence of IM on the RL algorithm, which is referred to as Beta (β) [3]. RL algorithms are extremely sensitive to hyperparameter choices [4], so we intend to explore experimentally the use of the IM strategy in the RL agent. In addition, evaluating how to find hyperparameters values, depending on the type of environment, is hard work. And demonstrate how IM influences the algorithm's sampling efficiency and exploration of the environment.

The sampling efficiency of a RL algorithm is the relationship between the number of interactions that an RL agent has with the environment, until it learns to reach its objective [5]. Thus, a better sampling efficiency occurs when our agent wins its environment through the smallest number of possible interactions.

In this work we propose a RL agent model that shows good results by overcoming the presented difficulties, and achieving its objective within the environment in a way that is satisfactory to standard reinforcement learning algorithms. This work also brings the application of a model that is easy to apply. The difficulties encountered in application development, especially in the development and calibration of hyperparameters and the intrinsic reward system are explained in order to guide the community that may be interested in applications of this approach, with explanations on important points, as well as the intrinsic reward regulation. Our codes and notebooks can be accessed via the link https://osf.io/2guhd/?view_only=99f4f351e0b04c0895df990760edc57b.

For a better understanding of this research, it is necessary to go deeper into the concepts of learning by reinforcement and intrinsic motivation. Thus, our next chapter will present the concepts we need to understand, then in chapter III we will present the research methodology, presenting the environment

and its characteristics and we will address the creation of the A2C algorithm along with the IM strategy and in chapter IV we have the results coming from our tests in search of a learning rate and the value of the beta hyperparameter. Finally, we performed the comparison of the RL agent using standard A2C and the A2C using IM and reached the conclusion of our work.

II. Background

A. Reinforcement Learning

Learning through Reinforcement is based on two main lines of reasoning [1], the first comes from psychology, through the study of an individual's learning through trial and error, and the study of animal learning. And the second line, with a computational bias, is the search for the optimal solution in value functions and dynamic programming.

In the classification of model-free algorithms, we have algorithms based on Policy Optimization. The policy of a Reinforcement Learning algorithm is the mathematical function that defines what action the algorithm should take in a given state [6]. These algorithms have strategies that act explicitly on the algorithm's policy. Acting directly in the agent's interest on the environment while the approximation function -of the gradient - remains stable, the biggest disadvantage of policy-based algorithms is their sampling efficiency [7].

Another important function present in reinforcement learning algorithms is the value function [2], represented by Function 1, this is the expected return of the sum of the discount γ for all the rewards r of a state S and action A, starting at a state S_0. It calculates the value of being in a state or of taking an action in a certain state. It is this function that chooses what action to take in an environment, and it's decisions are governed by the algorithm's policy.

$$V_\pi(s) = E_\pi[\sum_{t=0} \gamma r(S_t, A_t)|S_0 = s] \tag{1}$$

B. Advantage Actor-critic (A2C)

The reinforcement learning methods Actor-Critic are algorithms that separate in memory the structures that explicitly represent the algorithm's policy from it's value function [1]. The structure that represents the agent's policy is known as the Actor [6] because it performs the actions in the environment, and the value function is the Critic, acting as critical of the actions taken by the actor [1]. These structures act in different ways, each with a neural network independent of the other.

The Actor-Critic algorithm strategy is applied to some RL algorithms such as DDPG and SAC. In 2016 a new Actor-Critic strategy emerged. Called Asynchronous Advantage Actor-Critic (A3C) and also Synchronous Advantage Actor-Critic(A2C) [7].

The A3C strategy presented an algorithm that uses several agents at the same time, these are called workers [7]. Workers interact in parallel in different environments. The interaction with the environments returns the learning of each agent about their environment. Such interaction is intended to update the policy and value function of a single central agent called the Global Network. The fact that A3C runs several workers in different environments requires a high computational processing.

A2C is defined as having only one worker that interacts with a single environment. Therefore, it is an easy-to-implement option. The main feature of A2C and what sets it apart from traditional Actor-Critic methods is Advantage Function. The Advantage function present in A2C is a function that acts in an attempt to estimate how good the result of an action taken was better than what the agent expected. The advantage function helps the agent to define the quality of actions, being an important advance of the A2C algorithm.

Based on this, we can state that the A3C algorithm and its A2C derivation are an important RL strategy, and may be capable of good results in certain environments, as examples of these results in Atari environments, the Labyrinth [9].

C. Variational AutoEncoder

AutoEncoders are neural networks developed to create a representation of information [9], it is composed of an input data, a latent space and an output data. The AutoEncoder has two phases in it's operation, the encoding phase, which creates a representation of the input data. Then we have the decoding phase, which in turn decodes the data representation and recreates it. After the encoding phase, the AutoEncoder creates a representation of the data that is stored in it and is called latent space.

The VAE(Variational AutoEncoder) is a deep generative model, which also has encoding and decoding phases [9]. It uses deep learning techniques and probability distribution to generate it's own information. During the VAE coding phase, it is where the probability distribution used to create the information representation occurs. The distribution of probabilities is regulated during a training phase that must take place previously, ensuring that the latent space that has been coded has good properties. Such well-calibrated properties will allow the construction of new data. VAE therefore uses the probability distribution to transform information into a representation of it and then allows the information to be reconstructed. Concurrently with this process, a Loss Function called ELBO is applied. This function in the area of neural networks is used so that the model can measure how much the result was different from what was expected, therefore, the smaller the loss of information in the process, the greater the precision in the assembly of the original information.

Equation 3 shows the loss function of the VAE, where the first term is the reconstruction loss and the second encourages approximation of the probability distribution. In variational inference we try to estimate the posterior distribution of $p(Z|S)$, where S is the input data that generates a latent space $Z_1, \ldots, Z_n$. Because the probability distribution leads us to intractable integrals, we can approximate the $q(Z)$ distributions. The KL divergence (Equation 2) is the entropy between the anterior distribution and the posterior distribution, which is the second term present in our Equation 3.

$$r_{intrinsic} = KL((p(Z|S)||p(Z))) \quad (2)$$

$$L(\theta, \phi) = E_{q_\phi(Z|S)}[\log p_\theta(S|Z)] - KL(q_\phi(Z|S)||p(Z)) \quad (3)$$

Since the VAE is used, it's loss function is as important as the encoding and decoding, thus informing how faithful the decoding result is to the input that occurred in the VAE encoding.

D. Intrinsic Motivation

As an original term in psychology, intrinsic motivation refers to the intentions and motivations coming from the individual, which leads him to naturally seek novelties and challenges, regardless of external pressures [10]. It is an area based on studies carried out from the development of learning in babies [11], and more broadly, organisms tend to explore the space around them, as well as learn new skills. As mentioned earlier, intrinsic motivation seeks to study and assess the natural stimulus of an individual/agent's curiosity to explore an environment/problem, acquiring new experiences and learning new skills, without explicitly aiming to achieve a goal, but rather achieving it as a consequence.

In Reinforcement Learning, we have some strategies that try to recreate IM concepts [2] in learning agents. It is possible to classify the intrinsic motivation in RL into two types [2]: The first one is the knowledge acquisition, being a type of model application aimed at the RL agent to learn something in its environment, in this type of intrinsic motivation, the models increase the agent's efficiency in exploring the environment, and improve the abstraction of states. The following classification is called skill learning, aimed at allowing the agent to learn to abstract skills in order to achieve a goal or to be able to build a catalog of skills whose agent can access it and use these skills to overcome environments . Skill learning models are more focused on overcoming obstacles by skills. In this work we chose to focus on knowledge acquisition, due to its ability to explore little visited states in RL environments.

The use of intrinsic motivation in learning by reinforcement conceives the agent's own generation of stimuli, improving exploration and reducing dependence on stimuli coming only from the environment. Thus, the concept of extrinsic reward arises, which is the reward coming from the external environment, and the intrinsic reward which, unlike the extrinsic reward, are the rewards generated by the individual itself, internal stimuli of the organism that guide it to do something, that stimulates exploration, learning and the like.

Intrinsic motivation models classified as knowledge acquisition are able to improve agent performance in sparse environments [4]. The following topic reports the different types of environments that agents can encounter, as well as their characteristics.

E. Environments

In Actor-Critic Algorithm, the one who makes the decisions and who learns is called the agent [1], and what the agent interacts with and everything that is outside the agent is the environment. Environments can exist in many different ways, from text-based environments that run on the [12] command prompt, to those based on games like Atari [13], robotic movements in simulators and many others.

Environments have different characteristics that can make learning difficult or not. One of these features classifies environments into fully observable and partially observable. In fully observable environments, the agent receives complete information about the environment in which it finds itself, thus, right after taking a certain action, the agent receives information from a complete description of how the environment behaved as a whole. In partially observable environments we have a limitation on how much of the environment the agent has access to. After taking action by the agent, he receives only a part of the information external to it, not being able to know all the consequences that its action caused.

Another feature that RL environments have is their response to a certain action, there are two types of environments for this feature. The first is the deterministic environment, it is predictable, being static, making it easier for the agent to learn about the environment. The second type is called stochastic because of its greater randomness and unpredictability, thus making learning more arduous since the same actions and states can return different results.

Rewards are another extremely important feature in environments, they guide the agent in learning, some environments have well-defined reward systems, and every action taken returns a positive value as a reward, a negative value as a punishment or a null reward, guiding the actions of the agent until it reaches its goal within the environment. On the other hand, there are environments with a system of so-called sparse rewards. In sparse reward environments not every action immediately returns a positive reward to the agent. These immediate stimuli would confirm to the agent whether the action taken was good or not, and also quantified the quality of the action. Thus, rewards are given after several actions are taken, and there may still be times when these rewards, in addition to being sparse, are of small value, thus generating little stimulus to the agent and making the learning more difficult.

Each environment having its own difficulties, we see the need for methods and strategies that can be applied to Reinforcement Learning agents, helping them to overcome sparse environments and improving the agents' performance in the environments in which they are found.

III. Research Methodology

A. Choice of Environments

As mentioned earlier, there are a few types of environments, and each one has its own characteristics. This work follows a

specific line of investigation, which is to work with RL applications in environments with sparse and partially observable rewards.

One of the environments that follow the pattern created by OpenAi Gym is the set of environments called MiniGrid. This environment was developed by researchers with the intention of being an environment that has sparse reward environments and that requires the development of skills, and has a low processing cost, allowing them to be run on simpler computers. Grid-based and designed to be lightweight from the point of view of computer processing [14].

The Minigrid has two environments with sparse and partially observable rewards that were used in this work, the MultiRoom and the Door-Key. The MultiRoom represented in Fig. 1 is characterized by having, within a grid of 25x25 spaces, 3 rooms with doors between them and which must be opened so that the agent can travel through the environment until finding the arrival point, in this environment the point of arrival is the grid in green color while the agent is represented by a red triangle. The number of MultiRoom rooms and their size are configurable. The second environment used in this work and which is represented in Fig. 2 is the Door-Key, this environment divides the total space of the grid into two rooms with a door between them, the agent appears in one room and its objective is in the other. The room in which the agent appears there is always a key somewhere, where the agent must take it, carry it to the door and then open the door, thus being able to enter the room where his objective is, also characterized by a green grid.

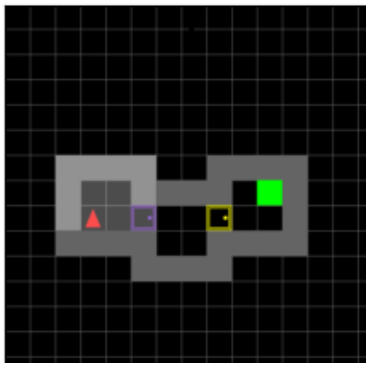

Fig. 1. Rendering of the MultiRoom environment with 3 rooms of size 4.

The MultiRoom and Door-Key environments are episodic and have some limitations in order to test the agents' learning efficiency. Each environment limits the number of actions the agent can take in each episode, so the agent doesn't have infinite actions to find the correct actions. And also the number of actions that the agent performs in each episode influences the reward that the agent will receive.

In this work, the MultiRoom environment was chosen with the configuration of 3 rooms and size of 4 grids each room, these configurations of the number of rooms and their size are parameterized in the creation of the environment. The reward system of this environment is directly influenced by its configuration. Therefore, the MultiRoom environment performs the following calculation to define the total allowed actions in each episode, it adds the number of rooms and multiplies by 20. Therefore, in our current environment configuration, we have 60 timesteps available in each episode.

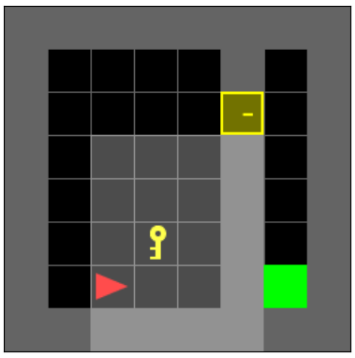

Fig. 2. Rendering the DoorKey-8x8 environment.

In the DoorKey environment, it was used in its 8x8 configuration and therefore allows the taking of actions for each episode by calculating the amount of grids configured multiplied by 10, in the case of the 8x8 configuration where the amount of grids in the environment is 64 , by multiplying it by 10 we have the amount of timesteps possible by the agent. Thus, the DoorKey environment allows us 640 actions per episode.

Both environments used in this research have the same set of possible actions, whose RL agent will use a number from 0 to 6 to choose the action, each action and its respective description are listed below:

- 0, turn left;
- 1, turn right;
- 2, move forward;
- 3, pick up an object;
- 4, drop the object being carried;
- 5, toggle (open doors, interact with objects);
- 6, done (task completed, optional).

The environments used in this work for the tests are of sparse rewards, as the agent, when taking actions during the episode, receives a 0 value reward, unless he manages to reach the objective represented in the green grid. The environment also penalizes the agent according to its delay in reaching the objective. This penalty happens because the reward generated by the environment when it reaches the goal is calculated according to Equation 4.

$$Reward = 1 - 0.9 * \frac{countsteps}{maxsteps} \tag{4}$$

Both environments are also partially observable, the agent has only a partial view of the environment, for its actions, in Fig. 1 and Fig. 2 we can see an area of maximum dimension 7x7 in gray tone standing out in front of the agent, this is the representation of the view that the agent is having in the environment. With this, comes the need for a strategy that can

help the agent to complete the objective of these environments, since the environment provides few stimuli that guide the agent towards its conclusion.

B. Algorithm Creation

To continue the investigations, it is necessary to create an RL algorithm that implements an intrinsic motivation methodology, using the intrinsic reward that overcomes the lack of rewards found in the chosen environments. Our research focuses on sparse reward environments, bringing to the investigation a solution that is more difficult for the standard RL agent to find. To overcome this difficulty, a strategy that helps the agent in these environments is necessary. The fact that IM helps the agent in the exploration of environments with their own stimuli made us choose this strategy, thus providing us with an algorithm that can make the RL agent generate its own stimuli, given the scarcity of these in the environment.

To meet the need for an algorithm that allows the agent to generate its own stimulus through intrinsic reward, we adopted the strategy that uses the Variational State as Intrinsic Reward (VSIMR) [15]. The strategy proposes to measure the distance between the probability distribution of a state S in relation to its latent representation Z, this distance is used as an intrinsic reward as we can see in Equation 2, where the Divergence KL is used to calculate this distance. The model tries to approximate the probability distribution of the latent space Z $p(Z|S)$, for that it uses variational distribution to find the probability of the posterior distribution. For this, it uses the VAE that receives an input S and creates a latent space Z. With that, the latent space represents the variations of the patterns in the state.

Creating an algorithm that follows this strategy requires an RL algorithm that has a Value Function and a VAE structure. In this work, we chose to use an A2C agent, which runs one episode at a time and not continuously. And a VAE structure was developed, capable of receiving the states of the environment as input to encode and then decode them. With the development of these, we were then able to apply the VSIMR strategy, like the algorithm shown in adapted Algorithm 1 [15]. In Fig. 3 we can visualize a graphical representation of the strategy.

Algorithm 1 Intrinsically motivated training loop for A2C.

for *Episode=0,1,2,....* **do**
  Initialize the dataset D and insert s_0 in D;;
  for *t=0,1,2...T* **do**
    Take an action and watch the next state s_{t+1} and the extrinsic reward $r_{extrinsic(st+1)}$;
    Compute $r_{intrinsic(st+1)}$ (st+1) = KL Divergence ;
    Store tuple (s_{t+1}, a_t,$r_{extrinsic(st+1)}$, $r_{intrinsic(st+1)}$) in D ;
    if *mod(t,N)* **then**
      Train the Actor and the Critic with the return Gt=$\Sigma_t r_{extrinsic(st)} + \beta KL - Divergence$;
      Train VAE with the states s collected in D ;
      Initialize the dataset D and insert s_t in D ;
    end
  end
end

In implementing the algorithm, we developed the A2C agent, which interacts with the MultiRoom and DoorKey environments, then we implemented the VAE and incorporated it into the A2C algorithm.

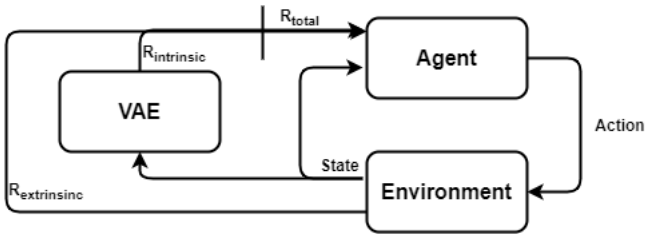

Fig. 3. Representation of the VSIMR strategy.

The implemented strategy starts the episode and starts a dataset D. This dataset is where data will be stored to feed the VAE training. On episode launch the initial state, called S_0 of the environment, is inserted into the dataset, so the agent can start taking actions within the environment. In this phase, the agent takes an action based on its π policy and finds itself in a state, called here S_t. After taking an action, the environment returns to the agent a state resulting from its action, here called S_{t+1}. The extrinsic reward of the state S_{t+1}. After the agent performs an action, it is time to compute the value of the intrinsic reward, sending the S_{t+1} to the VAE, which in turn encodes and decodes the S_{t+1} and applies the function of loss to compare the VAE input and output, one of the components of this loss function is the KL Divergence, which will be used by the agent as an intrinsic reward. To facilitate the application of the KL divergence as an intrinsic reward we use a Gaussian distribution to normalize the result of the KL divergence. After the calculation of the intrinsic reward, a tuple is stored, consisting of S_{t+1}. The action that the agent took, the extrinsic reward and the intrinsic reward in the D dataset. During the Actor and Critic training phase, both are trained with the return of the sum of the states' extrinsic

rewards plus intrinsic reward. VAE trains using the *dataset* D, after training the *dataset* is reset and S_t is inserted into D.

$$r_{total} = r_{extrinsic} + \beta r_{intrinsic} \tag{5}$$

The hyperparameter β is inserted during the calculation of rewards shown in Equation 5. It acts as a regulator for the influence that intrinsic motivation will have on the algorithm. The β is of paramount importance for this intrinsic motivation model, as it should not be too small a value to not influence the algorithm to explore new states, nor too high to surpass the extrinsic reward value. If the value of β is high enough to make the intrinsic reward more influential than the extrinsic reward, the agent loses the sense of the objective that the environment.

Once the algorithm is implemented using the IM strategy of using the variational state as an intrinsic motivation, we can start to interact our code with the environment and empirically search for data and results.

IV. Results

A. Beta Regulator Tests

With the algorithm already implemented, some tests were carried out where the agent did not reach its objective. During these tests we used, based on A2C Algorithm with atari [7] game environments, the agent learning rate hyperparameter at 0.001 and we turned to the analysis of the regulator β. The VSIMR model does not mention the rate used to regulate the intrinsic reward [15], therefore some tests with random rates and we can empirically deduce the importance of such a regulator, as it should remain a stimulus to the exploration of the environment since it should not override the extrinsic reward given by the environment. So we are left with the question. What is the best rate for regulating β?

With this question in mind, we adapted the algorithm so that it is run with several values in β in order to find rates that represent the best results and given the difficulty of the environment, to return information if the agent found its objective in MultiRoom environment. The result of this test we can visualize it in Fig.4. The graph shows us which values we have the best results. With the data generated by the analysis of the graph, further tests were carried out using the value 0.0002 for the hyperparameter β.

The still static definition of the learning rate and when finding a value for the hyperparameter β, the tests demonstrated an improvement of the agent towards achieving the objective in the environments. However, the Gaussian distribution that was being used in the value of the KL divergence that returns from the VAE was configured with a mean of 0.5 and a standard deviation of 0.5, so the agent always received positive stimuli with intensities of different values, with values close to zero when VAE returned little change in the states that were passed on to it, and values closer to 1 when the states showed greater changes. In order to try to improve the agent's performance we decided to change these values so that the Gaussian distribution to mean 0.25 and standard deviation of 0.75, making the intrinsic reward values between -0.5 to 1.0. This range of values was chosen empirically after observations of the agent's behavior and also some value tests, the purpose of which is to make values that represent little change in the state returned by the environment and passed on to the VAE to obtain the intrinsic motivation, had negative values in order to penalize, in a minimal way, the agent for finding states without much change in his observation of the environment. The purpose of this change in the range of intrinsic reward values was to encourage the agent to better explore the environment, aiming to complete the objective in a smaller number of possible actions, making the agent efficient by sampling. In order to find the best value for the hyperparameter Beta, we respectively tested the ranges: 0.0001 - 0.0009; 0.001 - 0.009; 0.01 - 0.09 and 0.1 - 0.9.

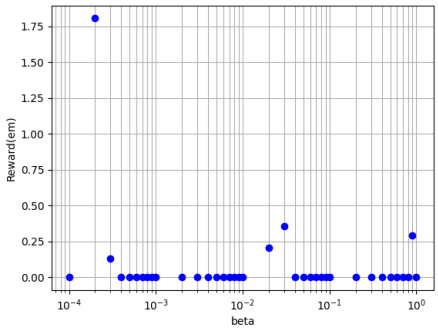

Fig. 4. Results of rewards for 1200 episodes, executed with each range described values in the hyperparameter β.

Tests performed in the configuration of the RL agent, which was achieved in this work through the analysis, showed an improvement in the DoorKey environment and that the agent managed to learn to reach the objective of the MultiRoom environment.

B. Tests with Learning Rates

Tests carried out showed us how crucial the relationship between these two hyperparameters is. When we achieved results in exploring the intrinsic motivation applied to partially-observable environments and sparse rewards, more tests were performed, now staying at the agent's learning rate, which initially is 0.003, when changed to 0.005 or 0.001, demonstrated in its results major changes in the way the agent was learning.

Our environments have 7 possible actions, each action represented by natural numbers and described above, having said that we can conclude by observing these actions, that some actions are not useful in certain environments of the set of environments present in the Minigrid. The MultiRoom has all the actions of the Minigrid environments, but the rules to reach the goal of overcoming the maze present in the MultiRoom are movement, represented by actions from 0 to 2. The other crucial action to beat the MultiRoom environment is to open the doors between the rooms, represented by action 5. The remaining actions 3, 4 and 6 will not show any progress in reaching the goal at the end of the maze, as their functions are

respectively to pick up an object, drop an object and point to the completion of the task. The MultiRoom environment does not have objects to be collected or dropped by the environment and the agent does not indicate the end of the task, it is determined when the agent reaches the green grid. Therefore concluding that in the MultiRoom environment actions 3, 4 and 6 are useless for the agent to obtain results in this environment.

The MultiRoom was used for tests with the learning rate, in order to visualize how much the agent learned certain actions that are most useful to reach the last room, such as moving forward (action 2) and interacting with objects to open doors (action 5).

In Fig. 5 and Fig. 6 are showed the graphs of the probabilities that the agent considered when taking each of the actions. The probabilities of actions range from 0.0 to 1.0 which represent the percentage of probability that the agent has influencing him to take a certain action. The figures were generated after the execution of the MultiRoom and have the action and its given color as caption, the actions represented by the dashed line represent the actions that do not help the agent to reach its goal and do not return to a different state after taking it of the action. The graph shown in Fig. 5 is the one that obtained the best result of the agent in the environment, showing in the graph a decrease in the probabilities of actions represented as not very useful for the agent and its exploration of the environment, and an increase in the probability rate of actions that move the agent through the environment. Fig. 6 shows the same agent using a learning rate of 0.005, in it we can see that action 3, which is an action that does not help the agent at all to explore the environment, much less reach its goal, had a great high in its probability. Observing the environment with a learning rate of 0.005, the agent demonstrated after 800 episodes taking too much action 3 and visually, when rendering the environment, it was parked on the grid where it appeared in the environment.

Finally, a test with a 0.001 learning rate showed that the agent, during the 1200 episodes in which they were tested in the MultiRoom, showed little learning and very similar probability rates demonstrating a high degree of randomness in the actions taken by the agent. The relationships between hyperparameters are extremely important and the definition of some values can be extremely careful and difficult to find.

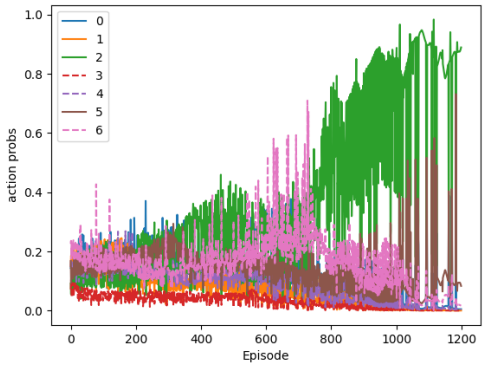

Fig. 5. Probability of choosing the agent's actions in the MultiRoom environment, using the learning rate of 0.003 and the hyperparameter β in 0.0002. Dashed lines are useless actions for the agent in this environment and solid lines are useful actions for the agent.

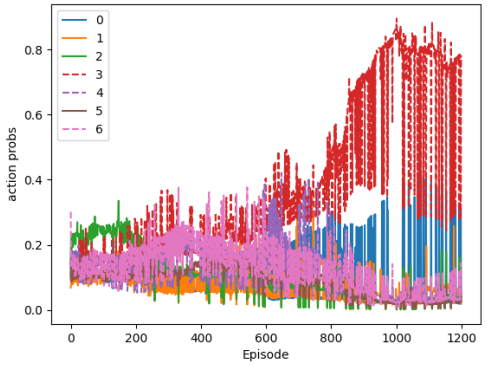

Fig. 6. Probability of the agent's choice of actions in the MultiRoom environment, using the learning rate of 0.005 and the hyperparameter β at 0.0002. Dashed lines are useless actions for the agent in this environment and solid lines are useful actions for the agent.

C. A2C+IM x A2C Standard

The adjustments made in the implementation made our agent achieve good performances in the environments, given its difficulties. With these results, it was possible to go deeper into the problem. Now that the agent can reach its goal, we made a comparison of the performance of the standard A2C agent with the A2C implementation using the intrinsic motivation strategy to reach its goal. In Fig. 7 we can visualize the performance of the algorithm using A2C with Intrinsic Motivation. The standard A2C algorithm returned no reward in 1200 episodes as showed in Fig. 8. We ran the A2C+IM algorithm first and then an algorithm only with the standard A2C, both in the MultiRoom environment with a configuration of 3 rooms of size 4. Running in just 1200 episodes, seeking in this way to prove the sampling efficiency that our implementation has, requiring few episodes so that the agent can learn to cross the rooms, open the doors and reach his goal. In Fig. 7 we have the blue line representing the real values of the rewards returned by the environment and in the orange line we have the average of these values demonstrating the growth of the agent's learning.

In order to test the agent's sensitivity to the storage of the environment. We tested the performance of the Doorkey environment in 1200 episode configurations, and with a gradual increase in stocacity, once in a fully deterministic environment. A second with the most stochastic environment, using two environments configurations that alternated with each other. And finally a third, where the agent had 1200 episodes to learn in a stochastic environment with 3 different settings. These graphs can be respectively in Fig. 9 and Fig. 10. In the graphs we can see the real values of the extrinsic

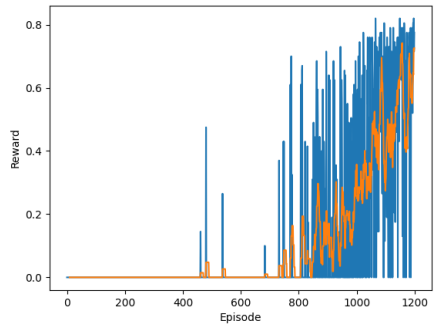

Fig. 7. MultiRoom A2C+IM results in 1200 timesteps (blue = extrinsic reward value, orange = extrinsic reward mean).

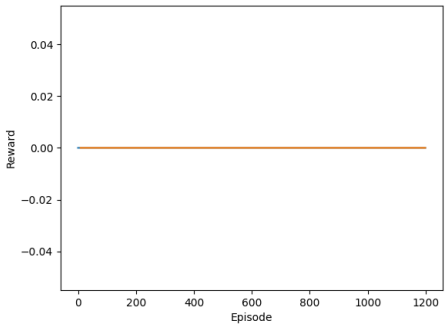

Fig. 8. Standard A2C results in 1200 timesteps (blue = extrinsic reward value, orange = extrinsic reward mean).

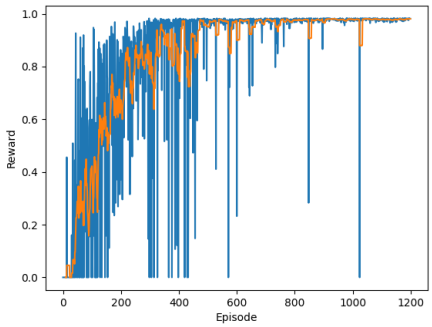

Fig. 9. A2C+IM results on DoorKey with Deterministic environment in 1200 timesteps (blue = extrinsic reward value, orange = extrinsic reward mean).

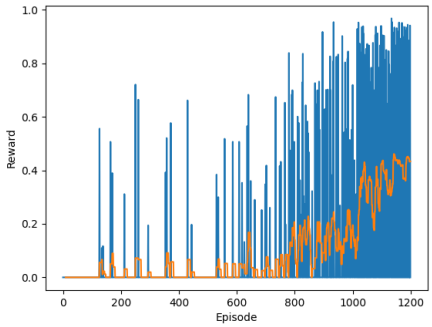

Fig. 10. A2C+IM results on DoorKey with Stochastic environment with 2 environment settings in 1200 timesteps (blue = extrinsic reward value, orange = extrinsic reward mean).

reward in blue and the average of the values in orange for better visualization. Analyzing the graphs, we can see that the A2C+IM strategy has a certain tolerance to stocacity, that even the agent interacting in the environment with two distinct configurations, he continues to learn.

V. Conclusion

The Intrinsic Motivation applied to the Learning Agent through Reinforcement is able to improve their performance and overcome difficulties in scarce environments, as presented in the tests described in this study. However, there are several points to be aware of when applying intrinsic motivation. The origin of this intrinsic motivation, in our case, we use variational inference to obtain our intrinsic reward, through the VAE. And the regulation of intrinsic motivation. It is important to emphasize that the regulation of the intrinsic reward is an extremely important adjustment and that it should neither be too low nor overlap the extrinsic reward, otherwise we run the risk of the agent not having the expected performance gain.

References

[1] R. S. Sutton, and A. G. Barto, Reinforcement Learning: An Introduction. 2. ed. London,England: The MIT Press, 2018.

[2] A. Aubret, L. Matingnon and S. Hassas, A survey on intrinsic motivation in reinforcement learning.arXiv preprint arXiv:1908.06976, 2019.

[3] J. Hare, Dealing with Sparse Rewards in Reinforcement Learning. Dissertation (Master)— University of Sheffield, Sheffield, nov. 2019.

[4] T. Zahavy et al. A Self-Tuning Actor-Critic Algorithm. 2020.

[5] Y. Yu, Towards sample efficient reinforcement learning.Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, v. 1, n. 1, p. 5739–5743, 2018.

[6] I. Grondman, A survey of actor-critic reinforcement learning: Standard and natural policy gradients.IEEE Transactions on Systems, Man, and Cybernetics, Part C(Applications and Reviews), v. 42, n. 6, p. 1291–1307, 2012.

[7] O. Nachum, Bridging the Gap Between Value and Policy Based Reinforcement Learning. 2017.

[8] V. Mnihr,A. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning.Proceedings of The 33rd International Conference on Machine Learning 2016, 2016.

[9] C. Doersch, Tutorial on Variational Autoencoders. 2021.

[10] E. L. Deci and R. M. RYAN, Intrinsic motivation. In:.The Corsini Encyclopedia ofPsychology. American Cancer Society, 2010. p. 1–2. ISBN 9780470479216. ¡https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470479216.corpsy0467¿.

[11] J. Piaget and M. Cook, The origins of intelligence in children. New York, NY, US: W W Norton Co, 1952.

[12] G. Brockman, OpenAI Gym. 2016.

[13] D. Silver, Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. 2017.

[14] M. Chevalier-Boisvert, L. Willems and S. Pal, Minimalistic Gridworld Environment for OpenAI Gym. [S.l.]: GitHub, 2018. ¡https://github.com/maximecb/gym-minigrid¿.

[15] M. Klissarov, R. Islam, K. Khetarpal and D.Precup. Variational State Encoding as Intrinsic Motivation in Reinforcement Learning. [S.l.]: Task-Agnostic Reinforcement Learning Workshop at ICLR 2019, 2019.