Curriculum Learning of Bipedal Locomotion on uneven terrain

Emeka Oguike, Jiayan Zhang, Nishan Acharya

1 Abstract

In recent times, the robotics research community around the world has shown considerable interest in the field of humanoid robotics. Boston Dynamic's Atlas, a bipedal walking robot, is a classic example of such humanoid robots. While such industry-grade robots are capable of performing a wide variety of human activities, they run into problems in environments with partially or completely different physical parameters than the one they were designed to operate in. Our approach to the locomotion control employs Curriculum Learning and Reinforcement Learning (RL) to train policies that aids a bipedal agent to learn a stable walking gait through increasing the roughness parameter toward any given initial target terrain.

2 Introduction

Biped humanoid robots are conceived with the final objective to have a human-like behavior. Deploying and using Biped robots in an outdoor scenario come with several challenges. A real-world environment is typically unstructured, unknown and can exhibit obstacles which have to be negotiated. Biped agents are thus designed with a physical structure that tries to emulate the versatile locomotive skills of human beings with the purpose of robust movements that allows for a better navigation of irregular terrains in comparison with robots with wheels. There is a significant amount of research efforts related to the development and control of biped agents, many of which require highly complex techniques and constraints to achieve good performances. Many endeavors have been at legged robots walking on flat surface and multiped walking robots on uneven terrain, however, there is no solution to biped walking robot on rough ground overcoming the curse of dimensionality in the continuous state and action space. Our approach is to develop an end-to-end pipeline that unifies the terrain curriculum generation and RL training process.

2.1 Related Work

Many of past efforts were done using classical approaches based on gait stability. For example [Jung Yup et al] propose an online gait trajectory generation

method which produces a smooth, continuous walking gait for agents in the simulated and real-world environment. [Dekker et al] use zero moment points (ZMP) dynamics derived from the robot model with a calculated ZMP trajectory to create a walking pattern for an agent.

Bipedal locomotion has also seen significant interest from the machine learning community as an interesting and challenging problem for reinforcement learning. More specifically, model-free learning of bipedal walking has mostly revolved around implementing several action policy learning algorithms based on Markov Decision Process. [Zhaoming et al] propose an iterative approach that allows for a successive redefinition of an agent's reward function after each iteration of learning. New policies are trained using a policy gradient algorithm which then mixes RL-based policy gradients with gradient updates defined by Deterministic Action Stochastic State (DASS) tuples. This method was applied in training a real-world bipedal robotic agent that achieved stable walking with different gait styles at various speeds.

[Arun Kumar et al] utilize a Deep Deterministic Policy Gradient (DDPG) and reward shaping to design and simulate a planar bipedal agent in Gazebo. The agent was able to achieve a running gait at a speed of 0.83m/s. These gait control methods achieve state of the art performance on the learning of walking, but the learned gaits can hardly generalize to changing environment conditions. [2]'s works on bipedal walking on uneven ground selectively parametrizes the policy and adapts periodic behavior to gradual shifts in task parameters. They take an initial gait as a departure point, and iterate between modifying the task slightly, and adapting the gait to this modification. Similarly, we propose to modify the roughness parameter gradually and adapt the gait accordingly to the target rough terrain.

3 Technical Approach

We formulate the problem of locomotion as a Markov Decision Process (MDP). An MDP can be represented using a tuple $(\mu_0, S, A, T, R, \gamma, R, S_0)$, in which:

- μ_0 is the set of initial states, and we consider it as a hyper parameter to our model.
- S is the state space. It is represented by the tuple of hull angle speed, angular velocity, horizontal speed, vertical speed, positions and velocities of the 4 joints, hips and knees, the legs contact with ground, and lidar sensor data of the terrain.
- A is the action space. It is an array of torques applied on each of the joints.
- $T : S \times A \times S \mapsto S$ is the transition function to the new positions and velocities resulting from the previous torque inputs and state.

- $R : S \times A \times S \mapsto R =$
  $\begin{cases} cost_{linear\ velocity} - cost_{control\ inputs} & \\ -cost_{com\ based\ impact} + alive\ bonus, & \text{if the robot is alive} \quad \text{is the reward} \\ -100, & \text{if the robot falls} \end{cases}$
  function given for moving forward, total 300+ points up to the far end, where the less amount of torque are applied to motor, the higher score it receives.

The agent first learns to walk on the plane, and then adapts its policy to optimize the policy to walk on the target terrain F. To generate the tasks of decreasing roughness, we specify a sequence of damping roughness parameters $\{\alpha\}$. As for the environment, the change in dynamics is the physical rule that the agent follows, where the gravity is our primitive variant.

3.1 Learning Algorithm

For this project, we plan to train a bipedal agent to learn a work gait using DDPG. The DDPG algorithm is a model-free, online, off-policy reinforcement learning method. More specifically, a DDPG-trained agent is an actor-critic reinforcement learning agent that searches for an optimal policy that maximizes the expected cumulative long-term reward. During this learning process, the DDPG updates the actor and critic properties at each time step, stores past experiences using a fixed-length experience buffer, and updates the actor and critic using a mini-batch of experiences randomly sampled from the buffer.

To estimate its policy and value function, a DDPG agent maintains four function approximators:

- The actor, $\pi(S|\theta)$, with parameters θ, takes observation S and returns the corresponding action that maximizes the long-term reward.
- The target actor, $\pi_t(S|\theta_t)$. The agent periodically sets the target actor parameters, θ_t, to the latest actor parameter values which improves the stability of the optimization.
- The critic, $Q(S, A|\phi)$, with parameters, ϕ, takes observation S and action A as inputs and returns the corresponding expectation of the long-term reward.
- The target critic, $Q_t(S, A|\phi_t)$. The agent periodically sets the target critic parameters, ϕ_t, to the latest critic parameter values to improve the stability of the optimization. .

Moreover the algorithm updates state value estimates with the equation:

$$y_i = R_i + \gamma Q_t(S_i', \pi_t(S_i'|\theta_t)|\phi_t)$$

Where S_i' is an observed state after taking action A_i. γ is the discount factor, and R_i is the observed reward.

The critic's parameters are updated by minimizing the loss, L, across all sampled experiences:

$$L = \frac{1}{M}\sum_{i=1}^{M}(y_i - Q(S_i, A_i|\phi))^2$$

Where M is the number of sampled experiences, y_i is the approximated value function for state S_i.

The actor's parameters are updated using a sampled policy gradient to maximize the expected discounted reward as shown below:

$$\begin{aligned}
&\nabla_\theta J \approx \frac{1}{M}\sum_{i=1}^{M} G_{ai} G_{\pi i} \\
&G_{ai} = \nabla_A Q(S_i, A|\phi) \quad \text{where } A = \pi(S_i|\theta) \\
&G_{\pi i} = \nabla_\theta \pi(S_i|\theta)
\end{aligned}$$

Where G_{ai} is the gradient of the critic output with respect to the action computed by the actor network, and $G_{\pi i}$ is the gradient of the actor output with respect to the actor parameters.

To transfer the expert policy learnt from one environment to another, the agent either chooses an action based on the output probabilities of taking different actions from the teacher policy, or refers to value functions derived by the teacher policy. For better exploration ability, the two methods can be combined together with choosing actions from the expert policy being the exploring term.

3.2 Curriculum Learning

More recently, several lines of research have explored how tasks can be sequenced into a curriculum for the purpose of learning a problem that may otherwise be too difficult to learn from scratch. For these complex tasks, it may be more beneficial to gradually acquire skills over multiple tasks in sequence, where each subsequent task requires and builds upon knowledge gained in a previous task until knowledge from all the sources are directly transferred to the target. Curriculum learning typically consists of 3 main elements:

- Task generation, which creates a suitable set of source tasks
- Sequencing, which focuses on how to order these tasks into a curriculum
- Transfer learning, whereby a model learns a policy, parameter set or value function on a target domain by transferring the knowledge from another source domain.

Compared to training without a curriculum, we would expect the adoption of the curriculum to expedite the speed of convergence and may or may not improve the final model performance.

In this project, the target task is a highly uneven terrain, and the sequence of tasks are range of terrains with identical topological profiles that gradually transitions from a flat surface to the target terrain by progressively scaling the features of the intervening terrains until these features match those of the target terrain.

4 Simulation

The agent in this project was simulated in OpenAI's Gym environment. OpenAI's Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Space Invaders.

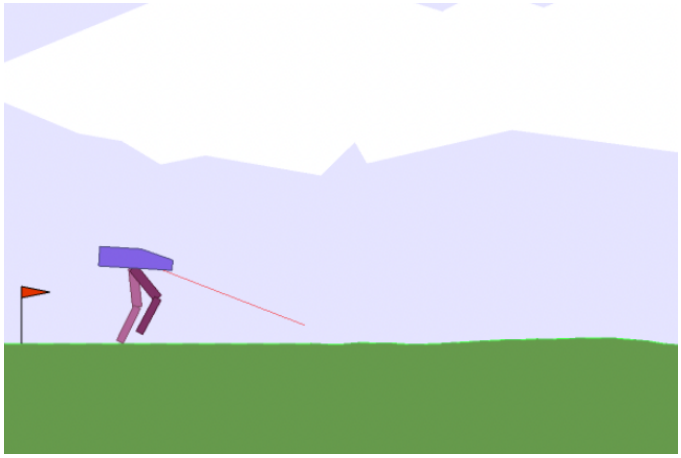


Fig 1

4.1 Simulated Environment

The agent was simulated in a pre-made 2D environment. The environment consisted of a terrain with options to add challenging features such as staircases, open pits and stumps at different points along the terrain.

The pre-made gym environment was customized to meet the requirements of the project. The simulation goal was to get the agent to move in the positive x-direction by a preset magnitude without falling over. A training episode ends if the agents accrues 30+ points or falls over. A target terrain was developed by adding rising stairs and descending stairs of a preset height at specified points along the terrain. A separate dynamic environment, similar to the previous one, was created to change every time the agent successfully completed an episode. This environment was initialized with a completely flat flat terrain and, after

each successful episode, the stair, identically positioned to the ones in the first environment, would appear with a gradual increase in height after each successful episode. The curriculum learning ends when the agent completes an episode with the stair of the same height as the those in the first environment.

While, our group had presented our project using Mujoco's 3D simulator, we decided against using Mujoco for two reasons: Firstly, Mujoco could only be installed on one of the group members computer. Because Mujoco doesn't have an implementation that ran on the new Mac M1 chips and could not be installed on the Tufts High Performance Cluster. Secondly, the much higher dimensionality of Mujoco's simulation (higher dimentionalities of the simulation's physical space; state space and action space) more computational resources and time.

4.2 Agent Design

The agent in this project consisted of a flat torso and two sets of legs. Each leg had two joint resulting in four controllable points on the agent. An image of the agent is depicted in Fig 1. The agent's state space consisted of the torso's angle, angular velocity, horizontal speed, vertical speed; the positions and angular velocities of all four joints; two boolean variables indicating the if the agent's two feet are in contact with the terrain; and finally eleven lidar range sensors that perceive the profile of the terrain.

The agent was given a positive reward every time step that is moved forward in the positive x direction in order to encourage it to move along the terrain; the agent was also penalized for actions that resulted in high torque values at the joint in order to limit the amount of energy expended in walking; the agent was also penalized in a manner proportional to the angle of its torso to encourage upper body stability; and finally the agent was given a reward of -100 every time it fell over. The equation of the reward function is shown below:

$$Reward = 130 \cdot X_{dist} - 5 \cdot |T_{angle}| - 0.00035 \cdot Tr_{total} \tag{1}$$

Where X_{dist} denote the x distance travelled by the agent in the a time step; Tr_{total} denotes the net torque applied to all joints on the agent and T_{angle} denotes the angle of the torso.

5 Experimental Results

The two agents in this project were trained using OpenAI's Pytorch implementation of DDPG for 20 epoch with 4000 steps per epoch. The parameter values used for the algorithm are clearly out lined in the "main.py" file in the project folder.

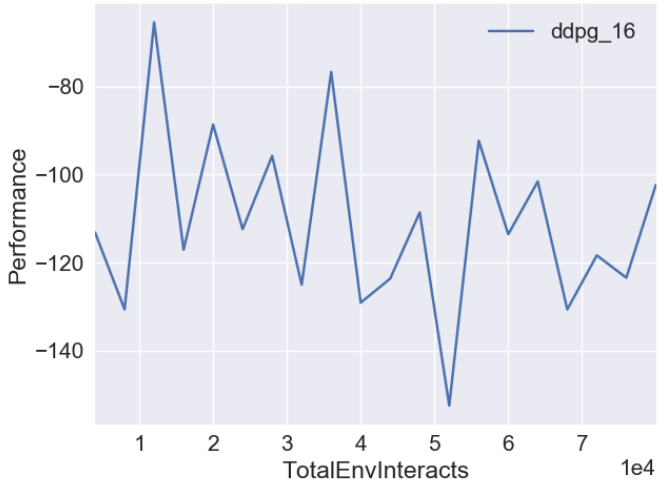


Fig 2: Average Reward vs. Episode Number for agent in trained in curriculum

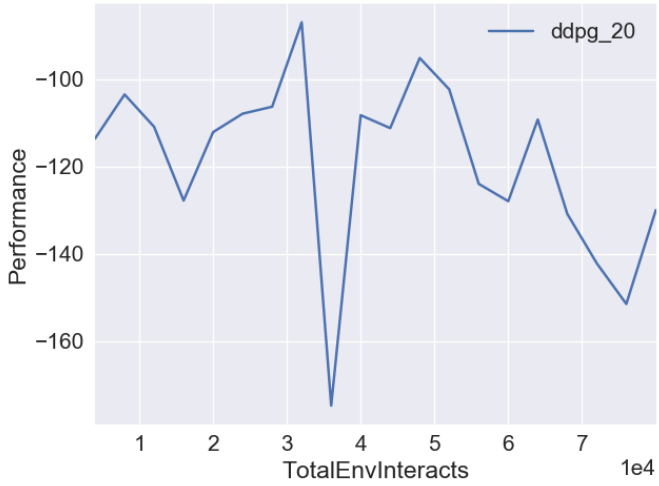


Fig 3: Average Reward vs. Episode Number for agent trained without curriculum

To evaluate our damping roughness approach, we compare the performance of an agent trained with the curriculum and one trained without the curriculum. Our metric for performance is the total reward per episode. Both plots start with approximately -100 because the agent receives -100 reward when it falls, which is a common case during the learning process. The policy trained with

the curriculum starts with a flat terrain, where the roughness is then gradually increased along the course of learning. Though the difficulty of task increases, its performance oscillates and remains around -100, as compared to the learning curve without our curriculum, where it continually meets dramatic drops. The policy trained with the curriculum exhibits substantially faster learning behavior. It is also noticeable that starting from around 40000 steps, both of the curves decline severely, which is a bottleneck for the degree of roughness.

However, trained with the curriculum, the policy adapts to the increased difficulty faster than the one trained without it, and outperforms with more alive bonus.

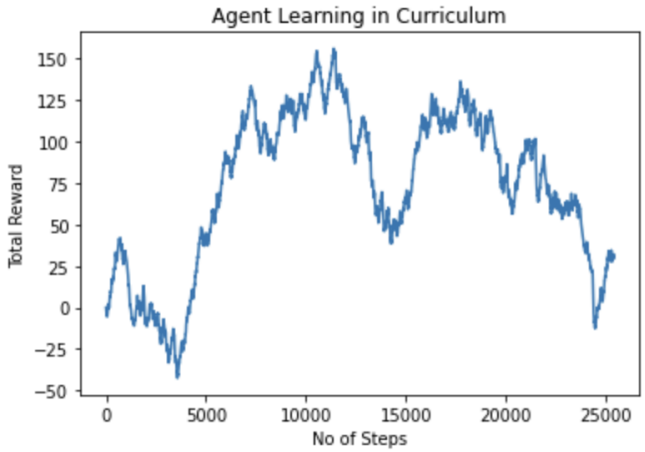


Fig 4: Average Reward vs. Number of Steps for agent trained in curriculum

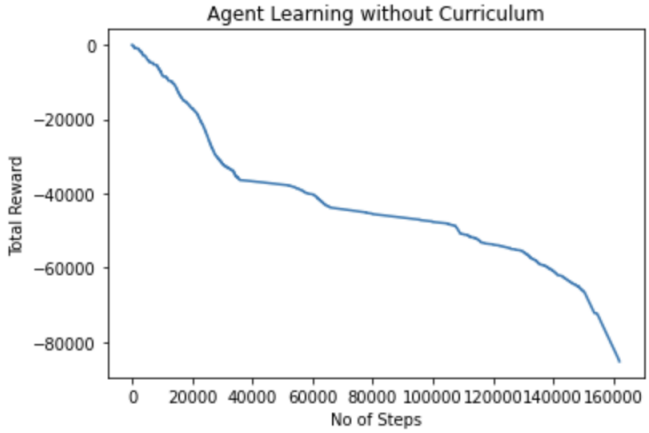


Fig 5: Average Reward vs. Number of Steps for agent trained without curriculum

We also analyzed the cumulative reward versus number of steps to track how the agent accumulated reward over the entire training process. Because the plots for reward vs number of steps were highly noisy for both agents, we decided on using the cumulative rewards as this depict the rate of progress of both agents. The results of the cumulative reward versus number of steps for the agent trained in the curriculum is depicted in Fig 4 and the cumulative reward versus number of steps for the agent trained outside the curriculum is depicted in Fig 5. In Fig 4, we can see that the agent's performance initially increases and the terrain is comparatively easier, and as the terrain gets more difficult, the rate of reward accumulated reduces. A "Video for Agent trained in the Curriculum" depicts the agents performance. We can see that this agent completes the curriculum in 25000 steps.

However, for the agent trained outside the curriculum, the agent accumulates negative reward at a fairly linear rate. While these values were unexpected for the agent trained without the curriculum, a "Video of the Agent without the Curriculum" depicts the agent constantly falling at the starting point even after 160000 steps! During certain training runs, for agents, with and without the curriculum, we noticed that the agents sometimes get stuck repeating the same movements at the beginning of the terrain. However, more often than not, the agent trained in the curriculum successfully completed the task. However, the agent trained without the curriculum never advanced past the starting point in the terrain.

6 Conclusion

In this project, we have compared the performances of two biped agents trained to walk on an uneven terrain: one trained in a curriculum whereby a series of intermediate terrains were gradually scaled, in terms topological complexity, to a target terrain; and another trained directly on the target terrain. The results of this effort show that the curriculum provides a viable path to training a biped agent to successfully navigate a complex terrain. The agent trained in the curriculum was able to successfully navigate the terrain in 25000 steps; while the agent trained directly on the curriculum never successfully navigated the training even after 160000 steps.

References

[1] MHP Dekker. Zero-moment point method for stable biped walking. *Eindhoven University of Technology*, 2009.

[2] Tom Erez and William D. Smart. Bipedal walking on rough terrain using manifold control. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1539–1544, 2007.

[3] Keyu Duan Dongbo Xi Yongchun Zhu Hengshu Zhu Senior Member IEEE

Hui Xiong Fellow IEEE Fuzhen Zhuang, Zhiyuan Qi and Qing He. A comprehensive survey on transfer learning. *Autonomous Robots*, 27(1):55–73, 2009.

[4] A Kumar, N Paul, and S Omkar. Bipedal walking robot using deep deterministic policy gradient (2018). *arXiv preprint arXiv:1807.05924*.

[5] Ill-Woo Park, Jung-Yup Kim, Jungho Lee, and Jun-Ho Oh. Online free walking trajectory generation for biped humanoid robot khr-3 (hubo). In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1231–1236. IEEE, 2006.

[6] Matthew J Powell, Huihua Zhao, and Aaron D Ames. Motion primitives for human-inspired bipedal robotic locomotion: walking and stair climbing. In *2012 IEEE International Conference on Robotics and Automation*, pages 543–549. IEEE, 2012.

[7] Michael Taylor, Sergey Bashkirov, Javier Fernandez Rico, Ike Toriyama, Naoyuki Miyada, Hideki Yanagisawa, and Kensaku Ishizuka. Learning bipedal robot locomotion from human movement. *arXiv preprint arXiv:2105.12277*, 2021.

[8] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan Hurst, and Michiel van de Panne. Iterative reinforcement learning based design of dynamic locomotion skills for cassie. *arXiv preprint arXiv:1903.09537*, 2019.

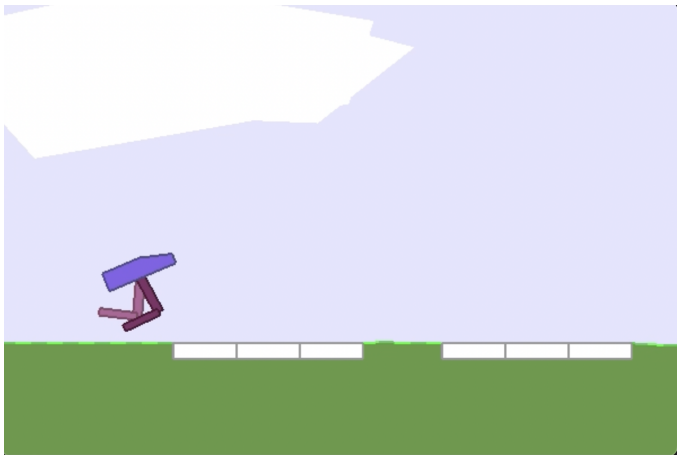
[9] Ruohan Zhang, Faraz Torabi, Lin Guan, Dana H Ballard, and Peter Stone. Leveraging human guidance for deep reinforcement learning tasks. *arXiv preprint arXiv:1909.09906*, 2019.

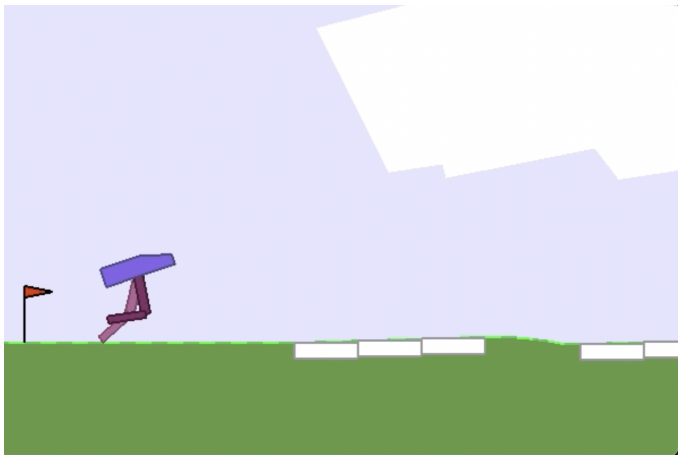
Appendices

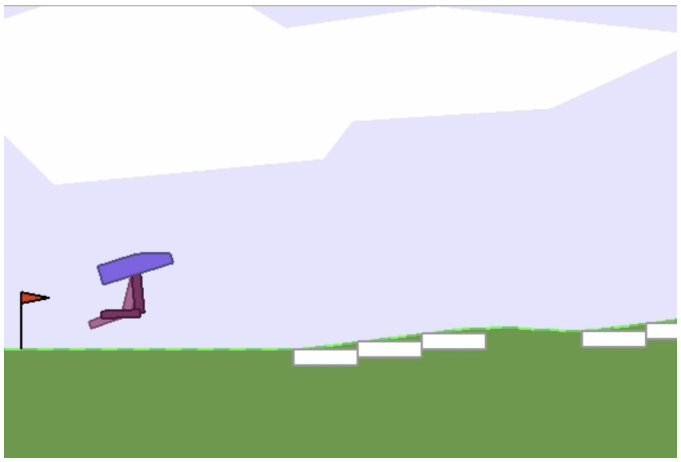
Appendix A Challenges

The main challenge we had in this project was setting up the simulation environments for the agents. Our group spent approximately 80+ hours downloading dependencies, changing source files, and finally discovering that most new computer architectures were incompatible with Gym and Mujoco. We attempted running both Gym and Mujoco environments on four different computers, the Tufts High performance cluster, and Google Colab, but none of these worked. The two environments only ran on one of our group member's computer that was 8+ years old. This severely limited the computational resources available to us our ability to code in parallel.

Appendix B Curriculum







Appendix C Important Links

- "Video for Agent trained in the Curriculum"
- "Video of the Agent without the Curriculum"