Sim-to-Real Transfer for Hopper Control via Reinforcement Learning and Domain Randomization

1st Alessandro Scavone
Department of Computer Engineering
Politecnico di Torino
Torino, Italy
s328782@studenti.polito.it

2nd Vincenzo Montana
Department of Computer Engineering
Politecnico di Torino
Torino, Italy
s322977@studenti.polito.it

***Abstract*—This paper addresses the sim-to-real transfer problem in robotics, focusing on Reinforcement Learning (RL) for a Hopper-v0 environment. We simulate a reality gap by shifting the torso mass of the Hopper model in a *source* environment compared to a *target* environment, then train policies with Proximal Policy Optimization (PPO). We discuss how the state and action spaces of Hopper influence policy learning, and explain why performance decreases when transferring from a mismatched simulator to the correct dynamics in the target environment. After outlining standard PPO training, we employ Uniform Domain Randomization (UDR) on thigh, leg, and foot link masses to improve robustness against the torso mass discrepancy. We present an extended analysis that selectively randomizes different parts of the hopper, comparing uniform and Gaussian sampling distributions. Results show that carefully tuned or selectively applied randomization strategies can yield higher transfer performance, albeit at the cost of increased variance.**

***Index Terms*—Reinforcement Learning, Sim-to-Real, Domain Randomization, Robotics, PPO, Hopper, Simulation**

I. Introduction

Reinforcement Learning (RL) has demonstrated notable success in learning control policies for simulated robotic tasks, especially using policy-gradient methods such as Proximal Policy Optimization (PPO) [2]. However, direct application of such policies to real hardware frequently faces the *reality gap*, wherein small discrepancies in simulator modeling—like incorrect link masses or friction—cause significant performance drops when deployed on the actual robot.

In this work, we focus on a simplified *sim-to-sim* setting to study such a gap, introducing a known mismatch in the torso mass of the Hopper-v0 environment. Specifically, the *source* domain has its torso mass reduced by 1 kg from the default, while the *target* domain remains at the correct (original) mass. We first demonstrate how a policy trained only in the source domain underperforms in the target environment relative to both *source* $\rightarrow$ *source* and *target* $\rightarrow$ *target* performance levels. This illustrates the natural drop that arises from training on an imperfect model. Next, we explore Domain Randomization (DR) strategies, focusing on how uniform or Gaussian randomization of the thigh, leg, and foot masses can partially remedy this gap. Our extended study investigates the effect of tuning or sweeping randomization parameters individually for each link, offering insights into the trade-off between higher mean returns in the target domain and larger performance variance.

If there is any funding support, it can be acknowledged here. If none, you may delete this.

II. Preliminaries and Related Work

A. Reinforcement Learning and PPO

In Reinforcement Learning (RL), an agent interacts with an environment $\mathcal{E}$ over discrete timesteps. At each timestep t, the agent observes a state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$ according to a policy $\pi(a|s)$, and receives a reward $r_t \in \mathbb{R}$ from the environment. The goal is to maximize the expected return, defined as the cumulative discounted reward:

$$R_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right]$$

where $\gamma \in [0, 1]$ is the discount factor.

Proximal Policy Optimization (PPO) is a policy-gradient method commonly used for continuous control tasks due to its stability and reliability [2]. PPO alternates between collecting trajectory data (rollouts) on-policy and optimizing the policy parameters θ using the following clipped surrogate objective:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is the advantage estimate, and ϵ is a hyperparameter controlling the clipping range.

B. Sim-to-Real Gap and Domain Randomization

Training an RL agent entirely in simulation avoids real-world risks and costs. However, discrepancies between the simulator and the real-world environment, known as the *sim-to-real gap*, can lead to significant performance drops upon deployment. This gap arises from inaccuracies in modeling the real-world dynamics.

Domain Randomization (DR) is a technique used to address this gap by training the policy on a distribution of environment parameters rather than a single fixed configuration. Formally, the simulator samples its parameters $\phi \sim p(\phi)$ from a predefined distribution $p(\phi)$. The objective of the policy is then to maximize the expected return over the parametrized environment:

$$J(\pi) = \mathbb{E}_{\phi \sim p(\phi)} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$$

where $\tau = (s_0, a_0, r_0, s_1, \dots)$ denotes a trajectory generated by the policy π in an environment parameterized by ϕ.

By exposing the policy to varied dynamics during training, DR encourages robustness, making the learned strategy more likely to generalize effectively to the real world [3].

III. Hopper Environment and Reality Gap Setup

A. State and Action Spaces

We use the Hopper-v0 environment from OpenAI Gym, which simulates a one-legged robot with three actuated joints: thigh, leg, and foot. The state space is continuous, consisting of the robot's joint angles, joint angular velocities, and the x-position of the center of mass (if included). Since each joint angle and velocity is real-valued, there are typically 11 dimensions in the observation vector (depending on how the environment is defined). The action space, likewise, is continuous, corresponding to torques applied to each of the three joints. Thus, both the state and action spaces are real-valued.

B. Source vs. Target Dynamics

In the default (target) Hopper environment, the torso mass is around $3.53\,\text{kg}$, while the thigh, leg, and foot masses are roughly $0.9\,\text{kg}$, $0.9\,\text{kg}$, and $0.4\,\text{kg}$ respectively. We create a **source** variant by reducing the torso mass by $1\,\text{kg}$ (to $2.53\,\text{kg}$), simulating an imperfect model. The mismatch in torso mass alone is sufficient to negatively affect learned dynamics if the policy is only exposed to the incorrect mass during training, effectively representing the reality gap.

IV. Standard PPO Training and Performance

A. Training Pipeline

We use stable-baselines3 (SB3) [4] for the PPO implementation and proceed as follows:

1) **Environment Creation:** Instantiate either the source or target Hopper variant.
2) **Hyperparameter Setup:** Specify learning rate, clipping range, entropy coefficient, and other PPO parameters.
3) **Policy Training:** Roll out the agent for a fixed number of timesteps (i.e. 700000), optimizing the PPO objective.
4) **Evaluation:** Test the final policy over multiple episodes (i.e. 50) and record the average return and standard deviation.

B. Comparison Across Source and Target

We train two baseline policies:

- A *source-only* policy: trained in the environment with a shifted torso mass.
- A *target-only* policy: trained in the environment with the correct mass.

Both are evaluated under three scenarios:

1) *source* $\rightarrow$ *source*: The same domain for training and testing, generally yielding high performance if the agent has adapted well to the training environment.
2) *source* $\rightarrow$ *target*: Transfer from a mismatched (lower-mass) environment to the correct one, typically resulting in performance degradation.
3) *target* $\rightarrow$ *target*: A scenario with perfect modeling, representing an upper bound for target performance.

As shown in Figure 1, when trained exclusively on the lower-mass (source) Hopper, the agent starts with a very low reward (near 0) but rapidly learns a viable gait, surpassing 1,000 reward by around the 10th evaluation. After this initial improvement, the returns remain somewhat volatile but tend to stabilize in the 1,500–1,800 range. This indicates that the policy is capable of learning an effective hopping strtegy in the modified environment; however, as discussed below, it may not transfer well to the higher-mass (target) Hopper without further adaptation.

We expect that training solely on the source domain leads to a notable drop in reward when switching to the target environment. Such expectations are confirmed by empirical data in Figure 2 where we compared the best source and target models obtained. This behavior occurs because the policy never encounters the correct torso mass during training and consequently develops suboptimal behaviors for the real (target) dynamics. On the other hand, training directly in the target environment yields strong results, but in a real-world setting, collecting such data would be expensive, risky, or time-consuming. Hence, the desire is to train primarily in the source domain generalizing as much as possible in order to achieve a robust transfer.

V. Uniform Domain Randomization (UDR) for Hopper

A. General Idea of UDR and training setup

Domain Randomization (DR) allows to avoid overfitting to the target environment, by sampling environment properties from a probability distribution at each episode. The main effort here needs to be spent on selecting the right amplitude of the range from which sampling the environment parameters. We can in fact intuitively understand that we want the agent to be able to generalize when facing with slightly different environment conditions, but at the same time we know that finding a clear pattern is essential to learn a valid policy, meaning that we have certain limitations when it comes to randomize the environment properties. For these reasons, we decided to select the edges of the uniform sampling distribution based on the value of the masses of each link. The best

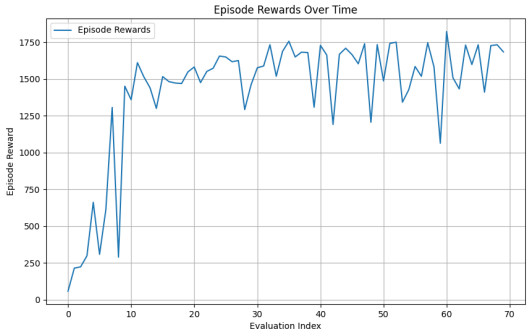

Fig. 1. Training curve of the source-only Hopper policy. Early episodes yield low returns, but the reward quickly increases beyond 1,000 by about the 10th evaluation. Over time, the policy continues to improve, generally stabilizing around 1,500–1,800 episode reward, though with noticeable fluctuations in performance.

Fig. 2. Evaluations of the best source and target models within the target environment context. Source model struggle when facing the torso mass gap, leading to poor cumulative reward

PPO hyperparameters (learning rate, clip range and entropy coeffient) are the same of the best model from the previous section. Instead of training exclusively with the $-1\,\mathrm{kg}$ torso offset (source), we randomize the other link masses (thigh, leg, foot) around their default values. Specifically, uniform samplin chooses each mass m_i from

$$m_i^{\text{sample}} \sim \mathcal{U}\big(m_i^0(1-\delta),\, m_i^0(1+\delta)\big)$$

where m_i^0 is the nominal mass of link i in the standard Hopper, and δ is the range fraction. The torso mass remains offset by 1 kg so that the *source* environment never exactly matches the real domain. Training on such distributions typically boosts *source* $\rightarrow$ *target* performance, although it may not equal *target* $\rightarrow$ *target* results. More in depth, we tried different δ values in the range $\{0.05, 0.10, 0.15, 0.20, 0.25\}$. We found that the best agent had a training $\delta = 0.20$, therefore we decided not to explore ranges wider than 0.25.

B. Training and Performances

In Figure 3, the reward results from the best model during training are depicted as the training progresses. Compared to the trend in Figure 1, there is an observable increase in noise, likely attributed to the randomization phase. Additionally, the policy achieves slightly lower reward values on average. This outcome is expected, as the stochastic nature of the environment parameters prevents the agent from overfitting to specific conditions. This added randomness ensures that the agent develops a more robust policy, capable of generalizing to a wider range of scenarios. The trade-off between higher rewards and generalization is evident here, emphasizing the balance introduced by the randomization strategy.

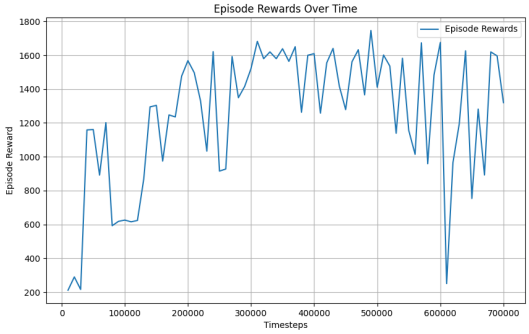

Fig. 3. Training curve of the source Hopper policy with UDR technique. We can notice much more noise caused by the randomization.

In Figure 4 we see an improvement provided by the best found model in the reward, with a mean reward slightly higher than 1,000 (compared to previous section outcomes, that had an average reward slightly lower than 1,000), however the gap with the upper bound provided by the target model remains marked and performances do not meet those ones.

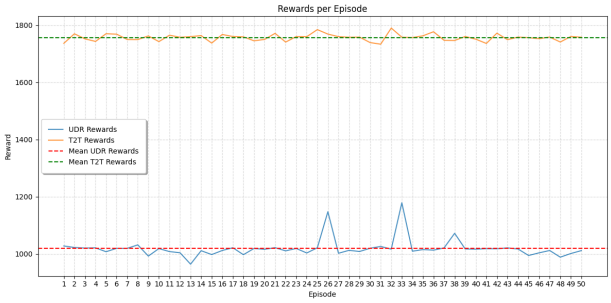

Fig. 4. Evaluations of the best source and target models. Here the source model was trained through UDR technique

C. Extended Analysis: Selective Randomization

Since using a single δ value across all three link masses (thigh, leg, foot) did not yield satisfactory transfer performance, we investigated whether randomizing one link at a time might provide a better alternative. Rather than performing a multi-dimensional search for all three link masses simultaneously (which would be time-consuming), we proceeded by tuning each mass separately and then combining the best values into a single setup. As before, we adopted the same PPO hyperparameters (learning rate, entropy coefficient, and clip range) identified in the standard PPO section.

Specifically, we conducted three sweeps, each focusing on one link at a time:

- *Sweep 1:* Vary $\delta_{\text{foot}} \in \{0.15, 0.20, 0.25\}$ while fixing $\delta_{\text{thigh}} = \delta_{\text{leg}} = 0$.
- *Sweep 2:* Vary $\delta_{\text{leg}} \in \{0.15, 0.20, 0.25\}$ while fixing $\delta_{\text{thigh}} = \delta_{\text{foot}} = 0$.
- *Sweep 3:* Vary $\delta_{\text{thigh}} \in \{0.15, 0.20, 0.25\}$ while fixing $\delta_{\text{leg}} = \delta_{\text{foot}} = 0$.

Each sweep was evaluated based on source-to-source average rewards. From these results, the top-performing values for each link were $\delta_{\text{foot}} = 0.20$, $\delta_{\text{leg}} = 0.20$, and $\delta_{\text{thigh}} = 0.25$. Table I summarizes these tests, including trials where each link is randomized in isolation (c_1, c_2, c_3 respectively setting only the best δ_{foot}, δ_{leg}, δ_{thigh}) and the case where all chosen values are combined (c_4).

TABLE I
UDR WITH DIFFERENT δ VALUES FOR FOOT, LEG, AND THIGH.

Config	S2S Mean	S2S Std	S2T Mean	S2T Std
c_1	1728.34	37.38	837.38	412.97
c_2	1715.93	5.01	712.86	76.94
c_3	1585.39	141.31	1255.89	153.79
c_4	1707.32	33.29	1131.15	149.06

Notably, randomizing only the thigh mass with $\delta_{\text{thigh}} = 0.25$ (c_3) led to the highest source-to-target average reward (1255.89) among the single-link trials. This finding seems to suggest that thigh segment plays a crucial role in the Hopper's forward propulsion and stability. Because the thigh contributes significantly to how the Hopper maintains balance and generates thrust, small improvements in thigh-mass generalization may yield disproportionately higher gaits in overall performance.

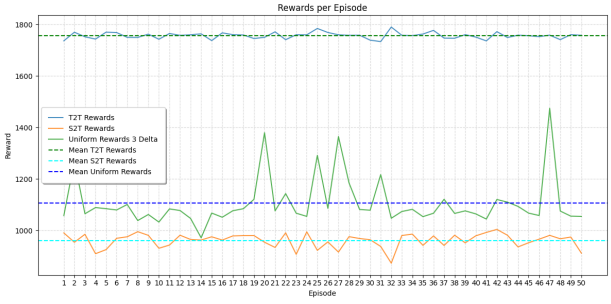

Fig. 5. Evaluation of the UDR model, considering for each mass the delta that performed better in the s2s evaluation, with respect to the t2t and the s2s models trained with PPO

Despite each link-specific δ value leading to good source-to-source performance when used individually, the combination of these values ($\delta_{\text{foot}} = 0.20$, $\delta_{\text{leg}} = 0.20$, and $\delta_{\text{thigh}} = 0.25$) yielded a lower average reward on the target environment (1131.15) than what might have been expected based on the best single-link randomizations. One plausible explanation is that each randomization introduces distinct variability in the robot's dynamics, so simultaneously randomizing all three links can compound or overlap these variations in a way that does not simply 'add up' constructively. As a matter of fact, individually optimal deltas may conflict when combined, thus preventing the policy from fully capitalizing on any one link's randomization scheme.

Another relevant observation concerns the standard deviation values in Table I. The trial with $\delta_{\text{foot}} = 0.2, \delta_{\text{leg}} = 0.0, \delta_{\text{thigh}} = 0.0$ displayed a notably large standard deviation of about 413, indicating that some runs (or episodes) achieved relatively high rewards while others were poor, suggesting instability in the learned policy. This could also highlight that the foot plays a crucial role in the hopper stability and control, so varying its mass drastically affects balance and locomotion efficiency, in particular when applied in the target environment. The agent struggles to generalize across these variations, leading to inconsistent gaits and a broader distribution of rewards. Moreover, this finding suggests that varying the foot mass alone is insufficient to bridge the reality gap (i.e., the torso mass difference between the source and target environments). Conversely, the setting $\delta_{\text{leg}} = 0.2, \delta_{\text{thigh}} = 0.0, \delta_{\text{foot}} = 0.0$ (c_2) exhibited a much lower standard deviation of approximately 77, suggesting a more consistent but ultimately lower overall average reward. Once all three deltas are combined, the standard deviation remains sizable (149.06), reflecting moderate variability in episode returns under the broader range of dynamics encountered. In practice, such variability can be undesirable if consistent performance is paramount, even if the mean reward improves. Thus, choosing which parameters to randomize, and how broadly, involves a balance between potentially higher average returns and the risk of larger variance in outcomes.

VI. Gaussian Domain Randomization

In addition to uniform sampling, we explored a Gaussian approach, where each mass is sampled from

$$m_i^{\text{sample}} = \max\{0, \mathcal{N}(m_i^0, m_i^0 \cdot \delta_i)\},$$

so that m_i^0 is the mean of the distribution, and $m_i^0 \cdot \delta_i$ is the standard deviation. Negative samples are clamped to 0 to preserve physical feasibility. The choice for this specific distribution was taken because of its different nature with respect to the uniform one. In fact, if sampling from a uniform distribution does not prioritize specific mass values, using a Gaussian probability function is more likely to provide values near the mean. Therefore we decided to select the standard values of the masses for the thigh, leg and foot as the mean value, then the standard deviation was tuned choosing appropriate δ values. The advantage of this approach is that we encourage the model to generalize over different values of each mass link, making also the agent experience occasional extreme conditions (high mass samples are possible, even though they are rare events).

As in the uniform case, we conducted three sweeps where only one link's δ_i was varied while the others stayed at zero, and then combined the best for each part.

A. Detailed Results and Observations

To systematically compare these Gaussian randomization configurations, we trained four models: three with each individual link mass randomized and one with all three link masses (thigh, leg, and foot) simultaneously randomized. Table II reports the source-to-source and source-to-target performance metrics for each configuration, showing that the final combined configuration (c_4) achieves notably higher source-to-target mean reward than any of the individual setups.

TABLE II
HOPPER GAUSSIAN DOMAIN RANDOMIZATION RESULTS. EACH CONFIGURATION (c_1–c_3) RANDOMIZES ONLY ONE LINK MASS, WHILE c_4 COMBINES ALL THREE.

Config	S2S Mean	S2S Std	S2T Mean	S2T Std
c_1	1681.69	18.37	997.36	170.42
c_2	1723.54	4.90	1061.00	43.75
c_3	1669.86	147.76	828.87	53.67
c_4	1663.72	14.10	1456.33	236.26

Here, c_1 corresponds to $\delta_{\text{foot}} = 0.25$ and zeros elsewhere, c_2 sets $\delta_{\text{leg}} = 0.35$, and c_3 randomizes only the thigh mass $\delta_{\text{thigh}} = 0.20$. Finally, c_4 merges those three parameters: $\delta_{\text{foot}} = 0.25$, $\delta_{\text{leg}} = 0.35$, and $\delta_{\text{thigh}} = 0.20$.

Also in this case, the foot configuration has higher standard deviation in the source-to-target leading to the same conclusions of the standard UDR case. The best single-mass configuration is the leg one c_2, achieving best s2t mean and best s2t standard deviation.

Fig. 6. Evaluations of the GDR model, considering for each mass the delta that performed better in the s2s evaluation, and the t2t and the s2s models trained with PPO

Combining Randomizations Improves Target Performance: While uniform domain randomization showed that combining multiple link randomizations could yield worse average returns in the target environment (in part due to cumulative variability), the Gaussian approach leads to the opposite effect in this setup. Configuration c_4 achieves the highest source-to-target mean (1456.33) among the four, surpassing each individual setting by a substantial margin. A plausible explanation for this discrepancy with the uniform case is that Gaussian sampling, which places more probability mass near the nominal values, provides a balanced mix of moderate and extreme variations for *all* three links simultaneously. This variety may help the policy develop a broader set of stabilization strategies that transfer better to the correct (target) torso mass. However, it also introduces a higher source-to-target standard deviation (236.26), showed also in 6, that could be reflecting the volatile nature of encountering occasional extreme values for multiple link masses at once. As we can see, when we used UDR, the variance was not surprisingly higher with respect to what we obtained during the single-mass delta applications, while when we used the Gaussian distribution, as mentioned, the variation was quite higher. This difference could be due to the fact that in uniform distributions variance is inherently high due to the equal probability across the entire range. In contrast, for Gaussian distributions, when perturbations for multiple masses are combined, their individual variances add up if the perturbations are independent. This additive property leads to a higher overall variance in the system when multiple deltas are applied together. Such variance highlights the inherent trade-off between improving mean performance and maintaining consistent returns.

Overall, these results emphasize that different sampling distributions (uniform vs. Gaussian) can produce distinct outcomes when multiple parameters are randomized. In this particular Gaussian scheme, combining the independently tuned deltas (c_4) ultimately yields the strongest transfer performance in terms of mean return, at the expense of increased variance.

VII. DISCUSSION OF RESULTS AND VARIANCE

A. The Variance Increase

As randomization ranges grow (or as we randomize each link's mass independently), the agent encounters an increasingly diverse set of training conditions. This diversity promotes a robust control strategy that can handle various mass configurations, thus raising the average reward in the real environment. However, encountering extremes (particularly in Gaussian-based sampling) can lead to policies that exhibit more variable behaviors when tested, yielding higher standard deviations. In practice, one must decide whether increased variance is acceptable if it comes with an improved mean return.

B. Careful hyperparameters tuning is necessary

Very broad parameter ranges can render the training problem more difficult, as the policy struggles to master widely differing dynamics. Consequently, too large a range may hurt average performance or cause slow convergence. Furthermore, higher variance might be undesirable if the robot's safe and reliable operation is crucial. Thus, domain randomization hyperparameters (distributions, ranges) must be carefully tuned to strike a balance between robustness and stability.

VIII. WALKER2D ENVIRONMENT

Beyond Hopper, we extended our investigation to the Walker2D-v4 environment, which features a two-legged bipedal robot that can walk forward by actuating six hinge joints. This environment is generally considered more complex

than Hopper due to its higher-dimensional state space (17 observations by default) and larger action space (6 torques), as detailed in the official Gym documentation.

A. Environment Characteristics and Dynamics

The Walker2D robot is comprised of seven main body parts: a single torso at the top and two parallel "legs", each one consisting of thigh, leg, and foot. The default observation space excludes the x-coordinate of the torso from the policy's inputs, yielding a 17-dimensional Box space of floating-point observations (positions and velocities). The action space is 6-dimensional where each dimension coorresponds to a torque applied at one of the six hinge joints (three in each leg).

Similar to Hopper, Walker2D terminates (i.e., issues a "done" signal) if the robot becomes *unhealthy*, meaning it either falls below a certain torso height range or exceeds a limit on torso angle. An episode also terminates after 1,000 timesteps by default. The reward structure primarily consists of three components: a forward reward proportional to the agent's velocity in the positive x-direction, a small penalty term that discourages large control inputs, and a *healthy reward* bonus for maintaining an upright posture.

B. Training Setup

In light of the higher dimensionality and complexity of Walker2D compared to Hopper, we increased the training duration to 1,000,000 timesteps per run (as opposed to 700,000 for Hopper). With the greater degrees of freedom and more intricate dynamics, Walker2D often requires additional training samples to achieve stable gaits and higher returns.

To create a sim-to-sim gap, we introduced a heavier torso mass in the target domain, adding 3.0 kg on top of the default Walker2D torso value while keeping the source domain at the original mass. This larger mismatch was chosen to amplify the effect of the reality gap, given preliminary experiments suggested that a smaller offset (e.g., 1 kg) did not sufficiently degraded performances. By adopting this more pronounced discrepancy, we could more clearly observe how a policy trained in the *source* domain (with default torso mass) would struggle upon transfer to the *target* domain (with significantly increased torso mass), also leading to greater variance in the source to target tests.

C. Observations and Learned Policy Performance

Following the same PPO training pipeline detailed earlier (but extending the number of training steps to 1,000,000 to accommodate Walker2D's additional complexity), we observed the performance summarized in Table III. Specifically, the best policy trained entirely on the *source* domain—with default Walker2D torso mass—achieved a mean episode return of about 1460 when evaluated on the same environment, alongside a standard deviation of approximately 204. However, when transferred to the *target* environment, which featured an additional 3 kg on the torso, the policy's mean return dropped to roughly 773 (with a standard deviation of about 168).

These results confirm that, although the agent acquires a viable gait under the original (source) torso mass, it struggles with the heavier (target) configuration—a more pronounced manifestation of the reality gap also noted in Hopper. While a discrepancy of 1 kg was enough to degrade performance there, the 3 kg mismatch in Walker2D heightened the issue. Although we hypothesize that the same domain-randomization approaches (uniform or Gaussian sampling of body-link parameters) used in Hopper could alleviate this drop, the larger state-action space and increased training costs of Walker2D leave a deeper exploration of these techniques for future work. Nonetheless, even these preliminary findings underscore that substantial discrepancies in torso mass can significantly undermine transfer performance in more complex locomotion tasks.

TABLE III
WALKER2D PERFORMANCE METRICS WITH DEFAULT MASS AS SOURCE DOMAIN AND +3.0 KG TORSO MASS AS TARGET DOMAIN.

Training Env.	Testing Env.	Mean Return	Std Return
Source	Source	1460.47	203.90
Source	Target	773.17	168.13

IX. UNIFORM AND GAUSSIAN DOMAIN RANDOMIZATION FOR WALKER2D

Also for the Walker2D, we explored the applicability of UDR and Gaussian Domain Randomization. As for the Hopper case, we initially applied the deltas to each mass individually in order to find, for each mass the one that provided better results in the source to source environment. Then, we performed one last training session where we applied the best delta found for each mass simultaneously. Given that the Walker2D has two legs, we opted to sample the same mass perturbation for both the left and right legs within each episode (e.g. sampled left foot mass equals sampled right foot mass). This decision was primarily motivated by preliminary studies, indicating that using the same sampled mass for both legs yielded better performance, likely due to the symmetrical dynamics it preserved, which may have facilitated learning and improved stability during training. Nevertheless by sampling different masses for the two legs would have increased the generalization of the model, but would have also required significantly more time for the training process to stabilize and yield good results. For what concerns the UDR we considered again 4 configurations: c_1 corresponds to $\delta_{\text{foot}} = 0.35$ and zeros elsewhere, c_2 sets only $\delta_{\text{leg}} = 0.35$, and c_3 only $\delta_{\text{thigh}} = 0.35$. Finally, c_4 merges those three parameters: $\delta_{\text{foot}} = 0.35$, $\delta_{\text{leg}} = 0.35$, and $\delta_{\text{thigh}} = 0.35$. Also in this case the training has been performed considering 1,000,000 timesteps.

The performance in terms of average reward on the source-to-target are very low with respect to the hopper case, and this can be due to the fact that 1,000,000 timesteps are not enough to train a model for the Walker that successfully transfers in the target environment (since in the performances in the source-to-source are definitely better). The best configuration

TABLE IV
WALKER2D UNIFORM DOMAIN RANDOMIZATION RESULTS. EACH CONFIGURATION (c_1–c_3) RANDOMIZES ONLY ONE LINK MASS, WHILE c_4 COMBINES ALL THREE.

Config	S2S Mean	S2S Std	S2T Mean	S2T Std
c_1	1590.71	418.89	401.68	121.77
c_2	1716.95	237.81	685.78	116.82
c_3	1418.24	194.16	552.22	46.81
c_4	1334.43	294.39	961.79	396.45

is the one that combines the 3 best deltas, obtaining a reward close to 1000. For what concerns the standard deviation of the first 3 configurations, in the s2t cases, it's lower with respect to the s2s. However, this does not necessarily indicate more stable performance, but rather it is likely due to the agent failing early in a consistent manner. The lower average rewards suggest that the agent is not learning effective policies, leading to shorter episodes that constrain variability. In contrast, in the final configuration, where the agent achieves higher rewards and episodes last longer, the deviation is larger. This suggests that as the agent learns better control strategies, performance varies more across episodes, resulting in a higher standard deviation.

We also tried the same approach for the Gaussian Domain randomization, with configurations c_1 corresponding to $\delta_{\text{foot}} = 0.3$ and zeros elsewhere, c_2 corresponding to $\delta_{\text{leg}} = 0.3$, and c_3 to $\delta_{\text{thigh}} = 0.25$. c_4 merges those three parameters: $\delta_{\text{foot}} = 0.3$, $\delta_{\text{leg}} = 0.3$, and $\delta_{\text{thigh}} = 0.25$, obtaining:

TABLE V
WALKER2D GAUSSIAN DOMAIN RANDOMIZATION RESULTS. EACH CONFIGURATION (c_1–c_3) RANDOMIZES ONLY ONE LINK MASS, WHILE c_4 COMBINES ALL THREE.

Config	S2S Mean	S2S Std	S2T Mean	S2T Std
c_1	1497.15	168.83	1097.37	397.65
c_2	1523.69	168.80	1087.27	394.95
c_3	1467.33	193.87	1221.84	357.84
c_4	1045.05	267.76	835.47	252.94

In this case the Gaussian distribution seems to provide better randomization than the uniform, since in the s2s case we obtain comparable results, but in the s2t case the mean rewards doubles the ones obtained in UDR. In the s2t though it has higher variance, but this should be inherent to longer test episodes due to better control strategies. Surprisingly, the c_4 configuration doesn't seem to generalize as good as in the hopper case, providing worse results than the Gaussian random sampling over the single masses. One explanation is that since the standard deviation is already very high for the single deltas (configurations $c_1 - c_3$), when we combine them together this could lead to extreme dynamics that might lead the walker to difficult adaptation and premature failure.

Overall, the Walker2D experiments reinforce our primary conclusions: mass discrepancies—particularly substantial ones—can significantly undermine policy performance. Domain Randomization techniques hold promise for bridging such a gap, but hyperparameter tuning (e.g., how wide to randomize, which body parts to randomize, and which distribution to use) becomes crucial, especially in higher-dimensional systems like Walker2D. This suggests that careful consideration of the randomization strategy is essential for achieving optimal generalization and performance in more complex environments.

X. CONCLUSION

In this paper, we investigated sim-to-real transfer for the Hopper-v0 environment by artificially shifting the torso mass in the training (source) domain. Proximal Policy Optimization (PPO) agents trained exclusively in this inaccurate model see substantial drops in performance when transferred to the target environment, highlighting the so-called reality gap. We confirmed that training with domain randomization —both uniform and Gaussian— on the remaining link masses significantly improves source-to-targt returns. A deeper analysis showed that selectively tuning individual mass distributions could further enhance mean performance but also introduced larger standard deviations, reflecting the trade-off between robust adaptation and predictable policies.

Future research directions include:

- **Adaptive DR distributions:** iteratively refining sampling parameters based on partial real-world feedback or online adaptation.
- **Correlated randomization:** jointly varying parameters that are physically interdependent.
- **Curriculum randomization:** gradually expanding from narrower to wider parameter ranges during training.

By investigating multiple DR approaches and distribution choices, we have shown that carefully designed randomization of an imperfect simulator can mitigate the reality gap for RL-based robotic control.

ACKNOWLEDGMENT

We thank the course instructors and teaching assistants for their guidance and support. No external funding agency was involved in this work.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[3] I. Peng, M. Andrychowicz, W. Zaremba, et al., "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018.

[4] A. Raffin, A. Hill, A. Gleave, et al., "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research*, 2021.