electronics

MDPI

Article

Automated Hyperparameter Tuning in Reinforcement Learning for Quadrupedal Robot Locomotion

MyeongSeop Kim [1], Jung-Su Kim [2] and Jae-Han Park [1,*]

[1] Applied Robot R&D Department, Korea Institute of Industrial Technology (KITECH), Ansan 15588, Republic of Korea; calintz123@kitech.re.kr
[2] Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul 01811, Republic of Korea; jungsu@seoultech.ac.kr
* Correspondence: hans1024@kitech.re.kr

Abstract: In reinforcement learning, the reward function has a significant impact on the performance of the agent. However, determining the appropriate value of this reward function requires many attempts and trials. Although many automated reinforcement learning methods have been proposed to find an appropriate reward function, their proof is lacking in complex environments such as quadrupedal locomotion. In this paper, we propose a method to automatically tune the scale of the dominant reward functions in reinforcement learning of a quadrupedal robot. Reinforcement learning of the quadruped robot is very sensitive to the reward function, and recent outstanding research results have put a lot of effort into reward shaping. In this paper, we propose an automated reward shaping method that automatically adjusts the reward function scale appropriately. We select some dominant reward functions, arrange their weights in a certain unit, and then calculate their gait scores so that we can select the agent with the highest score. This gait score was defined to reflect the stable walking of the quadrupedal robot. Additionally, quadrupedal locomotion learning requires reward functions of different scales depending on the robot's size and shape. Therefore, we evaluate the performance of the proposed method on two different robots.

Keywords: reinforcement learning; quadrupedal robot; reward shaping; automated machine learning

check for updates

Citation: Kim, M.; Kim, J.-S.; Park, J.-H. Automated Hyperparameter Tuning in Reinforcement Learning for Quadrupedal Robot Locomotion. *Electronics* **2024**, *13*, 116. https://doi.org/10.3390/electronics13010116

Academic Editor: Dah-Jye Lee

Received: 28 November 2023
Revised: 18 December 2023
Accepted: 21 December 2023
Published: 27 December 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Reinforcement learning (RL)-based quadrupedal locomotion can robustly control the robot with the same model even in various terrains. A model trained using both terrain data and robot internal information can show a stable gait pattern regardless of rough terrain or flat terrain [1–3]. In general, reinforcement learning uses multiple reward functions, and these have a very critical impact on the performance of the robot agent. Reinforcement learning for quadrupedal robots also uses various reward functions and requires a lot of effort to tune them. In addition, some reward functions have a particularly sensitive effect on performance and gait pattern, and stable gait control can be achieved by finding the optimal weight values of these reward functions depending on the robot model. Finding this optimal weight requires many attempts, and training a new model requires a new optimal weight. Similar problems exist in general machine learning tasks. Hyperparameters related to machine learning, such as model depth, shape, and number of parameters, have a significant impact on learning but require a lot of effort to find appropriate values. Accordingly, the need for research in the field of automatic machine learning (Auto-ML), which can automatically solve this problem, has recently emerged [4,5]. In this paper, we attempt to solve the task of finding the optimal weight of the reward function in the previously mentioned reinforcement learning problem of a quadrupedal robot from the perspective of Auto-ML. We select some dominant reward functions and define a score function that can evaluate whether the weights of these reward functions are

well set. The proposed method uses this function to select the model that records a high score among the agents where different several weights are used. Additionally, since the ratio of optimal weights is different for each robot model, two different quadruped robots are tested in the evaluation of the proposed method.

2. Related Works

RL-based quadrupedal locomotion can achieve high performance regardless of terrain by utilizing various algorithms. Previously, many model-based methods were adopted that generate footholds that guide the locomotion trajectory or combine them with optimization methods [6,7]. However, recently, a method has been used in which the desired position is output from the model in every feedback loop and the resulting torque is determined through a PD controller [1,3,8]. In addition, the sim-to-real method is being used by utilizing the restoration algorithm of exteroceptive sensing so that the robot learned in simulation can actually operate well in various terrains in the real world [2,9]. In this paper, a reinforcement learning method similar to [1] is adopted and used to verify the algorithm. In these RL-based gait learning methods, a wide variety of reward functions are used. The number of different reward functions requires multiple weights for each reward value, and their importance is set differently. In general, it is difficult to compare importance before the training results, so the sum of each reward function output is set to be within -1 to 1 [10]. Afterward, reward weights are adjusted appropriately according to the task. In this paper, we propose a method to find a reward weight suitable for each robot model based on the weight values of [1].

Finding a reward weight that can improve performance in reinforcement learning is called reward shaping, and various methods of reward shaping can be used depending on the task. Reward shaping requires expert knowledge regardless of the field, and requires a lot of effort to bring out the optimal agent performance. Therefore, the need for automation or a generalizable method of reward shaping was raised, and this is also one of the fields of Auto-ML. Research on methods to automate the learning process of reinforcement learning to make it faster and reduce manual work is called Automated Reinforcement Learning (Auto-RL). Such Auto-RL includes learning to learn for convergence of model parameters, neural architecture search (NAS) to find the optimal structure of the model, automated reward function to find an appropriate reward function, and reward shaping [11–14]. Among these, learning to learn and NAS are common research topics in Auto-ML and are generalizable methods, but they are insufficient to dramatically change the gait performance of quadrupedal robots. Therefore, this paper focuses on automated reward functions and reward-shaping methods. Among automated reward function methods, curriculum learning is a method that can be generalized. This was also used in [1,2,9]. However, curriculum learning does not guarantee performance on various robot models. Therefore, we plan to use a method that combines curriculum learning and automated reward shaping. Among automated reward shaping methods, a reward machine is a potential-based reward shaping method that can quickly explore the optimal policy without harming the optimal policy of the original Markov Decision Process (MDP) [15]. However, in complex and unexpected environments such as quadrupedal robots, it is difficult to bring about significant performance changes just by increasing exploration. For example, in the case of [1], experience is gained from more than 4000 robot models at the same time, but it is difficult to view simply a large number of agents as the main factor in obtaining a highly stable gait pattern. This can be seen as the comprehensive result of various terrain environments, multiple reward functions and their weights, and various hyper-parameters of the algorithm. This is because various gait patterns appear due to slight changes in the reward function.

Traditionally, the design of the reward function or reward shaping in reinforcement learning was one of the major factors that made it difficult to guarantee the performance of reinforcement learning. Accordingly, many studies have been proposed to reduce reward design efforts while ensuring the most stable performance improvement. One of the big problems in reinforcement learning related to reward is the sparse reward problem, and

there are studies that have solved this problem through reward shaping based on intrinsic reward [16,17]. The sparse reward problem often appears in robot grasping, and these studies showed reward shaping based on tactile force or guide demonstration. However, the sparse reward problem does not appear significantly in the reinforcement learning of walking robots. Therefore, we do not address intrinsic motivation but focus on reward shaping for gait stability according to the robot model. In another direction, reward-shaping studies for stable convergence of agent performance have also been actively proposed. In [18], reward shaping based on reward variance reduced the agent's overestimation and led to a more stable convergence of learning. In another direction, reward-shaping studies for stable convergence of agent performance have also been actively proposed [19]. On the other hand, this paper aims to control the ratio of conflicting reward functions in the learning of a quadrupedal robot and focuses on the stability of gait rather than improving the sum of reward. Additionally, if this ratio is not appropriate, a situation may arise where learning does not proceed at all, so this can be effectively resolved. Sometimes, reward shaping is not easy to generalize, and research is generally focused on methods for specific tasks. In a specific task, the value of the reward can vary depending on the current state or results of the action. At this time, the value of the reward can be determined by defining an evaluation function for the task, or it can be defined to provide intrinsic motivation [20,21]. In this paper, similarly to these, we define a score function for the walking stability of a quadrupedal robot and construct an algorithm to deduce the ratio of reward weights that highly estimate this score.

3. Reinforcement Learning Based Quadrupedal Locomotion

Reinforcement learning trains an agent to maximize the sum of rewards up to the terminal time in the environment structured by Markov Decision Process (MDP) [22]. The agent receives the current state s_t as input at time step t and outputs the action a_t. MDP's model presents the next state s_{t+1} and reward r_{t+1} through s_t and a_t. The goal of reinforcement learning is to learn a policy π that maximizes the sum of rewards $\sum_t^T \gamma^{t-1} \cdot r_t$ up to the last time step T. Here, γ is the discount factor where $\gamma = 0.98$.

A state s_t is an input of a controller and consists of numerical information obtained from the environment. The state information of the quadrupedal robot includes exteroceptive sensing and proprioceptive sensing information. In general, proprioceptive sensing information includes the robot's base coordinates, speed, attitude, and joint information of each leg and foot force. Exteroceptive sensing information includes information about the external environment that can be obtained through a camera or lidar mounted on the robot. This is generally part of the elevation map recognized through sensors as topographical information around the robot. In conclusion, state s_t includes joint values of each leg $q_t \in \mathbb{R}^{12}$, joint velocity $q'_t \in \mathbb{R}^{12}$, robot base and gravity information $b_t \in \mathbb{R}^9$, current action $a_t \in \mathbb{R}^{12}$, current command $c_t \in \mathbb{R}^3$ foot contact information $f_t \in \mathbb{R}^4$, and terrain information $M_t \in \mathbb{R}^{m \times n}$, where $s_t = (M_t, q_t, q'_t, b_t, a_t, f_t, c_t)$. Here, m, n are the grid size of topographical information. Robot base information b_t consists of position $b_{t,\text{pos}} \in \mathbb{R}^3$, angle $b_{t,\text{ang}} \in \mathbb{R}^3$, and gravity $b_{t,\text{gravity}} \in \mathbb{R}^3$. In addition, joint velocity q'_t, acceleration q''_t and joint torque τ_t can be also used for state information or reward functions. The action a_t is an output of the policy $\pi(\cdot|s_t)$. This policy is a neural network modeling a controller. The action a_t is transmitted to the PD controller without directly controlling the robot. PD gain values K_p and K_d are fixed and we only change the scale of action when the robot model is changed. The relationship between various state information, actions, and rewards in MDP is shown in Figure 1.

The reward function of a quadrupedal robot is generally designed to safely perform the desired gait using a small amount of torque. In this paper, we design the reward functions by referring to [1,2], and observe the change according to the weight adjustment. The quadruped robot agent is given the desired velocity v^d and angle θ^d where $c_t = \{v^d, \theta^d\} \in \mathbb{R}^3$. At this time, the reward function is designed as the difference between the current speed and angle of the robot and the desired values. We also add a negative

reward to limit the torque used by the robot while maintaining the desired speed for the agent. Putting these together, the reward function is defined as:

$$r_t = w_1 \exp(-\frac{1}{4}(v^d - b'_{t,\text{pos}})^2) - 10^{w_2} \cdot \tau_t + w_3 \exp(-\frac{1}{4}(\theta^d - b_{t,\text{ang}})^2) - w_4 q''_t \quad (1)$$

In Equation (1), w_1, w_2, w_3 and w_4 are the weights of each reward function. At this time, the desired speed and torque penalty reward functions are contrasted with each other. This means that more torque must be used to achieve higher speeds, but the available torque is limited. This paper deals with the optimal ratio of the two reward functions: torque and desired velocity. Several other reward functions also have an effect on learning, but finding the optimal ratio for all of them requires a lot of computing costs. Additionally, consideration of torque and desired velocity is also widely used in optimization-based methods, and they have the most decisive influence on gait control. Therefore, the method proposed in this paper fixes the weight values other than the two reward function's weights and presents a score function that can find an appropriate ratio between them. Basically, each reward weight (w_1, w_2, w_3, w_4) is initialized so that the reward sum is between -1 and 1. Afterward, the performance of w_1 and w_2 is evaluated according to the score function. The definition of this function is explained in detail in Section 4.

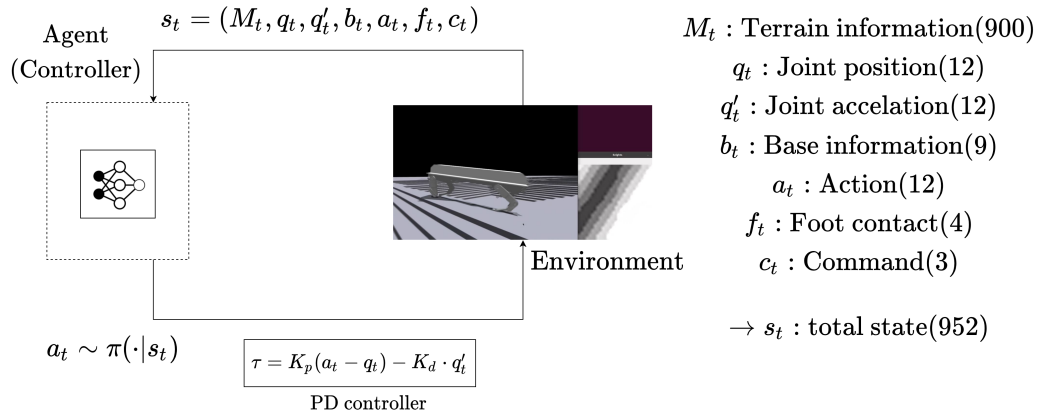


Figure 1. MDP environment configuration diagram of reinforcement learning-based quadrupedal locomotion control. Agent refers to a learned controlled model and receives state information as input and outputs a joint target. The joint target is transmitted to the PD controller to transmit torque to the robot model.

The control model is trained based on PPO (Poximal Policy Optimization) [23]. PPO is an on-policy algorithm and has an actor-critic structure. PPO prepares neural network parameters ϕ and ψ to estimate value and policy, respectively. The value network parameters for value estimation are updated as follows.

$$\psi \leftarrow \psi - \eta \nabla_\psi \mathbb{E}[\frac{1}{2}(r_{t+1} + \gamma_1 V_\psi(s_{t+1}) - V_\psi(s_t)]^2 \quad (2)$$

The above equation uses the temporal difference error of the value function to estimate the sum of rewards that the agent receives during one episode. The sum of rewards during one episode is called return and is defined as $G_t = r_t + r_{t+1} + \cdots + r_T$. The value function estimates the average of these returns, $\mathbb{E}[G_t]$. That is, $V^* \simeq \mathbb{E}[G_t]$, and V^* is the optimal value function. The policy π_ϕ that controls the robot learns to find parameters that can maximize this value. Equation (2) is learned using gradient descent, but ϕ is learned using gradient ascent to maximize the reward sum. The equation below updates the parameters of the policy network.

$$\phi \leftarrow \phi + \eta \nabla_\phi \mathbb{E}_{s_t, a_t}[min(\rho_t(\phi) A_t, clip(\rho_t(\phi), 1 - \epsilon, 1 + \epsilon) A_t)] \quad (3)$$

Here, ρ is a conservative function where $\rho_t(\phi) = \frac{\pi_\phi(a_t|s_t)}{\pi_{\phi,\text{old}}(a_t|s_t)}$. And $clip$ prevents the conservative function from playing too excessive a role and $\epsilon = 0.1$. Through this equation, the policy model learns to find the action that maximizes the advantage function A_t. The advantage function is used to lower the variance of the value function and is defined as $A_t = G_t - V_\psi(s_t)$. Additionally, the on-policy-based actor-critic algorithm can simultaneously collect data from multiple agents in a multi-threaded manner. When using n agents, we expressed as $s_t, a_t, r_t \rightarrow s_t^n, a_t^n, r_t^n$. The structure and process of the entire algorithm are expressed in Algorithm 1 and Figure 2.

Algorithm 1 PPO based Gait Learning with Automatic Reward Shaping

Initialize the network parameters : $\phi_1, \cdots, \phi_{j\times k}$, $\psi_1, \cdots, \psi_{j\times k}$
Prepare a grid table of w_1 and w_2
while Training Terminal **do**
  while Training L epoch **do**
    Obtain data from the simulation and store them in memory M
    $M \leftarrow \{s_t^1, a_t^1, s_{t+1}^1, r_{t+1}^1, \cdots, s_t^n, a_t^n, s_{t+1}^n, r_{t+1}^n\}$

    Estimate A_t
    $G_t = r_{t+1} + \cdots + \gamma^{T-2-t} \cdot r_{T-1} + \gamma^{T-1-t} \cdot V_\psi(s_T)$
    $A_t = G_t - V_\psi(s_t)$

    train policy parameter ϕ_l $(l = 1, \cdots, j \times k)$
    $\phi_l \leftarrow \phi_l + \nabla_{\phi_l} \mathbb{E}[\min(\rho_t(\phi_l) \cdot A_t,\ clip(\rho_t(\phi_l), 1-\epsilon, 1+\epsilon) \cdot A_t)]$

    train value parameter ψ_l $(l = 1, \cdots, j \times k)$
    $\psi_l \leftarrow \psi_l - \eta \nabla_{\psi_l} \mathbb{E}[\frac{1}{2}(r_{t+1} + \gamma_1 V_{\psi_l}(s_{t+1}) - V_{\psi_l}(s_t)]^2$

  end while
  Estimate gait score in case of every w_1, w_2
  Obtain N samples, where $N = T \times \frac{n}{j\times k}$
  (T is the length of the last episode and $\frac{n}{j\times k}$ is the number of agents for each weight combination).
  $J_{l,\textbf{score}} = \frac{1}{\frac{n}{j\times k}} \sum^{\frac{n}{j\times k}} [\frac{1}{T} \sum_{t=1}^{T} (-\tau_t - \exp(\frac{1}{4}(v^d - b'_{t,\text{pos}})^2) - b_{t,gravity}) + \sum_{t=1}^{T} (b_{t,pos} - b_{t-1,pos})^2]$
  pick the model ϕ and ψ with a high score.
end while

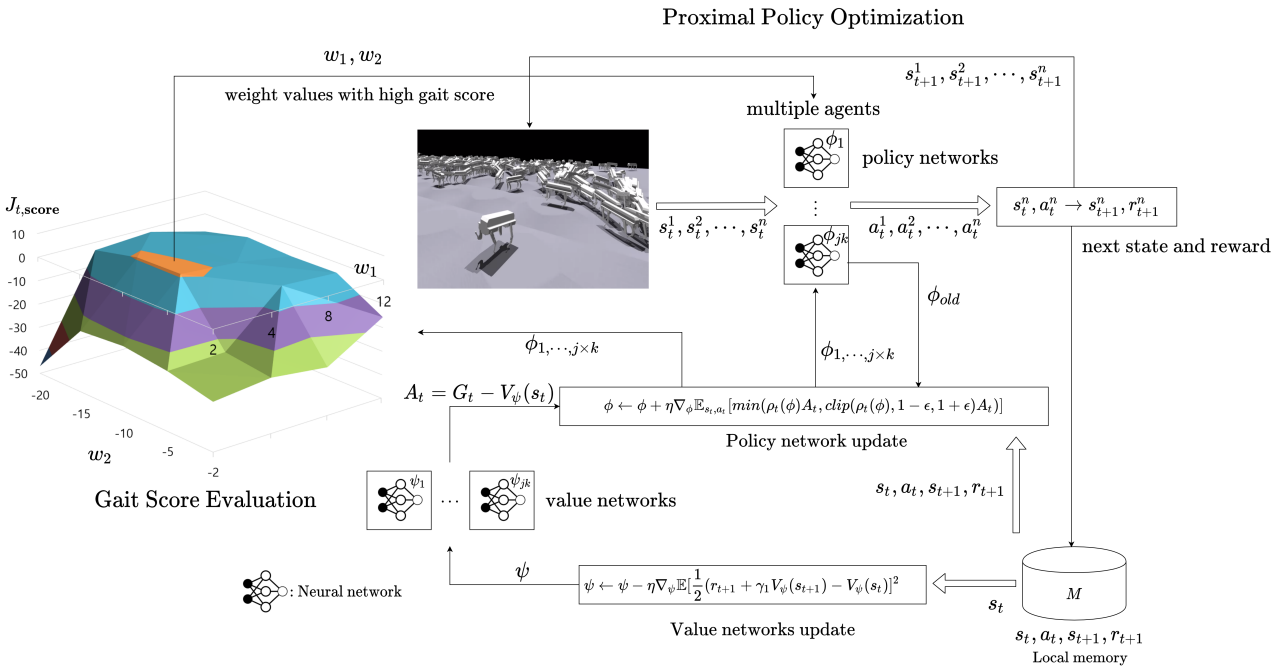


Figure 2. Overall algorithm structure combining PPO-based gait learning and automatic reward shaping method.

4. Automatic Reward Shaping for Stable Locomotion

In order to evaluate the currently determined weight, a numerical indicator is needed to check whether the gait of the trained quadruped robot is stable. In general, the goal of torque control for a quadrupedal robot is to reach close to the target velocity with less torque. Using excessive torque or making large leg movements is inefficient, unstable, and can result in unwanted gait patterns. Therefore, we determine a function that can appropriately adjust the weight of velocity and torque. The Equation (1) alone is not suitable for determining whether the model's gait pattern is stable. Firstly, this function is directly proportional to the currently determined weights and cannot distinguish between good and bad, and secondly, it does not include the shaking or tilt of the robot base. Therefore, we define the gait score function or score function as follows.

$$J_{l,\textbf{score}} = \frac{1}{T}\sum_{t=1}^{T}(-\tau_t - \exp(\frac{1}{4}(v^d - b'_{t,\text{pos}})^2) - b_{t,gravity}) + \sum_{t=1}^{T}(b_{t,pos} - b_{t-1,pos})^2 \quad (4)$$

The Equation (4) contains expressions for torque and desired velocity, as well as changes in base position and projection gravity currently applied to the robot. The closer the projection gravity applied to the robot is to 0, the more upright the robot is, and the change in base position is a value for the current robot speed itself. The change in base position indicates how fast the robot moves, and the higher it is, the larger the score function becomes, but there is a trade-off with the desired velocity. The score function defined that the higher the value, the faster and more stable the gait pattern is. In other words, RL-based locomotion using the current weights must maintain as fast a speed as possible while remaining close to the desired velocity, use as little torque as possible, and prevent the robot's center of gravity from collapsing as much as possible.

A better weight combination can be found based on the score function defined previously. Since the combination space of w_1, w_2 is two-dimensional, we will evaluate multiple weight combinations simultaneously within several grid combinations using a grid search method. In order to evaluate the weight values according to gait score, a network of as

many as the number of weight values is needed. When there are j w_1 and k w_2, the policy and value networks each require $j \times k$. Therefore, we prepare $j \times k$ models in advance and train this number each time. The number of agents is n, and since this number is more than $j \times k$, $\frac{n}{j \times k}$ is distributed for each weight combination. In Equation (4), T means the length of one episode. The gait score takes the average value of all samples of the last episode after learning of a set length L epoch is completed. This average is obtained based on the number of agents per weight combination, $\frac{n}{j \times k}$. The entire algorithm process can be checked through the Algorithm 1 and Figure 2.

The Table 1 displays information about each robot and the parameters used when learning the robots. In the table, the action scale represents the range of a_t. The heavy quadrupedal robot requires more torque than Anymal, so larger values were used. Additionally, the range of each weight value is different, and in the case of heavy robots, a wider range is explored. This is because heavier robots can use more torque, and therefore, may require greater constraints. However, in the case of the number of weight combinations, there are fewer cases of heavy robots, which is because the learning results are not very sensitive to small weight changes, so the area is set wide. Section 5 shows and explains the gait score pattern based on these parameters. Figure 3 shows the size of the neural network used in learning the quadrupedal robot in this paper. Each neural network consists of MLP only, and the size of the input and output is the same even if the robot changes. The size of the state information input to the two neural networks is the same at 952, the value network outputs a scalar value and the policy network outputs an action vector of size 12.

Table 1. Each robot's information and parameters for training the robots.

	Anymal	**Heavy Quadrupedal Robot**
total weight	25 kg	260 kg
leg weight	4.39 kg	15.47 kg
leg length	0.6 m	1.1 m
total size	$0.9 \times 0.5 \times 0.9\ \mathrm{m}^3$	$2.5 \times 0.9 \times 1.5\ \mathrm{m}^3$
action scale (a_t)	-0.5 to 0.5	-2.0 to 2.0
damping K_d	85.0	105.0
stiffness K_p	2.0	2.0
number of agents (n)	4088	4096
number of w_1 (j)	4	4
number of w_2 (k)	7	4
agents for each weight combination ($\frac{n}{j \times k}$)	146	256
maximum and minimum w_1	2, 12	2, 12
maximum and minimum w_2	-10 , -2	-20 , -2
	(1.00×10^{-10}), (1.00×10^{-2})	(1.00×10^{-20}), (1.00×10^{-2})
range of v^d	0.0 to 2.0	0.0 to 2.0
range of θ^d	-3.14 to 3.14	-3.14 to 3.14
maximum training epochs L	3000	3000

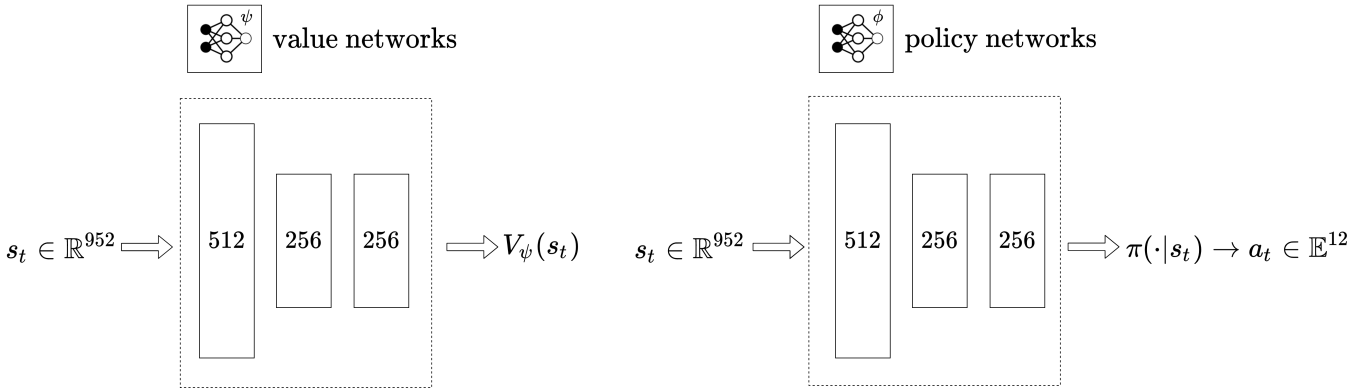


Figure 3. Structure of value network and policy network. Each neural network consists of MLP (Multi Layer Perceptron), and the policy network outputs joint targets to move the robot's legs.

5. Experiments and Evaluation

The verification of the proposed algorithm is performed on two robots. One is an Anymal robot [24] and the other is a large, heavy robot designed by us. Anymal has a light weight and a leg structure suitable for various terrains like Figure 4. On the other hand, a heavy quadrupedal robot weighs more than 200 kg and has a large body of more than 2 m in length and 1 m in height (check the Figure 5 and Table 1). Referring to Table 1, heavy robots have a taller and heavier body. Accordingly, the gain values K_d and K_p of the PI controller, and the action scale representing the range of a_t were adjusted. For the heavy robot, more torque is required and stable locomotion on complex terrain is more difficult. Considering these aspects, the torque grid of the heavy quadrupedal robot was designed to cover a wider area.

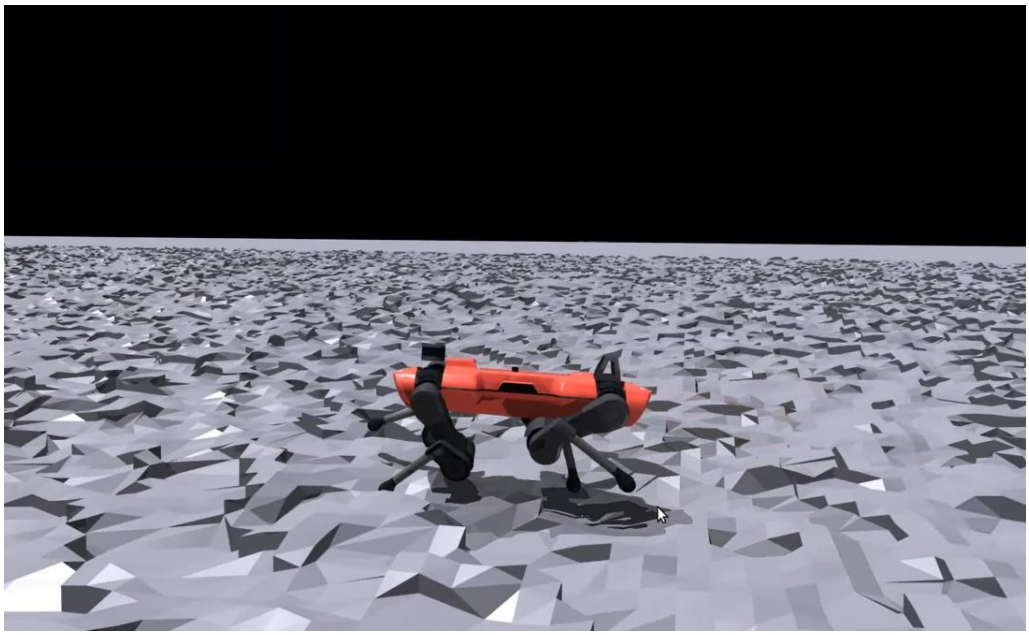


Figure 4. Anymal quadrupedal robot model in Isaac gym environment.

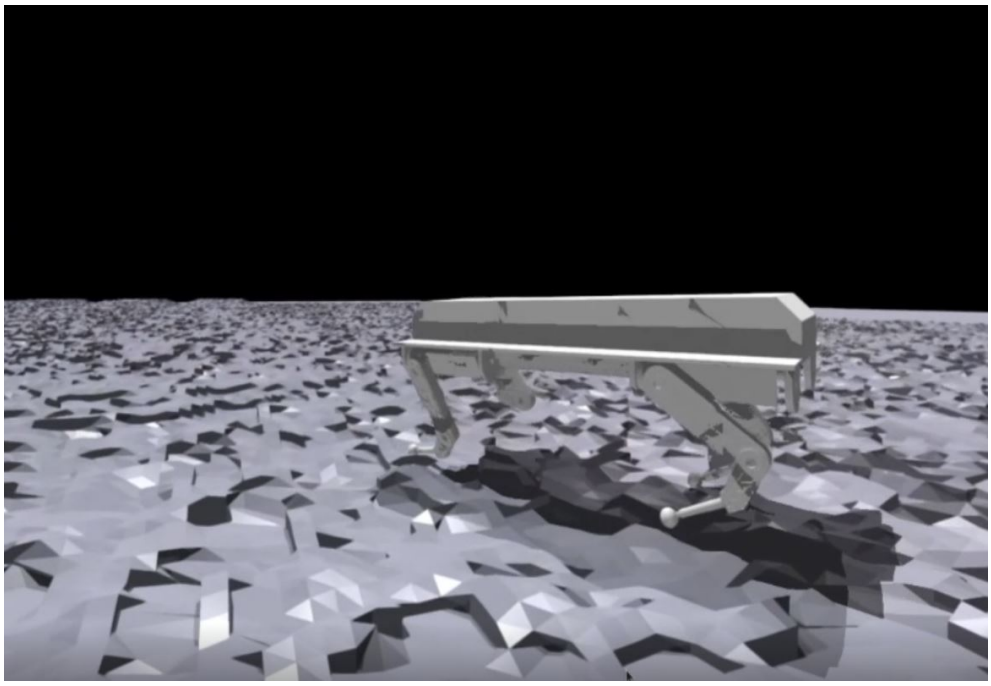


Figure 5. Self-designed heavy quadrupedal robot model in Isaac gym environment. With a weight of over 200 kg, more torque is required for locomotion.

We compare gait scores on two different terrains. One is a very uneven terrain (rough terrain) and the other is a terrain with many stairs (stair terrain). On rough terrain, the robot can easily fall, so it must locomote carefully, and on stair terrain, the robot must make large movements to cross high steps. Figure 6 shows two robots walking on rough terrain and stair terrain. On rough terrain, each quadruped robot tends to raise its legs high. It appears that the robot learned this on its own to avoid getting caught in the pitted terrain. In stair terrain, when the robot goes over the stairs, its body tilts according to the slope. Since the robot's posture and gait pattern appear differently depending on the terrain, there may be differences in the gait scores collected in the two cases. This is because the required torque and attitude are different depending on the terrain, resulting in differences in the values of each term in the Equation (4). In this section, we will also see how the gait score varies depending on the type of terrain.

Figures 7 and 8 show the gait score values of the Anymal robot according to the reward weight settings through a table and 3D graph. The red value in the table indicates the highest value among the scores. In the case of Anymal, the gait score tends to gradually increase towards this red area, and since this part is located in the center of the grid, it can be seen as indicating the need to appropriately adjust the two weight values. This shows that reward shaping using gait scores is efficient in terms of learning stable locomotion. However, determining the range of the grid can be problematic. However, as previously explained, the reward sum was adjusted only to a value that maintained -1 to 1, and reward adjustment on a larger scale was not necessary. Now, if you look at the difference between the two figures, there is a slight difference in the maximum value of the gait score. However, the shape of the graph and the area comprised by the peaks show a very similar form. We trained this agent for up to 3000 epochs (L), and after obtaining these gait scores, we can refer to the two gait score tables to select appropriate reward weight values. Therefore, for an agent learning on various terrains, the gait score according to the terrain can be obtained and the peak values among them can be selected to determine the reward weight. In this way, the Automatic reward shaping algorithm can be used when learning by integrating various terrains

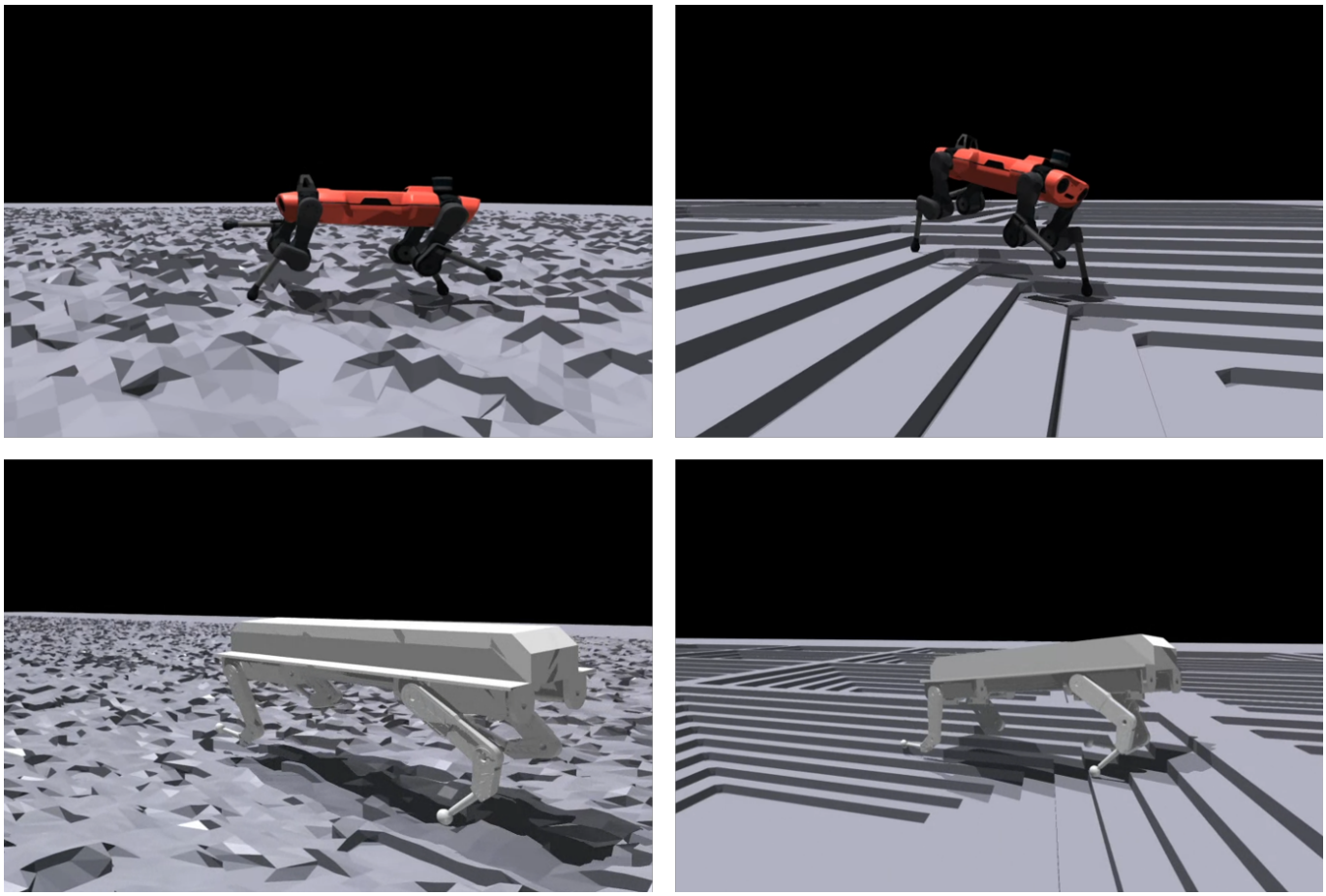


Figure 6. Gait score and automatic reward shaping test in different robots and different terrains. The top figures show Anymal's walking on rough terrain and stair terrain, and the bottom figures show the heavy quadrupedal robot walking on both terrains.

w_1

w_2	2	4	8	12
-10	0.6844	0.5052	0.6283	0.5617
-8	0.6329	0.7992	0.7176	0.5512
-6	0.8915	0.7132	0.9124	0.8812
-5	1.0905	1.1133	0.9038	0.8524
-4	0.8334	0.7242	1.0784	0.8308
-3	0.2362	0.5038	1.1595	1.2481
-2	-0.1164	0.2166	0.4501	0.5386

w_1: velocity weight, w_2: torque weight

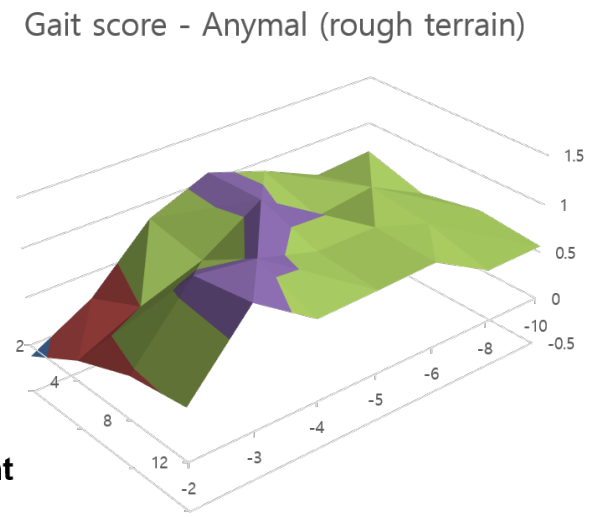


Figure 7. Collection of gait scores for reward shaping of training Anymal robot on rough terrain. The peak value of the gait score is indicated in red on the left table and they are matched with the purple part in the 3D graph.

w_1

w_2	2	4	8	12
-10	1.2303	0.8688	0.8865	1.1875
-8	1.2245	1.2763	1.0344	1.0032
-6	1.2617	1.2501	1.2076	1.1131
-5	1.3056	1.1103	1.2457	1.0125
-4	1.1547	1.3218	1.3207	0.9784
-3	1.1124	1.1528	1.2772	1.0544
-2	0.7619	0.7576	1.0739	1.1546

W_1: velocity weight, W_2: torque weight

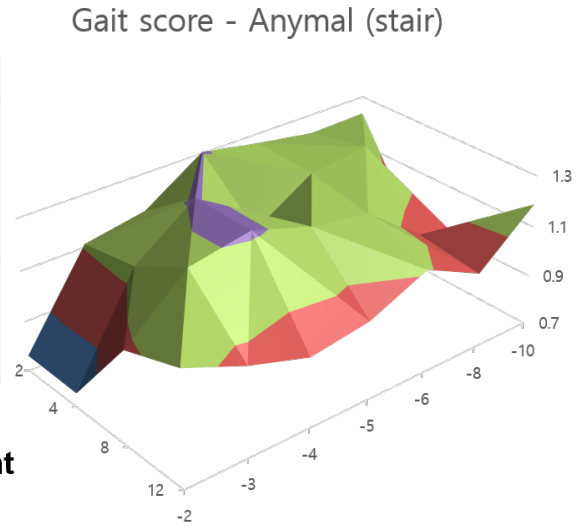


Figure 8. Gait score aspect of Anymal robot in stair terrain case. A similar pattern appears in the case of rough terrain, and in the case of peak values (marked in red and purple part in the 3D graph), there is an overlapping area in the two terrains.

Figures 9 and 10 show cases of heavy robots on each terrain. Somewhat contrary to Anymal, the 3D graph in the heavy quadrupedal robot takes the form of a cliff rather than a raised peak. A similar point is that here too, certain areas show high values, showing the necessity and effectiveness of reward weight adjustment. Another thing to note is that this robot shows very low performance if the velocity reward value is too low. This is because a heavy robot requires a lot of torque to walk, but if it focuses on using relatively less torque, it will show a learning pattern that prefers to stop in place. In particular, when $w_1 = 2$, the robot learned to stop in place in all cases. This shows that different ratios of reward weights depending on the robot model play a very important role in gait learning. Additionally, it requires a lot of effort to manually tune this, and the automatic reward shaping method shows that it is possible to find an optimal value in an appropriate range. As mentioned in the previous Anymal case, in the case of this robot, the agent can simultaneously learn various terrains and find an appropriate reward weight value through an automatic reward shaping method.

w_1

w_2	2	4	8	12
-20	-46.8768	-3.3822	-1.7791	-7.8304
-15	-23.9515	0.6516	-1.2241	-3.3322
-10	-22.4601	0.9225	-0.6941	-4.5373
-5	-24.9162	-24.1426	-3.1424	-4.3437
-2	-29.3376	-26.7701	-29.6405	-15.636

W_1: velocity weight, W_2: torque weight

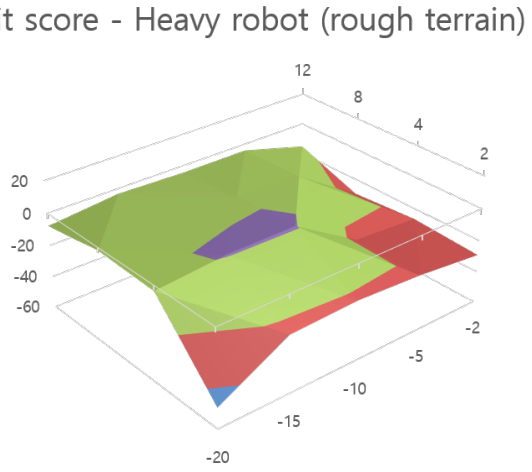


Figure 9. Gait score aspect of heavy robot in rough terrain case. It shows a cliff-shaped graph and the range of score values is also different. The peak value of the gait score is indicated in red on the left table and they are matched with the purple part in the 3D graph.

w_1

w_2	2	4	8	12
-20	-45.8343	-4.2414	-3.1384	-4.439
-15	-24.1316	-1.768	-2.9846	-3.6067
-10	-27.9178	1.0317	-4.496	-3.3114
-5	-25.7622	-23.8032	-3.4191	-3.7074
-2	-28.9157	-28.5221	-29.5954	-8.7838

w_1: velocity weight, w_2: torque weight

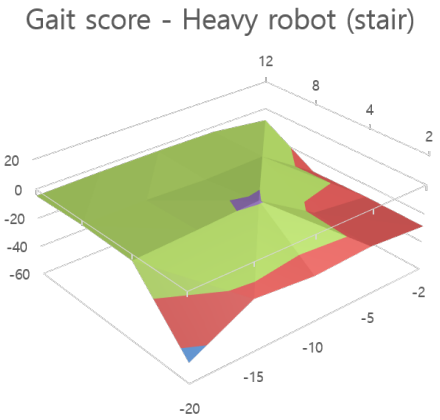


Figure 10. Gait score aspect of heavy robot in stair terrain case. It shows a graph similar to that of rough terrain, and there are overlapping points in the peak values (marked in red and purple part in the 3D graph). Even in the case of heavy robots, an overlap between rough terrain and stair terrain appears in the peak value.

6. Conclusions

In this paper, we presented a method to automatically tune the reward function regardless of the robot model when learning a quadrupedal robot using reinforcement learning. In general, reward shaping requires a lot of effort and know-how, but the automatic reward shaping method proposed in this paper allows us to find a reward weight combination close to stable locomotion with only a few range constraints. Additionally, it was shown that this method can be applied regardless of the robot model. If the computing cost is sufficient, the method can quickly proceed with reward shaping and reduce the incidental effort of reinforcement learning.

The method will be further developed in the future in terms of Auto-ML. The current method is somewhat inefficient in weight search. Searching in a wide grid can consume too much computing cost, so it may be efficient to apply a genetic algorithm based on the score function defined in the paper. Therefore, in future work, we will focus on considering the generalization of the algorithm and the efficiency of the search method. Additionally, since quadrupedal robots require verification in the real environment, we plan to check the gait controller once the actual platform is completed.

Author Contributions: Conceptualization, J.-H.P.; Methodology, M.K.; Software, M.K.; Formal analysis, J.-S.K.; Writing—original draft, M.K.; Writing—review & editing, J.-S.K. and J.-H.P.; Supervision, J.-S.K. and J.-H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Korea Research Institute for defense Technology planning and advancement (KRIT) and the grant number is 20-107-C00-007-03.

Data Availability Statement: The data presented in this study are openly available in [25], and https://developer.nvidia.com/isaac-gym.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RL	Reinforcement Learning
Auto-ML	Automatic Machine Learning
PPO	Proximal Policy Optimization
NAS	Network Architecture Search
MDP	Markov Decision Process
PD controller	Proportional Differential controller

References

1. Rudin, N.; Hoeller, D.; Reist, P.; Hutter, M. Learning to walk in minutes using massively parallel deep reinforcement learning. In Proceedings of the Conference on Robot Learning, Auckland, New Zealand, 14–18 December 2022.
2. Lee, J.; Hwangbo, J.; Wellhausen, L.; Koltun, V.; Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Sci. Robot. 2020*, **5**, eabc5986. [CrossRef] [PubMed]
3. Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, D.; Tsounis, V.; Koltun, V.; Hutter, M. Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **2019**, *4*, eaau5872. [CrossRef] [PubMed]
4. Parker-Holder, J.; Rajan, R.; Song, X.; Biedenkapp, A.; Miao, Y.; Eimer, T.; Zhang, B.; Nguyen, V.; Calandra, R.; Faust, A.; et al. Automated reinforcement learning (autorl): A survey and open problems. *J. Artif. Intell. Res.* **2022**, *74*, 517–568. [CrossRef]
5. Frank, H.; Kotthoff, L.; Vanschoren, J. *Automated Machine Learning: Methods, Systems, Challenges*; Springer Nature: Berlin/Heidelberg, Germany, 2019.
6. Tsounis, V.; Alge, M.; Lee, J.; Farshidian, F.; Hutter, M. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3699–3706. [CrossRef]
7. Bledt, G.; Powell, M. J.; Katz, B.; Di Carlo, J.; Wensing, P. M.; Kim, S. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018.
8. Tuomas, H.; Ha, S.; Zhou, A.; Tan, J.; Tucker, G.; Levine, S. Learning to walk via deep reinforcement learning. *arXiv* **2018**, arXiv:1812.11103.
9. Miki, T.; Lee, J.; Hwangbo, J.; Wellhausen, L.; Koltun, V.; Hutter, M. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robot.* **2022**, *7* eabk2822. [CrossRef] [PubMed]
10. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015** *518* 529–533. [CrossRef] [PubMed]
11. Wang, J.X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J.Z.; Munos, R.; Blundell, C.; Kumaran, D.; Botvinick, M. Learning to reinforcement learn. *arXiv* **2016**, arXiv:1611.05763.
12. Duan, Y.; Schulman, J.; Chen, X.; Bartlett, P.L.; Sutskever, I.; Abbeel, P. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv* **2016**, arXiv:1611.02779.
13. Barret, Z.; Le, Q. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578.
14. Afshar, R.R.; Zhang, Y.; Vanschoren, J.; Kaymak, U. Automated reinforcement learning: An overview. *arXiv* **2022**, arXiv:2201.05000.
15. Icarte, R.T.; Klassen, T.Q.; Valenzano, R.; McIlraith, S.A. Reward machines: Exploiting reward function structure in reinforcement learning. *J. Artif. Intell. Res.* **2022** *73* 173–208. [CrossRef]
16. Vulin, N.; Christen, S.; Stevšić, S.; Hilliges, O. Improved learning of robot manipulation tasks via tactile intrinsic motivation. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2194–2201. [CrossRef]
17. Hafez, M.B.; Wermter, S. Behavior self-organization supports task inference for continual robot learning. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September 2021; pp. 6739–6746.
18. Dong, Y.; Zhang, S.; Liu, X.; Zhang, Y.; Shen, T. Variance aware reward smoothing for deep reinforcement learning. *Neurocomputing* **2021** *458*, 327–335. [CrossRef]
19. Dong, Y.; Tang, X.; Yuan, Y. Principled reward shaping for reinforcement learning via lyapunov stability theory. *Neurocomputing* **2020**, *393*, 83–90. [CrossRef]
20. Carta, T.; Oudeyer, P.Y.; Sigaud, O.; Lamprier, S. Eager: Asking and answering questions for automatic reward shaping in language-guided rl. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 12478–12490.
21. Elthakeb, A.; Pilligundla, P.; Mireshghallah, F.; Yazdanbakhsh, A.; Gao, S.; Esmaeilzadeh, H. Releq: An automatic reinforcement learning approach for deep quantization of neural networks. In Proceedings of thte NeurIPS ML for Systems Workshop, Vancouver, BC, Canada, 7 March 2019.
22. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
23. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
24. Hutter, M.; Gehring, C.; Jud, D.; Lauber, A.; Bellicoso, C.D.; Tsounis, V.; Hwangbo, J.; Bodie, K.; Fankhauser, P.; Bloesch, M.; et al. Anymal-a highly mobile and dynamic quadrupedal robot. In Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, Republic of Korea, 9–14 October 2016.
25. Makoviychuk, V.; Wawrzyniak, L.; Guo, Y.; Lu, M.; Storey, K.; Macklin, M.; Hoeller, D.; Rudin, N.; Allshire, A.; Handa, A.; et al. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. *arXiv* **2021**. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.