

Received 30 November 2023, accepted 15 December 2023, date of publication 18 December 2023, date of current version 18 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3344393

RESEARCH ARTICLE

Comparative Analysis of Reinforcement Learning Algorithms for Bipedal Robot Locomotion

OMUR AYDOGMUS¹ AND MUSA YILMAZ^{2,3}, (Senior Member, IEEE)

¹Department of Mechatronics Engineering, Faculty of Technology, Firat University, 23119 Elazig, Turkey

²Center for Environmental Research and Technology, Bourns College of Engineering, University of California at Riverside, Riverside, CA 92521, USA

³Department of Electrical and Electronics Engineering, Batman University, 72100 Batman, Turkey

Corresponding author: Musa Yilmaz (musay@ucr.edu)

ABSTRACT In this research, an optimization methodology was introduced for improving bipedal robot locomotion controlled by reinforcement learning (RL) algorithms. Specifically, the study focused on optimizing the Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), Soft Actor-Critic (SAC), and Twin Delayed Deep Deterministic Policy Gradients (TD3) algorithms. The optimization process utilized the Tree-structured Parzen Estimator (TPE), a Bayesian optimization technique. All RL algorithms were applied to the same environment, which was created within the OpenAI GYM framework and known as the bipedal walker. The optimization involved the fine-tuning of key hyperparameters, including learning rate, discount factor, generalized advantage estimation, entropy coefficient, and Polyak update parameters. The study comprehensively analyzed the impact of these hyperparameters on the performance of RL algorithms. The results of the optimization efforts were promising, as the fine-tuned RL algorithms demonstrated significant improvements in performance. The mean reward values for the 10 trials were as follows: PPO achieved an average reward of 181.3, A2C obtained an average reward of -122.2 , SAC reached an average reward of 320.3, and TD3 had an average reward of 278.6. These outcomes underscore the effectiveness of the optimization approach in enhancing the locomotion capabilities of the bipedal robot using RL techniques.


INDEX TERMS Hyperparameter optimization, reinforcement learning, robot motion.

I. INTRODUCTION

In recent years, mobile robots have become increasingly integrated into our daily lives, and they come in various types. In general, robots can be classified into two main categories: wheeled robots and legged robots [1]. Furthermore, it's possible to combine wheeled and legged components to create various types of robots [2]. While wheeled robots are common, legged robots mimic nature and offer superior efficiency. Legged robots are well-suited for navigating rough terrains, climbing steps, crossing wide gaps, and traversing extremely uneven surfaces where wheeled robots would struggle due to ground irregularities [3]. However, these advantages come with the need for more complex control algorithms [4]. Traditional walking algorithms have limitations, leading to restricted motion [5]. Therefore, there is a growing need for learning-based walking algorithms

for legged robots [6]. In recent times, four-legged and two-legged robots have gained popularity [7], [8]. Biped robots have an exceptional ability to mimic human-like mobility, making them highly suitable for tasks that require human-like dexterity and agility, such as complex search and rescue missions. By profiting from their ability to mimic human characteristics, they can outperform conventional robots in a variety of fields [9].

The structure of this paper unfolds as follows: It commences with an introduction, providing insight into the motivation behind the presented work and offering information related to previous research on the control of bipedal robots and bipedal locomotion concepts. A comprehensive definition of the robot and its environment has been offered in the second section. Here, we provide a detailed account of the action and observation states, integral components used in the training process of Reinforcement Learning (RL) algorithms. The third section discusses the details of the RL algorithms used, including Proximal Policy

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Liu .

Optimization (PPO), Advantage Actor-Critic (A2C), Soft Actor-Critic (SAC), and Twin Delayed Deep Deterministic Policy Gradients (TD3). In the fourth section, the focus shifts to the analysis of hyperparameter optimization for RL algorithms, a key objective of this study. The concluding section offers a discussion that centers on the comparative performance of the RL algorithms. Additionally, we outline potential directions for future research, encapsulating the paper's findings and insights.

A. BACKGROUND AND RELATED WORKS

Bipedal robots generate more interest than other mobile robots that use conventional locomotion. They offer a significantly wider range of applications, such as search and rescue, industrial diagnostics, exploration of unknown areas, nuclear plants, and dangerous environments, among others [10]. Due to their reduced ground contact area and fewer driving effectors, bipedal robots may exhibit a potentially lower total energy consumption when compared to their multi-legged robots [11].

While the classical model-based control techniques have been suggested for bipedal robots, including model predictive control [12], robust control [13], and trajectory tracking control [14]. These control approaches are generally pre-calculated with lack flexibility. As a result, the bipedal robots often fall short in terms of stability, adaptability, and robustness. In addition, optimization-based control strategies are advanced for dynamic legged robots [4].

Obtaining a mathematical-based control approach for bipedal robot locomotion in diverse environments, while ensuring stability comparable to that of a human, is a challenging task. Consequently, reinforcement learning (RL)-based approaches for bipedal robots show promise in overcoming the complexity of control algorithms. Through RL, the robot can autonomously learn to walk with stability control on unknown and unexpectedly complex surfaces. Data-driven methods offer a versatile framework that empowers legged robots to exhibit a wide range of behaviors by incorporating reference motion terms into the learning process [6].

However, the development of RL-based algorithms presents a significant challenge in terms of computational time. Training on a real robot introduces various difficulties. The most rational approach is to create the robot within a simulation environment and conduct RL-based algorithm training. Several robot simulators are available in the literature, with popular options including Gazebo [15], Webots [16], CoppeliaSim [17], and IsaacSim [18]. These simulators offer realistic visualization with real dynamics. Nonetheless, to optimize hyperparameters that impact RL efficiency and reduce training time, it may be necessary to use a simplified robot structure. For this reason, Gym or Gymnasium, developed by OpenAI, aims to provide a simplified environment for faster training in RL studies. Also MuJoCo is a general purpose physics engine that aims to facilitate research and development in robotics and

various applications required physics engine. This tool developed by DeepMind. Thanks to these tools, they have been possible to develop many RL-based algorithms with low computational cost.

RL has shown promise in enhancing bipedal locomotion. Various studies have applied deep reinforcement learning (DRL) to this field. One research effort introduced a framework developed using MuJoCo that employs passive dynamics via DRL to generate multimodal bipedal locomotion. This approach successfully replicates natural walking patterns, underscoring DRL's potential in capturing complex bipedal dynamics [19]. A novel RL framework for 3D bipedal locomotion control policies was presented in a separate study. The authors utilized deep neural networks and RL algorithms to create adaptive control strategies adaptable to different terrains and disturbances, highlighting RL's potential in designing adaptable locomotion for bipedal walkers created using MuJoCo [20]. In its simplest form, a bipedal robot can be modeled with 4 degrees of freedom (DOF). Using the robot structure known as a bipedal walker in the literature, the walking algorithm provides an advantage in terms of computational power. Deep Q-learning and augmented random search have been used to teach a simulated two-dimensional bipedal robot in the Gym BipedalWalker-v3 environment [21]. Another study in the same environment presented an evaluation procedure via supervised learning, using a randomly initialized neural network to imitate the set and then execute the acquired policy against the environment [22]. PPO and Deep Deterministic Policy Gradients (DDPG) have been compared for same environment [23]. SAC has been used for Gym BipedalWalker-v3 environment, too [24].

B. MOTIVATION

The impact of hyperparameters on the performance of reinforcement learning algorithms is significant. Hyperparameters serve as adjustable settings that fine-tune the learning process and exert a substantial influence on the trajectory, stability, and ultimate success of RL-based tasks. The choice of appropriate hyperparameters can mean the difference between an RL algorithm swiftly converging to an optimal policy or becoming stuck at a suboptimal solution, or even worse, leading to undesirable drift. Furthermore, hyperparameters often require customization to suit specific robotic platforms and environmental conditions, presenting a non-trivial challenge in their selection. Poor hyperparameter choices can result in protracted training times, increased sample requirements, and limited adaptability to unexpected scenarios. These issues are particularly critical in applications such as bipedal robot control, where real-time adaptability and efficiency are of paramount importance. Therefore, understanding and optimizing hyperparameters is essential to unlock the full potential of RL in facilitating agile and adaptive robotic systems. Nevertheless, due to the numerous independent studies in the literature, it is challenging to

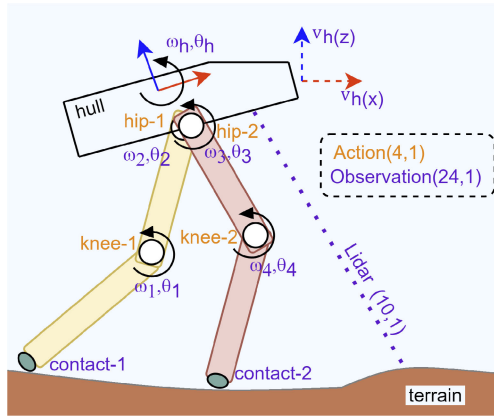


FIGURE 1. Action and observation states of robot.

determine which RL algorithm performs best with specific parameters.

For this reason, the present study aims to optimize various hyperparameters for different RL algorithms in a manner that is easily comprehensible and comparable in the literature. This approach will help elucidate which algorithm and hyperparameter combination is best suited for bipedal robots.

II. DEFINITIONS OF ROBOT AND ENVIRONMENT

In this study, a simple 4-joint walker robot environment was utilized to conduct a comparative analysis of various RL algorithms with the aim of optimizing their hyperparameters. The robot comprises a central hull and two legs, each equipped with both knees and hips, as illustrated in Fig. 1. The observation state encompasses several crucial variables as given in (1). These observation state variables are highlighted in purple in Fig. 1.

$$O_s = [\theta_h, \omega_h, v_x, v_z, \theta_{j1}, \omega_{j1}, \theta_{j2}, \omega_{j2}, C_{L1}, \theta_{j2}, \omega_{j2}, \theta_{j2}, \omega_{j2}, C_{L2}, \text{lidar}_{(0-9)}] \quad (1)$$

The observation state has the hull's angle speed (θ_h), angular velocity (ω_h), horizontal speed (v_x), vertical speed (v_z), joint positions, joint angular speeds (θ_j, ω_j), the state of leg-ground contact (C_{L1-2} expressed as a binary value: True or False), and data from 10 lidar rangefinder measurements for terrain feedback.

For this experiment, slightly uneven terrain, corresponding to the standard version in the GYM environment, was selected. Additionally, the standard reward function was employed during the training of the RL algorithms. The reward mechanism was designed to incentivize forward movement, providing a cumulative total of 300+ points as the robot progresses towards the far end. Conversely, if the robot falls, it incurs a penalty of -100 points. Furthermore, applying motor torque results in a marginal deduction of points.

III. RL ALGORITHMS

A. PPO

PPO is a popular reinforcement learning algorithm used to train agents in environments where they must learn a

policy to perform tasks and maximize cumulative rewards. PPO is known for its stability and efficiency in training DRL agents. It was introduced by Schulman et al. in 2017 [25]. The algorithm of the PPO algorithm is given in Algorithm 1. Additionally, you can access the details of all the algorithms mentioned in this paper from the OpenAI webpage.

PPO builds on the foundations of previous policy gradient methods and offers improvements in terms of both computational efficiency and sample complexity. PPO uses two neural networks: a policy network that generates actions based on the current state, and a value network that estimates expected cumulative rewards. What sets PPO apart is its trimmed surrogate objective function, designed to prevent the policy from making large and potentially harmful changes during updates. This clipping is implemented by a hyperparameter known as the "clipping parameter" that limits the update size of the policy. In addition to policy updates, PPO optimizes the value function using a standard value loss. Data is collected through interactions with the environment and multiple update cycles are applied iteratively to improve the agent's performance.

Algorithm 1 Proximal Policy Optimization (PPO)

Input: Initial policy π_θ and value function V_ϕ

Output: Updated policy π'_θ and value function V'_ϕ

while not converged **do**

 Collect a batch of trajectories using π_θ

 Compute advantages $A(s, a)$ based on the collected trajectories

 Update the policy by maximizing the PPO objective: **for each policy update do**

 Calculate the surrogate objective:

$$L(\theta) = \mathbb{E} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A(s, a) \right]$$

 Update policy parameters θ using gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta L(\theta)$$

end

 Fit the value function V_ϕ to the collected data:

$$\phi \leftarrow \operatorname{argmin}_\phi \mathbb{E} \left[(V_\phi(s) - V_{\text{target}}(s))^2 \right]$$

end

return π'_θ and V'_ϕ

B. A2C

A2C is another popular reinforcement learning algorithm used to train agents in environments where they must learn a policy to perform tasks and maximize cumulative rewards. A2C is a synchronous variant of the Asynchronous Advantage Actor-Critic (A3C) algorithm. It combines the advantages of both policy gradient methods and value

function estimation methods. The term ‘‘Actor-Critic’’ refers to the fact that it simultaneously learns both a policy (actor) and a value function (critic) [26]. The pseudo code of A2C is given in Algorithm 2.

A2C introduces the concept of the advantage function, measuring the relative merit of chosen actions compared to the average actions within a given state. This advantage guides policy updates, encouraging actions that lead to higher advantages and, in turn, enhanced expected returns. Simultaneously, the value function is updated to improve the accuracy of state value estimation, reducing the error between estimated and actual returns through Mean Squared Error (MSE) loss. A2C offers synchronous and asynchronous training options, allowing multiple agents to collect experiences in parallel. A2C’s strength lies in its robust sample efficiency, training stability, and adaptability to both discrete and continuous action spaces, making it a valuable choice for diverse reinforcement learning tasks across domains such as game playing, robotics, and autonomous control.

Algorithm 2 Advantage Actor-Critic (A2C)

Input: Initial policy π_θ and value function V_ϕ
Output: Updated policy π'_θ and value function V'_ϕ
 Initialize policy and value function parameters
 θ and ϕ
for each episode do
 Initialize an empty trajectory τ
 Initialize the environment with the initial state s_0
 while not done do
 Sample an action a_t from the policy π_θ
 Action a_t , observe reward r_t , next state s_{t+1}
 Add (s_t, a_t, r_t) to the trajectory τ
 Update the current state: $s_t \leftarrow s_{t+1}$
 end
 Compute returns and advantages for each time step in the trajectory: **for each time step t in τ do**
 Compute return G_t based on future rewards
 Compute advantage A_t using function V_ϕ
 end
 Update the policy and value function using the trajectory: **for each time step t in τ do**
 Update the policy using the advantage:

$$\theta \leftarrow \theta + \alpha_\theta \nabla_{\theta} \log(\pi_\theta(a_t|s_t))A_t$$

 Update function by temporal difference error:

$$\phi \leftarrow \phi + \alpha_\phi (G_t - V_\phi(s_t)) \nabla_{\phi} V_\phi(s_t)$$

 end
end
return π'_θ and V'_ϕ

C. SAC

SAC is a state-of-the-art reinforcement learning algorithm designed for training agents in continuous action space

environments. It builds upon the framework of DDPG and addresses some of its limitations. SAC is known for its stability, sample efficiency, and strong performance in a wide range of tasks [27]. The pseudo code of SAC is given in Algorithm 3.

This off-policy algorithm brings several key innovations to the table to enhance training efficiency and sample effectiveness. At its core, SAC adopts the actor-critic architecture, featuring a neural network-based actor for action selection and a critic for estimating the state-action value function. What sets SAC apart is its emphasis on entropy regularization, which encourages effective exploration by promoting action stochasticity. By maximizing entropy in the objective function, SAC achieves a delicate balance between exploration and exploitation. SAC’s off-policy nature allows the utilization of a replay buffer, breaking temporal data correlations and improving sample efficiency. The agent’s value function, or Q-function, is learned concurrently with the policy, and automatic temperature adjustment dynamically regulates the entropy term during training. This combination of features enables SAC to excel in environments with continuous action spaces, making it a robust choice for a variety of applications, including robotic control, autonomous systems, and game-playing agents. Careful hyperparameter tuning and experience replay buffers play a significant role in successful SAC training.

D. TD3

TD3 is an extension of the DDPG algorithm. TD3 is designed for solving reinforcement learning problems in environments with continuous action spaces, and it addresses some of the limitations of DDPG, such as issues related to overestimation bias in the Q-value estimates. TD3 is known for its stability, robustness, and improved performance in challenging continuous control tasks [28]. The pseudo code of TD3 is given in Algorithm 4.

TD3 addresses the challenges faced during agent training, focusing on improving stability, robustness, and sample efficiency. It retains the actor-critic architecture, where the actor generates actions and the critic estimates state-action values. Notably, TD3 introduces twin critics to mitigate overestimation bias, a common issue in Q-learning. These twin critics produce averaged Q-values for more accurate value estimates. Additionally, TD3 incorporates delayed policy updates, ensuring that actor network updates are less frequent compared to the critic network to enhance training stability. Exploration is facilitated through the addition of noise to actions. The algorithm also clips target Q-values to prevent instability. TD3 leverages off-policy learning, allowing the use of experience replay for efficient and stable training. With careful hyperparameter tuning and experience replay buffers, TD3 excels in addressing challenges in environments with continuous action spaces, making it a robust choice for a wide range of applications, including

Algorithm 3 Soft Actor-Critic (SAC)

Input: Initial policy π_θ , Q-functions Q_{ϕ_1} and Q_{ϕ_2} , and value function V_ϕ

Output: Updated policy π'_θ , Q-functions Q'_{ϕ_1} and Q'_{ϕ_2} , and value function V'_ϕ

Initialize policy parameters θ , Q-function parameters ϕ_1 and ϕ_2 , and value function parameters ϕ

Initialize target value function V_{target}

Initialize a replay buffer D

for each episode do

Initialize the environment with the initial state s_0

while not done do

Sample an action a_t from the policy π_θ

Execute action a_t , reward r_t , next state s_{t+1}

Store (s_t, a_t, r_t, s_{t+1}) in the replay buffer D

Update the current state: $s_t \leftarrow s_{t+1}$

Sample a minibatch of transitions from D

Update the Q-functions using soft Bellman:

$$Q_{\phi_1}(s, a) \leftarrow \mathbb{E}_{s', r \sim D} [r + \gamma \cdot \min(Q'_{\phi_1}(s', \pi'_\theta(s')), \dots Q'_{\phi_2}(s', \pi'_\theta(s')) - \alpha \log(\pi'_\theta(s', a)))]$$

Update the policy to maximize the entropy-regularized objective:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim D} [\nabla_a Q_{\phi_1}(s, a)|_{a=\pi_\theta(s)} \dots \dots \nabla_\theta \log(\pi_\theta(s, a))|_{a=\pi_\theta(s)}]$$

Update target function using a soft update:

$$V_{\text{target}}(s) \leftarrow (1 - \tau)V_{\text{target}}(s) + \tau \cdot \min(Q'_{\phi_1}(s, \pi'_\theta(s)), Q'_{\phi_2}(s, \pi'_\theta(s)) - \alpha \log(\pi'_\theta(s, \pi'_\theta(s))))$$

end

end

return $\pi'_\theta, Q'_{\phi_1}, Q'_{\phi_2}$, and V'_ϕ

robotic control, autonomous systems, and game-playing agents.

IV. OPTIMIZATION OF HYPERPARAMETERS

This study optimized hyperparameters for improved RL algorithm performance, focusing on identifying the best RL algorithm for bipedal robot locomotion. A Bayesian optimization technique known as Tree-structured Parzen estimator (TPE) was used for hyperparameter optimization powered by Optuna [29]. It's designed to efficiently search for the best set of hyperparameters for a given machine learning model. TPE is particularly useful when optimizing complex models with various hyperparameters. Instead of performing an exhaustive grid search or random search, TPE focuses the search on the most promising regions of the hyperparameter space [30]. The algorithm has been proposed by to determine hyperparameters [31].

The concept of conditional probability from Bayes' theory is introduced. Here, $p(x|y)$ denotes the conditional probability that the hyperparameter is x when the model's loss is y . In the initial step, a threshold, y^* , for the loss is selected based

Algorithm 4 Twin Delayed Deep Deterministic Policy Gradients (TD3)

Input: Initial policy π_θ , Q-functions Q_{ϕ_1} and Q_{ϕ_2}

Output: Updated policy π'_θ , Q-functions Q'_{ϕ_1}, Q'_{ϕ_2}

Initialize policy parameters θ , Q-function parameters ϕ_1 and ϕ_2

Initialize target Q-functions Q'_{ϕ_1} and Q'_{ϕ_2}

Initialize target policy π'_θ and a replay buffer D

for each episode do

Initialize the environment with the initial state s_0

while not done do

Sample an action a_t from the policy π_θ

Action a_t , observe reward r_t , next state s_{t+1}

Store (s_t, a_t, r_t, s_{t+1}) in the replay buffer D

Sample a minibatch of transitions from D

Compute the target Q-functions:

$$y = r_t + \gamma \cdot \min(Q'_{\phi_1}(s_{t+1}, \pi'_\theta(s_{t+1})), \dots Q'_{\phi_2}(s_{t+1}, \pi'_\theta(s_{t+1})))$$

Update Q-functions using Bellman error:

$$\nabla_{\phi_i} J(\phi_i) = \mathbb{E} [(Q_{\phi_i}(s_t, a_t) - y)^2]$$

Update the policy using the deterministic policy gradient:

$$\nabla_\theta J(\theta) = \mathbb{E} [\nabla_a Q_{\phi_1}(s_t, a)|_{a=\pi_\theta(s_t)} \nabla_\theta \pi_\theta(s_t)]$$

Add target policy smoothing noise:

$$a' = \pi'_\theta(s_{t+1}) + \text{clip}(\mathcal{N}(0, \sigma), -\text{clip}, \text{clip})$$

Update target Q-functions with a soft update:

$$\phi'_i \leftarrow \tau \phi_i + (1 - \tau) \phi'_i, \quad i = 1, 2$$

Update target policy with a soft update:

$$\theta' \leftarrow \tau \theta + (1 - \tau) \theta'$$

end

end

return $\pi'_\theta, Q'_{\phi_1}, Q'_{\phi_2}$

on the available data. Subsequently, two probability density functions, $l(x)$ and $g(x)$, are developed for data greater than the threshold and less than the threshold, respectively.

The concept of conditional probability from Bayes' theory is introduced. Here, $p(x|y)$ denotes the conditional probability that the hyperparameter is x when the model's loss is y . In the initial step, a threshold, y^* , for the loss is selected based on the available data. Subsequently, two probability density functions, $l(x)$ and $g(x)$ given in Eq. 2, are developed for data greater than the threshold and less than the threshold, respectively.

$$p(x|y) = \begin{cases} l(x) & \text{if } y < y^*, \\ g(x) & \text{if } y \geq y^* \end{cases} \quad (2)$$

where $l(x)$ represent the density function formed using a subset of observations denoted as $\{x^{(i)}\}$, where the corresponding loss function $f(x^{(i)})$ is guaranteed to be less than the threshold y^* . Meanwhile, $g(x)$ denotes the density function formed using the remaining observations. This parametrization splits the joint probability distribution $p(x, y)$ into two components: $p(y)$, which models the distribution of the parameters y , and $p(x|y)$, which models the distribution of the objective function values x given the parameters y . This separation allows the TPE algorithm to more efficiently explore and exploit the parameter space while searching for the optimal solution. The Expected Improvement (EI), given in Eq. 3, is a utility function commonly used in Bayesian optimization to quantify the potential improvement in the objective function. By breaking down $p(x, y)$ into $p(y)$ and $p(x|y)$, the computation of EI becomes more tractable and can be optimized effectively, making the TPE algorithm a popular choice in Bayesian optimization for tasks such as hyperparameter tuning and global optimization [32].

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y)p(y|x)dy$$

$$= \dots \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y)p(y)}{p(x)} dy \quad (3)$$

The Eq. 3 can be modified and simplified as given in Eq. 4

$$EI_{y^*}(x) = \frac{\gamma y^* l(x) - l(x) \int_{-\infty}^{y^*} p(y) dy}{\gamma l(x) + (1 - \gamma)g(x)} \propto$$

$$\dots \left(\gamma + \frac{g(x)}{l(x)}(1 - \gamma) \right)^{-1} \quad (4)$$

Therefore, the approach involves minimizing the ratio $g(x)/l(x)$ to obtain a new x . Subsequently, the new x is reinserted into the function, and the iterative process continues to adapt both the functions $l(x)$ and $g(x)$. This iterative cycle persists until the predetermined number of iterations is reached, resulting in the completion of the hyperparameter optimization [32].

Learning rates are key in RL, shaping learning speed and quality, and striking a balance between exploration and exploitation. The discount factor influences the agent's time preference for rewards, impacting long-term planning and risk-taking in RL tasks. Proper tuning of these parameters is essential for achieving success in various RL applications. In the study, the learning rate (η) and discount factor (γ) parameters were optimized for all RL algorithms to determine their most suitable values. The results, as illustrated in Figure 2, reveal that the best-performing η and γ values cluster in the same region. The contour plot uses darker colors to indicate higher rewards. Specifically, the most effective η value is $8.27e^{-4}$, and the optimal discount factor value is 0.96979, resulting in a corresponding reward value of 294.016.

As depicted in Figure 3, similar outcomes are observed for the A2C algorithm. Nevertheless, it's important to note that

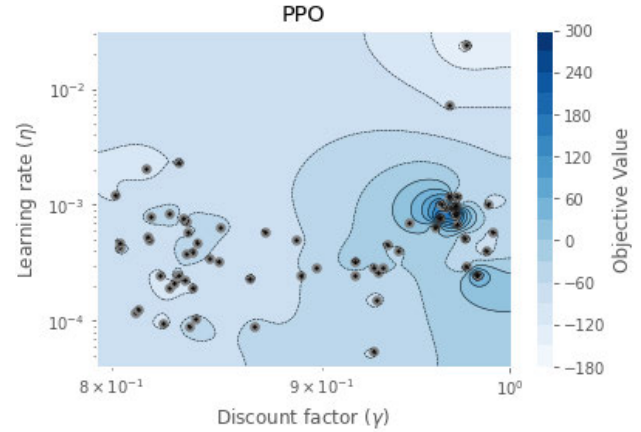


FIGURE 2. Contour plot for learning rate and discount factor with objective value of PPO algorithm.

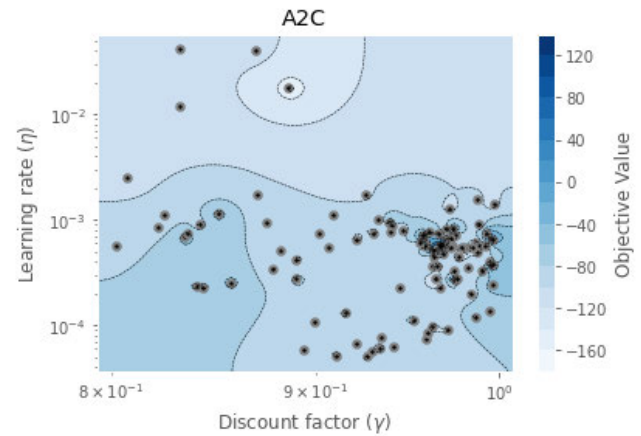


FIGURE 3. Contour plot for learning rate and discount factor with objective value of A2C algorithm.

the A2C reward value is notably lower, standing at 120.461, in comparison to PPO.

The optimized values for the SAC algorithm are concentrated in a tight cluster, as illustrated in Figure 4. During the optimization process of the SAC algorithm, the reward value surpassed 300, triggering an early stop at the 254th epoch using the EarlyStop callback. This resulted in a remarkably high reward value of 319.501.

Similar to A3C, the parameters yielding the best results are clustered in the lower right corner of the contour plot shown in Figure 4. In this context, TD3 achieved the best results with values of $9.78e^{-4}$ for η and 0.97857 for γ , respectively. The highest reward achieved with TD3 is 292.599.

In the field of reinforcement learning, several key concepts and techniques enhance the training of agents. Generalized Advantage Estimation (GAE) is one such technique that focuses on estimating the advantage function, providing a more accurate measure of an action's superiority or inferiority compared to the average action in a given state. The GAE incorporates a parameter known as (λ) to balance the consideration of short-term and

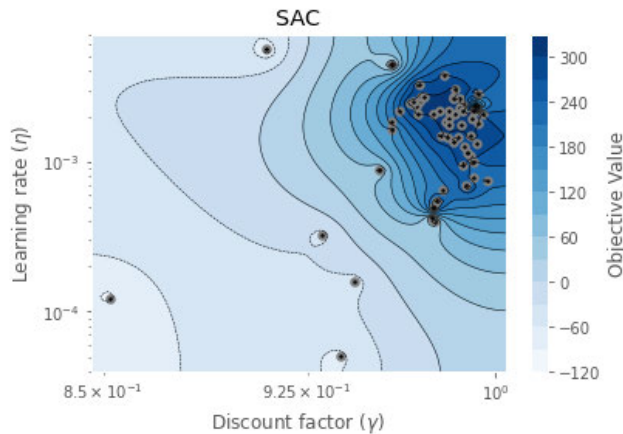


FIGURE 4. Contour plot for learning rate and discount factor with objective value of SAC algorithm.

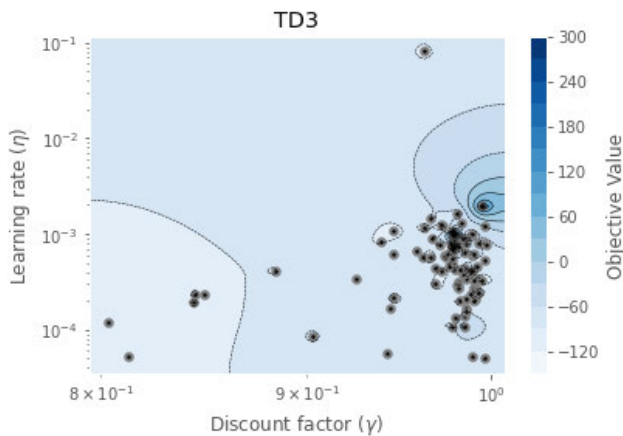


FIGURE 5. Contour plot for learning rate and discount factor with objective value of TD3 algorithm.

long-term rewards. Entropy coefficient, represented as (β) is a vital hyperparameter utilized in reinforcement learning algorithms like PPO and A2C. It governs the policy distribution's entropy, enabling the fine-tuning of the balance between exploration and exploitation. Lastly, Polyak update (τ), stabilizes training in deep reinforcement learning by maintaining two sets of parameters - the "target" and "current" networks. The Polyak update gradually modifies the target network's parameters, reducing variance and preventing rapid updates to ensure a more stable learning process.

In this study, the parameters (λ), (β), and (τ) were optimized. It is important to note that these parameters are specific to certain RL algorithms used in this study and are presented separately in Table 1. Additionally, the impact of all hyperparameters on the performance of the RL algorithms is visualized in Figure 6. Notably, as depicted in the figure, the γ parameter stands out with a more significant impact compared to the other hyperparameters.

TABLE 1. Optimal hyperparameters achieved through optimization process.

Hyperparameters	PPO	A2C	SAC	TD3
Learning Rate (η)	$8.27e^{-4}$	$5.64e^{-4}$	$2.52e^{-3}$	$9.78e^{-4}$
Discount Factor (γ)	0.96979	0.96685	0.96667	0.97857
GAE Parameter (λ)	0.92680	0.91653	-	-
Entropy Coef. (β)	$1.99e^{-5}$	$1.04e^{-4}$	-	-
Polyak update (τ)	-	-	$1.68e^{-1}$	$4.88e^{-2}$

TABLE 2. Test scores (Rewards) for optimal RL algorithms on bipedal walker.

Test N	PPO	A2C	SAC	TD3
Test-1	41.9	-111.4	320.7	281.3
Test-2	307.0	-131.7	320.4	279.0
Test-3	306.4	-105.8	320.0	277.0
Test-4	305.7	-113.8	320.7	278.7
Test-5	117.3	-122.6	320.6	276.1
Test-6	308.5	-109.9	319.5	280.3
Test-7	20.9	-120.6	319.5	279.5
Test-8	23.0	-129.7	320.9	279.3
Test-9	310.1	-147.1	320.8	276.9
Test-10	72.4	-129.8	319.4	278.0
Mean (1-10)	181.3	-122.2	320.3	278.6

The optimization process times were as follows: it took about 5 hours for 1000 epochs of PPO training, 6 hours for 1000 epochs of A2C training, 18 hours for 248 epochs of SAC training, and 43 hours for 1000 epochs of TD3 training. Notably, the SAC training process was terminated at the 248th epoch due to the earlyStop callback.

RL algorithms with optimized hyperparameters were tested 10 times in the same environment, and a performance analysis was conducted. The results are presented in Table 2. While PPO exceeded 300 points in a few of the tests, it achieved an average score of 181.3 points after 10 tests. On the other hand, A2C failed all of the tests. SAC, in contrast, consistently scored approximately 320 points in all tests, resulting in a satisfactory average of 320 points. TD3 consistently ranked as the second-best algorithm with a score of about 280. Despite PPO exceeding 300 points in some tests, it struggled to maintain the expected performance consistently. As evident, SAC emerged as the best RL algorithm for this study, showcasing both high stability and a high score.

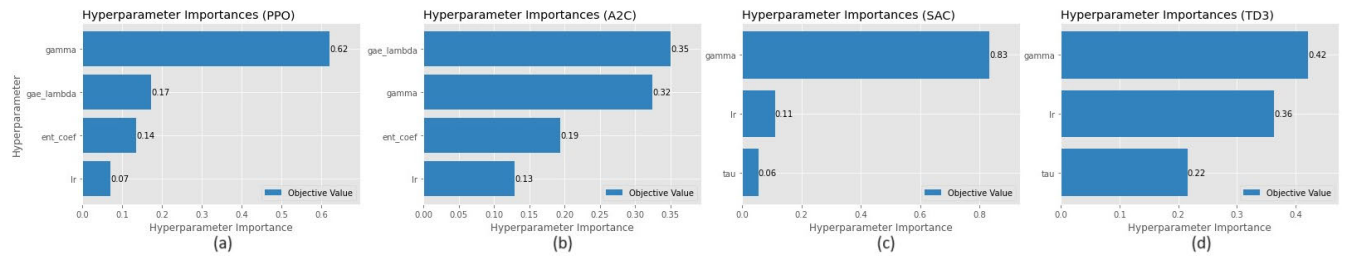


FIGURE 6. Importance of hyperparameters on objective value; a) PPO, b) A2C, c) SAC, d) TD3.

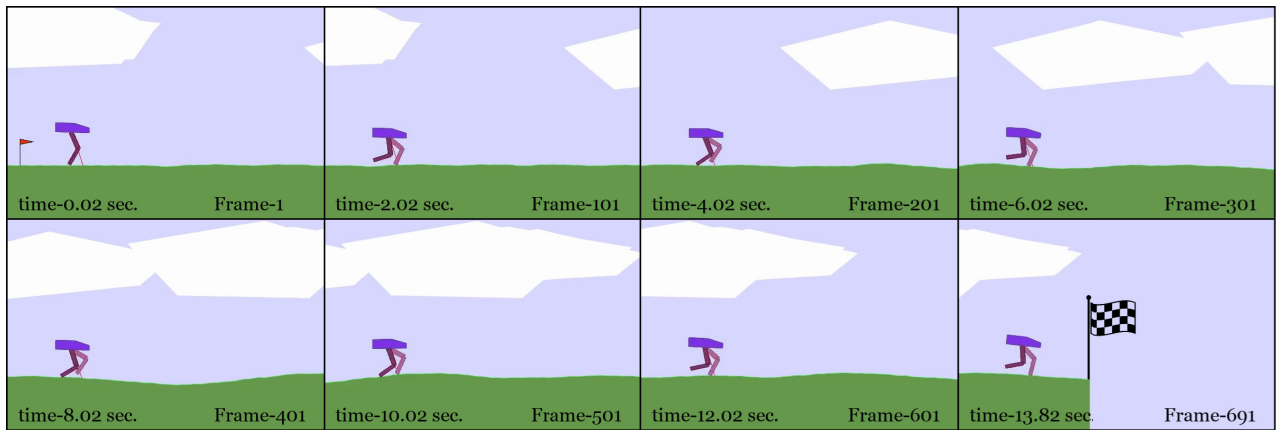


FIGURE 7. SAC performance for bipedal walker v3 (Frames 1-691 from Video File). Video link: <https://youtu.be/R2gQGgf14WQ>.

V. FINDINGS AND FUTURE RESEARCH DIRECTIONS

The video capturing the best results achieved by SAC was recorded, resulting in 691 frames. Using the SAC algorithm, the bipedal robot successfully completed the course from the starting point to the finishing point in approximately 14 seconds. Each frame was labeled with both the frame number and the corresponding time in seconds. Several images from this study can be seen in Figure 7. Also, the video is available in link: <https://youtu.be/R2gQGgf14WQ>.

The terrain used in this study consists of the standard environment prepared for the GYM, featuring slightly uneven terrain. However, when opting for a Hardcore environment, characterized by the presence of ladders, stumps, and pitfalls, the robot encounters difficulties and may fail. Therefore, if the robot is intended to navigate such challenging terrain, the procedures outlined in the study must be repeated with the Hardcore parameter set to True. The hardcore version of the terrain is shown in Fig. 8.

The insights presented in this paper extend their relevance to a broader spectrum of walking algorithms. To enable real-world application, the creation of life-size robot drawings and the implementation of all outlined processes in a simulation environment are paramount. Training the RL model with a simulated robot that mirrors

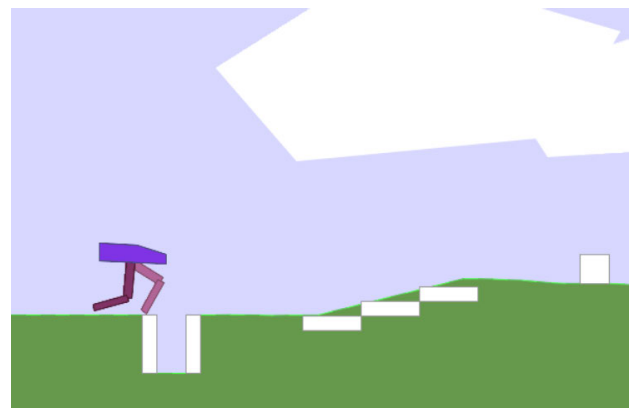


FIGURE 8. Hardcore environment with ladders, stumps, pitfalls.

real-world characteristics enhances the potential for effective real-time control applications. Additionally, assessing the viability of deploying bipedal locomotion strategies in challenging environments, featuring unpredictable terrain and dynamic obstacles, highlights the adaptability of the proposed approach. In the context of real-time implementation, synchronizing the sampling time between the simulation and the real-time controller is crucial, amplifying the utility of the controller developed in simulation within the real-time system. These considerations underscore the

foundational role of this work in real-time applications, paving the way for practical implementations across diverse contexts.

Exploring various robot structures, including quadrupeds and hexapods, goes beyond the bipedal walker, offering insights into the generalizability of the optimization methodology. Further, investigating real-world challenges like uneven terrain and obstacles enhances the practicality of the proposed RL optimization. Transfer learning, multi-agent systems, adversarial training, and online learning mechanisms provide avenues for understanding collaborative interactions, robustness, and adaptability. Additionally, studying sensor variability addresses how fine-tuned RL algorithms respond to uncertainties. These future research directions aim to emphasize current findings' significance and pave the way for a comprehensive understanding of RL-based locomotion in diverse robotic systems and environments.

VI. CONCLUSION

This study presented a comparative approach on RL-based algorithms used in the walking control of two-legged robots. Popular algorithms such as PPO, A2C, SAC, and TD3 RL algorithms were trained on the GYM bipedal walker robot and its environment. During training, parameters such as learning rate, discount factor, generalized advantage estimation, entropy coefficient, and Polyak update parameters were optimized with TPE optimization to achieve the best training results. This way, an attempt was made to obtain the highest possible performance from RL algorithms. Additionally, the importance of these optimized hyperparameters for each algorithm was analyzed. Ultimately, the discount factor was considered the most important parameter. Furthermore, the optimization distributions of learning rate and discount factor parameters were shown on the objective function, indicating where the clusters were mostly located. Subsequently, using the obtained optimal hyperparameters, all RL algorithms were subjected to 10 tests each, and the reward values from each test were compared. As a result of the study, the SAC-based RL algorithm was found to be the most suitable solution, achieving both high stability and the highest score. SAC managed to obtain a score of approximately 320 in all tests, while TD3 came closest with a score of around 280. PPO, on the other hand, exceeded a score of 300 in some tests but could not consistently achieve this value in every test, indicating insufficient stability, with an average score of 181. A2C performed poorly in all tests. In conclusion, it can be said that the SAC-based RL algorithm is the most suitable for bipedal robots.

REFERENCES

- [1] G. Boztaş and Ö. Aydoğmuş, "Implementation of pure pursuit algorithm for nonholonomic mobile robot using robot operating system," *Balkan J. Electr. Comput. Eng.*, vol. 9, no. 4, pp. 337–341, Oct. 2021.
- [2] M. Bjelonic, V. Klemm, J. Lee, and M. Hutter, "A survey of wheeled-legged robots," in *Proc. Climbing Walking Robots Conf.* Cham, Switzerland: Springer, 2022, pp. 83–94.
- [3] J. Carpentier and P.-B. Wieber, "Recent progress in legged robots locomotion control," *Current Robot. Rep.*, vol. 2, no. 3, pp. 231–238, Sep. 2021.
- [4] P. M. Wensing, M. Posa, Y. Hu, A. Escande, N. Mansard, and A. D. Prete, "Optimization-based control for dynamic legged robots," *IEEE Trans. Robot.*, 2023.
- [5] R. K. Mandava and P. R. Vundavilli, "An optimal PID controller for a biped robot walking on flat terrain using MCIWO algorithms," *Evol. Intell.*, vol. 12, no. 1, pp. 33–48, Mar. 2019.
- [6] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 2811–2817.
- [7] G. Satheesh Kumar, S. Aravind, R. Subramanian, S. K. Bharadwaj, R. R. Muthuraman, R. S. Mitchell, A. Bucha, M. Sriram, K. J. S. Hari, and N. J. Robin, "Literature survey on four-legged robots," in *Proc. ICMechD*, 2021, pp. 691–702.
- [8] K. Yamamoto, T. Kamioka, and T. Sugihara, "Survey on model-based biped motion control for humanoid robots," *Adv. Robot.*, vol. 34, nos. 21–22, pp. 1353–1369, Nov. 2020.
- [9] G. H. Z. Liu, M. Z. Q. Chen, and Y. Chen, "When joggers meet robots: The past, present, and future of research on humanoid robots," *Bio-Des. Manuf.*, vol. 2, no. 2, pp. 108–118, Jun. 2019.
- [10] Y. Xie, B. Lou, A. Xie, and D. Zhang, "A review: Robust locomotion for biped humanoid robots," *J. Phys., Conf. Ser.*, vol. 1487, no. 1, Mar. 2020, Art. no. 012048.
- [11] T. Mikołajczyk, E. Mikołajewska, H. F. N. Al-Shuka, T. Malinowski, A. Kłodowski, D. Y. Pimenov, T. Paczkowski, F. Hu, K. Giasin, D. Mikołajewski, and M. Macko, "Recent advances in bipedal walking robots: Review of gait, drive, sensors and control systems," *Sensors*, vol. 22, no. 12, p. 4440, Jun. 2022.
- [12] E. Daneshmand, M. Khadiv, F. Grimminger, and L. Righetti, "Variable horizon MPC with swing foot dynamics for bipedal walking control," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 2349–2356, Apr. 2021.
- [13] J. Arcos-Legarda, J. Cortes-Romero, and A. Tovar, "Robust compound control of dynamic bipedal robots," *Mechatronics*, vol. 59, pp. 154–167, May 2019.
- [14] H. Y. Park, J. H. Kim, and K. Yamamoto, "A new stability framework for trajectory tracking control of biped walking robots," *IEEE Trans. Ind. Informat.*, vol. 18, no. 10, pp. 6767–6777, Oct. 2022.
- [15] B. Gurevin, F. Gulturk, M. Yildiz, I. Pehlivan, T. T. Nguyen, F. Caliskan, B. Boru, and M. Z. Yildiz, "A novel GUI design for comparison of ROS-based mobile robot local planners," *IEEE Access*, vol. 11, pp. 125738–125748, 2023.
- [16] P. Wang, H. Mutahira, J. Kim, and M. S. Muhammad, "ABA—Adaptive bidirectional A algorithm for aerial robot path planning," *IEEE Access*, vol. 11, pp. 103521–103529, 2023.
- [17] Z. Xu, F. M. Zegers, N. Baharisangari, B. Wu, A. J. Phillips, W. E. Dixon, and U. Topcu, "Controller synthesis for multi-agent systems with intermittent communication and metric temporal logic specifications," *IEEE Access*, vol. 11, pp. 91324–91335, 2023.
- [18] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robot. Autom. Lett.*, vol. 8, no. 6, pp. 3740–3747, Jun. 2023.
- [19] S. Koseki, K. Kutsuzawa, D. Owaki, and M. Hayashibe, "Multimodal bipedal locomotion generation with passive dynamics via deep reinforcement learning," *Frontiers Neurobot.*, vol. 16, pp. 1–11, Jan. 2023.
- [20] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Reinforcement learning-based cascade motion policy design for robust 3D bipedal locomotion," *IEEE Access*, vol. 10, pp. 20135–20148, 2022.
- [21] J. Dibachi and J. Azoulay, "Teaching a robot to walk using reinforcement learning," 2021, *arXiv:2112.07031*.

- [22] R. Eliya and J. M. Herrmann, "Evolutionary selective imitation: Interpretable agents by imitation learning without a demonstrator," 2020, *arXiv:2009.08403*.
- [23] M. C. Bingol, "Evaluation of DDPG and PPO algorithms for bipedal robot control," *Osmaniye Korkut Ata Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, vol. 5, no. 2, pp. 783–791, 2022.
- [24] J. S. Ide, D. Mićović, M. J. Guarino, K. Alcedo, D. Rosenbluth, and A. P. Pope, "Soft actor-critic with inhibitory networks for faster retraining," 2022, *arXiv:2202.02918*.
- [25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
- [26] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
- [27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
- [28] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
- [29] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2623–2631.
- [30] M. Massaoudi, H. Abu-Rub, S. S. Refaat, M. Trabelsi, I. Chihhi, and F. S. Oueslati, "Enhanced deep belief network based on ensemble learning and tree-structured of Parzen estimators: An optimal photovoltaic power forecasting method," *IEEE Access*, vol. 9, pp. 150330–150344, 2021.
- [31] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 24, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2011.
- [32] G. Rong, K. Li, Y. Su, Z. Tong, X. Liu, J. Zhang, Y. Zhang, and T. Li, "Comparison of tree-structured Parzen estimator optimization in three typical neural network models for landslide susceptibility assessment," *Remote Sens.*, vol. 13, no. 22, p. 4694, Nov. 2021.



OMUR AYDOGMUS received the M.Eng. and Ph.D. degrees in electrical and electronics engineering from Firat University, Elâzığ, Turkey. He is currently a Professor with the Faculty of Technology, Firat University, specializing in mechatronics engineering. His research interests include collaborative robots, the integration of artificial intelligence in robotics, electric motor drive systems, and industrial automation applications. He has published numerous papers in peer-reviewed journals and conferences. He has also received several grants for his research. He is also a highly respected researcher in the field of mechatronics engineering. His work has made contributions to the development of new and innovative technologies in robotics and industrial automation.



MUSA YILMAZ (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in electrical education from Marmara University, İstanbul, Turkey, in 2004 and 2013, respectively. From 2015 to 2016, he joined the Smart Grid Energy Research Center (SMERC), University of California at Los Angeles (UCLA), Los Angeles, as a Visiting Scholar. He is currently an Assistant Professor with the Department of Electrical and Electronics Engineering and the Department of Energy Engineering, Batman University, Batman. He has also led his research team as a principal investigator in several European projects. He has authored more than 50 research articles, several book chapters, and frequently delivers invited keynote lectures at international conferences. His primary research interests include smart grid technologies and renewable energy. He has conducted extensive research in the field of smart grid and solar energy and along with Biosys LLC, he is an inventor of a ventilator class named "Biyovent." He has made significant contributions to academia and publishing. He has served as the Editor-in-Chief for the *Balkan Journal of Electrical and Computer Engineering* (BAJECE) and the *European Journal of Technique* (EJT). In addition, he is also the Co-Founder of INESEG, a publishing organization.

• • •