International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-4, April 2019

Hierarchical Reinforcement Learning for Humanoids

Abhishek Warrier, Arpit Kapoor, T. Sujithra

Abstract*: The control of humanoid robots has always been difficult as humanoids are multi-body systems with many degrees of freedom. With the advent of deep reinforcement learning techniques, such complex continuous control tasks can now be learned directly without the need for explicit hand tuning of controllers. But most of these approaches only focus on achieving a stable walking gait as teaching a higher order task to a humanoid is extremely hard. But there have been recent advances in Hierarchical Reinforcement learning, in which a complex task is broken down into a hierarchy of sub-tasks and then learned. In this paper, we demonstrate how a hierarchical learning inspired approach can be used to teach a higher order complex task, such as solving a maze, to a humanoid robot.

***Index Terms*: Humanoids, Hierarchical Reinforcement Learning, Reinforcement Learning, Deep Reinforcement Learning.**

I. INTRODUCTION

Humanoid Robots, by their very definition, are bipedal robots designed to resemble humans. But the control of humanoid robots is extremely hard due to their many degrees of freedom. Most modern humanoid robots usually have anywhere between 15-30 actuators, each of which needs to exert the right amount of torque, at the right time to achieve any kind of motion. Hand designing such controllers is quite difficult, which is why alternative approaches, such as the use of reinforcement learning, are now being explored. Reinforcement learning was originally considered as an extension of optimal control theory. In the reinforcement learning paradigm, at any point, the agent is in a state, and from each state it can choose an action. This action affects the environment, and thus in turn, the state of the agent. Thus the main objective is to obtain a policy, which is mapping from states to actions, that dictates the sequence of actions that need to be performed to take an agent from its starting state to its desired state. The main difference between Reinforcement learning and other traditional machine learning approaches is that after performing an action, the agent is given a reward that indicates how good the action was. The factors that determine the difficulty of a reinforcement learning problem are - 1) Observation/state space, 2) Action space, 3) Reward signal. The observation space refers to the information about the environment that is given to the agent. These observations may be partial (where the agent only gets information that it can perceive such as a camera image) or full (where the agent gets complete information.) Working with partial information usually makes it much harder but a much more realistic policy is obtained. The action space can be discrete/continuous. In discrete, an action is simply chosen from a finite set of values, whereas in continuous the action can be any value within a particular range. Humanoids and other robotics problems are usually continuous control problems as the exact value of torque/angle for each motor needs to be determined. Finally, the reward signal may be sparse/dense. In dense rewards a frequent feedback is given which makes training more sample efficient, but this requires sufficient domain knowledge and reward engineering. On the other hands, sparse rewards are easy to define but are much more difficult to learn from. Recently, many advances have been made that have made training deep neural networks easier. These advances in deep learning, combined with reinforcement learning has led to the advent of deep reinforcement learning. Deep reinforcement learning techniques make continuous control and sparse reward problems much more feasible. Another paradigm that has been more researched upon recently, is Hierarchical Reinforcement learning. In Hierarchical Reinforcement learning, a task is broken down into a hierarchy of sub-tasks, where each subtask handles a particular aspect of a problem. Each of these subtasks is much easier than the original task which makes learning feasible. In this paper, we attempt to apply Hierarchical Reinforcement learning to humanoids. Rather than simply making a humanoid walk, we attempt to solve a higher order task of solving a maze. Now it is very difficult to teach a humanoid to simultaneously learn how to walk and solve a maze. Thus we separate this task into two sub tasks, where a higher level task only needs to learn how to solve the maze and the lower level task only needs to learn how to walk.

Manuscript published on 30 April 2019.
* Correspondence Author (s)
Abhishek Warrier, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India,
Arpit Kapoor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India.
Dr. T. Sujithra, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India,

II. RELATED WORK

Although a lot of work on Reinforcement Learning techniques focus on general purpose algorithms that may applied to multiple domains, many works have also specifically focused on the application of reinforcement learning to humanoid robotics. One of the first papers that surveyed the application of RL techniques specific to humanoids was done by Peters et al [1]. The paper compares various approaches such as Policy Evaluation Methods, Policy Iteration Methods and Natural Policy Gradients and presents their relative merits. A novel algorithm, the Natural Actor Critic is introduced and demonstrated on the Cartpole problem to show that it can be scaled to high dimensional spaces.

A more comprehensive survey is conducted by Peters et al [2] in which improvement to the Natural Actor-Critic, the Episodic Natural Actor-Critic is introduced, and it is used to train a multi-DOF robot arm to hit a stationary baseball. To deal with the problem of high dimensionality, an alternative approach was taken by Tedrake et al [3] in which instead of a proper humanoid, they made use of a modified passive dynamic walker which was actuated. This simplified 6 DOF structure is much easier to train and manages to converge to a stable walking algorithm in 20 minutes when trained using a Stochastic Policy Gradient with TD-Error. But although this helped to gain insights into the problem of bipedal walking, the technique doesn't scale well to an actual humanoid. A major bottleneck in training humanoids or robots, in general, is that the algorithms used are not sampled efficient and require a large number of trials on a physical robot. A solution to this is to use model-based RL techniques in which the underlying representation of the problem is also learned. Morimoto et al [4] proposed such a model-based approach in which a Poincare Map is used to estimate the walking period and learn the effect of foot placement. The problem is considered a type of Semi-Markov Decision Process and Receptive Field Weighted Regression (RFWR) is used to approximate the model, policy and the value function. This proposed system was later successfully implemented on a bipedal robot to generate a walking pattern [5].

Probabilistic approaches to Reinforcement Learning have also been used such as in [6] in which a novel algorithm, PI2 is introduced. It is successfully implemented on a 34 DOF humanoid to perform complex tasks such as opening a door (albeit in simulation only). Another approach is to make use of the neurobiological concept of Central Pattern Generator (CPG) that produces a baseline rhythmic output independent of any sensory inputs in humans and other vertebrates. In [7] the problem of bipedal locomotion is designed as a partially observable Markov decision process (POMDP). They present a CPG Control Architecture which consists of a CPG Controller and a CPG Feedback Controller other than the robot. The CPG Controller consists of a neural oscillator that can interact with the sensory feedback provided by the CPG Feedback Controller to produce the desired output but designing such a controller usually requires a lot of trials and manual tuning. So instead, the sensory feedback controller is learned using policy gradients to generate a steady walking pattern. The fact that robots have continuous state-action spaces makes it hard for traditional reinforcement learning techniques to learn a complex controller for any robot, in general. Deep Deterministic Policy Gradient (DDPG), an algorithm proposed by [8] is an Actor-Critic approach based on Deterministic Policy Gradients (DPG). This paper introduces a model-independent algorithm which operates on continuous spaces and adapts ideas form Deep Q-Learning. This paper is quite pivotal as it allowed state of the art techniques to be extended to continuous action-spaces which makes it quite useful for application in Robotics. Since this paper, some improved techniques have been introduced which include Proximal Policy Optimization (PPO) [9] and Trust Region Policy Optimization (TRPO) [10]. Duan et al [11], presented a benchmark of various deep reinforcement learning algorithms that can be implemented for continuous control. Behaviours like locomotion can be quite sensitive to the choice of reward. It is common for most reinforcement learning algorithms to develop the reward functions with great care in order to motivate specific solution. Hees et al [12], proposed a method that enabled learning of complex locomotion behaviours with simple reward functions. In this paper, simple reward function was used to train the agents in rich, diverse environments which led to robust behaviours that generalize across diverse tasks. Also, a distributed implementation of PPO is introduced to achieve better performance. A couple additions are also made in Distributed PPO (DPPO) algorithm. Reward and input normalization in one such augmentation, whereas additional term in the loss to avoid huge infringement of trust region boundaries is another. Despite these great advances in Deep RL, one main drawback is that these techniques still require reward engineering and extensive domain knowledge. Recently, Andrychowicz et al presented the Hindsight Experience Replay (HER)[13], which enables sample-efficient learning from rewards which are sparse and binary. HER adds additional goals for replay to ensure that at any given point of time at least half of the replayed trajectories have a positive reward which speeds up learning. The stable walk is still one of the most challenging tasks for a humanoid robot to perform. In recent years, many researchers have explored the paradigm of deep reinforcement learning to demonstrate the bipedal walk. Kumar et al [14], present an architecture to demonstrate walk on a 5-link bipedal robot using Gazebo simulator. They make use of Deep Deterministic Policy Gradient (DDPG) to train the bipedal walk. The paper shows that with proper reward, the robot achieves a faster walking and it was even able to render a running gait with a speed of 0.83 m/s which was quite like that of a human. Moreover, Xie et al [15] demonstrate deep reinforcement learning using in formulating a feedback controller for walking on Cassie, a 7 DOF biped developed by Agility Robotics. They make use of PPO to optimize the policy for Markov Decision Process. Two-step motion of the robot is taken as the reference which is used to form the basis of the motion of the robot with respect to time. Time-scaling of the initial source motion is used to learn controllers for different walking speeds. It should also be mentioned that interpolation amongst policies can make the controller more robust. Much attention has been given to Passive Dynamic Walking (PDW), due to its energy efficient gaits. The challenge remains in controlling a passivity-based bipedal robot which are inspired by Passive Dynamic walkers. However, deep reinforcement learning algorithms may help in learning walking in such scenarios [16]. The paper presents a Deep Q Network based reinforcement learning controller that learns the walking policy for a passivity-based biped robot by taking the PDW as the reference trajectory.There have also been many new developments in hierarchical learning. In [17] the authors demonstrate a nested approach to HRL where information from the high level agent is fed into the observation of the low level agent. And In [18] the authors formulate a Hierarchical Actor-Critic approach that makes simultaneous multi-level training feasible. These both papers are the main inspiration for the approach given in this paper.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

III. METHODOLOGY

A. Problem Formulation

To study the application of Hierarchical RL in the context of humanoid robots, we will tackle the specific problem of teaching a humanoid to walk through and solve a maze. Such a complex policy can be broken down into a high level and low level policy.

1) The high level policy (π_h) is a discrete maze solver that takes the current map of the maze as its state and outputs a direction to move in.

2) The low level policy (π_l) is the humanoid walker that takes the position and angular speed of each joint and outputs the corresponding joint torques to control the humanoid.

Finally other than that, there is an optional intermediate policy.

B. Experimental Setup

1. Choice of Simulator

A major problem with reinforcement learning methods is that results are quite hard to reproduce [19]. This is because these techniques are sensitive to the smallest changes in the environment/algorithm. So the same algorithm, with slight implementation differences might give quite varying results. Thus we experimented with various simulation environments to find the optimal one for our task -

1) MuJoCo - Which stands for stands for **Mu**lti-**Jo**int dynamics with **Co**ntact, is a high performance physics simulator developed by the University of Washington. But the Humanoid environment provided in MuJoCo has a **376 dimensional** observation space and a **17 dimensional action space** that makes it extremely time consuming to even learn to walk a few steps. Furthermore, another drawback is that it is a proprietary software that requires a paid license.

2) Unity - Unity is primarily game engine that uses the Nvidia PhysX engine for its physics. But the ML-Agents library provided by Unity allows it to be used for training RL Agents. Furthermore, Marathon_Envs, a Unity package adds standard RL environments such as Ant, Humanoid, Hopper etc. The Humanoid environment provided by Marathon_envs has a **88 dimensional** observation space and a **21 dimensional action space** and also allows multiple agents to be trained in parallel. But although the humanoid manages to learn to walk straight, it is difficult to train a policy that can enable to walk in any direction.

3) Roboschool - Roboschool is a simulator, developed by OpenAI as an open source alternative to MuJoCo and it has slightly more realistic versions of the original MuJoCo environments. The roboschool humanoid environment has a **44 dimensional** observation space and a **17 dimensional action space** which is quite concise. But the main advantage is the inclusion of environments like RoboschoolHumanoidFlagrun-v1 where, the objective is to not only teach a humanoid to walk, but walk to a specific target. The policies trained in this environment, although time consuming, are quite robust where the humanoid can turn and even get up after a fall. Thus this was our final choice for the simulator. But Roboschool Envs are quite rigid and it was difficult to setup maze in it. So the final decision was to use both Unity and Roboschool in conjunction. The maze agent is implemented in Unity whereas the humanoid is implemented in Roboschool. Both simulators communicate with each other over sockets.

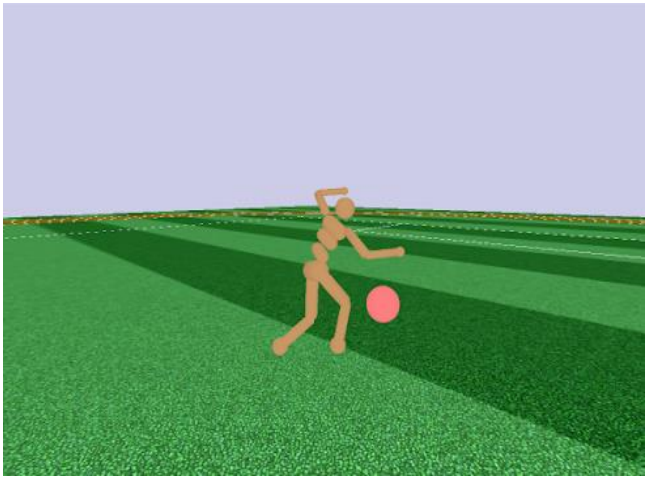

Figure 1. RoboschoolHumanoidFlagrun-v1

2. Choice of Algorithm

As stated earlier, the humanoid is a very high dimensional control task and training in such high dimensional continuous spaces has always been challenging, but recent advances with deep neural nets have made them much more viable with algorithms such as DDPG, A2C giving great results. But the main algorithm that we decided to use is Proximal Policy Optimization (PPO) which is the current state of the art, with its main advantage being its simplicity and efficiency. Furthermore, the original paper that introduced PPO [20], makes use of the Roboschool Humanoid Flagrun-v1 and Roboschool Humanoid Flagrun Harder-v1 environments for benchmarking, making it an apt choice. But for the higher level policy, i.e. the maze solving task is quite easy and can even be solved by the simplest of algorithms, such as Q-Learning as it has a discrete observation and action space. When the maze is being randomized one each iteration, better performance may be obtained by using a Deep Q-Network (DQN) instead.

C. Hierarchical Combination

Firstly, we observed that training the humanoid is quite difficult. So our solution is to train both the lowest level humanoid walker policy (π_l) which generates the torques for the joints of the humanoid and uses a continuous state as well as action space, and the high level policy separately. For the high level policy (π_h), which is a maze solver and has a relatively small state as well as action space, which is relatively easier to train. So for the low level walker, the RoboschoolHumanoidFlagrun-v1 environment was trained using PPO.

For the maze solver, both the continuous and discrete state-action spaces was tested. The discrete maze solver gave better performance as expected and was finalized as the high level policy. The discrete maze solve was trained both with PPO and Q-Learning, with Q-Learning giving better performance. So far, the maze solver and the humanoid walker have been trained independently of each other. The final part is using the maze solver policy as a high level selector policy and using it to control the direction motion of the humanoid walker.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

For this, a hierarchical policy integration algorithm has been devised which is given as follows in Algorithm 1.

Algorithm 1 Hierarchical Policy Integration

```
Let πh, πl, be the higher level and lower level trained policies
Let Sh be the current state of the higher level policy
Let Gh be the final goal
while Sh is not Gh do
    New state Sh'← πh(ah | Sh)
    Set goal of low level policy state gl← Sh'
    Let Sl be the current state of the low level policy
    while sl is not gl do
        Sl'← πh(al | Sl)
        Sl ← Sl'
    end while
end while
```

The actual implementation of this algorithm to our problem is as follows. Initially, both the humanoid and the maze agent are at their starting positions. Then, the current position of the maze agent is given to the Roboschool Humanoid Flagrun policy. The humanoid starts approaching toward the maze agent and the distance between it and the maze agent is constantly checked. When the distance goes below a threshold, the current state is passed to the maze agent policy that outputs a new direction to move in and updates the maze agent position. Now the humanoid now starts moving to the updated position. In this manner, the higher level maze policy slowly directs the lower level humanoid to solving the maze.

IV. RESULTS

Based on our observations, we conclude the following results –

1) We observed that the policies are effectively combined and the hierarchical behavior is as intended.

2) Switching from continuous maze to a discrete maze environment does radically improve performance.

3) Both of these continuous/discrete versions of MLAgents-PPO is outperformed by our q-learning implementation.

4) But the overall performance and success rate varies widely as the individual policies have not been trained to perfection and it is only occasionally able to solve the maze so there is still scope for improvement.

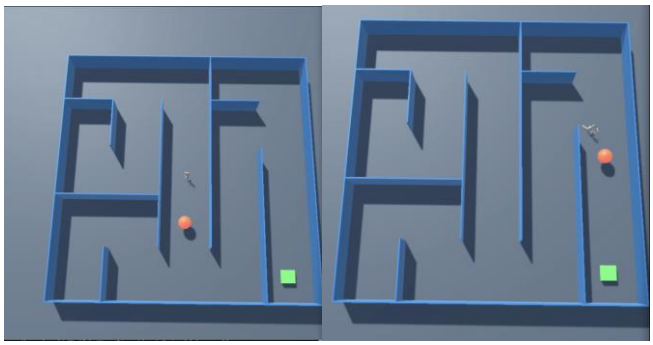

Figure 2. Humanoid and Maze Agent Combination Solving the Maze

The final combined policy is shown in Figure 2. Here, the green block is the goal for the higher level maze policy. The orange sphere is the current positon of the maze agent, which acts as the goal position for the lower level humanoid walker.

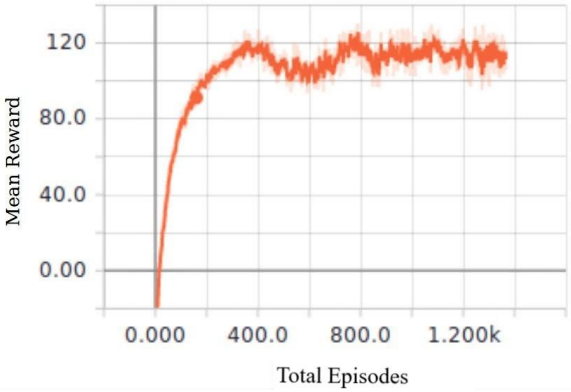

Figure 3. Mean Reward vs Total Episodes for Humanoid Policy

Figure 3 shows the plot of the episode mean reward against the total number of episodes for the humanoid walker policy. A steady increase in the mean reward is observed for the first 400 episodes, after which the reward value stabilizes to its optimal value.

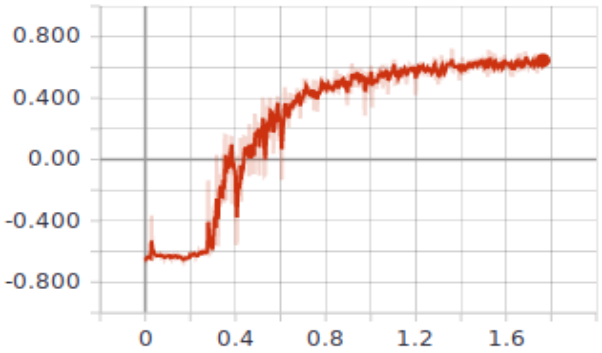

Figure 4. Cumulative Reward vs Time Steps for Discrete Maze Policy

Figure 4 shows the plot of the cumulative reward against the total number of time steps (relative) for the discrete maze solver. Initially, the cumulative reward barely improves but after a while, it starts steadily increasing till the value stabilizes.

V. CONCLUSION

The paper has demonstrated how a continuous high level task can be tackled using a hierarchical divide and conquer approach. The fact that the lower level policy is trained separately makes the overall task modular in nature. This means that the lower level policy could simply be replaced by an Ant/Hopper/Walker or any of the other continuous control agents to obtain respective maze solving agents. But yet in an ideal scenario, it would be better if efficient algorithms are developed where the entire hierarchy can be learned together at once.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication (BEIESP)

FUTURE WORK

Although in this paper, a humanoid was trained to solve a maze, it was not a humanoid robot. More precisely, the humanoids which are generally used in RL for benchmarking, are abstract dummy humanoid that do not correspond to any existing real humanoid robots. So the next logical task would be to create and experiment with simulations of popular humanoid robots such as Atlas, Darwin-OP, Nao, etc. and study/analyze their relative performance.

REFERENCES

1. J. Peters, S. Vijayakumar, and S. Schaal, "Reinforcement learning for humanoid robotics," in *Proceedings of the third IEEE-RAS international conference on humanoid robots*, 2003, pp. 1–20.
2. J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, *pp. 682–697, 2008.*
3. R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3d biped," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, vol. 3. IEEE*, 2004, pp. 2849–2854.
4. J. Morimoto, G. Cheng, C. G. Atkeson, and G. Zeglin, "A simple reinforcement learning algorithm for biped walking," in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, vol. 3. IEEE*, 2004, pp. 3030–3035.
5. J. Morimoto and C. G. Atkeson, "Learning biped locomotion," *IEEE Robotics & Automation Magazine*, vol. 14, no. 2, pp. 41–51, 2007.
6. F. Stulp, J. Buchli, E. Theodorou, and S. Schaal, "Reinforcement learning of full-body humanoid motor skills," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on IEEE, 2010*, pp. 405–410.
7. G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot, " *The International Journal of Robotics Research, vol. 27, no. 2*, pp. 213–228, 2008.
8. J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Intelligent Robots and Systems (IROS),2017 IEEE/RSJ International Conference on. IEEE*, 2017, pp. 23–30.
9. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprintarXiv:1707.06347*, 2017.
10. J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.
11. Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.
12. N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez,Z. Wang, A. Eslami, M. Ried milleret al., "Emergence of locomotion behaviours in rich environments,"arXiv preprint arXiv:1707.02286,2017.
13. M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welin-der, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 5048–5058.
14. A. Kumar, N. Paul, and S. Omkar, "Bipedal walking robot using deep deterministic policy gradient," *arXiv preprint arXiv:1807.05924*, 2018.
15. Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," *arXiv preprint arXiv:1809.07279*, 2018.
16. Y. Wu, D. Yao, X. Xiao, and Z. Guo, "Intelligent controller for passivity-based biped robot using deep q network," *Journal of Intelligent & Fuzzy Systems, no. Preprint*, pp. 1–15.
17. Brittain, M., & Wei, P. (2018). Hierarchical Reinforcement Learning with Deep Nested Agents. *arXiv preprint arXiv:1805.07008*.
18. Levy, A., Konidaris, G., Platt, R., & Saenko, K. (2018). Learning Multi-Level Hierarchies with Hindsight.
19. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2017). Deep Reinforcement Learning that Matters. arXiv, cs.
20. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

AUTHORS PROFILE

Abhishek Warrier B.Tech student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India. His research interests include Machine Learning, Robotics and Computer Vision.

Arpit Kapoor, B.Tech student, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India. His research interests include Machine Learning, Robotics and Computer Vision.

Dr T. Sujithra, Assistant Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, India. Her research interests include Wireless Sensor Networks, IOT and Network Security.