ISSN No:-2456-2165

Environment Interaction of a Bipedal Robot using Model-Free Control Framework Hybrid off-Policy and on-Policy Reinforcement Learning Algorithm

Mr. Pawan Kumar Mishra
(Assistance Professor)
Uttarakhand Tech. University, Dehradun

Prem Prakash
(M. Tech (CSE)
Uttarakhand Tech. University, Dehradun

Abstract:- In the reinforcement learning Algorithm, there are different kinds of environments, one of them is a continuous environment. In the continuous environment, the data sources change very rapidly. Due to this capricious in the data, it's difficult to train the agent using reinforcement learning. We have used the hybrid algorithm of DDPG and PPO. The DDPG is the off-policy algorithm that uses the old policy to train the agent, if we use the past observation to get obtains a new policy is not considered good as it brings instability in learning. DDPG is also data-efficient meaning if the collect number of the past old policy before if update a new current policy which makes them difficult to tune and unstable. PPO is the on-policy algorithm that focuses on keeping the new policy nearly close to the old policy. They have better sample complexity as they update multiple updates of mini batches of the data collected from the environment. And it's easily tuned. We have combined these two policies (on-policy and off-policy) by taking the data efficiency from the off-policy algorithms and using the high variance gradient of on policy, which helps in a large number of samples and distributed training the agent. However, combining the on-off policy algorithms is difficult to find the hyper parameters which are suitable to govern this trade-off. In this proposed algorithm we update the number of off-policy with each on-policy update. We used specialized clipping in objective function epsilon ' Ɛ ' to remove the incentives to keep the new policy as near as the old policy. In the experiment, we used the open AI gym Box2D benchmark Biped walker and biped walker Hardcore.

I. INTRODUCTION

The learning rate of a Biped Robot in a continuous action space using the model-free framework of reinforcement learning. As in the discreet action space, the agent could not explore as much the environment. Model-based frameworks need pre-information about the environment. As if the well-learned agent came across any different kind of environment the agent has the chance to collapse. Using the continuous action space of the agent is very difficult to intact the learning of the agent vital. As in the continuous environment, the action space of the agent can be large for one particular observation of the environment, and selecting the best action of the agent might take longer to train it. The problem associated with the training of the agent are:

- If the agent has no information about the environment
- If the agent has continuous action space
- The data set that uses to train the agent, some of the data set can be old which lets the change the new policy and take the training time longer.
- If the training is done on every single action which won't be having enough data of the environment to train the agents efficiently.
- We have worked on the measure or the goal of our research to find a way to out to overcome these issues in the training and learning techniques in machine learning.
- the objective is the interaction of a biped robot in a continuous environment using model-free reinforcement learning. The following objective of the research can be summarized as
- Dealing with the problem of the biped robot in the continuous action space environment.
- The agent should learn itself from the environment without any information about it faster.
- To increase the rewards and bring the learning stability of the agent in the different environments.
- To increase the rewards of the agent in the continuous environment.
- To improve Exploration and exploitations of the environment.

The theory and discussion of the research or the method we have used in this the interaction of a biped robot in the different environment to gain the maximum reward and find out the best reward function which brings the stability of the agent in an unknown environment.[19] The environment in our research is kept continuous which has multiple actions in closed action space. The main aim of the research is to keep the findings relevant which brings more information to improve this research and discussion on a wider scale. Since in the model-based reinforcement learning the knowledge of the environment is required to start the learning algorithms in the agent . which brings difficulty when if have no information or the how the other world is going to be, hence we needed to train the agent without any information about this environment. to train our agent we have used model-free reinforcement learning algorithms.

The DDPG reinforcement algorithm[25] is used to have the best sample size in the training data set for the agent. But this algorithm has the problem of using the older data which may differ in the new policy for the rewards function. Which makes the training of the agent bit unstable because sometimes the new policy and old policy deal with great differences, hence to overcome we have used another model-free reinforcement algorithm PPO which used the action based on the current policy[23]. And bring stability in the training of the agent. It uses the advantage function which does not let the new policy far from the old policy. By using the combination or hybrid of these two algorithms we can train our agent in more stable and less time. DDPG is the off-policy algorithms that train the agent after having the finite number of data in the databases and by collecting all the information from that data utilizes in the training of the agent. But few of the data in the databases might be old. On the other side, PPO is the on-policy-based reinforcement learning algorithm that used single data set to train the agent and have a rewards policy and select the new action. We have a hybrid of these off policy and on-policy reinforcement learning algorithms. IN this the off-policy brings a good sample data set for the information about the environment and the on policy deal with the stability and fast learning of the agent.

II. LITERATURE REVIEW

The author of the paper AO XI and Chao Chen,2020 has combined the model-based and model-free reinforcement learning framework. They have developed a hybrid algorithm to stabilize the movement of the agent in the moving environment. The model-based framework of reinforcement learning takes the job of an estimator, it estimates the number of observations and actions of the agent and the environment. then the data is passed through the Gaussian Processes which is also a model-based reduces the –observation-action pair from the data. then the learning is done with a model-free framework using the DDPG algorithm. This hybrid algorithm provides the two best parts combining the two different frameworks and the mismatch of any observation-action pair of the agent and the environment.[1]

In this paper, Rasool Fakoor, Pratik Chaudhari, Alexander J. Smola, 2019 proposed an algorithm P3O(Policy-on Policy-off Policy optimization) which combines the two kinds of algorithm on-policy and off-policy. The combination of these two algorithms improves the sample complexity and as well as the stability of the reinforcement learning algorithm. In the algorithms runs the on policy and for each on policy, the off-policy algorithms interleaved. They have used the effective sample so the old policy and new policy remain as close, getting far from the range of previous policy and new policy can bring instability. They have used the best of on-policy and off-policy in their algorithm. Off-policy improves the sample complexity of the reinforcement learning algorithm and on-policy brings stability.[14]

The algorithms used are given in the paper, John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, 2017 they have introduced a new algorithm of reinforcement learning based on policy gradient. This algorithm uses the stochastic gradient ascent to optimize the objective function on sampling data through interaction with the environment. They have performed multiple epochs and mini-update of a policy gradient. This algorithm is PPO (Policy Gradient Optimization) which is the better improvement of TRPO (trust Region Policy optimization), which gets much more sample data as well as easy to implement. They have tested on Atari games as well benchmark tasks.[27]

III. PROBLEM FORMULATION

To increase the efficiency and improve the stability of the agent in the unknown environment continuous action space.[27] The discrete action space has limited outputs of the action. Taking the example of the Lunar Lnader of the Open AI Gym environment, it is a tool kit of a library that provides an in build environment to test reinforcement algorithms, And Lunar Lander is one of the environments which provides both discrete action space and the continuous action space of the same environment, so it will easy to compare and understand the problem I am going to describe. Lunar Lander is an environment where there is a landing pad that is fixed to its position and there is a space shuttle or a lander. And the job is that we have to land on the landing pad. The discrete action space has four kinds of outputs these are moving the lander engine left, right, main, and nothing. These output actions are numbered from 0 to 3 (n-1 for the agent whose discrete action space is n starting from 0) and one of the outputs is selected whose probability value is maximum. Where in the Lunar Lander Continuous environment the continuous action space Box(2) means that the agent has two kinds of output, one output will focus on the main engine of the agent control, and the other output will focus on the moving of the agent control left or right. So in the continuous action space, the output can range from Box[-1 to 1, -1 to 1] (-n to n for the agent whose action space is n) whom to get the normalized values of the output in the given range we use TanH.

The action space of the biped robot switches from discrete action space to continuous action space.

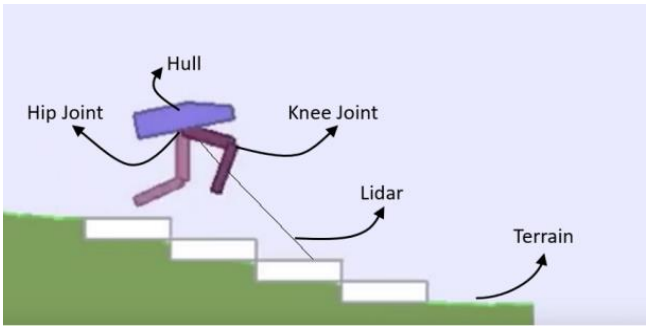


Fig. 1. The biped Robot

The main focus of this paper is to make the biped robot to adapt the environment and increase the speed of walking in the unknown environment without tripping and minimum energy is utilized. To walk for a biped Robot we have to

give some force to the joints of the biped robot. The force is given on the joints of the agent that is it has the two-leg and both the legs have two more joints (if we consider the multi-legged Robot with m leg and each leg has the n joints then the action space is equal to m X n) hence the action space in this biped Robot is 4, the action space in the continuous space is in range (-1, 1) to all four joints. The way the biped robot tends to walk when the training is not done, due to lack of stability and the balancing in the agent. The value of the continuous environment the action space ranged between -1 to 1 which means the action has held more the 100 actions in this range or even more. And at the time of observation, the agent or the biped robot in this has to select the best action from in-between ranged -1 to 1 then action which is selected will help to bring the stability in the joints of the biped robot and help it to improve its walk in the future.

IV. METHODOLOGY

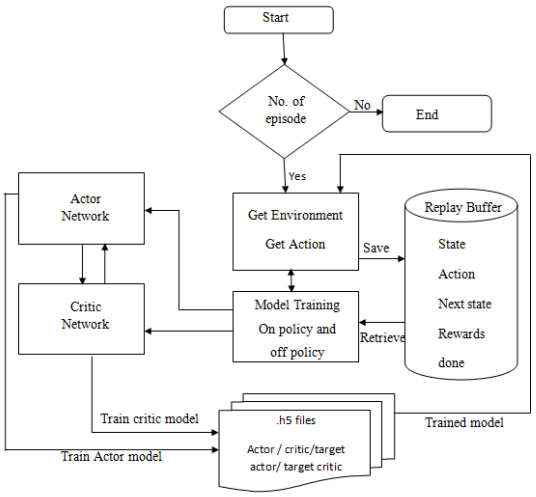


Fig 2.Flow Chart of Hybrid Model-Free Policy (HMFP)

A. *Hybrid Model-Free Policy (HMFP) Reinforcement Algorithm*

Step 1 input policy parameters Θ
Initial value function parameters' Φ
Initial the target parameters same as the main parameters

$$\theta_{\text{targ}} \leftarrow \theta, \ \phi_{\text{targ}} \leftarrow \phi$$

Step 2 for episode 1,2 , ….
Step 3 collect the data set D sample of the data by the policy $\pi_k = \pi(\Theta_k)$
Step 4 save the D (s,a, r,s',d) into the replay buffer
Step 5 calculates the advantage A on the current value function.
Step 6 update the policy using on-policy algorithms and using the clipping 'e '

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, A^{\pi_{\theta_k}}(s_t, a_t), \ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right)$$

Using adam optimizer and stochastic gradient ascent

Step 7 for every on-policy update doff-policy update j 1, 2… n epochs
Step 8 number of sample batch D_0= (s,r,a,s',d)
Step 9 compute target

$$y(r, s', d) = r + \gamma(1 - d) Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

Step 10 using gradient descent we update the Q- function

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_\phi(s, a) - y(r, s', d))^2$$

Step 11 using gradient ascent updating off-policy using off policy

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \mu_\theta(s))$$

Step 12 updating target network

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho)\phi$$
$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta$$

B. *Hybrid Model-Free Policy (HMFP)*

This algorithm is the hybrid algorithm of the on-policy and 0ff-policy taking the benefits of both the algorithms in which the use of clipping function from the on-policy which does not let the new policy go far from the old policy which make the learning the agent in the new environment much more stable. While sometimes the lack in information the sample efficiency is well known required which is given by the off-policy algorithm. using the replay buffer it stores the experience which used to update the new policy. The algorithm focus on the continuous environment space in which the action is bounded between the positive and negative integer eg. -1 to+ 1. Now in this, we can have more than hundreds of actions and anyone can be the best action concerning the state. to make the prediction we have used the two models which are actor and critic. The actor model of the algorithm selects the best of the action on the knowledge of the new and old state values. This action will make our agent move in the environment.

The actor loss is given by adding the actor policy of on and off-policy

Actor loss by on and off policy is given by using the formula

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a),\right.$$

$$\left.\text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right)A^{\pi_{\theta_k}}(s,a)\right)$$

The action is been selected using the sample applying the distribution which is defined by the action space we have used the normal distribution which gives the log probability.

$$\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The critic model is the name suggests is to critic the actor if the action which is taken is best for that state or not and give the information back to the actor they both are linked to each other.

When the action is selected and sent to the environment it receives another state, if that state achieved by that action is good for the agent then it is considered as the positive and positive reward is given and if that action didn't lead us to any positive, then the negative reward is applied. The motive of the policy is to get the maximum rewards in an episode. These rewards are sent to the critic network for the training.

The loss function of the critic is given by the mean squared bellman error

$$\nabla_\phi \frac{1}{|B|}\sum_{(s,a,r,s',d)\in B}(Q_\phi(s,a) - y(r,s',d))^2$$

C. Algorithms Used

a) DDPG

DDPG stands for deep deterministic Policy Gradient. This is the combination of the two other algorithms is DPG (Deterministic Policy gradient) and DQN (deep Q network).[25] The algorithm takes the policy gradient from the DPG algorithm, which policy gradient is which is learned from the model-actor and the q value is taken from algorithm DQN which is learning from the model critic network. In this the policy and q- values are learned together, that's why we need two model networks separately in the algorithm. But in this algorithm to learn the policy, we use the q-function, and to learn the q-function the Bellman equation is used. A large number of samples is collected before the agent is trained, then these values are sent to the critic to learn the q-function, that's why the DDPG algorithm is considered as the off-policy method. The action is selected without using the probability distributions. Just in the DQN, we have to take the argmax of all the action values which is q-function data or we can say Q-values., similarly this algorithm also uses the q-values of all the actions then the action is selected directly this makes the action continuous. The best action can be represented by

$a^*(s) = \arg\max_a Q^*(s,a)$

where a*(s) is the action selected over the state s
$\arg\max_a Q^*(s, a)$ is the maxima value of the q-value of all state-action pairs.

As the action that we got it directly from getting the maxima of the q-values of all the state-action pairs makes the policy deterministic. In DDPG having the deterministic policy, the agent can not explore the environment well after a few hundred episodes. Hence, the noise is been added to the algorithms to increase the high-scoring gallop. This algorithm is proposed for the continuous action space, in discrete action, the q-values can be calculated for each action and get the maximum q-value of the state-action pair. But in the continuous environment Q*(s, a) is differentiable wrt the agent value this helps not to use the loop to run the section of the code to calculate max (s, a) when a new action has to be selected by the agent in the environment.

The Bellman equation:

$$Q^*(s,a) = \underset{s'\sim P}{\mathrm{E}}\left[r(s,a) + \gamma \max_{a'} Q^*(s',a')\right]$$

Where s' ~ P is the next state
s' is the sample from the distribution P(.|s, a)
the bellman equation represents the best action values of the Q*(s, a), let us assume that there is a neural network $Q_¥$ (s, a) with the data set D(s, a,r,s',d),

where, ¥ is the parameter of the Q*(s, a)
D is the data set, s is the current state where the agent is right now an is the action it took, s' is the next state at which the agent reached, d is the if the all the state is covered or no new stated can be achieved the terminal state to calculate the loss function the mean–squared bellman error function is used to know how much the neural network is close to the Bellman equation

$$L(\phi,\mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}}\left[\left(Q_\phi(s,a) - \left(r\right.\right.\right.$$

$$\left.\left.\left. + \gamma(1-d)\max_{a'} Q_\phi(s',a')\right)\right)^2\right]$$

Where, (1-d) if d values are True (1) there is no reward as the state is terminal, (1-d) if d values are False (0) the exploration of the environment is on the main aim of the DDPG algorithm is to reduce the error of the mean squared bellman error as low as possible. We can achieve this by using one way is Reply Buffers in the replay buffer is simply a database where we going to store all the experiences of the agent from the environment. the size of the reply buffer has to be selected as per the number of the episode are we going to run, we cant store and save everything things. Using all the experience of the agent to learn very slow nut we cant make the agent learn from only a few data otherwise it will break. Another way to reduce

the mean squared bellman error is to use the Target Network.

$$r + \gamma(1-d)\max_{a'} Q_\phi(s', a')$$

As the error is reduced the q-function will going to look like the above equation. But we are training the parameter ¥ and this parameter ¥ also depend on the target equation which will be going make the MSBE not stable, to avoid this we have to use another network for actor and critic which will save the parameter ¥ while it is getting a train and after a delay, the target network is updated with the new value of parameter the scan be written as $¥_{targ.}$ The delay or the lag is done by Polyak averaging ' p '.

The updating equation is:

~~¥targ~~ p $¥_{targ}$ + (1 - p) ¥

The stochastic gradient descent is used to minimize the MSBE

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}}\left[\left(Q_\phi(s,a) - \left(r + \gamma(1 - d)Q_{\phi_{\mathrm{targ}}}(s', \mu_{\theta_{\mathrm{targ}}}(s'))\right)\right)^2\right],$$

after learning the q-function its is used to calculate the policy. The deterministic policy can be learned $\mu_¥$ (s) so that the maximum $Q_¥$(S.a) values are generated by that selected action. In the continuous action space, the q-function is differentiable wrt action and gradient ascent it to use.

$$\max_\theta \underset{s\sim\mathcal{D}}{\mathrm{E}}\left[Q_\phi(s, \mu_\theta(s))\right].$$

Q-function is constant

b) PPO

This algorithm is the improvement of the TRPO, where the TRPO used the second derivatives or the second-order which is very complex to understand ad to implement in code as well.[23] The PPO algorithm used the first-order method. In this, the algorithm does not let the new policy have the high margin values from the old policy which brings stability in the learning of the agent in the unknown environment. There are two ways the new policy remain close to the old policy.

We can use the PPO-Penalty in this the KL-constrained this method is used in the TRPO. In this, the objective function is given a penalty but as the training progress, the objective function changes the coefficient which makes it a weak constraint. Being the week we can use another method known as PPO –clip we do not use any constraint, but the put the clip cut the objective function if the new policy tends to go far from the old policy. this algorithm is on-policy because it does not update the policy using the old saved experience of the agent but start working on getting the action from the previous action. The objective function is :

$$\theta_{k+1} = \arg\max_\theta \underset{s,a\sim\pi_{\theta_k}}{\mathrm{E}}\left[L(s, a, \theta_k, \theta)\right],$$

We take the set of the mini-batch of the data set d :

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a),\right.$$

$$\left.\mathrm{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right)A^{\pi_{\theta_k}}(s,a)\right)$$

We have used another hyperparameter e epsilon which is used to keep the new policy as near as the old policy. When the advantage is positive the above equation is taken from in this equation given below. As the action is more chance to get selected the objective gains its value and we have clip the value not to increase from the threshold so that the old policy remain as closes the old policy and vice versa with the advantage function with negative.

When the advantage is positive ;

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1+\epsilon)\right)A^{\pi_{\theta_k}}(s,a).$$

When the advantage is negative:

$$L(s, a, \theta_k, \theta) = \max\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, (1-\epsilon)\right)A^{\pi_{\theta_k}}(s,a).$$

The latest policy is been used to explore the agent in the environment as the PPO uses stochastic policy. The exploitation may occur because the increase of the training and update rules the policy because less random.

V. RESULTS

The thesis experimented are carried on the openAI gym environment Biped Walker and Biped Walker hardcore. And compared to the work of AO XI AND CHAO CHEN in "Walking Control of a Biped Robot on Static and Rotating Platforms Based on Hybrid Reinforcement Learning"[2] though having a very vast area I have considered only a few permanents of their work and tried to improve the learning and stability to the biped Robot this is the part of the research done to see if biped Robot adapt the environment hybrid of model-free on-policy and off-policy to the hybrid reinforcement algorithm [19] of the model-based and model-free reinforcement learning algorithm.

The selection of the hyperparameter is very difficult when we have to combine the two different frameworks to get better results. These hyperparameters have suited well to give better results. Howe ever the more changes in the hyperparameter can let the different results.

A. The Result of the Biped walker

The Biped walker is the environment where the simple environment is given to the agent whiteout any hurdles. It

helps to make the agent walk. Increasing the movement speed.

Running both algorithms on the same environment Bipedwalker we compare these algorithms HRL and HMFP. We can see the HPL [2] took much time to complete the 1000 episodes of the environment. The reward in HRL was maximum at episode 446 and avg reward of 100 episodes is at 584 which later starts to decrease. In HMFP the rewards tend to increase concerning the episode.

The rewards of the 100 episodes were taken and plotted on the where we can clearly show that, the increase in the average of rewards with the increases number of an episode which means that agent the agent explores the environment. The graph average rewards in HRL decrease as the number of training increases.

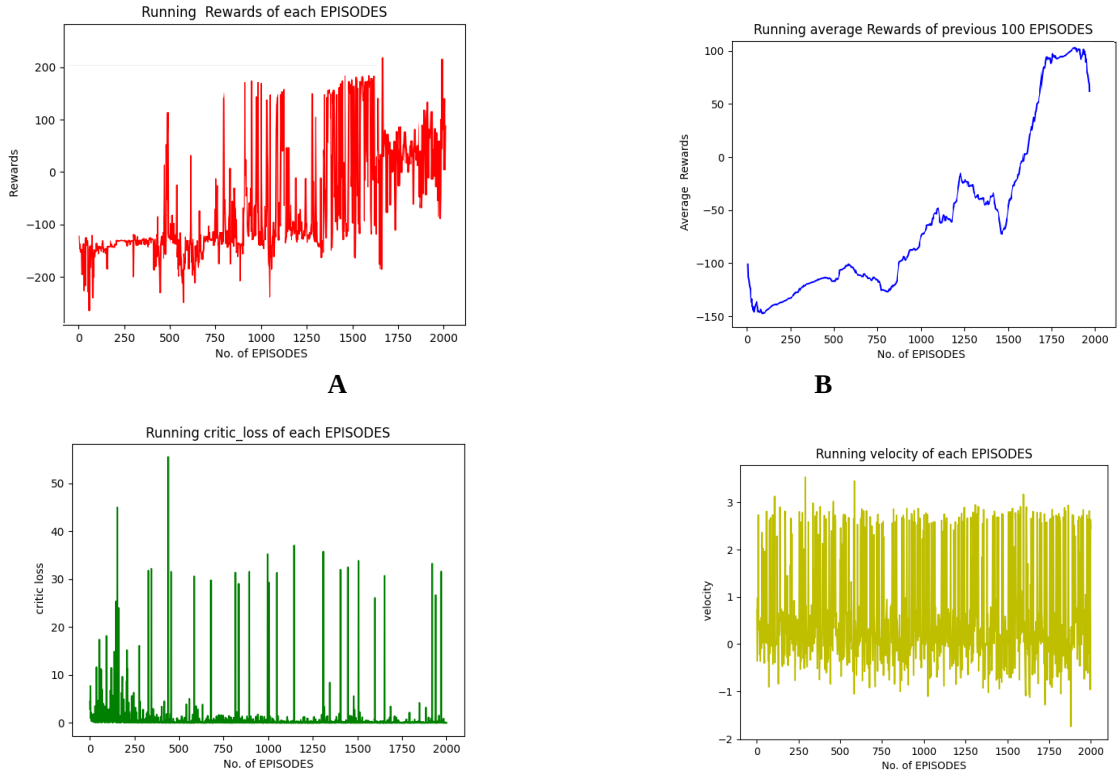
B. The Result of the Biped walker Hardcore

This environment is as similar as the previous environment Bipedwalker but in bipedwalkerHardcore come with huddles like a pothole, stairs which makes the path for the agent bit difficult. This environment keeps changing every episode so that the agent can be trained well in the new environment. The time of each episode is been increased than the previous environment due to the toughness.

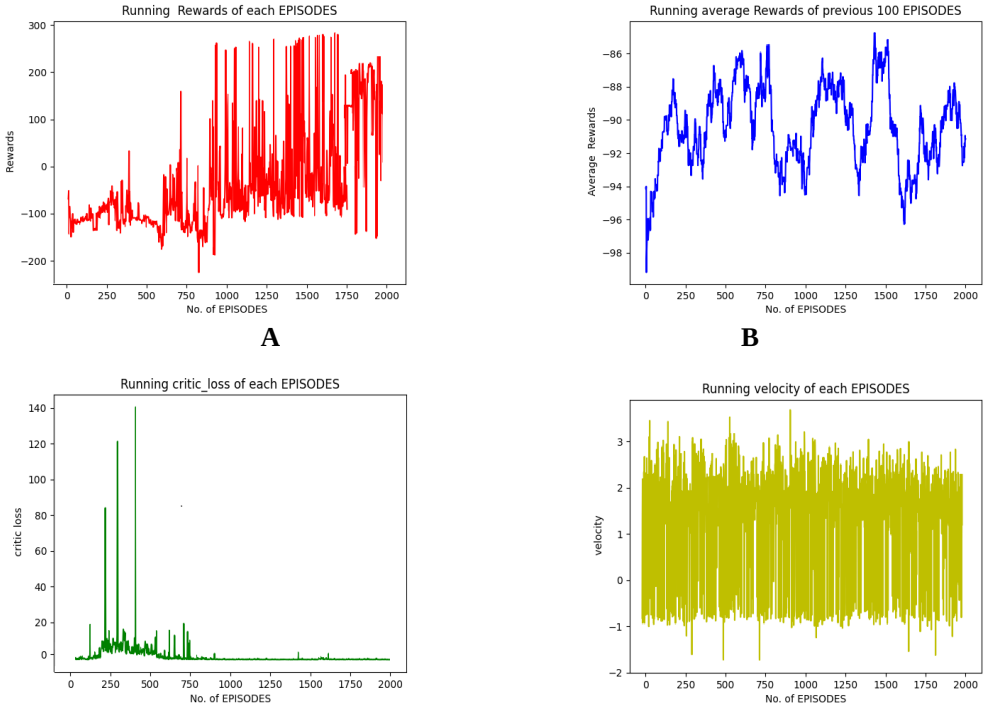
In the second environment, the HMFP gives better results than the HRL algorithm. The average rewards are consistently increasing.

VI. DISCUSSION

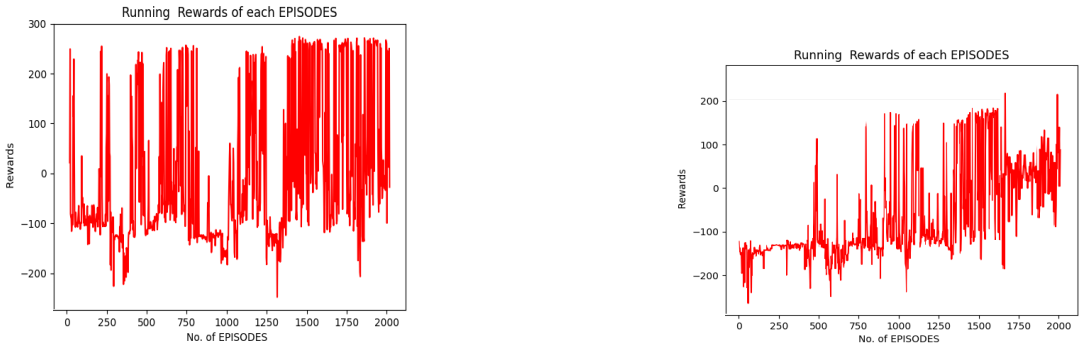
The thesis of AO XI AND CHAO CHEN need prior information about the environment so that their offline estimator work to reduce the number of state action from the set. Which may have an issue if that set of state actions would get important to a later state. This has easily been resolved using the hybrid of model-free on-policy and off-policy reinforcement learning algorithms. As when we applied the on policy reinforcement algorithm which has the low sample efficiency to let the agent explore the environment. The agent focuses on the single and the current policy of the algorithm. When we use the off-policy it improve the sample complexity for the on–policy reinforcement learning algorithm as it uses the replay buffer and stores that the number of experience. We have used those experiences of the off-policy to train the agent for 10 epochs for every single on-policy update in an algorithm. Since some of the numbers of experience may be too old to update a new policy here the on-policy reinforcement algorithm help to keep the algorithm stable. We have used the on policy chipping method using the advantage on the objective function. The hyper parameters e epsilon is used to chip the objective function when the advantage tends to go high or low from the previous policy from the nearby threshold of the previous old policy. This brings the faster and brings stability in the learning of the agent.

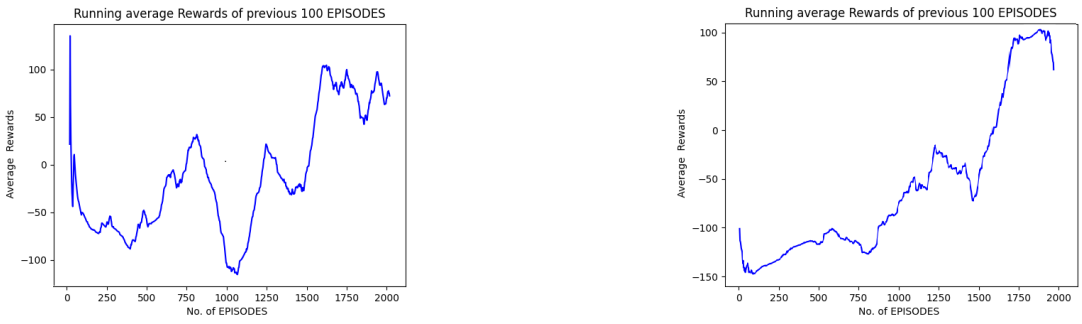


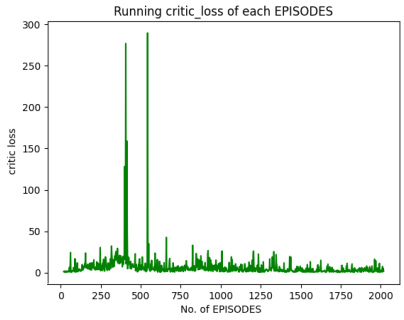
Graph 1 Line graph of Result HMFP BipedalWalker A) Rewards per episode B) Average Rewards(100ep) C) Critic Loss per episode D) Velocity per episode



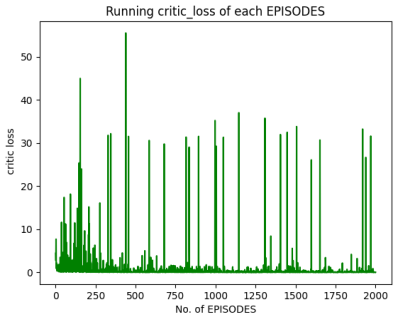
Graph 2 Line graph of Result HMFP Bipedal Walker Hardcore A) Rewards per episode B) Average Rewards(100ep) C) Critic Loss per episode D) Velocity per episode



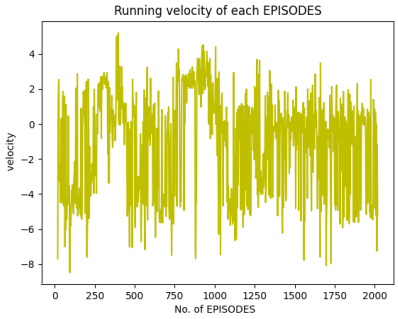




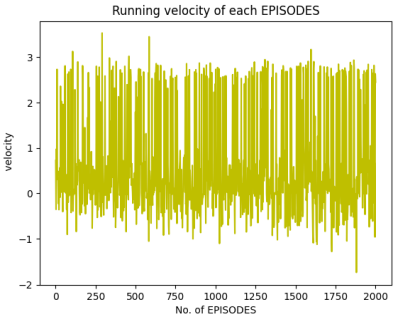
HRL



C HMFP

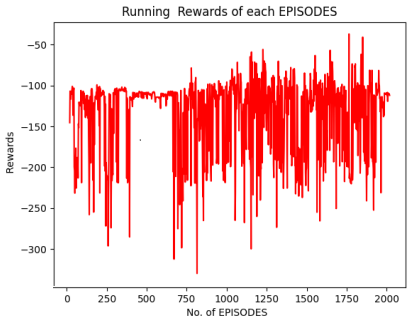


HRL

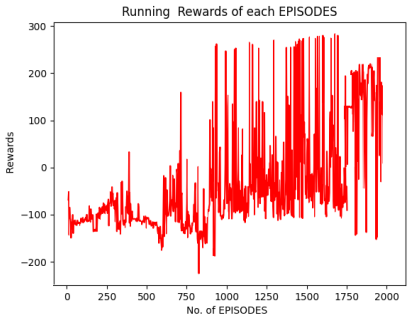


D HMFP

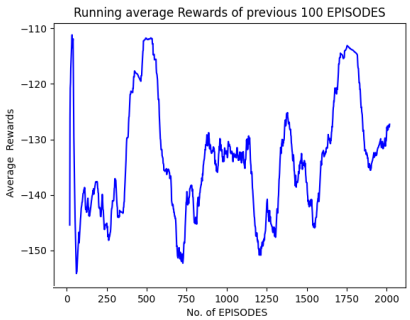
Graph 3 Line graph of Comparison between HRL and HMFP BipedalWalker A) Rewards per episode B) Average Rewards(100ep) C) Critic Loss per episode D) Velocity per episode



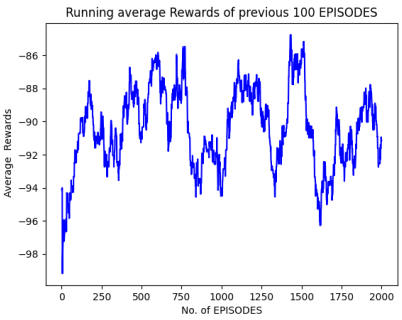
HRL



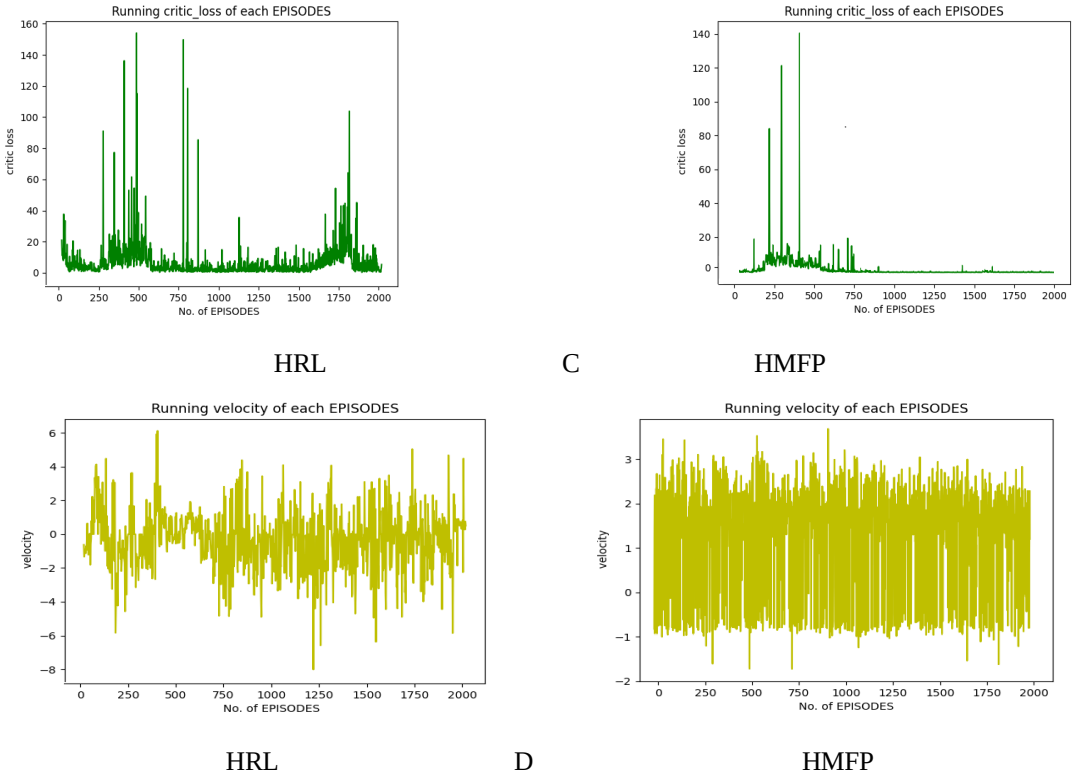
A HMFP



HRL



B HMFP



Graph 4 Line graph of Comparison between HRL and HMFP Bipedal Walker Hardcore A) Rewards per episode B) Average Rewards(100ep) C) Critic Loss per episode D) Velocity per episode.

Parameters	HMFP
Maximum reward	+248.024
Max Avg rewards (100 ep)	-87.0346
Critic Loss	0.00061
Avg velocity	2.84 m/s
Time taken	59 hour

Table 1 Trained HMFP Bipedal Walker

Parameters	HMFP
Maximum reward	+ 244.0028
Max Avg rewards (100 ep)	+ 98.2853
Critic Loss	0.0124
Max velocity	3.64 m/s
Execution time	52 hours approx.

Table 2 trained HMFP Bipedal Walker Hardcore

Parameters	HRL	HMFP
Maximum reward :	+242.0779	+ 244.0028
Max Avg rewards (100 ep)	+ 124.6484	+ 98.2853
Critic Loss	0.0256	0.0124
Max velocity	4.3 m/s	3.71 m/s
Time taken	61 hour	52 hour

Table 3 Comparison between HRL and HMFP BipedalWalkerHardcore

Parameters	HRL	HMFP
Maximum reward :	-42.7904	+248.024
Max Avg rewards (100 ep)	-112.2544	-86.4025
Critic Loss	0.00456	0.00091
Avg velocity	6.24 m/s	2.46 m/s
Time taken	67 hour approx	59 hour approx

Table 4 Comparison between HRL and HMFP Bipedal Walker Hardcore

VII. CONCLUSION

The study, Hybrid Reinforcement learning algorithm that consists of on-policy and off-policy model-free control framework was proposed and applied on continuous action space bipedal walker gym environment to increase the learning efficiency and gain the maximum rewards in less time. PPO algorithm is used as the on-policy framework which uses multiple epochs of stochastic gradient ascent to perform each policy update which has stability and reliability to the network. But it only updates a policy per action only once here DDPG is used as an off-policy this algorithm solves problems related to continuous action space and the raw pixel for observation. It collects a large number of samples to update the policy of the agent which increases the sample efficiency of the algorithm. However, some of this sample may be too old to update the new policy. So update off-policy with on-policy updates. Some epochs updated the number of off policies for each on-policy update in the proposed algorithms. The proposed Hybrid on-policy and off-policy reinforcement learning framework avoid the distribution mismatch problem when integrated with off-policy reinforcement learning with on-policy reinforcement learning. The simulation results show that the proposed Hybrid on-policy and off-policy reinforcement algorithm was able to control a bipedal walker robot and bipedal walker hardcore and achieve faster learning and efficiency in a different environment.

VIII. FUTURE WORK

In this work hybrid on-policy and off-policy model-free framework reinforcement learning. The PPO algorithm on policy is adopted to increase the reward policy of action of the agent and the DDPG algorithm off-policy increase the sample size of the experiment. Due to the reduction of the sample complexity and the increase of the efficiency of the state-of-the-art algorithm in increasing the rewards and faster learning in the unknown environment of a bipedal robot. This technology can be implemented over different kinds of robots. Which have more than two limbs (multi-legged robot)[26]?

Robotics has reached a milestone with the successful introduction of robots into humanoid manufacturing for searching new places in the different worlds. Robots in automotive manufacturing are also used for welding and painting, and these are two areas where robotic usage is almost universal to space as well as on the earth. There are other areas where the usage of robotics and this chapter is dedicated to brief descriptions of these fields along with a quick assessment of their status. More studies can be carried out on the learning effect of multi-legged robots if one of their limbs is lost[22]. The data-driven reinforcement algorithm can be implemented to hybrid on-policy and off-policy model-free control framework robot-environment interaction.

REFERENCES

[1.] Xi and C. Chen, "Walking Control of a Biped Robot on Static and Rotating Platforms Based on Hybrid Reinforcement Learning," in IEEE Access, vol. 8, pp. 148411-148424, 2020, doi: 10.1109/ACCESS.2020.3015506.

[2.] U. I. Khan and Z. Chen, "Natural Oscillation Gait in Humanoid Biped Locomotion," in IEEE Transactions on Control Systems Technology, vol. 28, no. 6, pp. 2309-2321, Nov. 2020, doi: 10.1109/TCST.2019.2939955.

[3.] L. Li et al., "Accelerating Model-Free Reinforcement Learning With Imperfect Model Knowledge in Dynamic Spectrum Access," in IEEE Internet of Things Journal, vol. 7, no. 8, pp. 7517-7528, Aug. 2020, doi: 10.1109/JIOT.2020.2988268.

[4.] Zehfroosh and H. G. Tanner, "A New Sample-Efficient PAC Reinforcement Learning Algorithm," 2020 28th Mediterranean Conference on Control and Automation (MED), 2020, pp. 788-793, doi: 10.1109/MED48518.2020.9182985.

[5.] M. Liu, Y. Wan, F. L. Lewis and V. G. Lopez, "Adaptive Optimal Control for Stochastic Multiplayer Differential Games Using On-Policy and Off-Policy Reinforcement Learning," in IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 12, pp. 5522-5533, Dec. 2020, doi: 10.1109/TNNLS.2020.2969215.

[6.] Asadulaev, I. Kuznetsov, G. Stein, and A. Filchenkov, "Exploring and Exploiting Conditioning of Reinforcement Learning Agents," in IEEE Access, vol. 8, pp. 211951-211960, 2020, doi: 10.1109/ACCESS.2020.3037276.

[7.] C. Wong, S. -R. Xiao and H. Aoyama, "Natural Walking Trajectory Generator for Humanoid Robot Based on Three-Mass LIPFM," in IEEE Access, vol. 8, pp. 228151-228162, 2020, doi: 10.1109/ACCESS.2020.3046106.

[8.] Kuchibhotla, P. Harshitha and S. Goyal, "An N-step Look Ahead Algorithm Using Mixed (On and Off)

Policy Reinforcement Learning," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), 2020, pp. 677-681, doi: 10.1109/ICISS49785.2020.9315959.

[9.] P. An, M. Liu, Y. Wan, and F. L. Lewis, "Multi-player H∞ Differential Game using On-Policy and Off-Policy Reinforcement Learning," 2020 IEEE 16th International Conference on Control & Automation (ICCA), 2020, pp. 1137-1142, doi: 10.1109/ICCA51439.2020.9264554.

[10.] J. Kim, "Multi-Axis Force-Torque Sensors for Measuring Zero-Moment Point in Humanoid Robots: A Review," in IEEE Sensors Journal, vol. 20, no. 3, pp. 1126-1141, 1 Feb.1, 2020, doi: 10.1109/JSEN.2019.2947719.

[11.] A. Saputra, J. Botzheim, and N. Kubota, "Evolving a Sensory–Motor Interconnection Structure for Adaptive Biped Robot Locomotion," in IEEE Transactions on Cognitive and Developmental Systems, vol. 11, no. 2, pp. 244-256, June 2019, doi: 10.1109/TCDS.2018.2863032.

[12.] P. Reyes and M. -J. Escobar, "Neuroevolutive Algorithms for Learning Gaits in Legged Robots," in IEEE Access, vol. 7, pp. 142406-142420, 2019, doi: 10.1109/ACCESS.2019.2944545.

[13.] Y. -J. Chen, W. -C. Jiang, M. -Y. Ju and K. -S. Hwang, "Policy Sharing Using Aggregation Trees for ${Q}$ -Learning in a Continuous State and Action Spaces," in IEEE Transactions on Cognitive and Developmental Systems, vol. 12, no. 3, pp. 474-485, Sept. 2019, doi: 10.1109/TCDS.2019.2926477.

[14.] Fakoor, R., Chaudhari, P. & Smola, A.J.. (2019). P3O: Policy-on Policy-off Policy Optimization. “Proceedings of The 35th Uncertainty in Artificial Intelligence Conference”, in “Proceedings of Machine Learning Research” 115:1017-1027 Available from http://proceedings.mlr.press/v115/fakoor20a.html.

[15.] Sun, Chongduo & Shao, Lei & Chen, FangJian & Li, Mingxing & Zhang, Chunyu & Zhang, Qiuju. (2019). Modeling and Analysis of a Modular Multilegged Robot with Improved Fault Tolerance and Environmental Adaptability. Mathematical Problems in Engineering. 2019. 1-17. 10.1155/2019/8261617.

[16.] O. Koç, G. Maeda and J. Peters, "Optimizing the Execution of Dynamic Robot Movements With Learning Control," in IEEE Transactions on Robotics, vol. 35, no. 4, pp. 909-924, Aug. 2019, doi: 10.1109/TRO.2019.2906558.

[17.] T. Song, D. Li, Q. Jin, and K. Hirasawa, "Sparse Proximal Reinforcement Learning via Nested Optimization," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 50, no. 11, pp. 4020-4032, Nov. 2020, doi: 10.1109/TSMC.2018.2865505.

[18.] K. Lobos-Tsunekawa, F. Leiva and J. Ruiz-del-Solar, "Visual Navigation for Biped Humanoid Robots Using Deep Reinforcement Learning," in IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 3247-3254, Oct. 2018, doi: 10.1109/LRA.2018.2851148.

[19.] Fang, Ling & Gao, Feng. (2018). Type Design and Behavior Control for Six-Legged Robots. Chinese Journal of Mechanical Engineering. 31. 10.1186/s10033-018-0259-9.

[20.] Pong, Vitchyr & Gu, Shixiang & Dalal, Murtaza & Levine, Sergey. (2018). Temporal Difference Models: Model-Free Deep RL for Model-Based Control.

[21.] X. Jiang, J. Yang, X. Tan, and H. Xi, "Observation-Based Optimization for POMDPs With Continuous State, Observation, and Action Spaces," in IEEE Transactions on Automatic Control, vol. 64, no. 5, pp. 2045-2052, May 2019, doi: 10.1109/TAC.2018.2861910.

[22.] Gu, Shixiang & Lillicrap, Timothy & Ghahramani, Zoubin & Turner, Richard & Schölkopf, Bernhard & Levine, Sergey. (2017). Interpolated Policy Gradient: Merging On-Policy and Off-Policy Gradient Estimation for Deep Reinforcement Learning.

[23.] T. Mannucci, E. van Kampen, C. de Visser, and Q. Chu, "Safe Exploration Algorithms for Reinforcement Learning Controllers," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 4, pp. 1069-1081, April 2018, doi: 10.1109/TNNLS.2017.2654539.

[24.] Lin and E. Boldbaatar, "Fault Accommodation Control for a Biped Robot Using a Recurrent Wavelet Elman Neural Network," in IEEE Systems Journal, vol. 11, no. 4, pp. 2882-2893, Dec. 2017, doi: 10.1109/JSYST.2015.2409888.

[25.] O' Donoghue, Brendan & Munos, Remi & Kavukcuoglu, Koray & Mnih, Volodymyr. (2016). PGQ: Combining policy gradient and Q-learning.

[26.] X. Chen, Z. Yu, W. Zhang, Y. Zheng, Q. Huang, and A. Ming, "Bioinspired Control of Walking With Toe-Off, Heel-Strike, and Disturbance Rejection for a Biped Robot," in IEEE Transactions on Industrial Electronics, vol. 64, no. 10, pp. 7962-7971, Oct. 2017, doi: 10.1109/TIE.2017.2698361.

[27.] Schulman, John & Wolski, Filip & Dhariwal, Prafulla & Radford, Alec & Klimov, Oleg. (2017). Proximal Policy Optimization Algorithms.

[28.] Ge, Shuzhi & Li, Yanan & Wang, Chen. (2017). Impedance adaptation for optimal robot–environment interaction. International Journal of Control. 87. 10.1080/00207179.2013.827799.

[29.] L. Lin, K. -S. Hwang, W. -C. Jiang and Y. -J. Chen, "Gait Balance and Acceleration of a Biped Robot Based on Q-Learning," in IEEE Access, vol. 4, pp. 2439-2449, 2016, doi: 10.1109/ACCESS.2016.2570255.

[30.] T. L. Brown and J. P. Schmiedeler, "Reaction Wheel Actuation for Improving Planar Biped Walking Efficiency," in IEEE Transactions on Robotics, vol. 32, no. 5, pp. 1290-1297, Oct. 2016, doi: 10.1109/TRO.2016.2593484.

[31.] Z. Yu et al., "Gait Planning of Omnidirectional Walk on Inclined Ground for Biped Robots," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 46, no. 7, pp. 888-897, July 2016, doi: 10.1109/TSMC.2015.2487240.

[32.] Lillicrap, Timothy & Hunt, Jonathan & Pritzel, Alexander & Heess, Nicolas & Erez, Tom & Tassa,

Yuval & Silver, David & Wierstra, Daan. (2015). Continuous control with deep reinforcement learning. CoRR.

Appendix
Hyper Parameters
No. of episodes (during traing the model) 2000
No. of episodes (during traing the model) 200
N (number of mini batch size for on policy) 20
Maximum Buffer Memory (for off policy) 1e6
Learning rate of Actor (alpha) 0.001
Learning rate of critic (beta) 0.002
Gamma 0.99
Tau (to update target network in off policy) 0.005
Batch size 64
Off policy update per on ploicy 64