Logo, company name

Description automatically generated

November 18, 2021

Group 5

Goh Kheng Xi, Jevan A0199806L

Irwin CHin de tao A0205393H

Skill acquisition and emerging strategies from MULTI-AGENT   
reinforcement learning

ESP3201 Final Report

Contents

[1. Abstract 2](#_Toc88172145)

[2. Introduction 2](#_Toc88172146)

[2.1 Multi-Agent Competition 2](#_Toc88172147)

[2.2 Procedurally Generated Environment 3](#_Toc88172148)

[2.3 Explore and Control 4](#_Toc88172149)

[3. Testing on MiniGrid’s Environment 5](#_Toc88172150)

[3.1 Door and Key Environment 6](#_Toc88172151)

[3.2 Custom Environment (Customised MiniGrid) 7](#_Toc88172152)

[3.3 Evaluation 8](#_Toc88172153)

[4. Curiosity-based models 8](#_Toc88172154)

[5. Testing on Adversarial Surprise Environment 8](#_Toc88172155)

[5.1 Solving Custom MiniGrid with Only Explore Policy 9](#_Toc88172156)

[5.2 Default AS Environment 10](#_Toc88172157)

[5.3 All Dark Rooms 11](#_Toc88172158)

[5.4 Larger Rooms 11](#_Toc88172159)

[5.5 Adding Locked Doors and Keys and Dynamic Obstacles 12](#_Toc88172160)

[5.6 Adding Flashing Lights Room 13](#_Toc88172161)

[5.7 Alice only 13](#_Toc88172162)

[6. Emergence Behaviour 14](#_Toc88172163)

[7. Conclusion 15](#_Toc88172164)

[References 16](#_Toc88172165)

# 1. Abstract

Many machine learning methods requires large volume of dataset to train the model and hence, this project studies reinforcement learning where rewards and penalties are introduced based on the state-space and an agent’s actions as the agent interacts with an environment. It is inspired by the hide and seek paper published by Open AI, where 2 teams compete in an adversarial game of hide-and seek [1]. The report summarises the motivations behind this project alongside the modifications and tests performed on several environment. The results were studied and analysed, forming new basis for further testing and studies.

# 2. Introduction

In reinforcement learning (RL), agents are trained to complete desired tasks through rewards and penalties. This is done with curriculum training, by sending the agent through progressively difficult environments [2]. However, the process of engineering environments is arduous, and the agent struggles to generalise new environments. Furthermore, it takes many iterations just to teach the agent one skill. This makes it difficult to train an agent that can complete complex sequence of actions which is often required in real-world tasks. In this project, we extend upon existing research on multi-agent competition and procedurally generated environments and assess the effectiveness of training such an agent in a customised complex environment.

# 2.1 Multi-Agent Competition

Open AI’s work on multi-agent hide and seek in 2019 showed that agents can learn certain actions without setting explicit rewards [1]. In a MuJuCo 3D environment, two teams are divided into hiders and seekers to play a simple game of hide and seek. The reward is team based, where hiders are rewarded with +1 reward if all hiders are hidden and incur a -1 penalty if any hider is within view of seekers. On the contrary, seekers are penalised with -1 if all hiders are hidden and rewarded with +1 reward if any hider is in sight. They are then trained using self-play and Proximal Policy Optimisation (PPO) [1].

A picture containing keyboard, indoor, computer, mouse

Description automatically generated

Figure . Hide and Seek played between two teams of agents

Surprisingly without explicit rewards, the multi-agent competition environment pushed both teams to interact with existing tools to maximise their reward in the game of hide and seek; hiders learn to block off paths by pushing and holding boxes in place, while seekers learn to use ramps to climb over blocked paths. With further training, the hiders even managed to steal the ramps before blocking off the path so that the seekers have no chance of climbing over.

The introduction of multi-agent competition to RL is especially fitting since humans can acquire secondary skills to fulfil a primary objective. Open AI’s research showed that this method not only works in teaching complex sequence of skills but also results in emerging strategies arising from multi-agent reinforcement learning (MARL) [1].

# 2.2 Procedurally Generated Environment

In a collaborative effort involving Facebook AI, University College London and Oxford University, MiniHack, a benchmark and testbed for MARL, was released in 2021 [3]. MiniHack aims to provide an easy way to modify environments, procedurally generate new complex environments and enable transfer learning for multi-agents training in a 2D environment [3]. This solves three problems faced by MARL research, namely the engineering effort required to design novel environments, the inability to test specific research problems in isolation and heavy compute power required to run 3D environments [3].

A picture containing text, scoreboard

Description automatically generated

Figure . MiniHack, a rich and diverse NetHack environment

Importing elements from the popular 2D grid world maze game NetHack, the environment exposes agents to over 500 monsters and 450 items [3]. This enables many interactions between the agent and the environment, some of which requires numerous sequences of actions. The customisability provided by MiniHack allows researchers to alter the complexity of environments with ease, while training on procedurally generated environments reduces possibility of overfitting and improves the generalisability on agents.

Nonetheless, the paper concedes that popular exploration algorithms such as Random Network Distillation are only able to complete simple navigation tasks, often incapable of making progress when the size of the maze is increased or when more objects are added. Furthermore, agents are only able to solve skill acquisition tasks that require interaction with a maximum of one single entity. This is possibly because even though exploration of the larger state-action space was required, the agent was often distracted by new states and found it impossible to learn any meaningful interactions. This is also known as the Noisy TV problem, which will be elaborated on in the subsequent sections. [4]

# 2.3 Explore and Control

Graphical user interface, application

Description automatically generated with medium confidence

Figure . Contrasting but Complementary reward for Explore and Control Policy

To solve the pitfalls of purely explore based policies, a team from UC Berkeley and Google Research proposed to introduce a control policy. While explore policies aim to maximise entropy and discover new states, control policies aim to minimise entropy by staying in already discovered states [4]. In their paper published in July 2021, Adversarial Surprise (AS) was introduced as a game where two policies take turn controlling the same agent [4]. Alice, the explore policy, aims to outdo Bob, the control policy’ by getting into a high entropy state, while Bob does the converse [4]. At the end of their round, the control of the agent is passed to the other policy and both parties aim to give their opponents the maximum ‘surprise’ by landing them in an undesirable state [4]. The agent traverses through MiniGrid environment, a predecessor of MiniHack that has less available entities and actions but still capable of procedurally generated environments [4].

Even though there is only a single agent present, there exists competition between 2 opposing policies, making this a multi-agent competition. The presence of a control policy solves the Noisy TV problem, while the presence of an exploration policy prevents the control policy from forcing the agent to always stay within explored states. The paper shows that the dual-policy agent can navigate a procedurally generated 4 room environment that requires interaction with multiple keys and doors, while further training results in emerging complex strategies between Alice and Bob [4]. This was similar to the outcomes observed in Open AI’s work on multi agent competition as mentioned in 2.3. As the paper lacked extensive research on the emerging strategies, we committed to extend into this area by adding complexity into the existing MiniGrid environments.

# 3. Testing on MiniGrid’s Environment

Before the MiniGrid environment was chosen, many other environments were explored which included Open AI’s hide and seek environment – MuJuCo. This was because we had initially contemplated to modify the simple Hide and Seek game to a complex game of Police and Thief. However, as the environment only works with Linux and MacOS systems and both of us only have access to Windows Machines, we gave up on setting the environment. We did set up a Linux virtual machine to install the environment, but without access to the GPU from the host PC, we were unable to run the compute heavy 3D environment. Furthermore, the MuJuCo installation process was long and complex, which resulted in many bugs and errors along the way.

MiniHack was found to work on Windows systems, but upon installation, many errors were present such as a required Docker file being corrupted. As the GitHub forum showed other users experiencing the same issues and we were unable to find a solution, we scaled down to its predecessor MiniHack, which is not only compute friendly but also easy to install. Even though it was released in 2017, it is still updated as of 2021 and is the go-to environment for latest research into exploration algorithms [5].

To create environments for the testing of AS, we first explored the in-built environments provided by MiniGrid to assess which features we wanted to include in our customised environment. We wanted to see if the agent trained with vanilla PPO can learn to pickup the key and unlock the door, a complex sequence of events that could be challenging for the agent. This is to show that AS is essential to handle a complex environment involving navigation and multi-skill acquisition.

## 3.1 Door and Key Environment

Graphical user interface, application

Description automatically generatedGraphical user interface

Description automatically generatedGraphical user interface

Description automatically generated

Figure a. 5x5 DoorKey Environment

Figure 4c. 8x8 DoorKey Environment

Figure 4b. 8x8 DoorKey Environment

The agent was trained on one of the MiniGrid’s DoorKey environment which consists of a wall that splits the map in half with a locked door and its corresponding key. From the 3 figures (Figure 4a, b, c) above, the red arrow represents the arrow, and the green square represents the goal location where upon reaching it, the agent will gain a reward of where maximum steps is and the run terminates. For each episode, the locations of the walls, door and key are generated randomly, but follow a similar format of having a door randomly in the wall and the vertical walls are in a random column; this is to ensure that the agent is robust. However, the randomly generated rooms are the same between each training sessions as a seed was used. Figure 4a, b and c show the first environment of every training session for the 5x5, 8x8, 16x16 maps respectively.

The agent was first trained on the 16x16 map using PPO, but after more than 1000 epochs, the model was evaluated with 30 runs and for the 30 runs, it was noticed that there was absolutely no progress, and the agent was simply behaving randomly in all the 30 runs.

The agent was then trained on the smaller 5x5 map using the same algorithm and after 100 epochs, the model was once again evaluated with 30 runs and in all the 30 runs, the agent managed to reach the goal state and incurred an average reward of 0.939. This agent was further trained to a total of 800 epochs and evaluated in the 8x8 map. Although in the 30 evaluation runs, none of the agents managed to reach the goal state, transfer learning behaviour was observed where it was noticed that the agent was trying to find the key and in 6 out of the 30 runs, the agent could be observed picking up the key. However, after the agent picked up the key, the agent does not know how to unlock the door. When this same agent was evaluated in the 16x16 room, it was still behaving randomly.

Graphical user interface, application, Word

Description automatically generatedGraphical user interface

Description automatically generated

Figure . Agent randomly drops key

Figure . Agent picks up key

As the reward was too sparse, a new reward function was introduced where the agent would receive a reward of 100 upon reaching the goal and the run terminates. In addition, for every step, a reward of -0.1 was introduced and the agent was trained on the 8x8 map for 1000 epochs. Then, the agent was once again evaluated for 30 runs, and it was observed that the agent was picking up the key more actively but was randomly dropping the key. In each of the 30 evaluated runs, the agent picked up the key at least once, but none managed to open the door. The agent greatly improved with this new reward function, but it was nowhere near the desired behaviour.

## 3.2 Custom Environment (Customised MiniGrid)

Graphical user interface

Description automatically generatedA customised environment described in Figure 7 was designed and coded. For the purpose of this project, this environment was named the “Customised MiniGrid”. For each episode, the environment was once again randomly generated to ensure the robustness of the agent, but the environment follows a similar format. In this newly designed environment, an obstacle will be blocking a randomly generated entrance to a randomly sized room which contains a key randomly placed in the room that unlocks the door that will spawn on a random location along the vertical wall.

Figure . Custom environment

For this environment, the agent needs to learn to pick up the obstacle first and drop it elsewhere (to pick up the key as the agent can only carry one item at a time). Then, it needs to find the key, pick it up, and then find and unlock the door without dropping the key. Finally, it needs to navigate to the goal state.

The agent was trained for 2000 epochs and evaluated for 10 runs. It was expected that in the 10 runs, the agent did not reach the end goal as the tasks were too complex and the reward and penalty functions were weakly formulated. In all the 10 runs, the agent managed to move the obstacles out of the way, but it only managed to pick up the keys in 1 of the runs. In fact, after moving the obstacle, the agent rarely enters the room to explore.

## 3.3 Evaluation

The poor performance of the agent is expected as RL is very susceptible to sparse rewards. Although rewards shaping is one of the solutions to the sparse rewards problem, it comes with its own drawbacks. For one, it is difficult to design the proper reward functions and ironically, requires expertise knowledge of the task which is the exact problem the model is trying to solve – to develop optimal strategy for the given task [6]. More importantly, the strategies evolved from such methods are constrained by the expertise knowledge used to design the reward functions and thus, the model might not discover new policies that could be better than what is already known. A possible solution is the curiosity-based model (Alice) which explores the environment based on intrinsic curiosity.

# 4. Curiosity-based models

Curiosity in reinforcement learning is a reward function that intrinsically rewards the agent as it explores new trajectories leading to high entropy states, which seeks to simulate the human nature of curiosity itself [7]. However, curiosity-driven learning has a major flaw in dealing with white noise, also known as the Noisy TV problem, as described in 2.3. When the environment contains meaningless noisy states much like a flickering television, an agent seeking novelty will be trapped in the noisy states, making meaningless exploration.

Even though, a curiosity-based agent would have sufficed for the modified environment in section 3.2 (Figure 7. Custom Environment) as there was no dynamic or noisy obstacles that would trap the novelty-seeking agent; to ensure the agent is as robust as possible, it was necessary to solve the Noisy TV problem by employing AS.

# 5. Testing on Adversarial Surprise Environment

This section describes the modification of algorithms and environment on the AS repository after the PPO MiniGrid training algorithm was unable to converge. The default AS Environment features connected rooms separated by a door, with the option of including “white noise” rooms of high entropy and dark rooms with no entropy.

## 5.1 Solving Custom MiniGrid with Only Explore Policy

A picture containing qr code

Description automatically generatedTo ascertain that the customised MiniGrid environment can be solved with just curiosity-based agents, we modify the AS default environment to only include dark rooms. We also freeze the control policy (Bob) and ensure that only the explore policy (Alice) was acting.

Figure . Locked Doors and Key with only explore policy

Next, the environment was modified such that each room generated has a 50% chance that its corresponding door is locked and if the door is locked, a key will be spawned in that room that the agent must picked up to unlock the door. This increases the complexity of the environment.

Chart, line chart

Description automatically generated

Figure . Average Number of Rooms Visited by Alice per episode against Number of Epochs

The average number of rooms visited by Alice per episode for each epoch was plotted in the graph above (Figure 9. Average Number of Rooms Visited by Alice per episode against Number of Epochs). The results of 600 epoch were collected over a period of about 5 hours and shows a much greater performance and potential compared to the 1000 epochs trained on the 16x16 environment in MiniGrid using vanilla PPO, which also has similar training duration of about 5 hours; yet, out of the 30 evaluation runs in MiniGrid, none of the agents could open a single room.

Due to time constraint, the training session was terminated early as evidence that curiosity agents are capable of learning complex skill sequences (picking up a door and unlocking the door).

## A picture containing graphical user interface Description automatically generatedChart Description automatically generated5.2 Default AS Environment

Figure . Results of Adversarial Surprise Training on Default Environment

Figure . Default Adversarial Surprise Environment

To challenge the agent, “white noise” rooms are introduced to the environment as flashing lights room in the Adversarial Surprise default environment where the floors of the “white noise” room changes colours rapidly to simulate high entropy and trap novelty-seeking agents.

For the default environment where the flashing lights rooms are generated at a 50% probability, the agent is trained for 300 epoch and as expected (as shown in the paper), it performed well and was observed to be able to traverse all the 4 rooms on the 300th epoch.

Figure 10 shows the average number of rooms that Alice (the explore policy) has explored per episode for each epoch. As 15 parallel environments are running in each episode, the number of rooms visited in 1 episode is averaged over the 15 parallel environments and this average is once again averaged over the 4 episodes of each epoch. This graph serves as a benchmark for some of our modified environment.

## 5.3 All Dark Rooms

Chart, line chart

Description automatically generatedA picture containing graphical user interface

Description automatically generatedTo show that the inclusion of a control policy (Bob) in AS is only solving the Noisy TV problem, the environment is set to all dark rooms with no entropy and ran with only the explore policy (Alice) to compare the results with that of default environment in section 5.2 (Figure 11. Default Adversarial Surprise Environment).

Figure . Comparing Curious Agent on Dark Rooms Only versus Adversarial Surprise Agent on Noisy Rooms

Figure . All Dark Rooms with Only Explore Policy

From Figure 13, when the rooms are all dark rooms, the curious agent is sufficient to explore the environment. In fact, it is performing slightly better than the adversarial surprise model with the “white noise” traps.

## 5.4 Larger Rooms

Graphical user interface

Description automatically generatedThe agent was trained on larger rooms where the maximum room size is 16x16 instead of 8x8, and as expected the average number of rooms visited per episodes were lower than that in the above environment.

Figure . Larger Rooms of Maximum 16x16

Chart

Description automatically generated

Figure . Larger Rooms Have Lower Average Number Of Rooms Visited By Alice

With reference to Figure 15, It was observed that the larger rooms undermined the agent’s ability to navigate through the rooms severely and no instances could be found to show that the agent entered the second room. However, the graph was relatively upward sloping although gradual which shows that the agent is still learning noticeable although slower.

## 5.5 Adding Locked Doors and Keys and Dynamic Obstacles

Graphical user interface

Description automatically generatedNext, the agent is pushed even further by introducing rooms that have a 50% probability of being locked and if it is locked, there will be a corresponding key in the room to unlock the door. Keys can only doors of the same colour as the key itself.

Figure . Rooms with Dynamic Obstacles and Locked Doors and Keys

Additionally, rooms with dynamic obstacles have a 50% chance to spawn and this event is independent of the probability that the room is locked which means that a room can be both locked and have dynamic obstacles at the same time. The moving objects cannot be overlapped and thus, it simply blocks the agent way instead of directly incurring a penalty when the agent encounters it. In the dynamic obstacles room, there is a switch to freeze the moving obstacles in place which unlike the switch in the noisy room, does not need to be picked up to toggle it.

Chart, line chart

Description automatically generated

Figure . Results on Locked Doors and Keys with Dynamic Obstacles

Figure 17 shows the average number of rooms Alice visited per episode for each epoch up till about 1200 epochs. Considering this environment is more complex than the customed environment in MiniGrid from section 3.2 (Figure 7. Custom environment), yet the model is performing better in terms of average number of rooms visited shows that AS model is better than just vanilla PPO.

## 5.6 Adding Flashing Lights Room

Graphical user interface

Description automatically generated with medium confidence

Figure . Addition of Flashing Lights Rooms

The final modification to the environment was the addition of the high entropy flashing lights room to the environment. In this environment, there are no dark rooms and the rooms are either with dynamic obstacles or flashing lights. Furthermore, each room has a 50% chance of being locked. The agent is then train on the environment for more than 7000 epochs.

Chart

Description automatically generated

Figure . Smoothed Results of Average Number of Rooms Visited by Alice

The graph above in Figure 19 shows the average number of rooms visited by Alice per episode for each epoch. The translucent wavy graph in the background shows the actual results of training while the solid blue graph shows the smoothed results and from the smoothed results, there is a gradual upward trend that shows the agent is still able to explore this complex environment despite the slow training.

## 5.7 Alice only

To demonstrate that this environment is impossible with only the exploration policy, the control policy Bob was frozen, and the agent is trained for not more than 400 epochs. The results were compared to the results in figure x where the control policy is active.

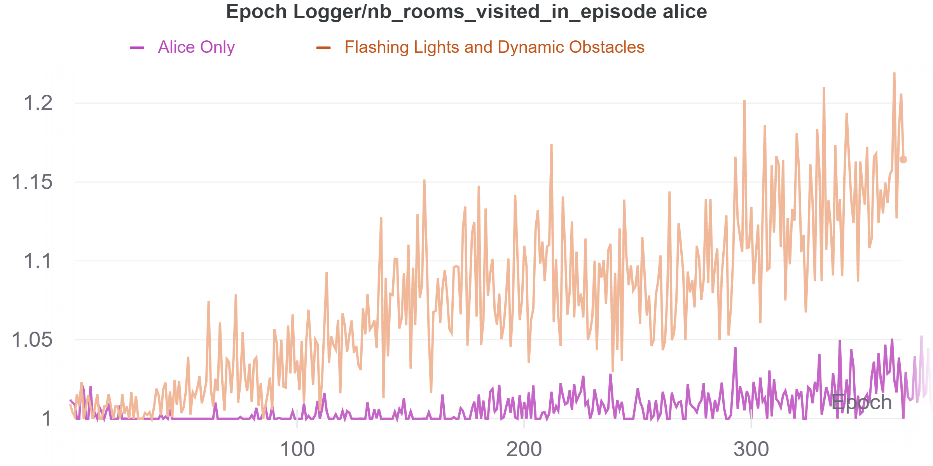


Figure . Explore Policy Only Performs Poorer Than Adversarial Surprise

As observed from the figure above, the average number of rooms explored by the agent with explore policy only remains relatively constant and much lower compared to the agent with both the explore and control policy.

# 6. Emergence Behaviour

The video clips of a randomly chosen episode were recorded every 5 epochs for the session in section 5.6. The video clips were then watched and studied to observe the emergence of complex behaviour. Emergence of strategies or skills are recorded, and the videos will be sent alongside this report. These behaviours might or might not be relevant to the main objective of traversing around the room, but it was interesting to see the complex strategies evolved similar to the Multi Agent Hide and Seek paper by Open AI.

It should be noted that the study of the video clips is time-intensive and hence, it is possible that we missed video clips which contain emergence of undiscovered skillsets. Therefore, it is also highly probable that the epochs mentioned are not the first instance observed of the described skillset.

The relevant files are described below, for your additional viewing if necessary.

285, 365 Epoch: Alice learns to turn off the switch in the dynamic obstacles room, possibly to navigate to the key easier. (video file: sec6\_1.mp4, video file: sec6\_2.mp4)

400 Epoch: Alice can be observed to acquire the skills of picking up the key to unlock the door and enters the next room. (video file: sec6\_3.mp4)

500 Epoch: Alice is seen stealing the switch to prevent Bob from toggling the flashing lights off. (video file: sec6\_4.mp4)

800 Epoch: Bob is seen turning the flashing lights off before stealing the switch, preventing Alice from toggling the lights back on. (video file: sec6\_5.mp4)

1310 Epoch: Bob steals the key so that Alice could not unlock the door. (video file: sec7\_6.mp4)

1335 Epoch : Bob stops the dynamic obstacles at the door so that the door is blocked and steals the key away from Alice. (video file: sec6\_7.mp4)

1495 Epoch: Alice close the door behind her so that Bob is trapped in the “white noise” room. (video file: sec6\_8.mp4)

2105 Epoch: Alice managed to traversed 3 out of the 4 rooms. (video file: sec6\_9.mp4)

For the rest of the epochs, the strategies mentioned above constantly appears, but no new strategies emerged (could be missed). Alternatively, the videos and further training data could be accessed from https://wandb.ai/jgoh/adversarial\_surprise2?workspace=user-jgoh.

# 7. Conclusion

Like most machine learning methods, reinforcement learning is time and computationally intensive. We did not focus on the fine tuning of the algorithm and instead, the advantages and limitations of the adversarial surprise algorithm was explored. Although for a static environment such as the customed environment we designed on MiniGrid, curiosity-driven models have shown to be slightly more effective than AS, in the real world, most environments are dynamic and noisy with many “white noises” to trap purely novelty seeking agents. Therefore, the inclusion of a control policy is necessary, as in AS.

However, AS is far from perfect as a new problem emerged with the use of this algorithm. As the training progress, it was noticed that there were long periods of epochs where the model was observed to have a very low average number of rooms traversed especially in the later epochs for the environment in 6.5. Further observation of the video clips shows that the agents were focused on competing between themselves instead of performing meaningful exploration into further rooms and this was synonymous with the observation made by the original paper.

Another method was thought of to improve the explorative behaviour of the adversarial surprise agent such that it explores the room faster while maintaining robustness in overcoming obstacles such as the “white noise” traps. The idea was to limit the maximum number of steps that the control policy (Bob) can take in a round to undermine the controlling effect and balance the amount of controlling effect such that it remains minimally robust to counter all the obstacles and able to rapidly explores the room. However, due to the structure of the code architecture of gym which the adversarial surprise stems from, this idea was not materialised but attempted.

Despite deviating greatly from the original idea of a Police and Thief multi agent competition, this project still seeks to solve the fundamental problems of reinforcement learning, which could be more meaningful than solving a specific complex environment. Improvement can be made to this project by introducing more than a single adversarial surprise agent simultaneously to an environment and provide the agent with a combined entropy score rather than individual entropy. This is to stimulate the collaborative behaviour of the agents as described in Open AI’s Multi Agent Hide and Seek paper.

# References

|  |  |
| --- | --- |
| [1] | B. B. I. K. T. M. Y. W. G. P. B. M. and I. M. , “Emergent Tool Use From Multi-Agent Autocurricula,” Open AI, 2019. |
| [2] | S. N. B. P. M. L. J. S. M. E. T. and P. S. , “Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey,” Cornell University, 2020. |
| [3] | M. Samvelyan, “MiniHack the Planet: A Sandbox for Open-Ended Reinforcement Learning Research,” Cornell University, 2021. |
| [4] | A. Fickinger, “Explore and Control with Adversarial Surprise,” Cornell University, 2021. |
| [5] | Facebook AI, “MiniHack: A new sandbox for open-ended reinforcement learning,” Facebook, 29 September 2021. [Online]. Available: https://ai.facebook.com/blog/minihack-a-new-sandbox-for-open-ended-reinforcement-learning. [Accessed 18 November 2021]. |
| [6] | K. Daaboul, “Reinforcement Learning: Dealing with Sparse Reward Environments,” Medium, 2020 August 2020. [Online]. Available: https://medium.com/@m.k.daaboul/dealing-with-sparse-reward-environments-38c0489c844d. [Accessed 18 November 2021]. |
| [7] | T. Simonini, “Curiosity-Driven Learning made easy Part I,” Towards Data Science, 16 October 2018. [Online]. Available: https://towardsdatascience.com/curiosity-driven-learning-made-easy-part-i-d3e5a2263359. [Accessed 18 November 2021]. |