Biped Walking Robot Using Reinforcement Learning

Arjun C R
Department of Computer Science and Engineering, SRM Institute of Science and Technology,
Chennai,
India

Akila V
Department of Computer Science and Engineering, SRM Institute of Science and Technology,
Chennai,
India

***Abstract*—This paper presents some results of a hardware biped robot learning to walk in a constrained environment using Deep Q-Learning Neural Network (Reinforcement Learning). The idea is to implement a simple deep Q-Learning model on a hardware biped robot, with six degrees of freedom, to learn to walk. The input state of the model orientation of the robot at a given instant and the output is an action representing the direction of the rotation of the stepper motor corresponding to a leg joint determined by the action selection policy. The rewards are determined by trial and error method. The robot tries to learn to walk without any simulations or pre-trained models. The hardware robot is suspended, preventing it from falling during the process of learning and it is made to learn to walk on a freely movable platform. The deep Q-learning model runs on the local machine and the hardware robot communicates with the system via USB Serial port communication.**

Keywords—Reinforcement Learning, Deep Q-Learning, Biped Walking Robot, Two Legged Robot, Hardware Biped Robot

I. Introduction

Exhibiting human like actions and behavior is essential for robots that are intended to help, service and co-exist with humans. Amongst many complex control problems that are involved in achieving human like behavior in robots, biped walking or the action of two legged waking is one of the most challenging tasks. So far simulations and pre-trained models have been the standard approaches to implement learning of biped walking by hardware robots. The objective of this paper is to create and implement a system architecture which will enable the hardware biped robot to learn to walk using reinforcement learning without any pre-trained models or simulations. This proposed method overcomes the bias in learning trajectories and policies that may occur in a simulation environment. Also, it is not possible to recreate all the dynamics of the real world in simulation. Previous studies and approaches on Biped walking of robot using reinforcement learning have explored a variety of methods as in [1], [2], [3], [4]. [5] is a model based reinforcement learning approach that achieves the given tasks much faster without the knowledge of the environment. However, [6] requires lot of time to learn the controller and [7] needs a highly stable mechanical system. Some bipeds implemented tried different body with short torso, rounded foot and without ankle joints. These biped robots have only 4 degrees of freedom and controlling biped walking trajectories with the famous ZMP approach [8], [9] is impossible or very difficult to implement hence, an alternative controller design must be used. The body dynamics of human given in [10] is used as the nominal trajectory in Poincare-Map-Based approaches. Interestingly many alternatives have focused on matching the natural dynamics of the biped to desired walking cycle timing. In [3] they have used phase oscillators to estimate the appropriate walking cycle timing [11].

A chapter on [12] describes a biologically inspired two layered biped learning system comprising a bottom level CPG (central pattern generator) and a top-level RL (reinforcement learning) module. In order to cope with the change in the environment, in the above mentioned approach, they have used a mental simulation based reinforcement learning approach [5], [13], [14] in which the learning system interacts only with the above mentioned mentally simulated model. So, if the environment model is identified properly, policy parameters can be improved within the environment, without the need for physical biped robots. However, this model is not suitable for implementing physical biped robots.

Although many alternatives of biped locomotion has been studied for a long time, it is only in the past 20 years, that physical robots have started to perform bipedal walking, this can be seen as the result of development of super fast computers. There were static walking robots [15]. The major drawback of this was that the control architecture in these models had to ensure that the center of gravity(COG) projection on the ground was always inside the area of foot support. This approach was inefficient because only slow walking speeds was possible, that too only on flat surfaces. The alternative was robots with dynamic walking [16], In these robots the COG projection can be outside the area of support of foot, but the zero momentum point (ZMP), which is the point where the total angular momentum is zero, cannot be outside the area of support of the foot. With dynamic walking robots faster walking speeds was possible, running [17], stair climbing [18], [19], execution of successive flips [20], and even walking with no actuators [21]. Planar biped robot under

a pure sensor driven controller such [22] is an example of real time physical robots with real time interaction with the virtual learning agent.

II. Materials And Methods

The connections between each hardware component is established using standard 10 cm, Male - Male, Male - Female, Female - Male and Female - Female jumper cables and single lead wires. The connection between the microcontroller and the local machine is established using a USB AB cable. The circuit is connected using 48 mm x 35 mm x 10 mm mini breadboards. These set of 3 breadboards is enclosed by the external container. The actuators used are 28BYJ-48 stepper motors which takes 2048 steps for one complete revolution. It is an unipolar five wired stepper motor and it requires a motor driver, ULN2003A is used, for connecting it to the microcontroller. The biped robot uses ADXL345 accelerometer. It can take 13-bit high resolution measurements enabling identification of inclination changes less than one degree. The microcontroller use is the Arduino Mega 2560 based on the ATmega2560. It can connect to a computer with AB/USB cable. The power source used is a 7.5 Volt, 1100 mAh two cell rechargeable Lithium-ion battery. It is housed inside the external container along with the microcontroller and the stepper motor drivers. The body or the central frame of the biped robot is a 90 mm x 90 mm x 45 mm cuboid shaped 4 mm thick cardboard box. This box encloses the breadboard circuit of the accelerometer sensor along with indication LED lights. These LED lights indicates the state of the communication with the local machine. The body also houses all the wiring for the six stepper motors and accelerometer from the external container. The legs of the biped robot are a set of 3-D printed models. The 3-D models were first developed using two software Blender 2.81 and Autodesk Inventor Professional 2020 based on the dimensions of the stepper motor, body, and the ball bearings. Each leg is divided into three components; upper leg and lower leg depicted in Fig. 1, the foot shown in Fig. 2 and body-leg interface shown in Fig. 3.

A. Reinforcement Learning

Reinforcement learning differs from supervised learning in the way that it does not need labeled input/output pairs to be present. Amongst several algorithms for reinforcement learning [23], this approach uses Deep Q Learning. Typically, the environment is stated in the form of a Markov decision process or MDP, because many reinforcement learning algorithms for this context utilize dynamic programming techniques.

1) Markov Decision Process: A Markov Decision Process is a tuple (S, A, T, R) where;

- S is the set of different states, in this case state is represented by the 3 - Dimensional orientation values plus its sign-inverted values (x, y, z, -x, -y, -z) of the robot at a given instant t.
- A represents a set with different actions that can be taken at each time t. There are twelve actions that the robot can take, move each of the six stepper motor in clockwise or anti-clockwise direction
- T is called the transition rule:
  - $T : (a_t \in A, s_t \in S, s_{t+1} \in S) \rightarrow P(s_{t+1} | s_t, a_t)$ where $P(s_{t+1} | s_t, a_t)$ is the probability that the future state is st+1 given that the current state is st and the action played is at. The distribution of probability of the future states at time t + 1 is given by T given the current state and the action taken at time t. Hence, we can predict the future state s_{t+1} by a drawing a random value from the distribution $T : s_{t+1} \sim T(a_t, s_t, .)$
- R is the reward function:
  - Reward gained for choosing action a_t in the state s_t is given by r_t.

After defining the MDP, it is important to remind that it relies on the following assumption: the probability of the future state

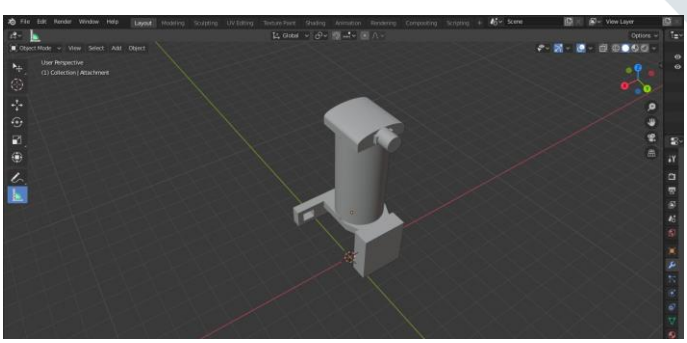

Fig. 1. Lower Leg

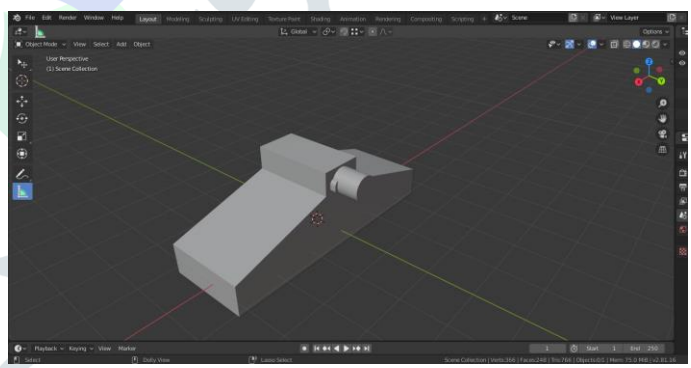

Fig. 2. Foot

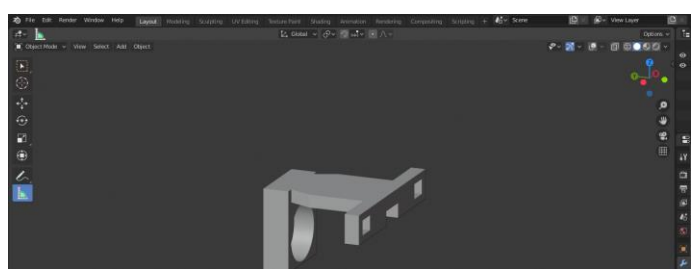

Fig. 3. Body - Leg Interface

s_{t+1} only depends on the current state s_t and action a_t , and doesn't depend on any of the previous states and actions. That is:

$P(s_{t+1} | s_0, a_0, s_1, a_1, ..., s_t , a_t) = P(s_{t+1} | s_t , a_t)$

Many complex MDP's are used in robot navigation such as [24].

2) Policy Function: The policy function π is exactly the function that, given a state s_t, returns the action at:

$\pi : s_t \in S \rightarrow a_t \in A$

Let's denote by Π the set of all the policy functions. Then the choice of the best actions to play becomes an optimization problem. Indeed, it comes down to finding the optimal policy π^* that maximizes the accumulated reward:

$\pi^* = \operatorname{argmax}_{\pi \in \Pi} \sum_{t \geq 0} R(\pi(s_t), s_t)$

3) Future Cummulative Reward: $R_t = R(a_t, s_t) + R(a_{t+1}, s_{t+1}) + ... + R(a_n, s_n) = r_t + r_{t+1} + ... + r_n$

However we can still improve the model. The elements r_t , r_{t+1}, ..., and r_n are values we are trying to estimate with the reward function R. At time t we are unsure of the reward in this future time, the more we look into the future the more uncertain it is. In other words, the larger is t' , the larger is the variance of the estimated reward r_{t+tj} . So, in order to fix that we have to discount each of the single rewards in the future, and the discount has to subsequently increase based on how further we are in to the future. So to do this we have to take the discounted sum of rewards:

$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} ... + \gamma^{n-t} r_n$

where $\gamma \in [0, 1]$. That way the higher is t', the smaller is $\gamma t'$, and therefore the more r_{t+tj} is discounted. γ is called the discount factor. The closer γ is to 0, the more the AI will try to optimize the current reward r_t . The closer γ is to 1, the more the AI will aim to optimize the future reward.

4) Q-Value: Each state and action pair (a,s) we have an associated numeric value Q(a,s):

$Q : (a_t \in A, s_t \in S) \rightarrow Q(a_t, s_t) \in \mathbb{R}$

We will say that Q(a, s) is "the Q-value of the action a played in the state s".

5) Temporal Difference: At the beginning t = 0, all the Q-values are initialized to 0. Now let's suppose we are at time t, in a certain state s_t . We play the action at and we get the reward r_t. Then we take a random draw from the $T(a_t, s_t, .)$ distribution, which leads us to the next state s_{t+1}:

$s_{t+1} \sim T(a_t, s_t, .)$

We can now introduce the temporal difference, which is at the heart of Q-Learning. The temporal difference at time t, denoted by $TD_t (a_t , s_t)$, is the difference between:

- $r_t + \gamma \max_a (Q(a, s_{t+1}))$, $\gamma \in [0, 1]$, that is the reward rt obtained by playing the action a_t in the state s_t , plus a percentage (which is our previous discount factor γ) of the Q-value of the best action played in the future state s_{t+1},
- and $Q(a_t , s_t)$, that is the Q-value of the action a_t taken in the state s_t , thus leading to

$TD_t (a_t, s_t) = r_t + \gamma max_a (Q(a, s_{t+1})) - Q(a_t, s_t)$

$TD_t (a_t , s_t)$ is like an intrinsic reward. The Q-values will be learned in such a way that:

- When $TD_t (a_t , s_t)$ is high, it receives "good surprise".
- When $TD_t (a_t , s_t)$ is low, it receives "frustration".

The temporal difference calculated is used in the last next step of the Q learning algorithm to reinforce (a,s) pair from time t-1 to t, with the help of the equation:

$Q_t (a_t , s_t) = Q_{t-1} (a_t , s_t) + \alpha TD_t (a_t , s_t)$

In this point of view, the Q-values measure the accumulation of the positive or negative temporal difference associated with the state action pair (a_t , s_t). In the surprise or positive case, reinforcement takes place, and in the frustration or negative case, weaking of the AI takes place. The objective is to learn Q-values that will fetch more positive values.

Based on this, the decision of action to be taken usually relies on the Q-value $Q(a_t, s_t)$. If the action a_t taken in the state s_t is has a large Q value $Q(a_t ,s_t)$ associated to it, the AI has higher chances of choosing a_t. And if the Q value associated is low the AI will have less inclination in considering the action a_t.

There are several ways of obtaining the best action to take. First, when being in a certain state s_t , we could simply take a with which we have the maximum of $Q(a, s_t)$:

$a = argmax_a (Q(a, s))$

But experience has shown that this is not the best option. A better solution is the softmax method. The softmax method consists of considering for each state s the following distribution:

$W_s : a \in A \rightarrow \frac{\exp(Q(s, a))^{\tau}}{\exp(Q(s, a^j))^{\tau}} \quad with \quad \tau \geq 0$

We get the action to take by picking a random value from that distibution:

$a \sim W_s (.)$

III. Proposed System

The idea is to let the robot learn to walk without any simulations or pre-trained models. This is ideal because it overcomes bias in learning trajectories and policies that may occur in a simulation environment. And also taking into consideration that all the dynamics of the real world cannot be recreated in a simulation, this approach seems to be a potential solution. However, letting the physical robot to learn without any heuristic data comes with a price. The problem is the time that the physical robot takes to fall, get up and learn to walk compared to the incredibly fast simulations that achieves the same in significantly less time. In order to compensate for this time factor, the proposed system involves a robot that learns to walk in a constrained environment which will speed up the learning process since, the process(actions) of falling and getting up is eliminated as described below, this makes the system comparatively faster than a free roaming robot learning to walk by falling and getting up.

The system consists of two parts, as depicted in Fig. 4,

one the neural network model that exists in the local machine and second the physical robot and its associated components. The neural network model uses a simple deep Q-learning algorithm with three hidden layers. The first hidden layer has twenty-four nodes, the second with eighteen nodes and the third with fourteen nodes. The input and output layer

had six and twelve nodes respectively. The neural network was fully connected with random initial weights and Adam optimizer [25]. The model parameters such as the learning rate, discounting factor(gamma) and temperature parameter are determined by experimentation. The modal uses Smooth L1-loss as the loss function. The temperature parameter is a random value obtained by trial and error, it determines how sure the RL agent is in taking the action. It is multiplied with the softmax value of the actions. The 3-Dimensional orientation of the robot at a particular instant is obtained from an accelerometer sensor and it along with its sign inverted values (positive to negative and vice versa) is used as the input state to the deep neural network. The accelerometer readings are of the form (x, y, z, -x, -y, -z) which represent the acceleration of the robot in the respective axis expressed in (m/s2) or in G-forces (g). Since the robot doesn't undergo very fast movements the value is usually in the range of -2 to 2 Table I . Using the orientation data the level of tilt of the robot is determined.

If the tilt exceeds a particular threshold value, which again is determined by experimentation, the reinforcement learning agent is given the maximum negative reward as going beyond the threshold tilt value corresponds to falling of the robot however, the robot doesn't fall as it is suspended and continues to walk. This is done overcome the significant delay of the robot actually falling down and getting back up. Maintaining the tilt within a certain range provides the agent with a small positive reward enabling to understand that it is supposed to maintain the balance. The output layer of the deep neural network consists of twelve neurons, corresponding to the twelve actions or Q-values, where each Q-value represents the rotation of the corresponding stepper motor by ten steps in that particular direction, six clockwise and six anti-clockwise respectively for each stepper motor. Each time the best action is determined by the action selection policy which takes the softmax of the twelve values which is then multiplied with the temprature parameter.

The hardware biped robot consists of two physical sections, one is the body of the biped walking robot with the two legs housing all the six stepper motors and the accelerometer sensor, second is the external container housing the Arduino microcontroller and the power source which is connected to local machine via USB Serial port communication at 250000 bit baud rate. The six joints of the legs I.e. hip, knee and ankle joints of both the leg are connected to the respective stepper motor on one side and a small 6 mm(inner diameter) ball bearing on the other side. The ball bearing is used to distribute the load of body on the leg thereby reducing the amount of strain on the stepper motor. The robot is suspended from the top support and stands on top of two discs. These discs are freely movable, hence when the robot tries to walk the discs acts as a moving platform. The suspension prevents the robot from falling and stepping out of the discs.

In order to prevent still state where the robot uses the suspension to stay still and not move there by ensuring continuous gain of positive reward, a small negative reward is given I.e. the robot is punished if it stays still instead of trying to walk.

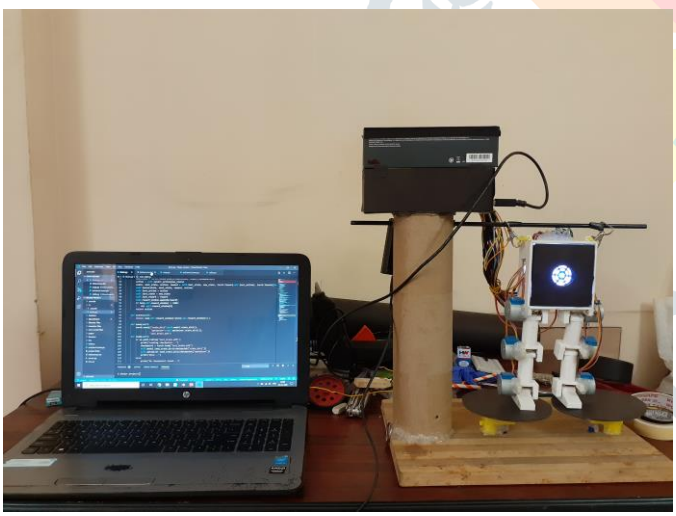

Fig. 4. Biped Robot with the walking environment

TABLE I
SAMPLE ACCELEROMETER READINGS

Xa	Ya	Za
1.05	0.04	0.11
1.02	-0.07	0.11
1.04	-0.02	0.19
0.98	0.01	0.07

IV. RESULT

A gradual increase in the rewards obtained by the RL agent can be seen after experimenting with different network parameters. Fig. 5 shows the reward window score curve after 3000 epochs starting with random initial weights and with temperature parameter(T) 700.

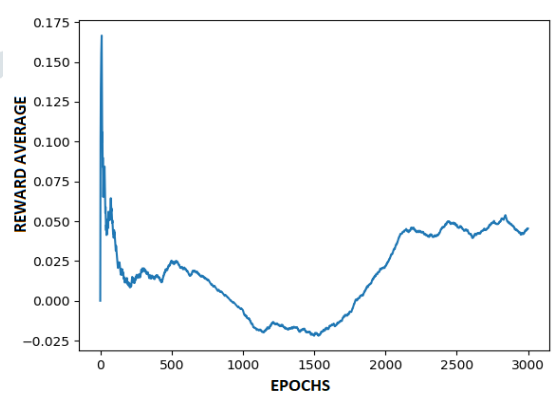

Fig. 5. 3000 epochs T = 700

The reward window score is value that can be used to visually represent whether the RL agent is learning the right set of actions, in this case learning to walk. It is the average of the last n rewards obtained by the RL agent, here the value of n used is 1000. This score is plotted against each epoch. The initial spike in all the curves are due to the stable standing state before the first action performed by the biped robot. In order to identify global maxima, several times the model was run with different initial weights, number of epochs and temperature parameters. Fig. 6 shows the reward curve with 4000 epochs and temperature parameter set to 7000.

As we can see from Fig. 7 and Fig. 8 setting the temperature parameter T to smaller value decreases the reward scores and also results in not so stable action selections, even though the number of epochs is higher (6000).

After exploring and tweaking other parameters such as the learning rate and discounting factor gamma, a model with a close to stable action selection and some gradual increase in the reward widow was obtained, as seen in Fig. 9, Fig. 10.

V. Future Work

The results from this approach shows that there is a potential of physical robots learning to walk without pre-trained models and simulations, however to be able to establish it concretely many factors have to be ensured and validated about the system. The motors and microcontroller used in this system are those that are available for many other use cases and not custom manufactured for this purpose. Also, there are several stages that lack precision and completeness in this implementation, when it comes to the hardware environment, as most of them were built with commercially available tools and materials. With respect to the reinforcement learning agent several different aspects are yet to be explored, such as different optimizers, network architecture, discounting factor, etc... Improving on these factors while also considering methods that can scale this approach well, allowing it to fit into larger and complex systems to achieve autonomous learning

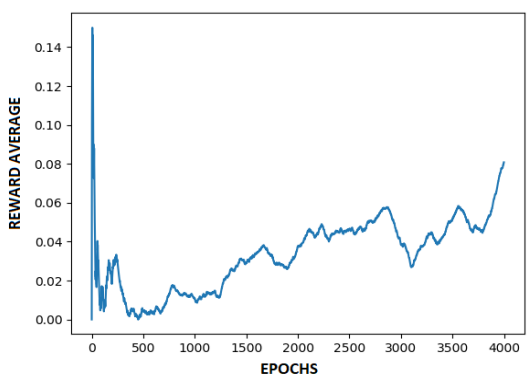

Fig. 6. 4000 epochs T = 7000

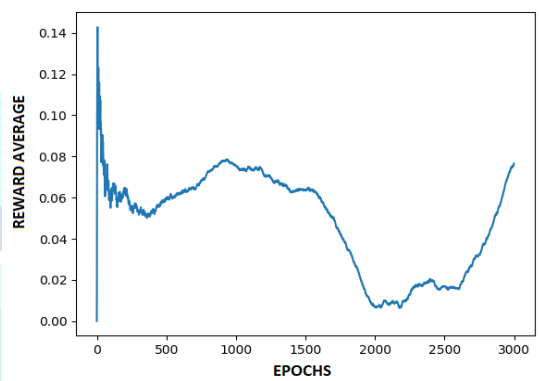

Fig. 8. 3000 epochs T = 100

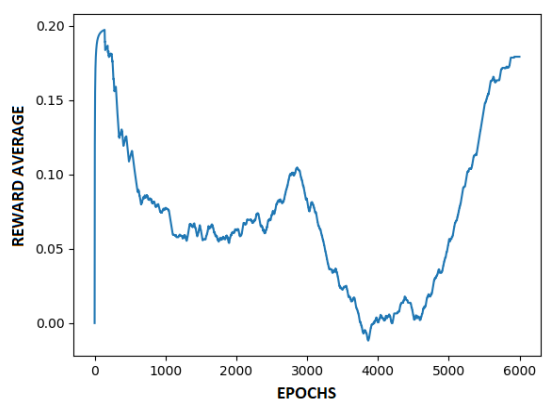

Fig. 7. 6000 epochs T = 100

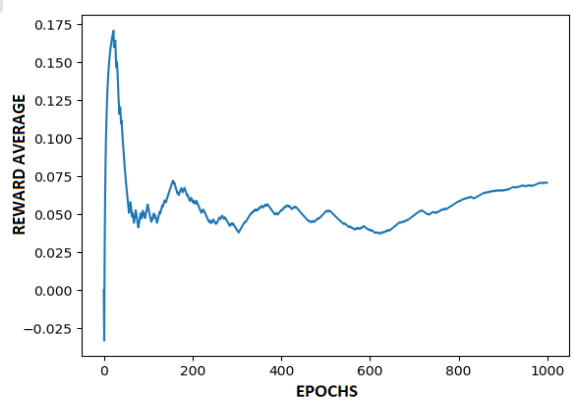

Fig. 9. 1000 epochs T = 15000

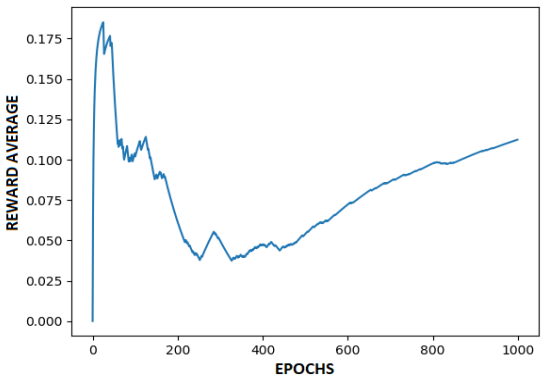

Fig. 10. 1000 epochs T = 30000

by robots without simulations will be ideal.

VI. Conclusion

The implementation of biped robot learning to walk with deep Q learning without pre-trained models and simulations was successful in the way that the reinforcement learning agent was able to obtain a gradual positive growth in the reward window score as mentioned in the results. The model was able to learn to maintain balance with the given 3 dimensional orientation input, also slowly increasing its stability with respect to its exploration of actions.

References

[1] C.-M. Chew and G. A. Pratt, "Dynamic bipedal walking assisted by learning," *Robotica*, vol. 20, no. 5, pp. 477–491, 2002. [Online]. Available: 10.1017/S0263574702004290

[2] J. Morimoto, J. Nakanishi, G. Endo, G. Cheng, C. G. Atkeson, and G. Zeglin, "Poincaré-Map-Based Reinforcement Learning For Biped Walking," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 2381–2386.

[3] J. Morimoto, G. Cheng, C. G. Atkeson, and G. Zeglin, "A simple reinforcement learning algorithm for biped walking," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 3, 2004.

[4] Y. Yuan, Z. Li, T. Zhao, and D. Gan, "DMP-Based Motion Generation for a Walking Exoskeleton Robot Using Reinforcement Learning," pp. 3830–3839, 2020. [Online]. Available: 10.1109/TIE.2019.2916396

[5] K. Doya, "Reinforcement Learning in Continuous Time and Space," *Neural Computation*, vol. 12, no. 1, pp. 219–245, 2000. [Online]. Available: 10.1162/089976600300015961;https://dx.doi.org/10.1162/089976600300015961

[6] H. Benbrahim and J. A. Franklin, "Biped dynamic walking using reinforcement learning," *Robotics and Autonomous Systems*, vol. 22, no. 3-4, pp. 283–302, 1997. [Online]. Available: 10.1016/s0921-8890(97)00043-2;https://dx.doi.org/10.1016/s0921-8890(97)00043-2

[7] R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3D biped," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004.

[8] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The development of Honda humanoid robot," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 2, 1998.

[9] J. . Yamaguchi, A. Takanishi, and I. Kato, "Development of a biped walking robot compensating for three-axis moment by trunk motion," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, vol. 1, 1993.

[10] T. Gélat and Y. Brenière, "Adaptation of the gait initiation process for stepping on to a new level using a single step," *Experimental Brain Research*, vol. 133, no. 4, pp. 538–546, 2000. [Online]. Available: 10.1007/s002210000452;https://dx.doi.org/10.1007/s002210000452

[11] K. Tsuchiya, S. Aoi, and K. Tsujita, "Locomotion control of a biped locomotion robot using nonlinear oscillators," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 2, 2003, pp. 1745–1750.

[12] J. Morimoto, "Humanoid Locomotion and the Brain," *Humanoid Robotics and Neuroscience: Science, Engineering and Society*, 2015.

[13] J. Morimoto and C. G. Atkeson, "Nonparametric representation of an approximated Poincaré map for learning biped locomotion," *Autonomous Robots*, vol. 27, no. 2, pp. 131–144, 2009. [Online]. Available: 10.1007/s10514-009-9133-z;https://dx.doi.org/10.1007/s10514-009-9133-z

[14] R. S. Sutton, "Planning by incremental dynamic programming," in *Proceedings of the Eighth International Conference on Machine Learning*. Morgan Kaufmann, 1991, pp. 353–357.

[15] I. Kato, S. Ohteru, H. Kobayashi, K. Shirai, and A. Uchiyama, "Information-Power Machine with Senses and Limbs," in *On Theory and Practice of Robots and Manipulators. International Centre for Mechanical Sciences (Courses and Lectures)*, vol. 201. Springer, 1974.

[16] A. Takanishi, G. Naito, M. Ishida, and I. Kato, "Realization of plane walking by the biped walking robot WL-10R, Robotic and Manipulator Systems," pp. 283–393, 1982.

[17] M. H. Raibert, "Hopping in legged systems — Modeling and simulation for the two-dimensional one-legged case," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-14, no. 3, pp. 451–463, 1984. [Online]. Available: 10.1109/tsmc.1984.6313238;https://dx.doi.org/10.1109/tsmc.1984.6313238

[18] A. Takanishi, H.-O. Lim, M. Tsuda, and I. Kato, "Realization of dynamic biped walking stabilized by trunk motion on a sagittally uneven surface," in *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, vol. 1, 1990, pp. 323–330.

[19] Y. Kurcmatsu, O. Katayama, M. Iwata, and S. Kitamura, "Autonomous trajectory generation of a biped locomotive robot," in *IEEE International Joint Conference on Neural Networks*, vol. 3, 1991, pp. 1983–1988.

[20] J. & Hodgins and M. Raibert, "Biped Gymnastics," 1990.

[21] T. McGeer, "Passive Dynamic Walking," *The International Journal of Robotics Research*, vol. 9, no. 2, pp. 62–82, 1990. [Online]. Available: 10.1177/027836499000900206;https://dx.doi.org/10.1177/027836499000900206

[22] T. Geng, B. Porr, and F. Wörgötter, "Fast Biped Walking with a Sensor-driven Neuronal Controller and Real-time Online Learning," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 243–259, 2006. [Online]. Available: 10.1177/0278364906063822

[23] C. Szepesvári, "Algorithms for Reinforcement Learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 4, no. 1, pp. 1–103, 2010. [Online]. Available: 10.2200/S00268ED1V01Y201005AIM009

[24] G. Theocharous, K. Rohanimanesh, and S. Maharlevan, "Learning hierarchical observable Markov decision process models for robot navigation," in *IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 1, 2001, pp. 511–516.

[25] Z. Zhang, "Improved Adam Optimizer for Deep Neural Networks," *IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pp. 1–2, 2018.