mathematics

MDPI

Article

Evaluating Domain Randomization in Deep Reinforcement Learning Locomotion Tasks

Oladayo S. Ajani, Sung-ho Hur * and Rammohan Mallipeddi *

School of Electronics Engineering, Kyungpook National University, Daegu 37224, Republic of Korea; oladayosolomon@gmail.com
* Correspondence: shur@knu.ac.kr (S.-h.H.); mallipeddi.ram@gmail.com (R.M.)

Abstract: Domain randomization in the context of Reinforcement learning (RL) involves training RL agents with randomized environmental properties or parameters to improve the generalization capabilities of the resulting agents. Although domain randomization has been favorably studied in the literature, it has been studied in terms of varying the operational characters of the associated systems or physical dynamics rather than their environmental characteristics. This is counter-intuitive as it is unrealistic to alter the mechanical dynamics of a system in operation. Furthermore, most works were based on cherry-picked environments within different classes of RL tasks. Therefore, in this work, we investigated domain randomization by varying only the properties or parameters of the environment rather than varying the mechanical dynamics of the featured systems. Furthermore, the analysis conducted was based on all six RL locomotion tasks. In terms of training the RL agents, we employed two proven RL algorithms (SAC and TD3) and evaluated the generalization capabilities of the resulting agents on several train–test scenarios that involve both in-distribution and out-distribution evaluations as well as scenarios applicable in the real world. The results demonstrate that, although domain randomization favors generalization, some tasks only require randomization from low-dimensional distributions while others require randomization from high-dimensional randomization. Hence the question of what level of randomization is optimal for any given task becomes very important.

Keywords: generalization; deep reinforcement learning; dynamic environments; locomotion; domain randomization

MSC: 68T01

Citation: Ajani, O.S.; Hur, S.-h.; Mallipeddi, R. Evaluating Domain Randomization in Deep Reinforcement Learning Locomotion Tasks. *Mathematics* **2023**, *11*, 4744. https://doi.org/10.3390/math11234744

Academic Editor: Moussa Labbadi

Received: 24 October 2023
Revised: 20 November 2023
Accepted: 22 November 2023
Published: 23 November 2023

1. Introduction

Several bio-inspired robotic applications and learning tasks often require the agents to adapt to the uncertainties in their environment. In simulated environments, Deep Reinforcement Learning (DRL) has proven to be successful at learning tasks across a wide range of domains such as games [1], rehabilitation [2], locomotion [3], production optimization [4], control [5–7], etc. However, in real-world environments, the deployment of DRL is often limited due to changing environmental conditions, an intrinsic feature mostly associated with real-world applications. Therefore, it is crucial that DRL agents are able to generalize over environmental conditions that they have never encountered during training [8]. Although training DRL agents directly in real-world environments seems to be a probable alternative, it is rarely practiced due to issues related to safety, time, and cost. The success of DRL in simulated environments is realized due to the advantage of time and stability featured in simulated environments. Furthermore, during the DRL training process, unsafe and practically impossible behaviors pertaining to real-world scenarios such as robot singularity can be explored in simulation. Therefore, sim-to-real transfer of DRL agents has been a major topic in the DRL community [9], where the motive is to develop

DRL agents that can generalize their performance over a range of real-world scenarios through the knowledge gained during training [9]. In DRL, the aim of generalization is to learn representations that are stable under dynamic environments [10–12] and/or avoid overfitting during training [13]. The literature on generalization under dynamic environments in DRL can be classified broadly into: (1) works that focus on developing dedicated DRL algorithms that are robust under changing environmental conditions [14,15]; and (2) works that introduce environmental uncertainties or domain randomization during training to realize agents that are more robust to the mismatches between simulation and reality [10,16,17]. Domain randomization is primarily defined as the use of an environment (source domain) with randomized properties or parameters during the training phase of an RL agent with the expectation that the resulting agent would generalize under uncertainties in the test environment (target domain) [18].

In the literature [8,12,19], comparative studies have shown that the use of dedicated generalization DRL algorithms are not superior to base DRL algorithms such as PPO and A2C in terms of generalization with the same training resources and conditions. This could be due to the significant challenges posed to featured optimization methods by the highly non-convex–concave nature of the objective functions employed to achieve robustness. In game-theoretical terms, these methods search for pure Nash Equilibria (trade-off solutions) that might not even exist [20]. According to one study [8], simply adding uncertainties into the environments during training can yield agents that can generalize to environments with similar uncertainties. Studies on achieving generalization by varying environmental conditions or domain randomization during training were reported in the field of games, where variations in the paddle and ball size were applied on selected Atari Games during training [17]. To facilitate DRL research in areas such as sim-to-real transfer as well as generalization, DeepRacer, an educational autonomous racing testbed, was proposed [21]. In this platform, simulation parameters such as tracks, lighting, sensor, and actuator noise can be randomized. Consequently, agents trained based on such domain randomization were reported to generalize in terms of multiple cars and tracks as well as to variations in speed, background, lighting, track shape, color, and texture. Through the robot arm reacher task in PyBullet, researchers [22] investigated the use of custom perturbations to introduce domain randomization. A number of works have also evaluated the use of environmental variation in robot navigation and locomotion tasks. For example, the authors of [10] evaluated two robot locomotion tasks and introduced dynamic environmental conditions by varying the operation parameters or characteristics of the robot body. The authors of [8] performed an empirical study on generalization using classical control environments (CartPole, MountainCar, and Pendulum, where features such as force, length, and Mass were randomly varied) as well as two locomotion tasks from Roboschool (HalfCheetah and Hopper, where robot operational features such as power, torso density, and friction were varied).

In the above works related to generalization in navigation and locomotion environments, the focus is more on varying the robot operation to mimic the environmental changes rather than varying the features of the environment itself, such as terrain. Furthermore, although the authors of [8] combined changes in terrain with changes in the robot operation and evaluated this using two locomotion environments from Roboschool, it is difficult to reach viable conclusions based on only two cherry-picked environments. Hence, the need for a rigorous evaluation using several complex navigation and locomotion environments is critical. The main contribution of the current work is to provide rigorous evaluations to analyze the generalization capabilities of base DRL algorithms where uncertainties are only considered with respect to the environmental terrain without modifying the robot operations for complex navigation and locomotion RL environments. So, unlike existing works, rigorous evaluations of the generalization ability of base DRL algorithms when trained with dynamic environmental conditions are presented for complex navigation and locomotion environments. The evaluations are performed using six complex benchmark PyBullet locomotion tasks [23] with varying environmental conditions. The variations

in the dynamics of the environment were studied under varying friction settings of the environmental terrain. An application scenario corresponding to the above experimental design is as follows: a robot trained to navigate a normal terrain or floor is to be deployed in a slippery terrain. Furthermore, the evaluations were carried out using two proven state-of-the-art DRL algorithms: (1) from the actor–critic class, Soft Actor–Critic (SAC); and (2) from the policy gradient class, the Twin Delayed Deep Deterministic policy gradient algorithm (TD3).

The remainder of the paper is organized as follows. Section 2 presents a detailed overview of the existing literature on generalization in DRL. In Section 3, highlights on the proven DRL algorithms used in this study are presented. Section 4 provides details regarding the benchmark locomotion environments used in this study and the dynamic changes introduced in each of the environments, while Section 5 provides details about the experimental methodology. In Section 6, the results and discussions from the experiments are presented, while Section 7 presents the conclusions and future directions.

2. Related Works

In the literature, several DRL approaches have been proposed to achieve generalization in RL. One such approach is meta-learning, where the ability to adapt to unseen test environments is achieved through the learning process performed on multiple training tasks. Furthermore, referred to as deep metaRL, it usually involves a recurrent neural network policy whose inputs also entail the actions selected and reward received in the previous time step [24]. Another class of DRL algorithms developed to achieve generalization are known as robust RL algorithms, where the algorithms are developed to be able to handle perturbations in the transition dynamics of the model. In one study [25], a robust RL algorithm known as Maximum a posteriori Policy Optimization (MPO) was developed for continuous control. In the approach, a policy that optimizes the worst-case, entropy-regularized, expected return objective was learned to derive a corresponding robust entropy-regularized Bellman contraction operator. Training DRL agents on a collection of risk-averse environments was evaluated in another study [10] using four benchmark locomotion environments where linear and RBF parameterizations were introduced to realize robustness in the DRL algorithm. Recently, introducing variations or uncertainties into RL environments during training to realize generalization has gained a lot of traction. This is due to the fact that recent studies have proved that vanilla DRL algorithms generalize better than their counterparts (EPO_{pt} and RL^2), dedicated Robust DRL algorithms for generalization [8]. In an empirical study [8], four classical control tasks were evaluated with changes in force, mass, and length introduced into the environments. In addition, this was extended for two locomotion environments where variations in the environments were introduced through changes in robot power, density, and ground friction. Those environments were evaluated using PPO and A2C and their corresponding Robust DRL algorithms (EPO_{pt} and RL^2) for the comparative analysis. The introduction of variation or uncertainties into DRL environments is mostly achieved in control tasks through domain randomization, where dynamic properties of either the associated system or its environment are varied. In a recent study [26], Dexterous In-Hand Manipulation for a five-fingered robot was learned, where environmental variations were introduced by randomly varying parameters such as the object dimension, surface friction, robot link and object masses, etc. In addition, a case for generalization was proven in this work by a successful sim-to-real transfer. Our work adopts a similar approach to achieve domain randomization through environmental variation. However, rather than introducing variations in the robot dynamics or a combination of the robot dynamics and changes in its environment, we explore randomization only by varying the surface friction of the terrain to investigate a scenario when a robot moves from a smooth terrain to a slippery terrain. We avoided changes in the robot dynamics because, in the real world, such dynamics cannot be changed directly but are rather a function of the terrain of the robot.

3. Algorithms

To access the effect of dynamic domains towards achieving generalization, we employ two typical proven DRL algorithms. Specifically, we choose Soft Actor–Critic (SAC) [27] and Twin Delayed Deep Deterministic (TD3) policy gradient [28], as they both lie between the two main families of RL algorithms (Policy Optimization and Q-learning) and hence, provides trade-off between Policy Optimization and Q-Learning. Such a trade-off is necessary here because, while policy optimization methods are considered stable and reliable, Q-learning methods are significantly more sample efficient when they carry out work because they can reuse data more efficiently. We use the stable baseline implementations of both algorithms [29] and highlight the main features of each algorithm in the following sections.

3.1. Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 is an off-policy algorithm that uses three crucial concepts to overcome the issue of overestimating Q-values (policy breaking) associated with Deep Deterministic Policy Gradient (DDPG) [30,31] as a result of overestimated Q-values. First, instead of learning a single Q-value, TD3 learns two Q-values and employs the one with the smaller value to estimate the target in the Bellman error loss functions. Secondly, TD3 updates the associated learning networks (policy and target networks) less frequently in contrast with Q-function. Lastly, by adding clipped noise to the target action as a form of regularization, TD3 makes it difficult for the policy to navigate towards Q-function errors by smoothing out Q along changes in action [28].

3.2. Soft Actor–Critic (SAC)

The SAC algorithm addresses a stochastic policy optimization problem in an off-policy approach. Similar to TD3, it also takes advantage of a clipped double-Q method and entails a unique feature of entropy regularization, which aims at maximizing the expected returns of the actor while maximizing entropy at the same time. The advantage here is that, while increasing entropy is associated with more exploration and hence faster learning, it can further help to prevent the policy from premature convergence [27].

4. Environments

The environments used for evaluations in this study are modified versions of six locomotion environments from PyBullet. The modification to the environments basically involves variations in the surface friction of the terrain. The resulting environmental terrains can be summarized as follows:

1. Normal Terrain (NT): The friction parameters of the environments under this terrain are kept constant with the default implementation values d from PyBullet. This implies that only the state variables are reset each time the environment is reset;
2. Random Terrain (RT): In this case, each time an episode is terminated, and the environment is re-initialized (reset), the friction coefficient f_c is sampled randomly from a k-dimensional uniform distribution (box) containing the default values d;
3. Extreme Terrain (ET): Here the friction coefficient f_c corresponding to the terrain is rest every time an episode is terminated, through uniformly sampling from one of the 2^k k-dimensional uniform distribution (box). Specifically, f_c is sampled from the union of two intervals that straddle the corresponding interval in RT.

The search spaces corresponding to NT, RT, and ET are illustrated schematically in Figure 1, where $k = 2$, the search space of RT is indicated by the bounded white box and those of ET are illustrated by the disconnected black boxes. The ranges of the actual friction parameter values corresponding to each environment are presented in Table 1.

Table 1. Range of the coefficient of friction for each environmental terrain.

Terrain	NT	RT	ET
Friction Coefficient	0.8	[0.5, 1.1]	[0.2, 0.5] ∪ [1.1, 1.4]

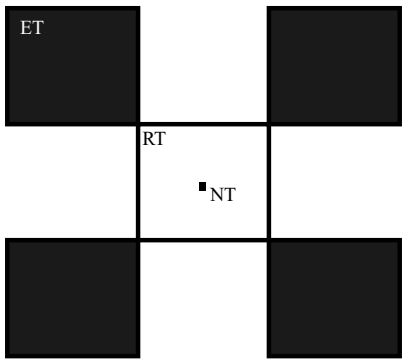
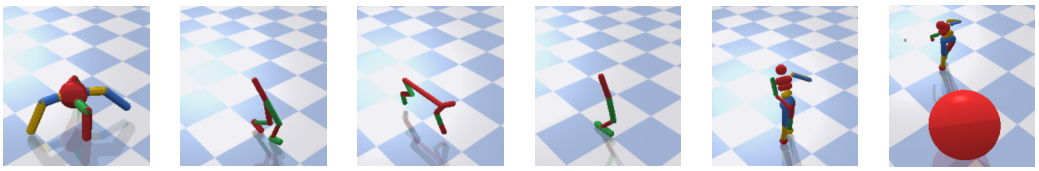


Figure 1. Schematics of the three (3) different environmental terrains.

Based on each of these environmental terrains, six locomotion tasks with different levels of difficulty and dynamics were trained. All the six tasks, Hopper [32,33], 2D Walker [32,33], Ant [34], HalfCheetah [34–36], Humanoid [34,37], and Humanoid Flagrun Harder (Flagrun) [38], were implemented based on the existing locomotion environments in PyBullet, which are modified and more realistic versions of those from Mujoco. For example, in the PyBullet version of the Ant environment, the ant is much heavier, thus ensuring that it has two or more legs on the ground to sustain its weight. Figure 2 shows a graphical illustration of each environment. The goal in all the tasks, except in Humanoid Flagrun Harder, is to learn to move forward quickly without falling. This requires that the humanoid move to a specific target whose position varies randomly while the humanoid is constantly bombarded by some cubes to push it off its trajectory. The detailed features, goal, and reward systems for each of the environments are as presented in [23].



(**a**) Ant (**b**) Walker2D (**c**) HalfCheetah (**d**) Hopper (**e**) Humanoid (**f**) Flagrun

Figure 2. Graphical illustration of the locomotion environments.

5. Experimental Design

To evaluate the effect of training based on each of the environmental terrain discussed in Section 4, we perform a series of train–test scenarios where each agent is trained on a specific terrain and tested on the same terrain (in-distribution), as well as all other terrains (out-distribution). Specifically, we train on two DRL algorithms (SAC and TD3), with NT, RT, and ET variation on six environments (Ant, 2D Walker, Hooper, HalfCheetah, Humanoid, and Humanoid Flagrun Harder). Consequently, we tested all the resulting agents on all the terrains (NT, RT, and ET) and compared them using three testing scenarios. Both SAC and TD3 were trained for 1×10^6 timesteps for all the training scenarios. For testing purposes, the best models returned during training were used and each of the testing scenarios were performed based on 25 runs of independent episodes (with an episode length of 1000 for each). To enable fairness, the hyperparameters used for each algorithms were the same (for all the environments and training scenarios) and their values were fixed as those provided in stable-baselines—a collection of improved implementations of RL algorithms [29].

Performance and Evaluation Metrics

Since RL agents are defined generally as goal-seeking, in RL, performance is measured in terms of the average cumulative reward of each Markov Decision Process (MDP) cycle or episode achieved by an agent based on the goal of the associated task. Consequently, we evaluate performance using the mean and dispersion (standard deviation) of the average cumulative reward.

To evaluate generalization performance as a result of domain randomization, three test scenarios were formulated to evaluate agents trained based on each of the aforementioned terrains (Section 4) to be tested and compared with those from other terrains. The three test scenarios are as follows.

1. **Default:** In this scenario, each agent trained on each of NT, RT, and ET, respectively, is tested on the default environmental parameters (same as NT). This provides insight into the effects of domain randomization when compared to agents trained on default conditions;
2. **Interpolation:** This scenario is expected to evaluate the performance of each agent on a dynamic terrain with a mild level of uncertainty. Hence, all the agents are tested based on the conditions of RT. An insight into the effect of mild-level domain randomization on the agents can be obtained by comparing results with those trained with no randomization (NT) as well as with a relatively high level of randomization (ET);
3. **Extrapolation:** In extrapolation, ET conditions are used for the testing environment. In this test scenario, insight into the generalization effect of domain randomization to totally unseen conditions (NT) as well as partially unseen conditions (RT) during training are evaluated and compared with those trained on similar terrain conditions as the test-bed (ET).

For each of the three test scenarios, the mean and dispersion (standard deviation) of the cumulative reward as well as the episode length for each of the corresponding agents on the associated test conditions are detailed in Section 6.

Finally, we perform evaluations based on a real-world applicable scenario in which a legged robot trained for operation under default environmental conditions as well as those trained with domain randomization are deployed for use in a slippery surface or terrain. This presents a case for performance in terms of generalization in unseen conditions during training. From the literature [39,40], the generally recommended Coefficient of Friction (CoF) for walking is 0.5 and above. Hence a CoF value below 0.5 implies that the surface is slippery. To evaluate how agents trained with or without randomization will generalize over such a scenario, we tested each of the resulting agents from NT, RT, and ET on an environment with a constant CoF value of 0.4.

6. Results and Discussion

Based on the train–test scenarios presented in Section 5 above, training and testing performance are presented in this Section. First, the performance during training using SAC and TD3 based on each of NT, RT, and ET are presented. Subsequently, we provide results and analysis for all the test scenarios.

6.1. Training

The accumulated rewards during training each of the respective agents based on the associated environmental terrain for all the six locomotion tasks are shown in Figures 3 and 4 for SAC and TD3, respectively. Although it can be observed that NT-based agents show slightly superior performance in instances such as Figures 3a and 4d,f, while RT-based agents show a slightly better training progression in Figures 3b and 4a,c. It is clear that all the agents are competitive in terms of the accumulated return irrespective of their environmental situations. This means that the agents were able to learn policies to adapt to the changes in their environmental conditions for all the tasks.

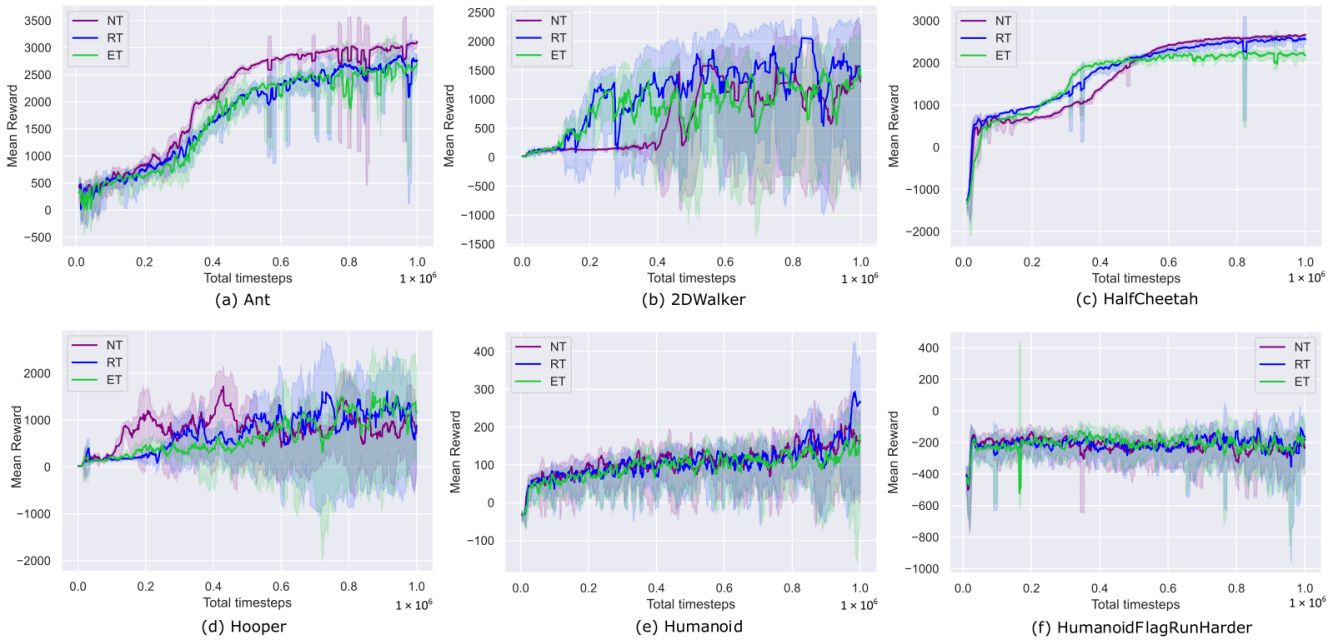


Figure 3. Performance comparison of SAC in terms of accumulated reward during training for all six locomotion tasks and associated environmental terrains.

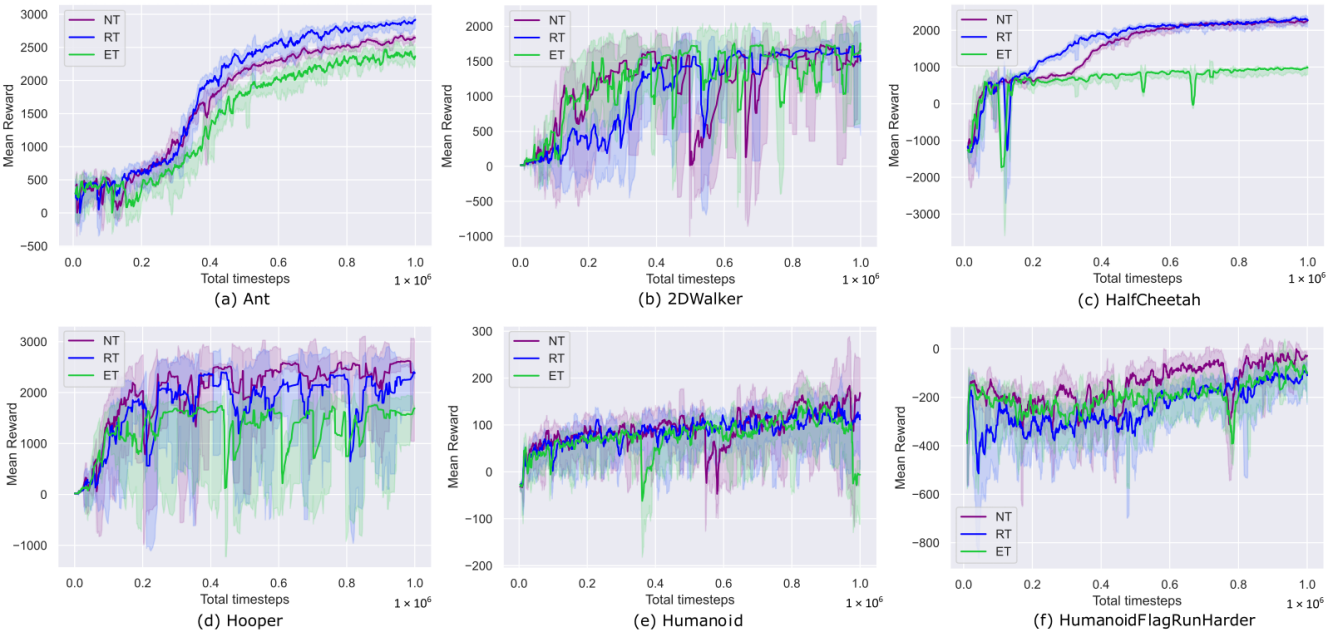


Figure 4. Performance comparison of TD3 in terms of accumulated reward during training for all six locomotion tasks and associated environmental terrains.

6.2. Testing

6.2.1. Default

The test performance of each resulting agent for both algorithms and all six locomotion tasks based on the NT are presented in Table 2. The presented data summarize the mean and standard deviation of the reward as well as the episode length. The expectation is that agents trained on NT will outperform other agents since they were trained on conditions similar to the test conditions. However, in terms of average returns over all the problem instances, NT only shows superior performance in five cases, while RT proves competitive by showing a superior performance on five instances. ET, however, outperforms NT and RT in only two cases.

It is clear that even on NT, domain randomization during training is both comparable and advantageous. In other words, domain randomization does not degrade the results when compared with those agents trained and tested on default conditions, instead it even results in a better performance as observed in seven instances (RT and ET combined).

Table 2. Performance of the implemented agents for each of the locomotion tasks on the NT testbed in terms of mean and standard deviation of the accumulated reward.

Environment	Algorithm	Reward (Mean & Std Dev)		
		NT	RT	ET
Ant	TD3	2773.37 ± 14.18	**3009.35 ± 17.24**	2536.21 ± 19.20
	SAC	**3231.36 ± 12.40**	2960.18 ± 22.17	2857.20 ± 6.23
Walker2D	TD3	**1871.02 ± 23.89**	1788.12 ± 7.51	1790.57 ± 6.77
	SAC	2011.16 ± 375.79	**2130.09 ± 23.60**	1930.08 ± 8.25
HalfCheetah	TD3	2353.24 ± 30.40	**2408.77 ± 32.01**	985.74 ± 8.71
	SAC	**2772.90 ± 25.19**	2696.64 ± 13.90	2349.12 ± 26.31
Hopper	TD3	**2741.39 ± 11.00**	2526.66 ± 8.97	1793.55 ± 18.34
	SAC	2404.11 ± 714.35	**2526.22 ± 261.19**	1831.47 ± 546.80
Humanoid	TD3	149.85 ± 62.51	139.39 ± 38.69	**171.96 ± 43.16**
	SAC	**205.75 ± 91.88**	180.90 ± 96.46	157.68 ± 82.71
Humanoid Flagrun Harder	TD3	−53.25 ± 46.76	**18.06 ± 20.81**	−28.83 ± 15.84
	SAC	−0.37 ± 18.79	−0.70 ± 24.30	**4.54 ± 26.59**

Best values are highlighted.

6.2.2. Interpolation

Here, the testbed is based on the same environmental conditions as in RT. Table 3 shows the average reward and episode length when testing each of the resulting agents for both algorithms and all six locomotion tasks based on RT. In all 12 instances, agents trained based on the ET conditions were the worst performing with only one instance where they outperformed those from RT and NT. The results from both NT and RT were rather competitive, with NT winning in six instances and RT in five instances.

Table 3. Performance of the implemented agents for each of the locomotion tasks on the RT testbed in terms of the mean and standard deviation of the accumulated reward.

Environment	Algorithm	Reward (Mean & Std Dev)		
		NT	RT	ET
Ant	TD3	1633.09 ± 45.25	**2999.95 ± 27.32**	2549.26 ± 33.37
	SAC	**3209.12 ± 31.24**	2952.29 ± 65.89	2849.54 ± 26.08
Walker2D	TD3	**1861.79 ± 20.29**	1792.21 ± 9.12	1791.41 ± 19.56
	SAC	1540.25 ± 304.29	**2005.30 ± 400.97**	1942.19 ± 28.44
HalfCheetah	TD3	2285.24 ± 80.77	**2378.99 ± 28.36**	993.48 ± 13.68
	SAC	**2710.21 ± 122.65**	2676.58 ± 80.08	2349.11 ± 32.35
Hopper	TD3	**2554.05 ± 498.93**	2420.55 ± 294.21	1786.45 ± 20.95
	SAC	1535.22 ± 1164.44	**1820.26 ± 952.38**	1552.99 ± 705.53
Humanoid	TD3	**169.09 ± 74.05**	146.93 ± 55.99	158.35 ± 67.99
	SAC	**232.37 ± 105.32**	165.46 ± 114.43	164.74 ± 98.91
Humanoid Flagrun Harder	TD3	−50.36 ± 41.57	**−5.75 ± 22.64**	−25.36 ± 20.42
	SAC	−42.96 ± 31.91	−2.59 ± 13.05	**7.59 ± 20.38**

Best values are highlighted.

6.2.3. Extrapolation

The results of testing agents on environments with the same conditions as ET are presented in Table 4. Relative to the average return, ET shows a superior performance in six problem instances, while those trained on RT also showed superior performance on four problem instances. NT-based agents, however, only outperformed ET and RT in two problem instances. This shows that, generally, agents trained with domain randomization are able to generalize more in dynamic environments or environments with varying conditions irrespective of whether they had seen similar conditions during training (ET) or not (RT), compared with those trained without any form of randomization (NT). Furthermore, it is clear that ET-trained agents proved to be more stable than other agents in terms of their dispersion as they were trained on similar conditions to the testbed.

Table 4. Performance of the implemented agents for each of the locomotion tasks on the ET testbed in terms of mean and standard deviation of the accumulated reward.

Environment	Algorithm	Reward (Mean & Std Dev)		
		NT	**RT**	**ET**
Ant	TD3	1383.20 ± 429.61	**2838.00 ± 232.40**	2480.66 ± 131.91
	SAC	2599.88 ± 872.06	2114.00 ± 1073.54	**2797.41 ± 90.80**
Walker2D	TD3	1117.09 ± 75.48	1486.92 ± 600.37	**1741.68 ± 269.48**
	SAC	896.93 ± 664.27	1224.27 ± 849.52	**1685.02 ± 484.83**
HalfCheetah	TD3	1897.13 ± 309.04	**2262.60 ± 92.93**	1004 ± 20.63
	SAC	1950.22 ± 797.80	**2332.00 ± 203.63**	2295.29 ± 114.13
Hopper	TD3	1571.73 ± 1283.61	1501.59 ± 1159.81	**1780.24 ± 32.79**
	SAC	**1613.85 ± 1234.47**	1316.51 ± 1213.35	1250.74 ± 876.91
Humanoid	TD3	**135.01 ± 62.76**	104.25 ± 47.94	134.65 ± 63.33
	SAC	175.89 ± 54.27	141.20 ± 56.02	**199.88 ± 70.61**
Humanoid Flagrun Harder	TD3	−55.41 ± 40.60	**5.54 ± 23.26**	−32.42 ± 25.68
	SAC	−44.01 ± 24.48	−13.49 ± 15.31	**12.94 ± 24.94**

Best values are highlighted.

In Table 5, we present a summary of the deductions from all the test instances, showing the number of wins and losses for each of the associated agents under all testing conditions. From the results, it can be observed that there are 13 instances of the train–test scenarios where NT outperforms both RT and ET. This implies that there are instances where agents trained with domain randomization do not always outperform those trained without domain randomization. For example, in Tables 2–4, ET fails on the HalfCheetah task. This usually occurs because the associated agents could not capture the varying dynamics of the environment effectively, hence, the agent was not able to learn the task appropriately.

Table 5. Summary of Performance of testing all agents on NT, RT, and ET testbeds.

Metric	Reward		
	NT	**RT**	**ET**
Win (+)	13	14	9
Loss (−)	23	22	27
Draw (≈)	–	–	–

6.2.4. Comparison between SAC and TD3

Although both SAC and TD3 are high-performing, state-of-the-art DRL algorithms, their performance has not been widely studied in the context of domain randomization. In order to investigate their performance in the context of domain randomization, we compare the cumulative rewards featured by each algorithm on each of the 18 train–test scenarios

presented in Tables 2–4. From the 18 instances, it can be observed that SAC obtained the best cumulative reward on 13 instances while TD3 obtained the best cumulative reward on only 5 instances. This implies that SAC functions better than TD3 in the context of domain randomization. This can generally be attributed to the featured stochastic policy optimization in SAC [27].

6.2.5. Real-World Scenario

Although the advantages of domain randomization during training for reinforcement learning is illustrated from the aforementioned testing scenario, we present yet another scenario that is applicable in the real world. Here, we assume that the terrain is slippery, and hence, the testing CoF of the testbed is set at 0.4. Figures 5 and 6 show the box plots for all the locomotion tasks based on SAC and TD3, respectively. For each of the agents (NT, RT, and ET), we also show the average returns for each of the tasks. The superiority of the agents trained with domain randomization is clearly demonstrated in this scenario. For instance, in Figure 5, agents trained with domain randomization (RT and ET) had higher average returns in all the tasks than those trained on default conditions (NT), except for the humanoid environment (Figure 5e) where NT outperformed ET by a relatively small margin. Similarly, in Figure 6, one of the domain-randomization-based agents (either RT or ET) always outperforms the NT-based agents, and in most of the cases where NT-based agents outperform one of either RT or ET (Figure 6d–f), the margin is always very small. In Table 6, we present a summary of the results for all the instances evaluated based on the real-world scenario. It is clear that domain randomization (as in RT or ET) facilitates generalization; however, it is not exactly clear what type or what level of domain randomization is required for a certain task because, while RT-trained agents were superior in some task instances, ET-trained agents were superior in others.

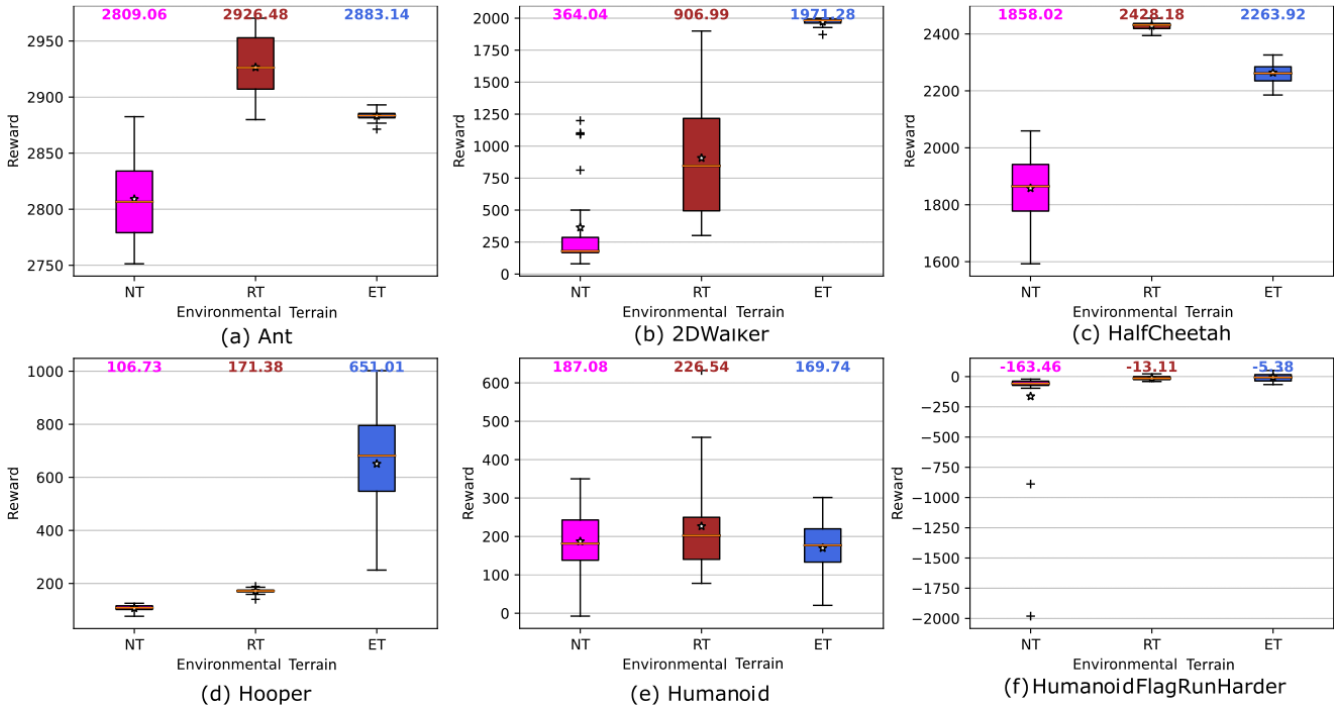


Figure 5. Box plots showing the performance results of each NT-, RT-, and ET-trained agent on the real-world testbed with mean reward values highlighted using SAC.

Table 6. Summary of Performance of testing all agents on the real-world-scenario testbeds.

Metric	Reward		
	NT	RT	ET
Win (+)	0	6	6
Loss (−)	12	6	6
Draw (≈)	–	–	–

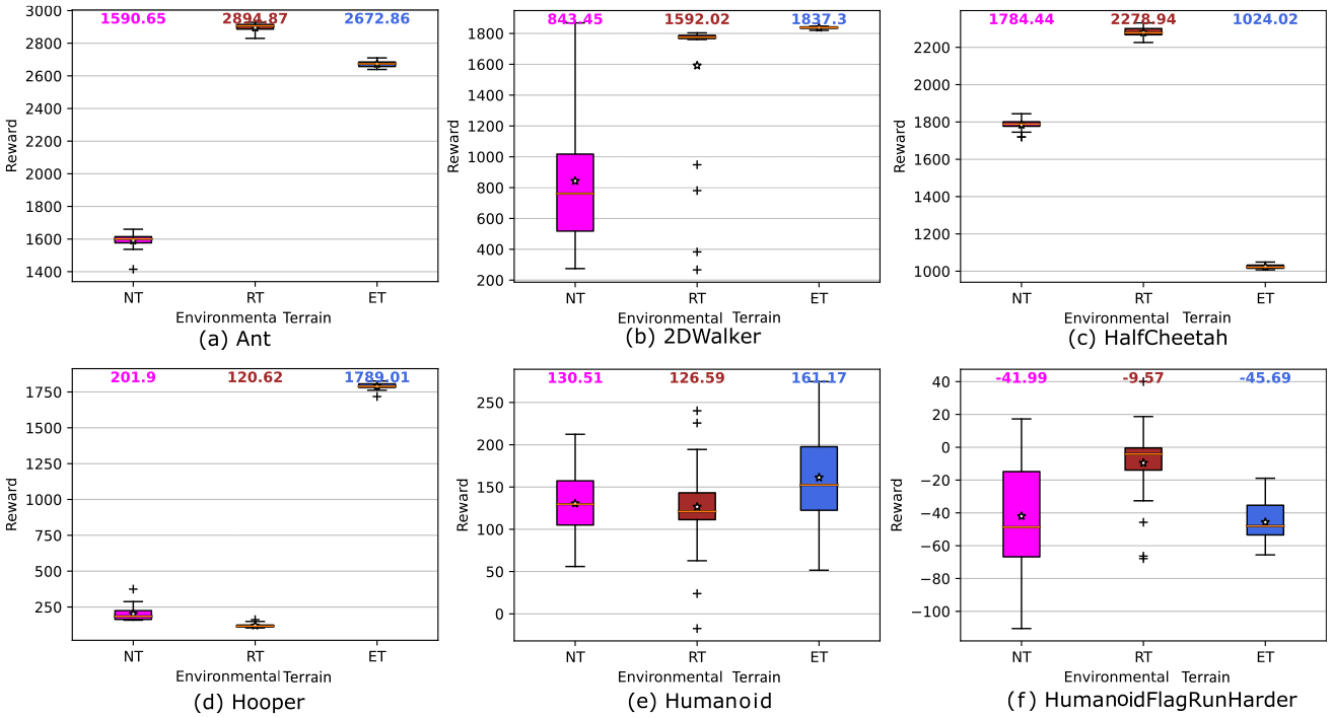


Figure 6. Box plots showing the performance results of each NT-, RT-, and ET-trained agent on the real-world testbed with mean reward values highlighted using TD3.

7. Conclusions and Future Work

An assessment of domain randomization in DRL for locomotion tasks is presented in this work. Specifically, we evaluated two state-of-the-art deep RL algorithms (SAC and TD3) on six locomotion tasks based on three different terrain conditions (NT, RT, and ET). The adopted framework is motivated by previous studies on generalizations from the OpenAI Retro contest [19] as well as the CoinRun benchmark [12], which concluded that vanilla deep RL algorithms trained with environmental stochasticity may be more effective for generalization than specialized algorithm. Similar to the authors of [8], we introduced a system of testbeds and experimental protocol to evaluate the capability of DRL algorithms trained with or without domain randomization to generalize to environments both similar to and different from those seen during training. Furthermore, we introduce a real-world scenario where the performance of all the trained DRL agents are compared on a common real-world scenario (slippery terrain).

Overall, agents trained with domain randomization have better generalization performance than those trained without any form of domain randomization in terms of the accumulated returns. However, the question of what type and what level of domain randomization is necessary and sufficient for a specific task is outstanding. Therefore, in the future, we plan to introduce an optimization framework that incorporates a range of parameters for domain randomization as hyperparameters. This is important because, as demonstrated by the results, different environments or tasks benefit from different levels of domain randomization, and the optimal setting of domain randomization parameters would lead to better generalization results.

Author Contributions: Conceptualization, O.S.A. and R.M.; methodology, O.S.A. and R.M.; formal analysis, O.S.A. and R.M.; data curation, O.S.A. and R.M.; writing—original draft preparation, O.S.A. and R.M.; writing—review and editing, O.S.A., S.-h.H. and R.M.; supervision, S.-h.H. and R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Acknowledgments: This research was supported by the Korea Electric Power Corporation (KEPCO) (R21XO01-17).

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

References

1. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.A. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
2. Erickson, Z.M.; Gangaram, V.; Kapusta, A.; Liu, C.; Kemp, C. Assistive Gym: A Physics Simulation Framework for Assistive Robotics. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 10169–10176.
3. Peng, X.B.; Coumans, E.; Zhang, T.; Lee, T.; Tan, J.; Levine, S. Learning Agile Robotic Locomotion Skills by Imitating Animals. *arXiv* **2020**, arXiv:2004.00784.
4. Zhang, K.; Wang, Z.; Chen, G.; Zhang, L.; Yang, Y.; Yao, C.; Wang, J.; Yao, J. Training effective deep reinforcement learning agents for real-time life-cycle production optimization. *J. Pet. Sci. Eng.* **2022**, *208*, 109766. [CrossRef]
5. Peng, Z.; Hu, J.; Shi, K.; Luo, R.; Huang, R.; Ghosh, B.K.; Huang, J. A novel optimal bipartite consensus control scheme for unknown multi-agent systems via model-free reinforcement learning. *Appl. Math. Comput.* **2020**, *369*, 124821. [CrossRef]
6. Fu, Q.; Li, Z.; Ding, Z.; Chen, J.; Luo, J.; Wang, Y.; Lu, Y. ED-DQN: An event-driven deep reinforcement learning control method for multi-zone residential buildings. *Build. Environ.* **2023**, *242*, 110546. [CrossRef]
7. Ajani, O.S.; Mallipeddi, R. Adaptive evolution strategy with ensemble of mutations for reinforcement learning. *Knowl.-Based Syst.* **2022**, *245*, 108624. [CrossRef]
8. Packer, C.; Gao, K.; Kos, J.; Krähenbühl, P.; Koltun, V.; Song, D. Assessing Generalization in Deep Reinforcement Learning. *arXiv* **2018**, arXiv:1810.12282.
9. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 1–4 December 2020; pp. 737–744.
10. Rajeswaran, A.; Lowrey, K.; Todorov, E.; Kakade, S. Towards Generalization and Simplicity in Continuous Control. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
11. Zhang, A.; Ballas, N.; Pineau, J. A Dissection of Overfitting and Generalization in Continuous Reinforcement Learning. *arXiv* **2018**, arXiv:1806.07937.
12. Cobbe, K.; Klimov, O.; Hesse, C.; Kim, T.; Schulman, J. Quantifying Generalization in Reinforcement Learning. In Proceedings of the 2019 International Conference on Machine Learning (ICML), Long Beach, CA, USA, 10–15 June 2019.
13. Whiteson, S.; Tanner, B.; Taylor, M.E.; Stone, P. Protecting against evaluation overfitting in empirical reinforcement learning. In Proceedings of the 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), Paris, France, 11–15 April 2011; pp. 120–127. [CrossRef]
14. Finn, C.; Abbeel, P.; Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Proceedings of the 2017 International Conference on Machine Learning (ICML), Sydney, Australia, 6–11 August 2017.
15. Duan, Y.; Schulman, J.; Chen, X.; Bartlett, P.; Sutskever, I.; Abbeel, P. RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. *arXiv* **2016**, arXiv:1611.02779.
16. Vacaro, J.; Marques, G.; Oliveira, B.; Paz, G.; Paula, T.; Staehler, W.; Murphy, D. Sim-to-Real in Reinforcement Learning for Everyone. In Proceedings of the 2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE), Rio Grande, Brazil, 23–25 October 2019; pp. 305–310. [CrossRef]
17. Kansky, K.; Silver, T.; Mély, D.A.; Eldawy, M.; Lázaro-Gredilla, M.; Lou, X.; Dorfman, N.; Sidor, S.; Phoenix, D.; George, D. Schema Networks: Zero-shot Transfer with a Generative Causal Model of Intuitive Physics. *arXiv* **2017**, arXiv:1706.04317.
18. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 23–30.
19. Nichol, A.; Pfau, V.; Hesse, C.; Klimov, O.; Schulman, J. Gotta Learn Fast: A New Benchmark for Generalization in RL. *arXiv* **2018**, arXiv:1804.03720.
20. Kamalaruban, P.; Huang, Y.T.; Hsieh, Y.P.; Rolland, P.; Shi, C.; Cevher, V. Robust reinforcement learning via adversarial training with Langevin dynamics. *arXiv* **2020**, arXiv:2002.06063.

21. Balaji, B.; Mallya, S.; Genc, S.; Gupta, S.; Dirac, L.; Khare, V.; Roy, G.; Sun, T.; Tao, Y.; Townsend, B.; et al. DeepRacer: Educational Autonomous Racing Platform for Experimentation with Sim2Real Reinforcement Learning. *arXiv* **2019**, arXiv:1911.01562.
22. Zhao, W.; Queralta, J.P.; Qingqing, L.; Westerlund, T. Towards Closing the Sim-to-Real Gap in Collaborative Multi-Robot Deep Reinforcement Learning. In Proceedings of the 2020 5th International Conference on Robotics and Automation Engineering (ICRAE), Singapore, 20–22 November 2020; pp. 7–12.
23. Coumans, E.; Bai, Y. PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning. 2016–2021. Available online: http://pybullet.org (accessed on 6 April 2021).
24. Wang, J.X.; Kurth-Nelson, Z.; Soyer, H.; Leibo, J.Z.; Tirumala, D.; Munos, R.; Blundell, C.; Kumaran, D.; Botvinick, M. Learning to reinforcement learn. *arXiv* **2017**, arXiv:1611.05763.
25. Mankowitz, D.J.; Levine, N.; Jeong, R.; Abdolmaleki, A.; Springenberg, J.T.; Mann, T.; Hester, T.; Riedmiller, M.A. Robust Reinforcement Learning for Continuous Control with Model Misspecification. *arXiv* **2020**, arXiv:1906.07516.
26. Andrychowicz, O.M.; Baker, B.; Chociej, M.; Józefowicz, R.; McGrew, B.; Pachocki, J.W.; Petron, A.; Plappert, M.; Powell, G.; Ray, A.; et al. Learning dexterous in-hand manipulation. *Int. J. Robot. Res.* **2020**, *39*, 20–23. [CrossRef]
27. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the 2018 International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018.
28. Fujimoto, S.; Hoof, H.V.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. *arXiv* **2018**, arXiv:1802.09477.
29. Hill, A.; Raffin, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; Traore, R.; Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; et al. Stable Baselines. 2018. Available online: https://github.com/hill-a/stable-baselines (accessed on 27 September 2021).
30. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M.A. Deterministic Policy Gradient Algorithms. In Proceedings of the 2014 International Conference on Machine Learning (ICML), Beijing, China, 21–26 June 2014.
31. Lillicrap, T.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2016**, arXiv:1509.02971.
32. Erez, T.; Tassa, Y.; Todorov, E. Infinite-Horizon Model Predictive Control for Periodic Tasks with Contacts. In *Robotics: Science and Systems*; MIT Press: Cambridge, MA, USA, 2011.
33. Levine, S.; Koltun, V. Guided Policy Search. In Proceedings of the 2013 International Conference on Machine Learning (ICML), Atlanta, GA, USA, 17–19 June 2013.
34. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.I.; Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv* **2016**, arXiv:1506.02438.
35. Wawrzynski, P. Learning to Control a 6-Degree-of-Freedom Walking Robot. In Proceedings of the EUROCON 2007—The International Conference on "Computer as a Tool", Warsaw, Poland, 9–12 September 2007; pp. 698–705. [CrossRef]
36. Heess, N.M.O.; Hunt, J.J.; Lillicrap, T.P.; Silver, D. Memory-based control with recurrent neural networks. *arXiv* **2015**, arXiv:1512.04455.
37. Tassa, Y.; Erez, T.; Todorov, E. Synthesis and stabilization of complex behaviors through online trajectory optimization. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 4906–4913. [CrossRef]
38. Liang, J.; Makoviychuk, V.; Handa, A.; Chentanez, N.; Macklin, M.; Fox, D. GPU-Accelerated Robotic Simulation for Distributed Reinforcement Learning. In Proceedings of the 2nd Conference on Robot Learning (CoRL 2018), Zurich, Switzerland, 29–31 October 2018.
39. Miller, J.M. "Slippery" work surfaces: Towards a performance definition and quantitative coefficient of friction criteria. *J. Saf. Res.* **1983**, *14*, 145–158. [CrossRef]
40. Li, K.W.; Chang, W.R.; Leamon, T.B.; Chen, C.J. Floor slipperiness measurement: Friction coefficient, roughness of floors, and subjective perception under spillage conditions. *Saf. Sci.* **2004**, *42*, 547–565. [CrossRef]