A Trajectory Control for Bipedal Walking Robot Using Stochastic-Based Continuous Deep Reinforcement Learning

Surriani, Atikah
Department of Electrical and Information Engineering, Faculty of Engineering, Gadjah Mada University

Wahyunggoro, Oyas
Department of Electrical and Information Engineering, Faculty of Engineering, Gadjah Mada University

Adha Imam Cahyadi
Department of Electrical and Information Engineering, Faculty of Engineering, Gadjah Mada University

https://doi.org/10.5109/7151701

出版情報：Evergreen. 10 (3), pp.1538-1548, 2023-09. 九州大学グリーンテクノロジー研究教育センター
バージョン：

A Trajectory Control for Bipedal Walking Robot Using Stochastic-Based Continuous Deep Reinforcement Learning

Atikah Surriani[1,*], Oyas Wahyunggoro[1,] Adha Imam Cahyadi[1]

[1]Department of Electrical and Information Engineering, Faculty of Engineering, Gadjah Mada University, Yogyakarta Indonesia

*Author to whom correspondence should be addressed:
E-mail: atikah.surriani.sie13@ugm.ac.id

(Received January 26, 2023; Revised August 2, 2023; accepted September 17, 2023).

Abstract: The bipedal walking robot is an advanced anthropomorphic robot that can mimic the human ability to walk. Controlling the bipedal walking robot is difficult due to its nonlinearity and complexity. To solve this problem, recent studies have applied various machine learning algorithms based on reinforcement learning approaches, however most of them rely on deterministic-policy-based strategy. This research proposes Soft Actor Critic (SAC), which has stochastic policy strategy for controlling the bipedal walking robot. The option thought deterministic and stochastic policy affects the exploration of DRL algorithm. The SAC is a Deep Reinforcement Learning (DRL) based algorithm whose improvement obtained through the augmented entropy-based expected return allows the SAC algorithm to learn faster, gain exploration ability, and still ensure convergence. The SAC algorithm's performance is validated with a bipedal robot to walk towards the straight-line trajectory. The number of the reward and the cumulative reward during the training is used as the algorithm's performance evaluation. The SAC algorithm controls the bipedal walking robot well with a total reward of 384,752.8.

Keywords: deep reinforcement learning, bipedal walking robot, soft actor-critic (SAC), deep deterministic policy gradient (DDPG), twin delayed DDPG (TD3), bipedal walking robot, trajectory control

1. Introduction

Robotics and control systems have become the most attractive research area for the researchers for decades[1,2,3,4),5,6)]. One of the challenging problems is controlling bipedal walking robots. The bipedal walking robot is an advanced anthropomorphic robot that can mimic the human ability to walk[7)]. It is designed and modeled in human like and walking by the knowledge of the human gait. Controlling the bipedal walking robot is difficult due to its nonlinearity and complexity[2,9)]. It refers to the intricate and dynamic nature of their movements, such as multi-body systems consisting of multiple interconnected links and joints and underactuated, meaning they have fewer actuators (motors) than degrees of freedom, which makes controlling them challenging. It makes the bipedal walking robot difficult to be modelled except using an advanced mathematical theorem[3,10)]. There are many control problems to be solved for the bipedal walking robot, the most challenging part is trajectory controlling. Stability control for the bipedal walking robot to take steps to walk following the trajectory is critically important. It occurs to avoid the bipedal robot falling when walking towards the path.

Recent trend for robotics and control system research is machine learning-based algorithm[11,12)], particularly Reinforcement Learning (RL). Reinforcement Learning (RL) is a promising method to increase the performance of robots in many fields[13)] especially for bipedal walking robots. Reinforcement learning is an appropriate algorithm to control bipedal walking robots to imitate human gait, because it is a type of machine learning algorithm wherein the is a machine learning method that can often be applied to perceive a robot's position and lead it to the desired position[14)]. This method enrolls the robot to learn as a human being through trial and error.[15)]. Moreover, the trial-error process is granted by a reward as the return that indicates the strategy of policy value of the process's performance. These advantages make many researchers use DRL to control the bipedal walking robot.

Previous research used many methods to control the bipedal walking robots, A. Mayub and Fahmizal applied fuzzy logic to control the bipedal walking robot[8)]. It presented the Center of Point (CoP) and the Zero Moment Point (ZMP) model to control the bipedal walking robot by using Force Sensitive Resistor (FSR) on the foot, but a fuzzy logic system involves a considerable degree of subjectivity. It leads to misinterpretation, especially for complex high dimensional data like bipedal walking robots. T.P.A. Wiggers proposed the Hybrid Zero Dynamics (HZD) to control the bipedal walking robot[16). The HZD implementation can be complex and challenging. HZD involves the design and control of hybrid systems, which combine continuous dynamics (e.g., walking) with discrete events (e.g., foot impacts). Managing the interactions between these continuous and discrete elements can increase the complexity of the control system. J.Y. Kim, I.W. Park, J.H. Oh controlled the bipedal humanoid robot using a combination of several controllers, such as the predicted motion control, the walking pattern control, and the real-time balance control[17). These controllers were applied to make the bipedal humanoid robot walk on an uneven and inclined floor. Using a combination of several controllers (predicted motion control, walking pattern control, and real-time balance control), offers several advantages. However, it also comes with certain disadvantages that should be considered, such as the design of tuning efforts and complexity. Kumar, et al[10) and D. Rastogi[20) applied Deep Deterministic Policy Gradient (DDPG) to control the bipedal walking robot. DDPG can perform better in high dimensional environment and can work very well in continuous action and state, yet it suffers with the hyperparameter and the overestimation bias[11,22). P.B. Khoi et al presented Twin Delayed DDPG (TD3)[1) to control the bipedal walking robot. TD3 can handle the overestimation bias on DDPG, but TD3 is still using the deterministic policy. T. Tiong and I. Saad proposed Twin Average Delayed DDPG (TAD3)[21) which had the double-actors and the double-critics. TAD3 was claimed to have a better performance than DDPG and TD3, but still suffered with the hyperparameter.

Deep reinforcement learning methods is used to apply the deterministic policy and stochastic policy[13,24). Many studies operate robots using deterministic policy, under full observable condition. However, the deterministic policy has only limited action choice and it mostly need the noise injection into action to gain the exploration ability[14,25). In fact, many real robotic and control problems must encounter randomness and explore better to obtain the best action. In this case, it might be chosen to learn using stochastic behaviors that have more exploration ability[24). This research performs Soft Actor Critic (SAC) that is one of continuous DRL algorithms, which uses the stochastic policy in the learning process. This stochastic policy can improve the adaptation ability of the bipedal walking robot to head on many situations[12,23). The stochastic policy has also a powerful performance during the exploration process[26). This research selects the off-policy DRL approaches such as DDPG, TD3 and SAC. The selection of off-policy DRL over on-policy methods offers several advantages that contribute to more efficient and stable learning. The utilization of a separate behavior policy, experience replay mechanisms, and improved data efficiency are pivotal in enabling the agent to explore and exploit the environment effectively. Off-policy algorithms like DDPG, TD3 and SAC have shown remarkable success in addressing challenging tasks with high-dimensional action spaces, making them valuable tools for researchers and practitioners seeking to develop robust and adaptive agents.

The purpose of this research is to control the bipedal humanoid robot to walk as a human gait in the straight-line trajectory by using SAC Algorithm. The implementation of the SAC algorithm will be evaluated with another DRL algorithm whose deterministic policy based method. This research concerns this issue because comparing deterministic and stochastic policy Deep Reinforcement Learning (DRL) methods is essential because each approach offers distinct advantages and trade-offs, and the choice of policy representation can significantly impact the performance and behavior of the learning agent. Understanding the differences between deterministic and stochastic policies allows researchers and practitioners to make informed decisions based on the specific requirements and characteristics of the problem at hand.

Based on the introduction, the contributions of this work are explained as follows: 1) This research applies SAC deep reinforcement learning for the bipedal walking robot. 2) This research applies SAC which uses the stochastic policy and the entropy for training strategy, despite DDPG and TD3 that used the deterministic policy. This research presents the comparison of these continuous deep reinforcement learning approaches to find the best algorithm for the bipedal walking robot. The Evaluation of the results is based on the reward and the cumulative reward that is part of the reinforcement learning goal[18,29). The study is started by Section II that explains the RL, SAC. In Section III the Methodology explains about the bipedal walking robot parameter, the configuration, the interface to the algorithm and the controller architecture that can be implemented for the bipedal walking robot. Section IV provides the numbers of the simulation results which show the behaviors and the performance comparison of continuous DRL such as SAC, DDPG and TD3 for the bipedal walking robot. Section V presents the conclusions of this research that is the comparison result of the continuous DRL approaches.

2. Reinforcement Learning

Reinforcement Learning is formulated using Markov Decision Process (MDP)[8,30). MDP consists of a few tuples such as (A, S, p_s, r). The action, A, and the state, S, are the continuous domain, the transition probability state, p_s: $S \times S \times A \rightarrow [0, \infty)$ that represents the probability density of the next state of the environment, $\boldsymbol{s}_{t+1} \in S$ given the present state $\boldsymbol{s}_t \in S$ and action $\boldsymbol{a}_t \in A$. The environment gives reward to the agent on each transition as $r : S \times A \rightarrow [r_{min}, r_{max}]$. The action for this research conducts two policy terms, such as deterministic policy $\mu(\boldsymbol{s}_t)$ and stochastic policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$. The term $\rho_\pi(\boldsymbol{s}_t)$ is denoted as the state and $\rho_\pi(\boldsymbol{s}_t, \boldsymbol{a}_t)$ as the state-action marginals of the trajectory distribution of the policy.

Fig. 1 shows the diagram of reinforcement learning.

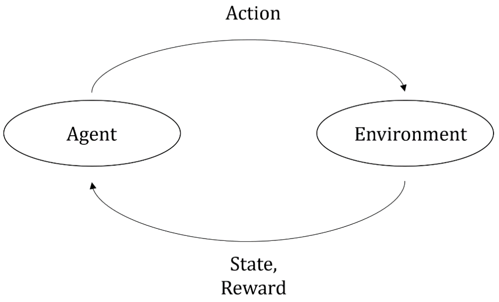

Fig. 1: Diagram of Reinforcement Learning.

Based on Fig. 1, the diagram of reinforcement learning shows the agent gives the action to the environment. After receiving the action from the agent, the environment gives the observation and the reward to the agent[30]. Basic RL has goal to maximize the expected reward, R, that is defined as follows,

$$R = \sum_t E_{(s_t,a_t)\sim\rho_\pi}[r(\boldsymbol{s}_t, \boldsymbol{a}_t)]. \quad (1)$$

2.1 Deep Deterministic Policy Gradient (DDPG)

Deep deterministic policy gradient (DDPG) is the extended algorithm of Deterministic Policy Gradient (DPG)[23]. Moreover, the DDPG algorithm also adopted the replay buffer, R_b, concept from Deep Q-Network (DQN)[31]. This feature tends to conquer the experience of managing unnecessary computatios. Reply buffer works with minibatch take the essential old and new experiences. This concept significantly increases stability. Reply buffer is used with minibatch for efficiency of sampling learning[32]. DDPG used the actor and critic networks. The actor network consists of the actor $\mu(\boldsymbol{s}_t)$ with parameter θ^μ ; target network $\mu'(\boldsymbol{s}_t|\theta^{\mu'})$with $\theta^{\mu'}$. Where, $\theta^{\mu'} \leftarrow \theta^\mu$ it is updated by the sample policy gradient,

$$\nabla_{\theta^\mu} J \approx \frac{1}{M}\sum_i \nabla_a Q(\boldsymbol{s}_t, \boldsymbol{a}_t|\theta^Q)\big|_{s=s_i.a=\mu(s_i)} \nabla_{\theta^\mu}\mu(\boldsymbol{s}_t|\theta^\mu)\big|_{s_{i.}} \quad . (2)$$

DDPG uses Ornstein-Uhlenbeck noise (OU Noise), N in the action, thus it can be defined as $\boldsymbol{a}_t = \mu(\boldsymbol{s}_t) + N$[33]. The OU Noise, N, is associated with exploration noise that gains exploration performance of the deterministic policy. The critic parameter consists of $Q(\boldsymbol{s}_t, \boldsymbol{a}_t)$ with parameter θ^Q, target network $Q'(\boldsymbol{s}_t, \boldsymbol{a}_t|\theta^{Q'})$ with $\theta^{Q'}$, where $\theta^{Q'} \leftarrow \theta^Q$. This critic network updates,

$$y_i = r_i + \gamma Q'(\boldsymbol{s}_{i+1,}\mu'(\boldsymbol{s}_{i+1}|\theta^{\mu'})|\theta^{Q'}). \quad (3)$$

DDPG applied a soft updating rule in network target, which means that only the essential one can be used in the learning process for the main weight.

2.2 Twin Delayed Deep Deterministic Policy Gradient (TD3)

Fujimoto et al improved the DDPG by using two Q-value functions mention as a clipped variant of double Q-learning[34]. It is designed to solve the overestimation bias in the actor-critic network. The actor is adopted from DDPG, which applied the exploration noise. The critic network consists two Q-values that define as $Q_i(\boldsymbol{s}_t, \boldsymbol{a}_t)$ with parameter $\theta^Q{}_i$, target network $Q'_i(\boldsymbol{s}_t, \boldsymbol{a}_t|\theta^{Q'})$ with $\theta^{Q'}{}_i$, *where* $\theta^{Q'} \leftarrow \theta^Q{}_i$, where $i = 1,2$. Applying these Q-values, Fujimoto et al defined the soft target network as clipped double Q-learning as follows,

$$y_1 = r + \gamma \min_i Q'_i(\boldsymbol{s}_{t+1}, \mu(\boldsymbol{s}_{t+1})). \quad (4)$$

The improvement is also conducted in the policy by delaying policy updates. The policy is updated at a lower frequency than the value estimation to minimize the error and divergent behavior. TD3 used the soft target smoothing that makes the strategy to exploit the action with high Q-value.

2.3 Soft Actor-Critic (SAC)

The soft actor-critic (SAC) algorithm is the deep reinforcement learning that was developed by T. Haarnoja et al[25] in 2018. The application of the algorithm for this research is based on[35]. First main difference of SAC compared to other algorithm is augmenting to maximize the entropy[36]. Conventional RL has the goal to maximize the expected reward as denoted at (1). The SAC algorithm defines a stochastic policy term by appending the conventional RL objective function (1) with the expected entropy-based for the policy over $\rho_\pi(\boldsymbol{s}_t)$,

$$J(\pi) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim\rho_\pi}[r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha H(\pi(.|\boldsymbol{s}_t))]. \quad (5)$$

The entropy objective, H, is defined based on Ziebart[37]. The temperature parameter, α, is the parameter that chooses the importance of relativity between the entropy and reward. This augmentation based on entropy energy makes the policy have stochastic behaviour. SAC algorithm applied stochastic policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$, while DDPG and TD3 applied deterministic policy $\mu(\boldsymbol{s}_t)$. Thus, the action is defined as, $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$. The policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ with parameter θ_π, computes using the soft policy iteration that can guarantee to converge at optimal policy during training. The actor network consists of the policy that is associated to a Gaussian distribution $\aleph = (\delta_t, \sigma_t)$ with mean, δ_t, and covariance, σ_t, given by neural networks[35]. Thus, the action can be written as follows, $\boldsymbol{a}_t = f(\aleph, \boldsymbol{s}_t)$ where $\aleph = (\delta_t, \sigma_t)$ that is the input noise vector from the environment that follows the Gaussian[26]. Based on the modified reward function (5), Haarnoja et al the derived the soft value function and the soft Q-value function as follows,

$$V^{\pi}(\boldsymbol{s}_t) = E_{\boldsymbol{a}_t \sim \pi}[Q(\boldsymbol{s}_t, \boldsymbol{a}_t) - \log \pi(\boldsymbol{a}_t|\boldsymbol{s}_t), \tag{6}$$

and

$$Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) = r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \gamma E_{\boldsymbol{s}_{t+1} \sim p}[V^{\pi}(\boldsymbol{s}_{t+1})]. \tag{7}$$

The soft value function or the soft state value and the soft Q-value or soft state-action value indicate the reward in the future[26]. The reward also consist of the expected sum of the reward and the entropy[38]. Soft Q-value, $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ with target parameter $\theta_{Q^{\pi}}$, is updated and applied to train the policy. The soft state value $V^{\pi}(\boldsymbol{s}_t)$ with target parameter $\theta_{V^{\pi}}$ is updated and required to estimate the soft Q-value. The SAC algorithm also worked with the double Q-value function which can stabilize the learning process and performance[14,36].

3. Methodology

The research is performed by designing the environment of bipedal walking robots and the agent of deep reinforcement learning. The interface between the environment and the agent is shown in Fig. 2. Based on Fig 2, the research method or the agent of the system is designed based on the reinforcement learning formula. This research applied continuous Deep Reinforcement Learning algorithms such as soft actor-critic (SAC) based on the research of T. Haarnoja et al[25].

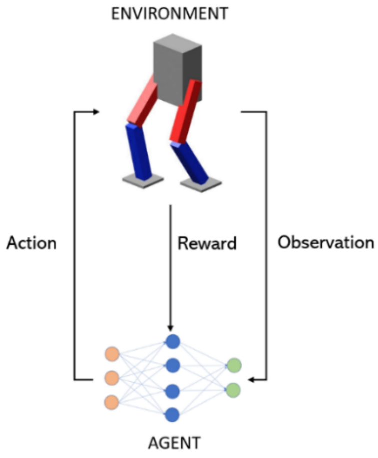

Fig. 2: The Interface of Bipedal Walking Robot with DRL.

The flowchart of the system can be viewed in Fig 3. Fig 3 views the flowchart of the system that illustrates the system's mechanisms especially the training process using SAC algorithm.

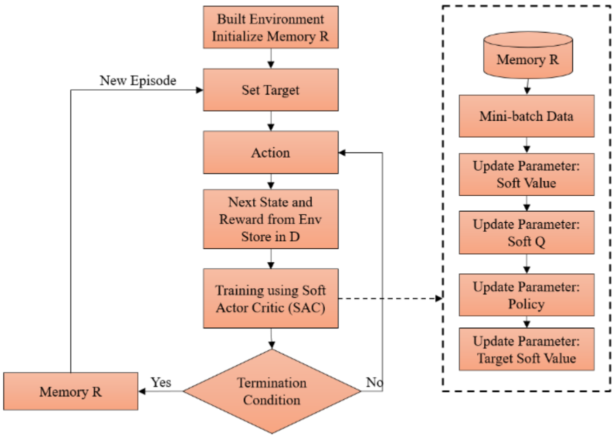

Fig. 3: The System Flowchart.

The SAC method is applied in order to compare with deep deterministic policy gradient (DDPG) from research of Lillicrap et al[39] and twin delayed deep deterministic policy gradient (TD3) on the research of Fujimoto et al[34] which employed deterministic-based policy. Castro has been applied DDPG and TD3 for controlling bipedal walking robot[29]. The comparison diagram of each algorithm can be viewed at Fig 4.

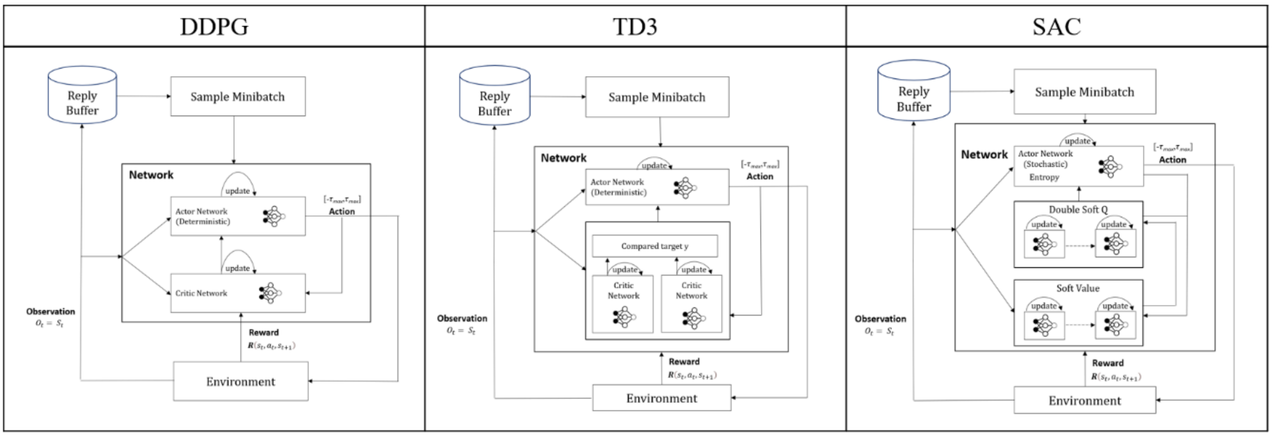

Fig 4: The comparison Diagram of DDPG, TD3, and SAC.

3.1 Bipedal Walking Robot Environment

Regarding the Castro's model[15,26]. The bipedal walking robot is standing up right, in the neutral position at 0 rad. The goal of the bipedal walking robot is walking like a human gait at the straight-line trajectory. The trajectory is shown in Fig 5.

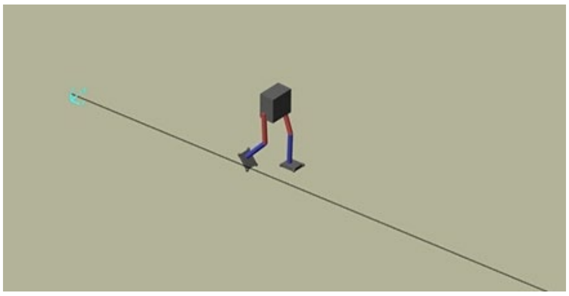

Fig. 5: Trajectory of Bipedal walking Robot.

This trajectory represents a spline curve in 2-D or 3-D, passing through specified interpolation points. The curve is set open continuously based on cubic interpolation between adjacent points. The configuration of the bipedal walking robot based on J. Englsberger et al[40]. has six joints, three joints are in the right foot and three joints are in the left. The joint is in each hip, knee and ankle. It is built in 3D with the torso's height is 0.35, the torso's length is 0.2 and the torso's width is 0.24. The frontal joint of the bipedal walking robot has as $\theta_{L_1}\,\theta_{L_2}\,\theta_{L_3}$ as the left roll joint for the hip, the knee and the ankle and $\theta_{R_1}, \theta_{R_2}, \theta_{R_3}$ as the right roll joint for the hip, the knee and the ankle. The bipedal walking robot is standing up right, in the neutral position at 0 rad. The action is defined as $\boldsymbol{a}_t = [\theta_{F_1},\ \theta_{F_2},\ \theta_{F_3}\ ]$, where F is denoted the right leg or the left leg force. The range of the torque (τ) input is -3N.m to 3N.m, but the actual inputs of each joint are normalized at range -1 and 1.

The bipedal walking robot receives the action of the torque from the agent and gives several data of the observation $\boldsymbol{o}_t{}^T$ and the reward $\boldsymbol{r}_t$ to the agent. The observation has 29 data as follows,

$$\boldsymbol{o}_t = [\boldsymbol{p}\ \ \dot{\boldsymbol{p}}\ \ \boldsymbol{\varphi}\ \ \boldsymbol{\omega}\ \ \boldsymbol{\theta}_L\ \dot{\boldsymbol{\theta}}_L\ \boldsymbol{\theta}_R\ \dot{\boldsymbol{\theta}}_R\ \ \boldsymbol{a}_{t-1}\ \boldsymbol{F}]. \quad (8)$$

Where $\boldsymbol{p} = [p_y\ p_z]$ consists of positions in y and z axes, $\dot{\boldsymbol{p}} = [\dot{p}_x\ \dot{p}_y\ \dot{p}_z]$ denoted as linear velocities at x-y-z axes, $\boldsymbol{\varphi} = [\varphi_r\ \varphi_p\ \varphi_y]$ denoted as roll, pitch and yaw, $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$ conducts of angular velocities at x-y-z axes. It also observes the angle and speed of both leg, thus it has $\boldsymbol{\theta}_L = [\theta_{L_1}\ \theta_{L_2}\ \theta_{L_3}]$ consists of the left foot angles of hip, knee and ankle, $\boldsymbol{\theta}_R = [\theta_{R_1}\ \theta_{R_2}\ \theta_{R_3}]$ consists of the right foot angles of hip, knee and ankle, $\dot{\boldsymbol{\theta}}_L = [\dot{\theta}_{L_1}\ \dot{\theta}_{L_2}\ \dot{\theta}_{L_3}]$ consists of the left foot speed of hip, knee and ankle, $\dot{\boldsymbol{\theta}}_R = [\dot{\theta}_{R_1}\ \dot{\theta}_{R_2}\ \dot{\theta}_{R_3}]$ consists of the right foot speed of hip, knee and ankle, and $\boldsymbol{F} = [F_L\ F_R]$ is denoted as forces of both of right and left legs. It also includes $\boldsymbol{a}_{t-1}$ the action from the previous experience of the leg torques. This environment is Fully observable MDP, thus $\boldsymbol{o}_t = \boldsymbol{s}_t$.

The reward $r_t$is defined in accordance of the study of N. Heess, et al[13], as follows,

$$\mathrm{r_t} = \dot{p}_x - 3{p_y}^2 - 50{p_z}^2 + 25\,\frac{\mathbf{Ts}}{\mathbf{Tf}} - 0.02\,\sum_{\mathbf{i}} {\boldsymbol{\tau}^{\mathbf{i}}_{\mathbf{t-1}}}^2. \quad (9)$$

Where, $\mathrm{r_t}$ is reward, $\dot{p}_x$ is the velocity of the bipedal robot or the movement through the X-axis translation. p_y a is the displacement of the bipedal walking robot towards the line trajectory. p_z is the vertical translation displacement which is normalized with respect to the robot center of mass. $\boldsymbol{\tau}^{\mathbf{i}}_{\mathbf{t-1}}$ is the torque of joint i from previous time. **Ts** is the environment of time during training. **Tf** is the final time of the training. The training can be terminated at these following conditions, first when $p_z \leq 0{,}1$ or the robot is falling down which is the COM of the torso value is less than 0.1m at Z-axis, second, $p_y > 1$, or the bipedal walking robot move away at Y-axis more than 1 m, third, when the value for the roll, $\varphi_r > 0.7854$ rad, the pitch, $\varphi_p > 0.7854$ rad, or the yaw, $\varphi_y > 0.7854$ rad.

3.2 Network Architecture

The network architecture is designed based on S. Castro's model[18,29], that is referred to the research of

Lillicrap et al[30] for DDPG network and Fujimoto et al[34] for TD3. This research designs the agent's architecture based on Haarnoja et al[35]. The SAC hyperparameter is shown in Table 1.

Table 1. The SAC Hyperparameter.

Parameter	**Value**
Optimizer	Adam [42]
Mini-batch size	256.00
Discount factor (γ)	0.99
Experience Buffer	1.10^6
Target Smoothing	5.10^{-3}

Some parameters are set arbitrarily to fit the network architecture. The network consists of the actor network and critic network which is used deep neural network. The actor network of SAC algorithm applies the Gaussian distribution which need the input of observation or state, gives as the mean value and a standard deviation value for each action, $\aleph = (\delta_t, \sigma_t)$. The standard deviations are set as nonnegative and mean values are set within the range of the action, $\delta_t = [\boldsymbol{a}_{t-min}, \boldsymbol{a}_{t-max}]$. Hence, the output layer that turn over the standard deviations is required the ReLU layer, to enforce nonnegativity, while the output layer that turn over the mean values is required a scaling layer, to keep the mean values inside the scale of the output range. The details architecture SAC algorithm through the trajectory control of Bipedal Walking robot is shown at Fig 6.

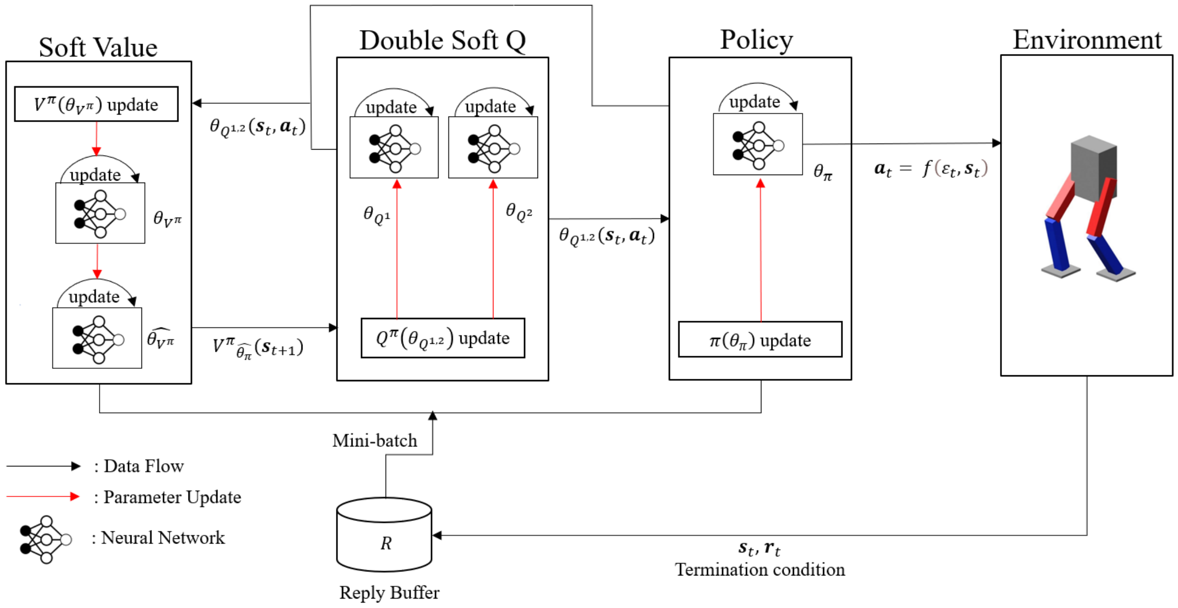
Soft Value
Double Soft Q
Policy
Environment
$V^{\pi}(\theta_{V^{\pi}})$ update
update
$\theta_{V^{\pi}}$
update
$\widehat{\theta_{V^{\pi}}}$
$\theta_{Q^{1,2}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$
$V^{\pi}_{\widehat{\theta_{\pi}}}(\boldsymbol{s}_{t+1})$
update
update
θ_{Q^1}
θ_{Q^2}
$Q^{\pi}(\theta_{Q^{1,2}})$ update
$\theta_{Q^{1,2}}(\boldsymbol{s}_t, \boldsymbol{a}_t)$
update
θ_{π}
$\pi(\theta_{\pi})$ update
$\boldsymbol{a}_t = f(\varepsilon_t, \boldsymbol{s}_t)$
Mini-batch
: Data Flow
: Parameter Update
: Neural Network
R
Reply Buffer
$\boldsymbol{s}_t, \boldsymbol{r}_t$
Termination condition

Fig. 6: SAC Architecture of Trajectory Based of Bipedal Walking Robot.

Based on Fig 8, three separate networks are used in the SAC algorithm to optimize learning process, these are soft state value ($V^{\pi}(\boldsymbol{s}_t)$) parameterized by $\theta_{V^{\pi}}$, double Soft Q-value, $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$ parameterized $\theta_{Q^{\pi}}$, and Policy Function (π) parameterized by θ_{π}. The soft state value ($V^{\pi}(\boldsymbol{s}_t)$) is responsible for estimating the expected cumulative reward starting from a given state under the current policy π. The double soft Q-value, $Q^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, estimates the expected cumulative reward when taking a particular action in a given state under the current policy π. It predicts the expected Q-value, which indicates the expected cumulative reward from the current state and action onwards. The parameter θ represents the learnable weights of the soft Q-function. Policy (π) is responsible for selecting actions given a state. It decides which actions to take based on the state and the current policy. The soft state value network is trained to minimize the mean-squared Bellman error. This error measures the difference between the predicted state value and the target value, which is the sum of the immediate reward and the estimated value of the next state. The double soft Q-value is trained using the soft Bellman backup that be defined by haarnoja[33]. The policy network is trained to maximize the expected reward, which includes the Q-values and an entropy term.

3.3 Evaluation

These DRL algorithms in this study will be evaluated based on the reward generated from each algorithm's training. This study also calculates the success rate, training success of each algorithm and the success rate of the bipedal robot in completing the trajectory in 2000 episodes. It is defined as follows:

$$SR = (SE/TE)\,100. \qquad (10)$$

Where,
SR = Success Rate (%)
SE = Number of Successful Episode
TE = Total Number of Episode.

4. Result and Discussion

4.1 Discussion

The first reinforcement learning algorithm is based on the MDP formulation introduced by Bellman[40]. This evaluation is to understand how the DRL algorithm works with prior off-policy such as DDPG, TD3 and SAC to control bipedal walking robot. Silver et al[21] improved the exploration ability for the deterministic policy by designing an off-policy learning technique using a differentiable function approximator and adding a small amount of random noise to the action. These techniques are implemented inside the action of DDPG and TD3 algorithm.

SAC algorithm employed stochastic policy that has gained exploration ability. In SAC algorithm stochastic characteristic is obtained by adding the expected entropy of policy over $\rho_\pi(\boldsymbol{s}_t)$, to the RL's objective. According to Haarnoja et al this improvement considers enhancing learning performance over the RL objective [34]. This entropy-based stochasticity tends to lead with an infinite horizon that can interfere if the method becomes non-convergent. In contrast to conventional reinforcement learning algorithms that optimize for the maximum expected reward alone, the Soft Actor-Critic algorithm optimizes for both the expected cumulative reward and the policy's entropy. This entropy-based stochastic policy is related to the soft Q-function network which estimates the value of the next state, and the entropy bonus. The entropy bonus is the term that encourages exploration by maximizing the policy entropy. Moreover, in policy or actor network consist of this entropy. The entropy term is used to encourage exploration by maximizing the entropy of the policy. This combination of optimization processes in SAC algorithm aims to find a policy that balances exploration and exploitation. Furthermore, the stochastic policy can be represented as a probability distribution over actions given a state. In this SAC algorithm stochastic policy uses a Gaussian distribution. By using a Gaussian distribution, the policy can produce continuous actions in the continuous action space. Action selection in the stochastic policy is done by sampling from this probability distribution. The sampling process introduces randomness or uncertainty, allowing the policy to explore different actions at each state. This stochasticity enables the agent to explore the action space effectively and helps in developing a more flexible and adaptive policy. The stochastic policy is one of the key features of the SAC algorithm, making it highly effective in dealing with environments with continuous and complex action spaces like bipedal walking robot. By performing more efficient exploration, the algorithm can address the exploration challenge commonly encountered in continuous reinforcement learning tasks. Whereas DDPG and TD3 applied deterministic policy which can lead to suboptimal learning if the agent gets stuck in a local optimum. To address this, DDPG incorporates noise into the action selection process.

Overestimation bias is a common issue that arises in deep reinforcement learning algorithms when they use function approximators, such as neural networks, to estimate action values. In DDPG, during training, the critic network is updated using a temporal difference (TD) target that is calculated based on the Bellman equation. However, the use of function approximation and bootstrapping in DDPG can lead to overestimation of Q-values in certain situations, which can be detrimental to learning. Overestimation bias in DDPG occurs due to the maximization step during the TD target calculation. The TD target is computed using the maximum Q-value among all possible actions in the next state, even if the actor network may not accurately approximate the true Q-values. As a result, the maximization step can lead to optimistic Q-value estimates, especially when the actor's policy has not fully converged or when there is high variance in the action-value estimates. In TD3, introduces three key such as target policy smoothing, clipped double Q-Learning, and delayed policy updates modifications to DDPG to mitigate overestimation bias. In SAC algorithm, takes a different approach to address overestimation bias and is based on entropy regularization. It directly maximizes the expected reward along with the entropy of the policy. The entropy term encourages exploration and prevents premature convergence to suboptimal policies. By maximizing the expected reward and entropy jointly, SAC finds a balance between exploration and exploitation, leading to more stable and robust learning in continuous action spaces.

4.2 Result

The evaluation of the training result is based on the episode accumulated reward and the average reward. These measurements represent the reward maximization during training as the basic purpose of each algorithm. It can be used to analyze these algorithms' performance if the algorithms used the same reward function. The result of simulations is obtained from the bipedal walking robot simulation to perform the human gait at the straight-line trajectory. The performance of the SAC algorithm is evaluated and compared to DDPG and TD3. The bipedal walking robot is set to do the training at 2,000 episodes. Fig. 7 shows the episode reward comparison of each continuous DRL algorithm.

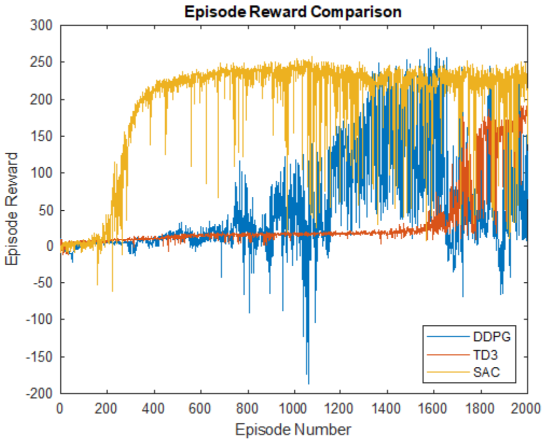

Fig. 7: The Episode Reward Comparison.

Fig 7 shows that SAC has the higher episode reward compared to DDPG and TD3. The curve of the episode reward of the SAC algorithm increases significantly at 300 episodes, while the DDPG's curve increases at 700 episodes, and TD3's curve increases at 1,600 episodes. This result indicates stochastic policy has a better exploration ability than the deterministic policy. This is represented by the simulation result of a bipedal walking robot, where stochastic policy obtains higher rewards during training. Fig 8 shows the details of the average reward of each continuous DRL approach. In view of Fig 8, it is known that the SAC algorithm can adapt faster at range 200 and 400 episodes and swiftly fix the step to maintain the failure at the first step. The learning speed also increases significantly. It occurs due to SAC algorithm ability to use the selected entropy policy and the optimal policy to gain the robustness during exploration.

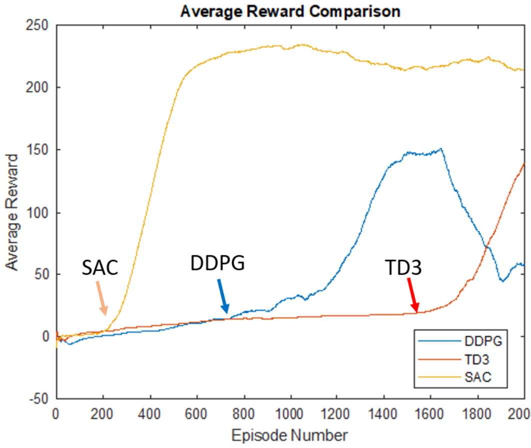

Fig.8: The Episode Reward Comparison.

Regarding Fig 7 and Fig 8, the reward begins to increase at 1,500 episodes as if the average step increases. In view of Fig 7 and Fig 8 can be described that the SAC algorithm works better than DDPG and TD3. It has better episode reward and average reward. Thus, it can be concluded that the SAC takes faster episodes to train, however TD3 remains to have more episodes during training than DDPG and SAC. The further analysis in this research compares the total reward of each continuous algorithm which can represent the implicit value of each algorithm's performance that cannot be presented by the comparison curve of the episode reward and the average reward. The comparison of the total reward is presented in Table 2.

Table 2. Total Reward Comparison.

DRL Continuous Algorithm	Total Reward
SAC	384,752.98
TD3	174,803.00
DDPG	378,711.00

This result occurs because the exploration ability increases with the stochastic policy instead of the deterministic policy. It occurs due to the stochastic domain that applied in the SAC algorithm guarantees that all states are estimated. The SAC method can suggest another mechanism to collect the experience from the environment with the current policy and to update the function approximators using the stochastic gradients from batches sampled from a replay pool. Further analysis that is the maximum entropy inside SAC can mitigate the main problem on the other DRL algorithm that is the overestimation bias. This algorithm is robust to avoid the noise at the training process. The analysis involves more parameter analysis related to the success rate of training that indicates the success rate of the robot to reach the goal. It is shown in Table 3.

Table 3. Success rate of DRL Algorithm

DRL Algorithm	Training Complete (%)	Succes Rate of Training (%)
SAC	100	90
TD3	100	20
DDPG	100	45

From Table 3 it represents that SAC algorithm has the higher success rate of training at 90%. Fig 9 shows the display of walking movement of bipedal robot. Fig 13 (a) shows the start walking of bipedal walking robot through the straight-line trajectory, and Fig 9 (b) shows the next step of the bipedal walking robot movement.

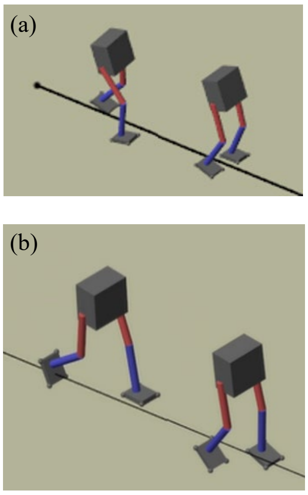

Fig. 9: The Movement of Bipedal Walking robot through the Trajectory: (a) Start of The Walking Movement, (b) Next Step Walking as The Human Gait.

In view of Fig 9, the bipedal walking robot can learn to walk through the straight-line trajectory and achieve the best total reward. It indicates SAC algorithm that using the stochastic policy has the robust training process and gain the exploration ability to get the best reward.

5. Conclusion

This study presents the implementation and comparison of stochastic-based continuous deep reinforcement learning algorithm that known as SAC with the deterministic-based algorithms. The implementation employed several algorithms of continuous deep reinforcement learning approaches that are SAC that has stochastic behaviour, and deterministic-based policy algorithm such as DDPG and TD3 for controlling the trajectory of the bipedal walking robot. The performance comparison performance of the stochastic-based DRL over the deterministic-based DRL is shown based on the results through the bipedal walking robot system. Based on the simulation results, the comparison presents that SAC has the better episode reward of 384,752.98 than DDPG of 378,711.00 and TD3 174,803.00. It is also shown based on the average reward of SAC that has the better result than DDPG and TD3. Based on the average SAC obtains higher average reward than DDPG and TD3. The stochastic-based, SAC, also makes the learning process increase significantly at 300 episodes, while the DDPG's curve increases at 700 episodes, and TD3's curve increases at 1,600 episodes. The SAC algorithm has the higher success rate of training at 90%.

From this research, promising direction of future work is applying in real robot because the algorithm is good for continuous action space. The researcher also can add the obstacle through the trajectory to add the complexity of system.

Nomenclature

CoP	Center of Point
A	Action
S	State
p_s	Transition Probability State
r	Reward
R	Expected Reward
$\boldsymbol{a}_t$	Action over time step
$\boldsymbol{s}_t$	State over time step
$\boldsymbol{s}_{t+1}$	Next State
r_{min}	Reward Minimum
r_{max}	Reward Maximum
R_b	Reply Buffer
N	OU Noise
$\nabla_{\theta^\mu} J$	Policy Gradient
$Q(\boldsymbol{s}_t, \boldsymbol{a}_t)$	State Value
$Q'(\boldsymbol{s}_t, \boldsymbol{a}_t)$	Critic Update
H	Entropy
$\aleph$	Gaussian Distribution
V^π	Soft Value Function
Q^π	Soft Q-value function
$\mathrm{R}_\mathrm{r}^\mathrm{w}$	Transformation from the world reference to the body robot
$\mathrm{R}_\mathrm{f}^\mathrm{r}$	Transformation from the robot reference to the foot reference
ZMP	The Zero Moment Point
$\boldsymbol{p}$	Matrix position in y and z axes
$\dot{\boldsymbol{p}}$	Matrix of Linear velocities at x-y-z axes
$\boldsymbol{F}$	Forces of both of right and left legs
$\boldsymbol{a}_{t-1}$	Previous Action
$\dot{p}_x$	Velocity of the bipedal robot or the movement through the x-axis
p_y	The displacement of the bipedal walking robot.
p_z	Vertical translation displacement
Ts	The environment of time during training
Tf	final time of the training

Greek symbols

$\mu(\boldsymbol{s}_t)$	Deterministic Policy
$\pi(\boldsymbol{a}_t\|\boldsymbol{s}_t)$.	Stochastic Policy
ρ_π	Trajectory distribution of Policy
$\mu'(\boldsymbol{s}_t\|\theta^{\mu'})$	Update Actor with Update Parameter $\theta^{\mu'}$
θ^μ	Actor Target Parameter
$\theta^{\mu'}$	Actor Update Target Parameter
θ^Q	Critic Target Parameter
$\theta^{Q'}$	Critic Update Target Parameter

δ_t	Mean
σ_t	Covariance
θ_{Q^π}	Target Parameter Soft Q-Value
θ_{V^π}	Target Parameter Soft Value
τ	Torque
θ_{L_1}	The left roll joint for the hip
θ_{L_2}	The left roll joint for the knee
θ_{L_3}	The left roll joint for the ankle
θ_{R_1}	The right roll joint for the hip
θ_{R_2}	The right roll joint for the knee
θ_R	The right roll joint for the ankle
θ_{F_1}	Action of right/left hip
θ_{F_2}	Action of right/left knee
θ_{F_3}	Action of right/left ankle
$\boldsymbol{\varphi}$	Matrix of roll, pitch and yaw
φ_r	Roll
φ_p	Pitch
φ_y	Yaw
$\dot{\boldsymbol{\theta}}_L$	Matrix of left foot speed of hip, knee and ankle
$\dot{\boldsymbol{\theta}}_R$	Matrix of right foot speed of hip, knee and ankle
γ	Discount factor

References

1) P.B. Khoi, N.T. Giang, and H. Van Tan, “Control and simulation of a 6-dof biped robot based on twin delayed deep deterministic policy gradient algorithm,” *Indian J. Sci. Technol.*, **14** (*31*) 2460–2471 (2021). doi:10.17485/ijst/v14i30.1030.

2) P.S. Yadav, V. Agrawal, J.C. Mohanta, and M.D.F. Ahmed, “A theoretical review of mobile robot locomotion based on mecanum wheels,” *Evergreen*, **9** (*2*) 396–403 (2022). doi:10.5109/4794163.

3) V. Shrivastava, V. Diwakar, M. Sehgal, M. Verma, and E. Neha, “Modelling and analysis of hexapod walking robot,” *Evergreen*, **9** (*2*) 378–388 (2022). doi:10.5109/4794162.

4) H. Srivastava, A. Kaushal, H. Kumar, A. Tripathi, and A.K. Sharma, “A design and development of baggage sorting robotic system at the airport,” *Evergreen*, **9** (*1*) 86–92 (2022). doi:10.5109/4774219.

5) M.H. Mohamad Nor, Z.H. Ismail, and M.A. Ahmad, “Broadcast control of multi-robot systems with norm-limited update vector,” *Int. J. Adv. Robot. Syst.*, **17** (*4*) 1–12 (2020). doi:10.1177/1729881420945958.

6) S.H. Alsamhi, O. Ma, and M.S. Ansari, “Convergence of machine learning and robotics communication in collaborative assembly: mobility, connectivity and future perspectives,” *J. Intell. Robot. Syst. Theory Appl.*, **98** (*3–4*) 541–566 (2020). doi:10.1007/s10846-019-01079-x.

7) Laxmi Kant Sagar, and D Bhagwan Das, “Fuzzy expert system for determining state of solar photo voltaic power plant based on real-time data,” *Evergreen*, **9** (*3*) 870–880 (2022). doi:10.5109/4843118.

8) A. Mayub, and Fahmizal, “Center of pressure feedback for controlling the walking stability bipedal robots using fuzzy logic controller,” *Int. J. Electr. Comput. Eng.*, **8** (*5*) 3678–3696 (2018). doi:10.11591/ijece.v8i5.pp3678-3696.

9) X. Wu, S. Liu, T. Zhang, L. Yang, Y. Li, and T. Wang, “Motion control for biped robot via ddpg-based deep reinforcement learning,” *2018 WRC Symp. Adv. Robot. Autom. WRC SARA 2018 - Proceeding*, 40–45 (2018). doi:10.1109/WRC-SARA.2018.8584227.

10) A. Kumar, N. Paul, and S.N. Omkar, “Bipedal walking robot using deep deterministic policy gradient,” *ArXiv*, (2018). http://arxiv.org/abs/1807.05924.

11) A. Arunika, J.F. Fatriansyah, and V.A. Ramadheena, “Detection of asphalt pavement segregation using machine learning linear and quadratic discriminant analyses,” *Evergreen*, **9** (*1*) 213–218 (2022). doi:10.5109/4774236.

12) P. Panwar, P. Roshan, R. Singh, M. Rai, A.R. Mishra, and S.S. Chauhan, “DDNet- a deep learning approach to detect driver distraction and drowsiness,” *Evergreen*, **9** (*3*) 881–892 (2022). doi:10.5109/4843120.

13) N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S.M.A. Eslami, M. Riedmiller, and D. Silver, “Emergence of locomotion behaviours in rich environments,” *ArXiv*, (2017). http://arxiv.org/abs/1707.02286.

14) D. Kim, S.H. Kim, T. Kim, B.B. Kang, M. Lee, W. Park, S. Ku, D.W. Kim, J. Kwon, H. Lee, J. Bae, Y.L. Park, K.J. Cho, and S. Jo, “Review of machine learning methods in soft robotics,” *PLoS One*, **16** (*2 February*) 1–24 (2021). doi:10.1371/journal.pone.0246102.

15) C. Liu, A.G. Lonsberry, M.J. Nandor, M.L. Audu, A.J. Lonsberry, and R.D. Quinn, “Implementation of deep deterministic policy gradients for controlling dynamic bipedal walking,” *MDPI Biomimetics*, **4** (*28*) 1–20 (2019). doi:10.3390/biomimetics4010028.

16) T.P.A. Wiggers, “Learning Bipedal Locomotion over Challenging Terrain,” University of Amsterdam, 2021.

17) J.Y. Kim, I.W. Park, and J.H. Oh, “Walking control algorithm of biped humanoid robot on uneven and inclined floor,” *J. Intell. Robot. Syst. Theory Appl.*, **48** (*4*) 457–484 (2007). doi:10.1007/s10846-006-9107-8.

18) S.S. Mousavi, M. Schukat, and E. Howley, “Deep reinforcement learning: an overview,” *Lect. Notes Networks Syst.*, **16** 426–440 (2018). doi:10.1007/978-3-319-56991-8_32.

19) K. Arulkumaran, M.P. Deisenroth, M. Brundage, and

A.A. Bharath, “Deep reinforcement learning: a brief survey,” *IEEE Signal Process. Mag.*, **34** (*6*) 26–38 (2017). doi:10.1109/MSP.2017.2743240.

20) D. Rastogi, “Deep reinforcement learning for bipedal robots divyam rastogi,” (2017).
21) T. Tiong, and I. Saad, “Reinforcement learning with deep deterministic policy gradient,” *Proc. - 2021 Int. Conf. Artif. Intell. Big Data Algorithms, CAIBDA 2021*, 82–85 (2021). doi:10.1109/CAIBDA53561.2021.00025.
22) T. Haarnoja, “Acquiring Diverse Robot Skills via Maximum Entropy Deep Reinforcement Learning,” University of California, 2018. https://escholarship.org/content/qt25g6573w/qt25g6573w_noSplash_f9a057588134c4eda05c80fdf4ba8212.pdf.
23) D. Silver, G. Lever, D. Technologies, G.U.Y. Lever, and U.C.L. Ac, “Deterministic Policy Gradient Algorithms,” in: T.J. Eric P. Xing (Ed.), Proc. 31 St Int. Conf. Mach. Learn., Beijing china, 2014. http://proceedings.mlr.press/v32/silver14.pdf.
24) J. Schulman, “OPT IMIZING EXPECTATIONS : FROM DEEP RE INFORCEMENT LEARNING TO STOCHAST I C COMPUTAT ION GRAPHS,” University of California, 2016. http://joschu.net/docs/thesis.pdf.
25) T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, “Reinforcement learning with deep energy-based policies,” *34th Int. Conf. Mach. Learn. ICML 2017*, **3** 2171–2186 (2017).
26) E. Prianto, M. Kim, J. Kim, and J. Bae, “Path planning for multi-arm manipulators using deep reinforcement learning : soft actor – critic with hindsight experience replay,” *MDPI Sens.*, **20** (*5911*) 1–22 (2020). doi:10.3390/s20205911.
27) Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Aral, N. Koyachi, and K. Tanie, “Planning walking patterns for a biped robot,” *IEEE Trans. Robot. Autom.*, **17** (*3*) 280–289 (2001). doi:10.1109/70.938385.
28) S. Castro, “Modelling an Simulation of Walking Robots,” 2019. https://www.mathworks.com/videos/modeling-and-simulation-of-walking-robots-1576560207573.html.
29) S. Castro, “Walking robot control: from pid to reinforcement learning » racing lounge - matlab & simulink,” 2019–2022 (2021). https://blogs.mathworks.com/racing-lounge/2019/04/24/walking-robot-control/ (accessed November 4, 2021).
30) A. Berthet, “Review of Deep Reinforcement Learning Algorithms,” Paris, France, 2020.
31) O. Kilinc, Y. Hu, and G. Montana, “Reinforcement learning for robotic manipulation using simulated locomotion demonstrations,” **1** 1–12 (2020).
32) M. Fang, C. Zhou, B. Shi, B. Gong, J. Xu, and T. Zhang, “DHER : HINDSIGHT EXPERIENCE REPLAY FOR DYNAMIC GOALS,” in: ICLR, 2019: pp. 1–12.
33) J. Achiam, and P. Abbeel, “Deep deterministic policy gradient,” *Open AI Spinn. Up*, 1–12 (2021). https://spinningup.openai.com/en/latest/algorithms/ddpg.html%0Aupdate_every.
34) S. Fujimoto, H. Van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” *ArXiv*, (2018).
35) T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft actor-critic algorithms and applications,” (2018). http://arxiv.org/abs/1812.05905.
36) T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor,” *35th Int. Conf. Mach. Learn. ICML 2018*, **5** 2976–2989 (2018).
37) B.D. Ziebart, “Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy,” Carnegie Mellon University, 2010. doi:10.1159/000029047.
38) J.I. Kim, M. Hong, K. Lee, D. Kim, Y.L. Park, and S. Oh, “Learning to walk a tripod mobile robot using nonlinear soft vibration actuators with entropy adaptive reinforcement learning,” *IEEE Robot. Autom. Lett.*, **5** (*2*) 2317–2324 (2020). doi:10.1109/LRA.2020.2970945.
39) T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” in: 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc., ArXiv, London UK, 2016.
40) J. Englsberger, C. Ott, M.A. Roa, A. Albu-Schäffer, and G. Hirzinger, “Bipedal walking control based on capture point dynamics,” *IEEE Int. Conf. Intell. Robot. Syst.*, 4420–4427 (2011). doi:10.1109/IROS.2011.6048045.
41) M. VUKOBRATOVIĆ, and B. BOROVAC, “Zero-moment point — thirty five years of its life,” *Int. J. Humanoid Robot.*, **01** (*01*) 157–173 (2004). doi:10.1142/s0219843604000083.
42) D.P. Kingma, and J.L. Ba, “Adam: a method for stochastic optimization,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, 1–15 (2015).
43) R.S. Sutton, and A.G. Barto, “Reinforcement Learning: An Introduction (2nd Edition, in preparation),” 2018. http://ir.obihiro.ac.jp/dspace/handle/10322/3933.