2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Symmetry Considerations for Learning Task Symmetric Robot Policies

Mayank Mittal*, Nikita Rudin*, Victor Klemm, Arthur Allshire, and Marco Hutter

***Abstract*— Symmetry is a fundamental aspect of many real-world robotic tasks. However, current deep reinforcement learning (DRL) approaches can seldom harness and exploit symmetry effectively. Often, the learned behaviors fail to achieve the desired transformation invariances and suffer from motion artifacts. For instance, a quadruped may exhibit different gaits when commanded to move forward or backward, even though it is symmetrical about its torso. This issue becomes further pronounced in high-dimensional or complex environments, where DRL methods are prone to local optima and fail to explore regions of the state space equally. Past methods on encouraging symmetry for robotic tasks have studied this topic mainly in a single-task setting, where symmetry usually refers to symmetry in the motion, such as the gait patterns. In this paper, we revisit this topic for goal-conditioned tasks in robotics, where symmetry lies mainly in task execution and not necessarily in the learned motions themselves. In particular, we investigate two approaches to incorporate symmetry invariance into DRL -- data augmentation and mirror loss function. We provide a theoretical foundation for using augmented samples in an on-policy setting. Based on this, we show that the corresponding approach achieves faster convergence and improves the learned behaviors in various challenging robotic tasks, from climbing boxes with a quadruped to dexterous manipulation.**

I. Introduction

Deep reinforcement learning (DRL) is becoming an important tool in robotic control. Without prior knowledge or any assumptions on the underlying model, these methods can solve complex tasks such as legged locomotion [1]–[3], object manipulation [4], [5], and goal navigation [6]. However, this very black-box nature of DRL does not leverage the knowledge of the symmetry in the task and often results in policies that are not invariant under symmetry transformations [7], [8]. This problem is not limited to the current DRL algorithms. Humans and animals also exhibit asymmetric execution of various tasks by, for example, always using the dominant hand or foot for tasks requiring higher dexterity. Robots, however, should avoid such limitations and achieve optimal task execution in all cases.

In robotics, we can think of symmetry at two levels: 1) motion execution, which pertains to the behavior of mirrored body parts during periodic motions, and 2) task execution, which pertains to the behavior used to achieve mirrored objectives. This distinction is crucial since achieving symmetry in task execution does not necessarily imply or demand symmetry in motion execution. To illustrate, consider tasks for quadrupedal locomotion (Fig. 1). A typical locomotion task may display both symmetries by learning a trotting gait for all commanded directions [2]. However, when faced with the challenge of climbing a tall box, the robot needs to deviate from symmetry at the motion level [6]. Nevertheless, it can still maintain symmetry at the task level; for instance, climbing a box in front of or behind the robot is considered equivalent. While we anticipate that behaviors for symmetrical goals will exhibit similarities, the solutions obtained using DRL are not. Usually, the trained policies exploit the behavior learned for only one of the goals. For instance, instead of climbing the box backward, the robot may first turn around and then ascent the box. Unfortunately, this behavior consumes more time and energy, rendering it sub-optimal. During the learning process, once an asymmetry in a behavior arises, it tends to get magnified with further training. Hence, it is important to incorporate symmetry considerations inherent to the task into DRL to learn superior and more efficient behaviors.

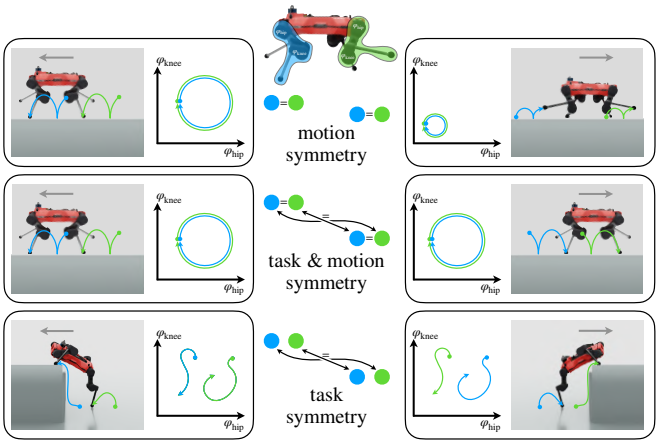


Fig. 1: Motion and task symmetry for quadrupeds. While motion symmetry involves similar movements of the legs, it does not guarantee that the robot behaves the same when commanded different goals (walking forward and backward). In contrast, task symmetry ensures consistent behaviors for such goals, potentially resulting in periodic symmetric motions for walking on flat ground or entirely asymmetrical aperiodic patterns for tasks such as climbing a box.

A. Related Work

Achieving symmetric motions has been of long-standing interest in character animation and, recently, robotics, where symmetrical gaits are usually considered more visually appealing and efficient. In model-based control, symmetric

* M. Mittal and N. Rudin contributed equally.

This work was supported by the Swiss National Science Foundation through the National Centre of Competence in Digital Fabrication (NCCR dfab). It has also received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme grant agreement No. 852044.

All authors are with the Robotic Systems Lab, ETH Zürich, 8092 Zürich, Switzerland. A. Allshire is with the University of Toronto, Canada. M. Mittal, N. Rudin, and A. Allshire are also with NVIDIA.

Contact: {mittalma, rudinn, vklemm}@ethz.ch

motions are typically enforced by hard-coding gaits [9] or by reducing the optimization problem by assuming perfect symmetry [10]. Similarly, in robot learning, the structure of the action space can be modified to ensure a symmetric policy. For instance, central pattern generation (CPG) for locomotion pushes the policy towards symmetrical sinusoidal motions [2], [11]. For periodic motions, motion phases as a function of time can also be used to learn policies for only half-cycles and repeat them during execution [12], [13]. Alternatively, based on the robot's morphology, the policy can control only half of the robot, with the other half simply repeating the selected actions [14]. While these ideas are simple, they constrain the policy by some explicit switching mechanism based on time or behavioral patterns.

To avoid this issue, recent works have looked at introducing invariance to symmetry transformations into the learning algorithm itself. Inspired by the success of data augmentation in deep learning, one way to induce this invariance is by augmenting the collected experiences with their symmetrical copies [12], [15]. An alternate approach is adding a penalty or loss function to the learning objective [7], [16]. It is also possible to design special network layers to represent functions with the desired invariance properties [8], [17]–[19]. Abdolhosseini *et al.* [12] compared these different approaches for bipedal walking characters. They showed that in many cases, using a symmetry loss function is more effective than data augmentation and performs at par with customized network architectures.

It is important to note that most of the above works have studied symmetry under the lens of symmetrical motions, or more specifically, gait patterns. This may not always be desired or feasible for a wider range of tasks, such as manipulating objects or climbing over surfaces, where symmetry appears at the task level and not on how symmetrically located actuators move. This paper aims to revisit the idea of symmetry from this task perspective and understand its efficacy on different real-world robotic problems.

B. Contributions

We investigate the notions of symmetry in DRL for goal-conditioned tasks. Specifically, we explore two approaches for embedding symmetry invariance into on-policy RL: data augmentation and mirror loss function. While these methods have previously appeared in literature, their applications have primarily centered around walking animated characters, rather than robotic tasks with goal-level symmetries. Our analysis aims to highlight often-overlooked intricacies in the implementations of these approaches. In particular, we discuss the ineffectiveness of naive data augmentation and introduce an alternate update rule that helps stabilize learning from augmented samples.

Our study compares the two approaches on four diverse robotic tasks: the standard cartpole, agile locomotion with a quadruped, object manipulation with a quadruped, and dexterous in-hand object manipulation. Notably, in contrast to prior work [12], our experimental findings show that data augmentation is the most effective way to achieve task-symmetrical policies. We demonstrate the sim-to-real transfer of policies learned with this method for agile locomotion using the platform ANYmal [20]. Although the robot is not perfectly symmetrical, we show that the policy trained using data augmentation results in nearly symmetrical behaviors for climbing boxes in front of and behind the robot.

II. Preliminaries

A. Reinforcement Learning

This work considers robotic tasks modeled as multi-goal Markov Decision Processes (MDPs) with continuous state and action spaces. For notational simplicity, we consider the goal specification a part of the state definition. We denote an MDP $\mathcal{M}$ as $(\mathcal{S}, \mathcal{A}, T, r, \gamma, \rho_0)$, where the symbols follow their standard definitions [21]. Our goal is to obtain a policy π that maximizes the expected discounted reward, $J(\pi) = \mathbb{E}_{\tau\sim p_\pi(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right]$, where the trajectory $\tau = (s_0, a_0, s_1, a_1, s_2, \dots)$ is sampled from $p_\pi(\tau)$ with $s_0 \sim \rho_0(\cdot),\ a_t \sim \pi(\cdot|s_t)$, and $s_{t+1} \sim T(\cdot|s_t, a_t)$. As before, we employ the definitions from [21] for the state-action value function $Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, \dots}\left[\sum_{l=0}^{\infty}\gamma^l r(s_{t+l}, a_{t+l})\right]$, the value function $V^\pi(s_t) = \mathbb{E}_{a_t}\left[Q^\pi(s_t, a_t)\right]$, and the advantage function $A^\pi(s_t, a_t) = A^\pi_t = Q^\pi(s_t, a_t) - V^\pi(s_t)$.

In DRL, the total expected reward can only be estimated through trajectories collected by executing the current policy π_{θ_k}, where θ_k are the policy's parameters at the learning iteration k. Following this, modern policy gradient approaches, such as TRPO [22] and PPO [23], use importance sampling to rewrite the policy gradient as:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim p_{\pi_{\theta_k}}}\left[\sum_{t=0}^{\infty}\eta_t(\theta) A_t^{\pi_{\theta_k}} \nabla_\theta \log \pi_\theta(a_t|s_t)\right],$$
$$\text{where } \eta_t(\theta) = \frac{p_{\pi_\theta}(s_t, a_t)}{p_{\pi_{\theta_k}}(s_t, a_t)} = \frac{p_{\pi_\theta}(s_t)}{p_{\pi_{\theta_k}}(s_t)}\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}. \quad (1)$$

In practice, the term $\frac{p_{\pi_\theta}(s_t)}{p_{\pi_{\theta_k}}(s_t)}$ is computationally intractable. However, it can be neglected by assuming the divergence between the policy distributions π_θ and π_{θ_k} is sufficiently small [22]. In PPO, this is achieved by using a clipped surrogate loss, $\mathcal{L}^{\text{PPO}}(\theta)$ [23]. Additionally, the value function is fitted using a supervised learning loss.

B. MDP with Group Symmetries

For an MDP $\mathcal{M}$ with symmetries, a set of transformations exists on the state-action space, such that the reward function and transition dynamics are invariant to them [24], [25]. More formally, we define a symmetric MDP with an N-fold symmetry if it contains a set of symmetric transformations $\mathcal{G} = \cup_k G_k = \{g_0, g_1, g_2, \dots g_{N-1}\}$, where $g_0 := (\mathbb{I}, \mathbb{I})$ is the identity transformation, and $g_i := (L_{g_i}, K_{g_i}), \forall i \in \{1, \dots, N-1\}$ are distinct non-identity transformations. The operators $L_g : \mathcal{S} \to \mathcal{S}$ and $K_g : \mathcal{A} \to \mathcal{A}$ can be seen to define similar transformations but in different spaces.

III. Approaches for Symmetry in RL

In literature, there are three main ways to incorporate symmetry into DRL: 1) using a symmetry loss function, 2) performing data augmentation, and 3) designing specialized network architectures. While the first two approaches only approximate the symmetry equivariance, specialized networks tend to guarantee it by embedding the equivariances into the layers themselves. However, this constrains the policy to always be equivariant, which can be detrimental in robotic applications since robots are not perfectly symmetrical. Additionally, perfectly symmetrical policies struggle with neural states, where $s = L_g[s], \forall g \in \mathcal{G}$, unless the environment introduces its own bias [26]. For instance, consider a quadruped starting to walk from a stance gait. A symmetric policy cannot lift the right front foot to take the first step since that means the other feet should also be raised under $L_g[s]_{g\in\mathcal{G}}$. However, this is not possible since $\pi(s) \neq \pi(L_g[s]), \forall g \in G - \{g_0\}$.

In practice, we only want to encourage the policy to learn similar behaviors for equivalent goals while letting it adapt the individual actuation or motion-level commands to deal with the asymmetries in the robot's design and neutral states. Keeping this in mind, we mainly look at the symmetry loss function and data augmentation approaches.

A. Using Mirror Loss Function

In the method proposed by Yu *et al.* [7], they add an explicit auxiliary loss to the learning objective that penalizes asymmetricity in the policy. Based on this approach, we can write the policy learning objective for all symmetry transformations in $\mathcal{G}$ as:

$$\mathcal{L}(\theta) = \mathcal{L}^{\text{PPO}}(\theta) + w \sum_{g\in\mathcal{G}} \mathcal{L}_g^{\text{sym}}(\theta), \text{where} \tag{2}$$

$$\mathcal{L}_g^{\text{sym}}(\theta) = \mathbb{E}_{\tau\sim p_{\pi_{\theta_k}}} \left[\sum_{t=0}^{\infty} \| K_g[\pi_\theta(s_t)] - \pi_\theta(L_g[s_t]) \|_2^2 \right], \tag{3}$$

and w is a scalar hyperparameter that governs the trade-off between minimizing the RL objective and the symmetry loss. Tuning this parameter w depends on the task and can adversely affect the training if set to a high value. Although not explicitly mentioned in prior works [7], during implementation, the quantity $K_g[\pi_\theta(s)]$ is treated as a label and is not back-propagated through, despite its differentiability.

From an intuitive standpoint, the symmetry loss (Eq. 3) encourages the policy to be symmetrical over its entire state-action spaces. However, achieving this objective can be challenging in high-dimensional problem spaces.

B. Symmetry-Based Data Augmentation

Data augmentation is commonly used in deep learning to make networks invariant to visual or geometrical transformations [27], [28]. A natural approach for symmetry augmentation within RL is augmenting the collected trajectories with their symmetrical copies [12]. However, this results in having to evaluate $\pi_{\theta_k}(\cdot|\cdot)$ and $A^{\pi_{\theta_k}}(\cdot,\cdot)$ in Eq. 1 on samples not generated from the rollout policy. Computing these quantities using such "off-policy" samples can introduce high variance in the gradients and diminish the method's effectiveness [12].

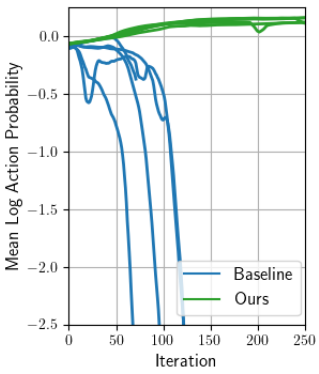


Fig. 2: The log action probabilities computed using baseline (Eq. 1) and our proposed (Eq. 6) approaches. We plot the mean obtained over the symmetry-augmented samples from each training iteration. The plot shows 5 runs with different seeds for the Cart-Pole task. The baseline method leads to training instabilities caused by low action probabilities. Meanwhile, our approach maintains stable convergence for all runs.

To deal with this issue, we approach symmetry augmentation from another perspective. At iteration k, let us construct policies $\pi^g_{\theta_k}$, such that $\pi^g_{\theta_k}(K_g[a]|L_g[s]) = \pi_{\theta_k}(a|s), \forall g \in \mathcal{G}, s \in \mathcal{S}, a \in \mathcal{A}$. Based on these augmented policies, we can write the RL objective for π_θ (Eq. 1) as learning from trajectories collected from these policies, *i.e.*, $\tau^g = (s^g_0, a^g_0, \dots)$:

$$\nabla_\theta J(\pi_\theta) = \sum_{g\in\mathcal{G}} \mathbb{E}_{\tau^g\sim p_{\pi^g_{\theta_k}}} \left[\sum_{t=0}^{\infty} \eta^g_t(\theta) A^{\pi^g_{\theta_k}}_t \nabla_\theta \log \pi_\theta(a^g_t|s^g_t) \right],$$

$$\text{where } \eta^g_t(\theta) = \frac{p_{\pi_\theta}(s^g_t, a^g_t)}{p_{\pi^g_{\theta_k}}(s^g_t, a^g_t)} = \frac{p_{\pi_\theta}(s^g_t)}{p_{\pi^g_{\theta_k}}(s^g_t)} \frac{\pi_\theta(a^g_t|s^g_t)}{\pi^g_{\theta_k}(a^g_t|s^g_t)}. \tag{4}$$

In data augmentation, the samples are collected by rolling out π_{θ_k} and not $\pi^g_{\theta_k}$, *i.e.*, $\tau^g = (s^g_0, a^g_0, \dots) = (L_g[s_0], K_g[a_0], \dots)$ with $s_t, a_t \sim p_{\pi_{\theta_k}}(s_t, a_t), \forall t \geq 0$.

Additionally, for the policies $\pi^g_{\theta_k}$ and the symmetric MDP $\mathcal{M}$, it can be shown that $\forall s \in \mathcal{S}, a \in A, g \in \mathcal{G}$:

$$A^{\pi^g_{\theta_k}}(L_g[s], K_g[a]) = A^{\pi_\theta}(s,a) \neq A^{\pi_\theta}(L_g[s], K_g[a]), \text{and}$$
$$p_{\pi^g_{\theta_k}}(L_g[s]) = p_{\pi_{\theta_k}}(s) \neq p_{\pi_{\theta_k}}(L_g[s]). \tag{5}$$

Thus, using Eq. 4 and Eq. 5, we obtain:

$$\nabla_\theta J(\pi_\theta) = \sum_{g\in\mathcal{G}} \mathbb{E}_{\tau\sim p_{\pi_{\theta_k}}} \left[\sum_{t=0}^{\infty} \frac{p_{\pi_\theta}(L_g[s_t])}{p_{\pi_{\theta_k}}(s_t)} \frac{\pi_\theta(K_g[a_t]|L_g[s_t])}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t) \nabla_\theta \log \pi_\theta(K_g[a_t]|L_g[s_t]) \right]. \tag{6}$$

Comparing Eq. 6 to simply applying Eq. 1 on augmented samples, we can see that the denominator of the action probability ratio are different. Using Eq. 1, we would get $\frac{\pi_\theta(K_g[a_t]|L_g[s_t])}{\pi_{\theta_k}(K_g[a_t]|L_g[s_t])}$, while with Eq. 6, we have $\frac{\pi_\theta(K_g[a_t]|L_g[s_t])}{\pi_{\theta_k}(a_t|s_t)}$. In other words, Eq. 6 keeps the action probability of the original samples. In contrast, we would need to compute the action probability for augmented samples for the other case. This difference is crucial since $\pi_{\theta_k}(K_g[a_t]|L_g[s_t])$ can be arbitrarily small for not perfectly symmetric policies, leading to instabilities in the training, as shown in Fig. 2.

However, even with the above change, the issue with computing the probability ratio $\frac{p_{\pi_\theta}(L_g[s_t])}{p_{\pi_{\theta_k}}(s_t)}$ remains unresolved. It can only be disregarded if the constructed policies $\{\pi^g_{\theta_k}\}_{g\in\mathcal{G}}$ are sufficiently close to the policy $\pi^g_{\theta_k}$ used for generating rollouts. While this may not hold for any policy π_{θ_k}, from our experiments in Sec. IV-D, we find that the probability ratio term can be ignored in the case of randomly initialized policies with sufficiently small weights and bounded updates. However, the ratio is important when policies are initialized non-symmetrically.

Conceptually, we can interpret Eq. 6 as follows: When we observe a high return for a specific action a taken from a given state s, we want to boost the likelihood of choosing that action in the future. In the case of symmetry, we also want to amplify the likelihood of the equivalent action $K_g[a]$ taken from the equivalent state $L_g[s]$.

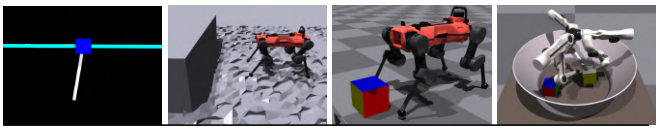
IV. Experiments and Results

A. Tasks

We consider four tasks, implemented using NVIDIA Isaac Gym [29], with inherent task symmetry (Fig. 3):

- *CartPole*: A classic control environment where the goal is to balance a pole attached by an unactuated joint to a cart. The input to the system is the desired cart velocity. As a reward, the agent receives an L-1 penalty between the pole's current and upright position.
- *ANYmal-Climb*: An agile quadrupedal locomotion task from [6], where the quadruped ANYmal [20] needs to reach a target pose on a box over a defined time. The agent observes its state along with a robot-centric height map and receives a sparse delayed reward signal.
- *ANYmal-Push*: A loco-manipulation task where the robot needs to push a cube to a desired position. The cube's initial and target positions are spawned radially around the robot. The agent observes the robot's and object's states and receives a dense tracking reward.
- *Trifinger-Repose*: An in-hand cube reposing task for the Trifinger platform [30]. The agent needs to pick the cube from the table and manipulate it to its desired pose. The task setup is similar to that in [4].

The two quadrupedal tasks use curriculums to guide the training. For the climbing task, we use an initial move-in-direction reward that encourages the robot to move toward the target pose (phase A). This reward is later removed so that the robot can optimize its motion freely (phase B), as done in [6]. Once the robot starts climbing the box successfully, we randomize its initial orientation (phase C). Instead of always facing the boxes (yaw $= 0$), the orientation is sampled uniformly with yaw $\in [-\pi, \pi]$. Note that this curriculum intervention is necessary to achieve effective climbing behaviors. When training policies with randomized orientations from the beginning, they converge to a sub-optimal sideways climbing motion, which fails to solve the task for higher boxes. For the *ANYmal-Push* task, the curriculum moves the cube target further away as the robot pushes the cube successfully.



Task		Space	Transformations			
CartPole	$\mathcal{S}$	$(\dot{x}, \theta, \dot{\theta})$	$(\dot{x}, \theta, \dot{\theta}), (-\dot{x}, -\theta, -\dot{\theta})$			
	$\mathcal{A}$	$(\dot{x}_{\text{des}})$	$(\dot{x}_{\text{des}}), (-\dot{x}_{\text{des}})$			
ANYmal-Climb	$\mathcal{S}$	$\mathbb{R}^{282}$	Identity,	reflect-x,	reflect-y,	$\curvearrowright 180°$
	$\mathcal{A}$	$\mathbb{R}^{12}$	Identity,	reflect-x,	reflect-y,	$\curvearrowright 180°$
ANYmal-Push	$\mathcal{S}$	$\mathbb{R}^{51}$	Identity,	reflect-x,	reflect-y,	$\curvearrowright 180°$
	$\mathcal{A}$	$\mathbb{R}^{12}$	Identity,	reflect-x,	reflect-y,	$\curvearrowright 180°$
Trifinger-Repose	$\mathcal{S}$	$\mathbb{R}^{41}$	Identity,	$\curvearrowright 120°$,	$\curvearrowright 240°$	
	$\mathcal{A}$	$\mathbb{R}^{9}$	Identity,	$\curvearrowright 120°$,	$\curvearrowright 240°$	

Fig. 3: We consider four robotic tasks: a continuous cart-pole, quadruped climbing a box, quadruped manipulating a cube, and in-hand cube reposing. In the table, we specify their state and action spaces along with the available symmetry transformations.

B. Metrics

Prior work [12] uses metrics that typically characterize the gait symmetricity. However, this does not serve as a proper measure for a policy's symmetricity during task execution. For instance, in the *ANYmal-climb* task, we do not require that the front and back legs follow similar trajectories, but rather that when climbing forward and backward, the front legs behave similarly to the back legs, respectively.

Thus, we use two metrics that directly characterize the policy's performance in the task and measure its symmetricity: 1) the *average episodic return*, which is the undiscounted reward accumulated by the policy over an episode, and 2) the *symmetry loss* from Eq. 3, which measures the discrepancy in the policy for equivalent state-action pairs.

C. Training Performance

We compare PPO with symmetry loss, symmetry augmentation, and a combination of both against the standard version of the algorithm [23]. For PPO with symmetry loss, we consider different weights w to understand its implications. We use the weight from the best policy for the combined symmetry augmentation and loss method.

From Fig. 4, we observe that PPO with symmetry augmentation obtains the highest return and fastest convergence while having a low symmetry loss. Optimizing the symmetry loss directly helps induce symmetry but comes at the cost of performance and slower convergence. Increasing the weight w reduces the symmetry loss but hinders learning as the gradients from the losses in Eq. 2 compete against each other.

Additionally, for the ANYmal-Climb task, we can notice how different methods recover once phase C begins. At the start of this phase, the sudden change in the robot's orientation causes all the policies to fail since they now need to perform the climbing motion in different directions. The policy training with symmetry augmentation recovers nearly immediately as it is inherently symmetric from being trained on other orientations through the augmented samples. It must only adapt to intermediate orientations not previously seen in the earlier phases. On the other hand, policy training with the vanilla PPO takes much longer to recover and converges to

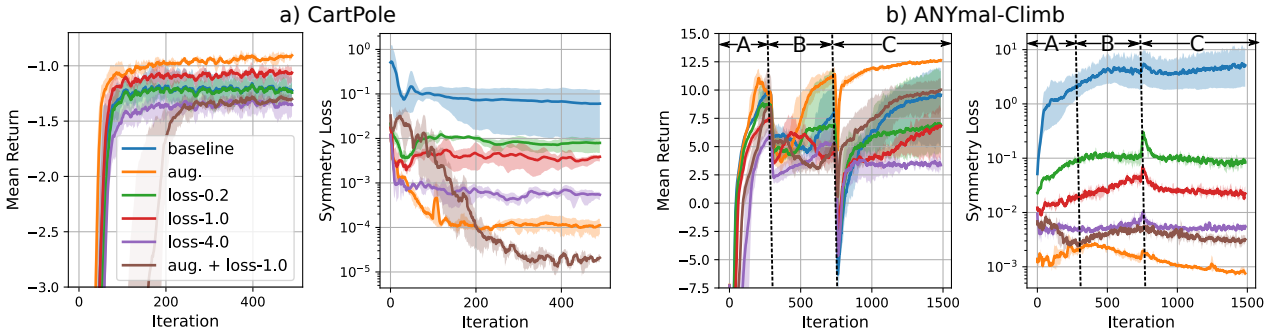


Fig. 4: Comparison of different methods for the CartPole and ANYmal-Climb tasks – vanilla PPO (baseline), PPO with symmetry augmentation (aug.), PPO with symmetry loss (loss-w), and a combination of the two. We plot the mean and standard deviation over three seeds. For the ANYmal-Climb task, we use a curriculum denoted as phases A, B, and C in the plot. We observe that symmetry augmentation yields the best performance consistently over all the tasks.

a different behavior (Sec. IV-F). Lastly, the policies trained using symmetry loss do not consistently recover in this phase.

Notably, the symmetry loss weighs symmetricity equally for all equivalent state actions. During training, the policy explores new actions for each symmetric state independently. If better actions are found for one of the states, the symmetry loss will push the policy to adopt equivalent actions for all equivalent states without considering the respective rewards. On the other hand, the augmentation approach will push the policy towards the best-performing actions since all transitions are compared to the same value function. Interestingly, using both symmetry loss and augmentation does not necessarily improve the performance or convergence, showing that symmetry augmentation does not benefit from the additional gradients provided by the loss.

D. Effect of network initialization

As discussed in Sec. III-B, symmetry augmentation assumes that the rolled-out policy is sufficiently symmetric, and hence, the slightly off-policy samples do not cause issues during training. A symmetric policy is expected to maintain that characteristic throughout training. However, when training commences from an arbitrary policy, there is no guarantee that it will converge to exhibit symmetric behaviors. To assess the severity of this problem, we compare the training of policies initialized with randomized weights drawn from a uniform distribution with varying scales.

For small weights, the actions from the policy are typically small as well, and as such, the policy is roughly equivalent to its symmetric counterparts. More concretely, for Gaussian distributions, policies $\pi_\theta(a|s)$ and $\pi_\theta(L_g[a]|K_g[s])$ are similar for small means and large enough standard deviation. With larger weights, the disparity between the two distributions increases, and they diverge from each other.

Fig. 5 shows that, indeed, the scale of initial weights influences the performance and symmetricity of the policies trained with data augmentation. Higher weights lead to lower performance and higher symmetry loss. Since directly optimizing over the symmetry loss does not assume any symmetricity of policy, we speculate that it is not affected by the initialization effect, and combining both augmentation and loss approaches can help recover the symmetricity even when the policy is initialized with high weights. Our findings,

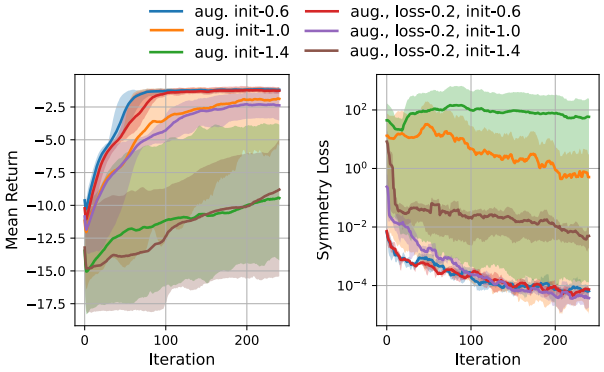


Fig. 5: Effect of network initialization scales (init-n) for the CartPole task. We plot the mean and standard deviation over three seeds. Symmetry augmentation (aug.) struggles when initialized weights are high. Adding a small symmetry loss helps mitigate the issue but does not improve the performance.

shown in Fig. 5, affirm that using a small symmetry loss coefficient greatly enhances the symmetricity of policies initialized with high weights. However, it is worth noting that this enhancement does not translate into improved performance compared to policies with low-weight initializations.

E. Evaluation of Symmetry in Learned Behaviors

To evaluate the performance of policies trained with and without symmetry augmentation, we create equivalent versions of each task and compute their total episodic return for each equivalent goal. For example, in the *ANYmal-Climb* task, we compare the episode returns for the goal of climbing a box forward and backward. For symmetric policies, the variation between the obtained returns for each goal should be low. Table I shows that for all the tasks, policies trained with augmentation consistently achieve higher average returns while having much lower variation in the returns between symmetric versions of the task. This result shows that learning with symmetry augmentation does lead to more optimal and symmetrical behaviors.

F. Qualitative Behavior Analysis

Finally, we describe the different behaviors learned by the policies for all tasks. We refer the reader to the supplementary video for more details.

Environment	Vanilla-PPO		PPO + aug.	
	Return	Variation	Return	Variation
CartPole	-2.507	0.353	**-1.928**	**0.003**
ANYmal-Climb	15.544	1.022	**17.462**	**0.124**
ANYmal-Push	16.331	2.255	**18.373**	**0.424**
Trifinger-Repose	2153.343	75.752	**2285.125**	**7.884**

TABLE I: We take a set of equivalent goals for each task and report the average episodic returns over 500 runs for each goal. The variation is the maximum difference in the returns between equivalent goals. Since rewards are symmetric, a higher variation implies less symmetric behavior between equivalent goals.

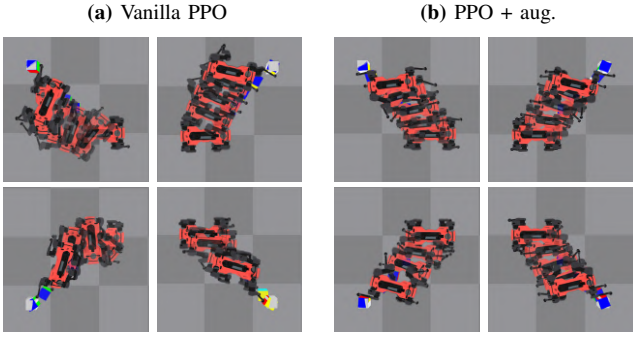


Fig. 6: Observed trajectories for equivalent goals in the ANYmal-Push task. Using data augmentation, the behavior is more symmetrical and the robot uses all of its legs for manipulation.

1) CartPole: Even for this relatively simple task, the behavior of policies trained without augmentation depends on where the pole is initialized. When the pole starts flat on the right side, the policy immediately moves the cart to spin the pole upwards. For the same position on the left, the policy lets it swing towards the other side first, leading to sub-optimal task returns. Policies trained with augmentation exhibit equally optimal behavior from both sides.

2) ANYmal-Climb: Policies trained without augmentation usually learn to climb only in one direction, always using the same leg first. When the robot is initialized in another direction, the policy prefers to turn on the spot before climbing. Since we set the initial and target orientations as the same, the policy turns again on the box to reorient itself. This leads to sub-optimal policies that turn twice instead of directly climbing backward. Training with augmentation mitigates this issue and the policies can climb forward and backward while using any of the legs to initiate the climbing.

3) ANYmal-Push: In this loco-manipulation task, the asymmetry in the learned policy with vanilla-PPO is more prominent since the robot uses only some of its legs for walking while the others for manipulating the object. Regardless of the uniform sampling of the object and its target around the robot, policies typically push the object with only two of its limbs and turn around to use only those two limbs for manipulation (Fig. 6). With symmetry augmentation, the robot uses all the limbs depending on whichever is closest to the object. It does this without any hand-crafted rewards to encourage a certain end-effector to move towards the object.

4) Trifinger-Repose: The policies trained without augmentation learn different finger gaits for rotationally equivalent goals. For example, the robot may flip the cube on the table before picking it up, while sometimes directly picking it up. In contrast, policies trained with augmentation produce the same pattern for rotationally similar goals and also complete the task faster.

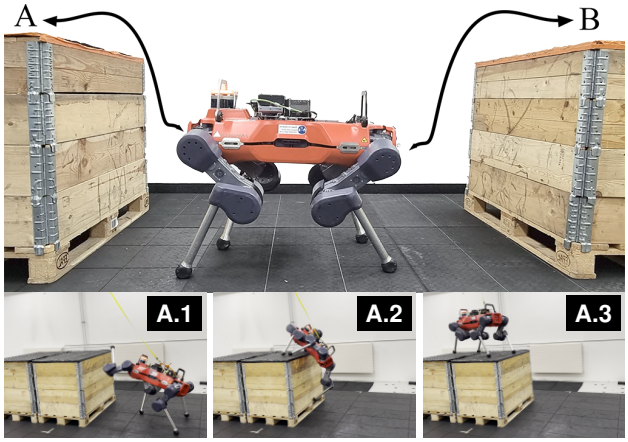


Fig. 7: Hardware deployment for the ANYmal-Climb task. The panel below shows the execution of the policy trained with symmetry augmentation to reach A. Please check the supplementary video for comparisons with behaviors obtained using vanilla-PPO.

G. Hardware Deployment

We conduct hardware deployment on ANYmal-D for the ANYmal-Climb task (Fig. 7). We find that policies from vanilla-PPO result in fast re-orienting behaviors that often cause perception failures and missteps. In contrast, policies trained with augmentation avoid these unnecessary rotations and display more predictable and robust behaviors. It is worth highlighting that even though the real robot is not perfectly symmetrical (uneven payload and wear-and-tear of the actuators), the policies trained with augmentation are resilient to these asymmetries and achieve successful box climbing maneuvers. One possible explanation for this success lies in the approach's emphasis on encouraging symmetry while allowing the policy to adapt naturally to the robot's asymmetries during training.

V. Discussion

We investigated two approaches for inducing symmetry invariance in on-policy DRL methods for goal-conditioned tasks. We presented an alternate update rule for symmetry-based data augmentation that helps stabilize the learning in practice. We compared the two approaches on various robotic tasks and showed how data augmentation leads to faster convergence with virtually symmetric and more optimal policies. Through hardware deployment for the quadrupedal agile locomotion task, we demonstrated that the policy learned with data augmentation transfers well even when the hardware is not perfectly symmetrical.

While this work mathematically motivates and empirically justifies the importance of initializing with small weights for data augmentation, a more rigorous treatment is for future work. Further investigation is also needed to understand how to perform augmentation when the symmetry in the MDP and the transformations are not explicitly available. For instance, for the latent vector obtained from an autoencoder.

REFERENCES

[1] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, 2022.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[3] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *Robotics: Science and Systems*, 2021.

[4] A. Allshire, M. Mittal, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg, "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[5] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, *et al.*, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[6] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2497–2503.

[7] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Trans. Graph.*, vol. 37, no. 4, jul 2018.

[8] E. Van der Pol, D. Worrall, H. van Hoof, F. Oliehoek, and M. Welling, "Mdp homomorphic networks: Group symmetries in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4199–4210, 2020.

[9] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. van de Panne, "Locomotion skills for simulated quadrupeds," *ACM Transactions on Graphics*, vol. 30, no. 4, 2011.

[10] A. Majkowska and P. Faloutsos, "Flipping with Physics: Motion Editing for Acrobatics," in *Eurographics/SIGGRAPH Symposium on Computer Animation*, 2007.

[11] G. Bellegarda and A. Ijspeert, "Cpg-rl: Learning central pattern generators for quadruped locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 547–12 554, 2022.

[12] F. Abdolhosseini, H. Y. Ling, Z. Xie, X. B. Peng, and M. Van de Panne, "On learning symmetric locomotion," in *ACM SIGGRAPH Conference on Motion, Interaction and Games*, 2019, pp. 1–10.

[13] L. Liu, M. van de Panne, and K. Yin, "Guided learning of control graphs for physics-based characters," *ACM Transactions on Graphics*, vol. 35, no. 3, 2016.

[14] N. Rudin, H. Kolvenbach, V. Tsounis, and M. Hutter, "Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning," *IEEE Transactions on Robotics*, vol. 38, pp. 317–328, 2021.

[15] Y. Lin, J. Huang, M. Zimmer, Y. Guan, J. Rojas, and P. Weng, "Invariant transform experience replay: Data augmentation for deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6615–6622, 2020.

[16] M. Abreu, L. P. Reis, and N. Lau, "Addressing imperfect symmetry: a novel symmetry-learning actor-critic extension," *arXiv preprint arXiv:2309.02711*, 2023.

[17] R. Wang, R. Walters, and R. Yu, "Incorporating symmetry into deep dynamics models for improved generalization," in *International Conference on Learning Representations*, 2021.

[18] D. Wang, R. Walters, and R. Platt, "$\mathrm{SO}(2)$-equivariant reinforcement learning," in *International Conference on Learning Representations*, 2022.

[19] D. Ordonez-Apraez, M. Martin, A. Agudo, and F. Moreno-Noguer, "On discrete symmetries of robotics systems: A group-theoretic and data-driven analysis," *Robotics: Science and Systems*, 2023.

[20] M. Hutter, C. Gehring, A. Lauber, F. Günther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Blösch, *et al.*, "Anymal-toward legged robots for harsh environments," *Advanced Robotics*, vol. 31, no. 17, pp. 918–931, 2017.

[21] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Cambridge, MA, USA, 2018.

[22] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, vol. 37, 07–09 Jul 2015, pp. 1889–1897.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] B. Ravindran and A. G. Barto, "Symmetries and model minimization in markov decision processes," University of Massachusetts, USA, Tech. Rep., 2001.

[25] M. Zinkevich and T. R. Balch, "Symmetry in markov decision processes and its implications for single agent and multiagent learning," in *International Conference on Machine Learning*, 2001, p. 632.

[26] S. Yan, Y. Zhang, B. Zhang, J. Boedecker, and W. Burgard, "Geometric regularity with robot intrinsic symmetry in reinforcement learning," in *RSS 2023 Workshop on Symmetries in Robot Learning*, 2023.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[28] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *Advances in neural information processing systems*, vol. 33, pp. 19 884–19 895, 2020.

[29] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu based physics simulation for robot learning," in *Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[30] M. Wuthrich, F. Widmaier, F. Grimminger, S. Joshi, V. Agrawal, B. Hammoud, M. Khadiv, M. Bogdanovic, V. Berenz, J. Viereck, M. Naveau, L. Righetti, B. Schölkopf, and S. Bauer, "Trifinger: An open-source robot for learning dexterity," in *Conference on Robot Learning*, vol. 155, 2021, pp. 1871–1882.