BSc Data Science & Artificial Intelligence

Multi-Task Simulation for

Evaluating Generalization under Dynamic Task Execution

Viktoria Georgieva

Supervisors:
Álvaro Serra-Gómez, Thomas Moerland

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

July 7, 2026

Abstract

A locomotion policy that covers multiple terrains is needed when deploying robots in real-world settings. These conditions demand that many unrelated obstacles be overcome, even though the skills required do not always overlap and sometimes even conflict with one another. Training such a policy, however, is a difficult task due to gradient interference, difficulty imbalance, and other challenges we have identified and continue to discover. This thesis investigates the conditions under which these policies can retain acceptable task-specific performance when exposed to multiple different distributions simultaneously, as well as the practical implications that complement the learning process. For this purpose, five locomotion tasks were implemented covering four terrain types: a plane, rough terrain, stairs, and a gap. Single-task setups gave a reliable reference point for comparing the multi-task policy against specialist performance. Task conditioning and phased curriculum were used to counteract task interference and simplicity bias, contributing to the multi-task learning (MTL) policy's non-trivial performance across all five tasks. Evaluation was conducted for all task-policy pairs to produce a *cross-evaluation matrix*, which revealed a consistent hard-to-easy asymmetry. Policies previously trained on harder terrains were more likely to generalize to easier ones, but the reverse did not always hold. These findings suggest that in multi-task settings, environment design, which we often place secondary to algorithmic choice, is as important for policy generalization as the learning algorithm itself.

Contents

1 Introduction **1**
- 1.1 Contributions . . . 2
- 1.2 Thesis Overview . . . 2

2 Related Work **3**
- 2.1 Single-Task and Multi-Task Reinforcement Learning for Locomotion . . . 3
- 2.2 Stability Challenges in Multi-Task Reinforcement Learning . . . 3
- 2.3 Policy Optimization for Locomotion Learning . . . 4
- 2.4 Evaluation Practices for Locomotion Learning . . . 5
- 2.5 Embodied agents and locomotion in simulation . . . 6

3 Methods **6**
- 3.1 Research Design . . . 6
- 3.2 Benchmark Environment Design . . . 7
- 3.3 Learning Setup . . . 10
  - 3.3.1 Single-Task Baselines . . . 11
  - 3.3.2 Unified Multi-Task Setup . . . 12
- 3.4 Implementation and Reproducibility . . . 15
- 3.5 Analysis Plan . . . 16

4 Experiments **16**
4.1 Evaluation Protocol . . . 16
4.1.1 Experimental Procedure . . . 19
4.2 Results . . . 20
4.2.1 Training Dynamics . . . 20
4.2.2 Single-Task Performance . . . 23
4.2.3 Multi-Task Retention and Interference . . . 23

5 Conclusions and Further Research **27**
5.1 Discussion . . . 27
5.2 Limitations . . . 29
5.3 Further Research . . . 29

References **33**

A Comparison of Robotics Simulators **34**

B Benchmark Reward Configuration **34**

C Learning Setup Reproducibility Details **35**
C.1 Software and Compute Setup . . . 35
C.2 PPO and Stability Settings . . . 36
C.3 Seeded Single-Task Checkpoints . . . 37
C.4 Unified Multi-Task Phases . . . 37

1 Introduction

In recent years, we have witnessed impressive advancements in embodied AI, with the market for robotic solutions projected to continue expanding into domains we once deemed too difficult to automate. To a certain extent, this growth is due to robotics becoming useful for improving precision and efficiency in sectors such as manufacturing, logistics, and healthcare. However, the recent momentum is primarily attributed to breakthroughs in enabling technologies (artificial intelligence, sensing, and computing, among others), with modern robotics becoming a major beneficiary. Unlike traditional AI applications (e.g., chatbots, recommendation systems, or image classifiers), embodied AI stretches past the virtual domain. This potential for real-world integration is what attracts public attention, but also places pressure on researchers and industry to deliver more capable systems sooner and in greater quantities.

As robots have entered the physical world, single-task proficiency is no longer enough for a system to operate successfully in settings where multiple objectives compete. This is especially relevant when deploying embodied agents in high-stakes environments, where small errors can have serious consequences. In robotics, system capability entails a broader set of competencies, particularly when agents are expected to generalize outside controlled environments [1, 2].

Because demand is growing, while the aforementioned technologies are still maturing, this mismatch inevitably opens new gaps for researchers to address. The call for improved adaptability necessitates the development of more versatile systems, enabling multiple tasks to be pursued simultaneously and evaluated at scale [2]. Although more attention is usually given to algorithms, environment design has an equally strong effect on performance. In fact, it is one of the most reliable predictors of system behavior [3], allowing us to anticipate outcomes prior to real-world deployment.

Rightfully, academia has started to account for this dependency through frameworks that leverage large-scale policy learning. Orbit, for example, provided reusable environments and simplified setups under different tasks and physics configurations [4]. Humanoid-Gym and Holosoma showed how a similar approach can support parallel locomotion training for humanoid robots [5, 6]. Such frameworks, aside from being valuable for improving training efficiency, reflect an understanding of simulation as more than a place to run experiments. However, they mainly provide infrastructure and reusable templates. Less is known about how the structure of the locomotion task set influences retention, transfer, and generalization when one policy is exposed to multiple terrains.

This creates a gap between training infrastructure and environment design. Therefore, instead of stopping at algorithmic performance, this study investigates how environment design choices determine which skills are preserved when a policy is trained across different terrain distributions with partially conflicting objectives. To examine this relationship, I raise two research questions:

RQ1. *How do specific environment design choices, such as task-family structuring and shared interfaces, affect policy generalization?*

RQ2. *Does multi-task training across structured task families improve generalization compared to training tasks in isolation?*

In this work, *task families* are understood as groups of locomotion tasks that share terrain type and/or locomotion objective: flat velocity tasks (Family A), random rough walking and stair

climbing tasks (Family B), and agility terrain tasks (Family C). Based on this definition and the aforementioned questions, I formulate two hypotheses:

H1. *Structuring locomotion tasks into families with a shared observation and action interface will improve transfer between tasks that require similar control strategies.*

H2. *A unified multi-task policy will retain better cross-task performance than single-task specialists, but may trade off peak performance on some individual tasks.*

To verify these hypotheses, this thesis requires a controlled benchmark and an evaluation protocol to be implemented. The following contributions define these components.

1.1 Contributions

This work makes two main contributions:

1. It introduces a unified benchmark environment for multi-task quadruped locomotion, comprising five benchmark tasks and three structured task families. This benchmark provides a setting in which the same robot, action space, and observation interface are shared, so differences between policies can be analyzed in terms of task structure and learning strategy.

2. It defines an evaluation protocol that compares single-task and multi-task policies on task performance, retention, and transferability. In addition to determining whether a policy performs well on a specific task, the protocol reveals which previously acquired behaviors are preserved.

These contributions define both the experimental setting and the evaluation methodology needed to study when multi-task reinforcement learning improves quadruped locomotion and whether already learned skills adapt reliably to unseen conditions.

1.2 Thesis Overview

In pursuit of this, Section 1 first reveals the motivation behind this study, which draws on advancements in multi-task reinforcement learning and legged locomotion. The background needed to understand and address the research gap mentioned earlier is then summarized in Section 2. To build on this work, Section 3 describes the methodology. This includes the training setups, the selected robot, the benchmark task design, the observation and action spaces, and the success metrics, as well as justification of the practical choices made during development. Section 4 reports the experimental results, including training dynamics and final specialist performance. The evaluation protocol is also described, showing how policy performance is assessed. After this, the MTL phased curriculum and cross-task evaluation reveal interference patterns across tasks where transfer trade-offs occurred. Finally, Section 5 summarizes findings, answers the RQs, discusses limitations, and makes suggestions for future work.

2 Related Work

To formulate my research questions, I explored recent work in reinforcement learning for embodied control and how environmental conditions affect policy generalization. Although manipulation is an important branch of embodied AI, this study focuses specifically on legged locomotion. Unlike manipulation, where agents change the environment around them, locomotion involves interacting with the terrain itself. That makes such tasks highly relevant for studying adaptation, because any small change in terrain design propagates into policy performance.

Another aspect of locomotion learning that provides insights into the transferability of learned skills is the evaluation strategy. Currently, there is no broad consensus on what counts as a reliable indicator of policy quality. However, this question has received much attention in the literature and bears directly on reproducibility.

In this section, I review recent academic work that addresses these issues and discuss the main lessons on which this study builds.

2.1 Single-Task and Multi-Task Reinforcement Learning for Locomotion

Before moving to the contributions, a clear distinction between specialist and multi-task policies needs to be made. Hwangbo et al. are often credited as one of the earliest examples of specialist-policy learning for sim-to-real control [7]. For a long time, single-task learning was treated as a default for training and validating controllers. Although we no longer consider this setup sufficient for general forms of autonomy, it prepared the ground for transitioning to policies that adapt to varied tasks and conditions.

However, the need for a robotic system to perform reliably in changing terrain meant that this baseline could not capture environmental diversity well enough if tied to fixed tasks. Rapid motor adaptation (RMA) reinforced this point by showing that policies can adjust online to distribution shifts [8]. Nonetheless, the major challenge in multi-task reinforcement learning, namely optimization under shared parameters, remains just as relevant. As discussed by Teh et al. [9], poor choices of sampling, objectives, or update dynamics inflict gradient interferences, leading to retention loss and undesirable trade-offs.

Some of these difficulties were mitigated when massively parallel simulation and curriculum training were introduced, speeding up data collection and improving generalization [10]. These advances made it possible to train locomotion policies faster and on a single workstation. Even so, faster training did not immediately remove the underlying problem. Stable transfer still requires attention to evaluation and checkpoint selection.

2.2 Stability Challenges in Multi-Task Reinforcement Learning

To understand why retention remains an ongoing issue in robotics, one needs to look at why these discrepancies arise in the first place. It is not uncommon for learned skills to improve one task but degrade performance on another. For example, a policy optimized for fast locomotion on flat ground might underperform when deployed in rough terrain, where small, cautious steps are needed to keep the robot moving. A shared policy may be pulled toward locally dominant objectives because task-specific updates can point in different directions. Over time, this manifests as negative

transfer and, in sequential settings, as catastrophic forgetting [11]. In this thesis, *forgetting* refers to a drop in performance on a previously learned task after another task or distribution has been emphasized. One approach to reducing such interference is to modify gradients before the update step, as proposed in PCGrad [12]. Adaptive task sampling makes allocation explicit by considering which tasks should be emphasized and what sampling strategy is most appropriate. These make a particular difference in locomotion settings with sparse rewards [2].

Similarly, recent work on scheduled multi-task training argues that *simplicity bias* may occur when task difficulty varies widely. Easier tasks may dominate optimization unless harder tasks are assigned a higher priority [13]. In light of this, task sampling could be understood as a stability mechanism in itself. It controls which tasks generate updates at different stages to counteract the undertraining of difficult tasks. This motivates the phased training strategy used later in this thesis.

Curriculum learning constitutes the framework behind this same design. It presents tasks in an order of increasing difficulty, so that early skills can support later learning [14]. In reinforcement learning, this can be implemented through adjusting the environment, task distribution, or initial difficulty [15]. When solving locomotion tasks, starting on difficult terrain often results in early falls and sparse learning signals. Rudin et al. [10] showed that when combined with massively parallel simulation, terrain curricula can accelerate quadruped locomotion learning. In line with this, this thesis uses curriculum learning to control terrain difficulty and reduce failure during early training.

Progressive architectures offer another route by separating knowledge over tasks [16], but this comes at the cost of a larger model and weaker parameter sharing. As noted earlier, addressing these challenges does not end with optimization. Here, environment design becomes part of the discussion. Its role, however, is sometimes neglected, although how tasks are grouped and ordered is proven to affect what is retained during training [17, 18]. In response, this thesis makes an important assumption: stability is as much an exposure issue as it is an optimization one and therefore deserves equal attention.

2.3 Policy Optimization for Locomotion Learning

Reinforcement learning problems in continuous control (including locomotion tasks) are typically formalized as Markov Decision Processes (MDPs) [19]. At each timestep t the policy π_θ maps the current observation o_t to an action a_t:

$$a_t \sim \pi_\theta(a_t \mid o_t). \tag{1}$$

The policy parameters θ are optimized during training to maximize the expected cumulative discounted reward $\mathbb{E}\left[\sum_t \gamma^t r_t\right]$, where r_t is the reward at step t and $\gamma \in (0, 1)$ is the discount factor.

Additionally, the policy is represented by an actor network, accompanied by a critic whose role is to estimate the value function and reduce variance in the gradient updates [20]. Without additional constraints, however, large gradient updates can shift the policy distribution too far in a given iteration. Hence, previously collected data may compromise subsequent training in case it becomes invalid [21].

Proximal Policy Optimization (PPO) [22] is an on-policy actor-critic algorithm that constrains how much the policy changes during updates. It does so by optimizing a clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right], \tag{2}$$

where $r_t(\theta)$ is the probability ratio between the updated and previous policy, $\hat{A}_t$ is the estimated advantage, and ϵ controls the size of the allowed policy update.

This clipping mechanism makes policy improvement more conservative by preventing large policy shifts that can destabilize training. As a result, PPO is often a preferred choice for locomotion policy learning [7, 2]. Because it is an on-policy algorithm, rollout data always reflects the current policy, a property that is especially relevant to terrain curriculum training. When difficulty increases, the agent's behavior changes in response to the added complexity. Therefore, the older data is no longer representative of the new action probabilities [10]. PPO avoids this by collecting fresh trajectories at each update step, ensuring that the training signal matches the agent's current capabilities.

Off-policy algorithms such as SAC [23] utilize experience replay, which improves sample efficiency. However, this advantage becomes less significant in cases when massively parallelized environments generate millions of transitions per hour, making data collection cheaper than gradient-update computation. Compared to its predecessor, TRPO [21], PPO offers a lower computational cost per update with comparable constraint guarantees. Therefore, it is the more practical choice considering that policies are trained for thousands of iterations across thousands of environments. Furthermore, PPO's effectiveness on individual locomotion tasks has been demonstrated by Hwangbo et al. [7] on rough terrain, Rudin et al. [10] on varied terrain types, and Kumar et al. [2] in multi-task settings. This track record, combined with PPO's stability under noisy contact rewards (common in legged locomotion), justifies its use as the training algorithm in this thesis.

2.4 Evaluation Practices for Locomotion Learning

Although a unified evaluation framework has not yet emerged, researchers seem to agree on one point. Raw reward values do not provide a complete picture of policy quality. It is possible to achieve a high mean reward while exploiting behavior that works during training but fails once conditions change. In locomotion, this often means learning a gait that keeps the robot stable but becomes unnatural to avoid falls. A robot may learn to keep its base very low and crawl forward to survive in the simulation, but such behavior is unlikely to be useful in real-world scenarios.

A first step is to be cautious even when evaluating on benchmark suites, since averaging scores can be misleading. Agarwal et al. [24] advocate for statistically safer comparisons that report uncertainty, including confidence intervals and more robust aggregate metrics. This already points to a broader lesson. Evaluation should first and foremost ask how reliable a result is, regardless of how high the reported reward may be.

Equally important is experimental design, including choices about seeds, baselines, compute budget, and reporting practices. Henderson et al. [25] show that deep reinforcement learning papers often report final performance without discussing how sensitive results are to these choices, while Patterson et al. [26] argue that such decisions should be treated as part of the experimental design itself. In experiments, these affect outcomes more than is usually acknowledged, compromising reproducibility. This is why the present study places a strong emphasis on decisions such as checkpoint selection and cross-evaluation. These practices determine whether a policy is evaluated only by its best reported score or as a system that remains reliable no matter the setup.

2.5 Embodied agents and locomotion in simulation

New possibilities in robotics opened up when massively parallel simulation was introduced as an alternative to CPU training. Before that, robotic learning was primarily realized through physics engines such as *MuJoCo* [27], *Bullet* [28], *Gazebo/ODE* [29], *DART* [30], or *RaiSim* [31], and was supported by manual tuning and experimentation. In fact, these tools did allow researchers to run several environments in parallel on CPU cores, but the scale was much smaller, training took longer, and the cost was higher.

Conducting this research is only possible because the practical barrier to entry has been lowered. Modern GPU simulators made it feasible to train and evaluate policies even when resources are limited. With this in mind, I built my study around the *Isaac Sim/Isaac Lab* ecosystem; Appendix A provides a comparison with earlier simulators. Isaac Gym is a useful reference point here, as it began the transition to parallel training [32]. Isaac Lab (Isaac Gym's successor) carried this idea in a more modular form of simulating and testing various tasks and learning algorithms [33]. Most recently, the shift has become even more visible as recent simulation frameworks now support larger and more reusable training setups. They made it easier to train policies at scale, but do not fully explain how tasks and physical conditions should be defined.

Another related line of work addresses generalization through domain randomization, which varies parameters to improve policy adaptability to deployment uncertainty [34]. The present study takes a different yet complementary route. It varies the locomotion task structure instead of randomizing the physical parameter. It studies task design and its effects on training, aiming to determine whether a unified policy can maintain high performance when exposed to terrains that favor different specialist behaviors. From that premise, this thesis proceeds.

3 Methods

Prior work showed that deep reinforcement learning can learn locomotion skills in simulation [7, 10], but multi-task training still suffers from poor transfer, interference, and forgetting [9, 11]. The present study recognizes and studies these problems through controlled evaluation.

The multi-task policy is trained with task conditioning and adds a task ID to the observation vector. Sampling profiles are used to balance the learning of difficult tasks. Methodologically, this study is considered *empirical*. It does not prove the optimality of a training schedule but analyzes which tasks transfer well under new conditions and whether a conditioned policy can retain several skills learned simultaneously.

3.1 Research Design

The research was designed as an experimental study where reinforcement learning policies for quadruped locomotion were trained and evaluated. Simulation was used to enable controlled repetition and safe testing before deployment on a physical robot. This is a common practice in robotics research and industry, as simulation reduces both development cost and operational risk [33, 4]. The robot platform selected was *Unitree Go2*. It is a commercially available quadruped suitable for studying legged locomotion and supporting sim-to-real deployment. Isaac Lab's manager-based locomotion environment served as the starting point.

The study compared single-task reinforcement learning against a multi-task setup trained on the same collection of tasks. It examined the relationship between *training regime* (independent variable) and the resulting *locomotion skills* (dependent variable). This directly supported the research questions by testing (1) whether one policy could learn across multiple terrains and task families without losing much of its task-specific performance, (2) whether skills transferred to unseen experimental conditions, and (3) where multi-task learning introduced performance trade-offs. Policy performance was measured using task success, failure rate, command-tracking quality, and episode survival.

The single-task setup was treated as the specialist baseline and was expected to create stronger task-specific behavior, as the policy optimized only one objective distribution. By contrast, multi-task training exposed the policy to more diverse experiences aimed at improving transferability between families, but introduced interference as a challenge to be addressed. To isolate the effect of the training design, the task collection was kept consistent across both setups.

3.2 Benchmark Environment Design

The benchmark environment was organized around five locomotion tasks. This organization followed general conventions established by recent work on multi-task evaluation, which groups tasks by shared structure and difficulty [1, 2].

In this thesis, I adapt this idea to legged locomotion. Table 1 summarizes the high-level structure of the benchmark (operationalizing H1).

Task	Family	Terrain	Objective	Main terrain parameter
A1	A	Flat ground	Forward velocity tracking	Plane
A2	A	Flat ground	Omnidirectional velocity tracking	Plane
B1	B	Random rough terrain	Robust velocity tracking	Random height variation
B2	B	Inverted pyramid stairs	Forward stair climbing	Step height $[0.04, 0.16]$ m
C2	C	Gap terrain	Forward gap crossing	Gap width $[0.00, 0.20]$ m

Table 1: Benchmark task overview. The table summarizes the task family, terrain type, objective, and main terrain parameter used to instantiate each benchmark task.

All tasks were implemented as extensions of Isaac Lab's Unitree Go2 locomotion environment. The base simulator, robot asset, actuator model, contact handling, command manager, and manager-based reward/termination structure were inherited from Isaac Lab. The thesis contributions operated at two levels. At the *task design level*, this included selecting terrain type and defining command ranges per task, as well as decomposing the benchmark into the five tasks introduced earlier and distributing them across families. At the *interface level*, default Isaac Lab reward weights were adjusted to match the task demands. The task-specific configuration is reported in Appendix B, Table 10. The observation space was standardized to the same 235-dimensional interface to enable multi-task training and cross-evaluation.

More specifically, A1 and A2 share the same terrain type (flat ground) but have different command ranges. A1 constrained the robot to forward velocity tracking, while A2 extended this to forward/backward, lateral, and yaw velocity commands. B1, B2, and C2 each introduce a distinct terrain type in addition to flat, increasing task difficulty. B1 tests robustness to irregular ground

height, B2 tests stair climbing, and C2 tests locomotion over discontinuous terrain. B1 has an omnidirectional command range (same as A2), while B2 and C2 are forward-biased.

Table 2 shows the command ranges, which determine whether the policy had to learn forward-only locomotion, full omnidirectional tracking, or forward-biased behavior. In each episode i and timestep t, the actual command values $v^{\text{cmd}}_{x,i,t}$, $v^{\text{cmd}}_{y,i,t}$, and $\omega^{\text{cmd}}_{z,i,t}$ were sampled from these ranges.

Task	Command profile	$v^{\text{cmd}}_{x,i,t}$ range	$v^{\text{cmd}}_{y,i,t}$ range	$\omega^{\text{cmd}}_{z,i,t}$ range
A1	Forward-only	$[0.35, 1.00]$	$[0.00, 0.00]$	$[0.00, 0.00]$
A2	Omnidirectional	$[-1.00, 1.00]$	$[-1.00, 1.00]$	$[-1.00, 1.00]$
B1	Omnidirectional	$[-1.00, 1.00]$	$[-1.00, 1.00]$	$[-1.00, 1.00]$
B2	Forward-only	$[0.40, 0.90]$	$[0.00, 0.00]$	$[0.00, 0.00]$
C2	Forward-biased	$[0.40, 0.80]$	$[-0.10, 0.10]$	$[-0.20, 0.20]$

Table 2: Command ranges used in the benchmark tasks. Linear velocities are in m/s and yaw velocity is in rad/s. The command profile indicates whether the task required forward-only tracking, full omnidirectional tracking, or forward-biased motion.

The unified multi-task environment was then built from this fixed task set. Namely, all four terrain types were combined, and task conditioning was introduced by appending an identifier to each observation.

This grouping is meaningful because it clearly distinguishes between command and terrain complexity. It also creates a challenge for multi-task training, as tasks placed different pressures on the shared policy. For instance, *Family A* keeps the terrain simple, rewarding stable velocity tracking. By contrast, *Family B* and *Family C* increase the physical difficulty and required more agility. This means that a conservative gait that excelled on flat terrain might underperform on stairs and gaps, and vice versa. Because the tasks differed in command structure and requirements, the shared policy had to adapt while also preserving gait quality under the previously encountered conditions. Thus, the selection was both sufficient for the thesis scope and covered the main forms of variation relevant to simulated locomotion.

As a prerequisite to multi-task learning, all benchmark tasks were implemented with compatible 235-dimensional observation spaces, consisting of 48 proprioceptive and command features and a 187-dimensional height scanner. The same policy architecture could then be evaluated across different environments without changing the input dimension or observation semantics. The task objectives were formulated as command-following locomotion problems, meaning that the agent had to track the commanded base velocity to remain stable. Episodes terminated when the agent lost its balance or an illegal base contact was triggered. Therefore, succeeding at a particular task required both survival and sufficiently accurate tracking criteria to be satisfied. The tasks also shared the same action space, with the policy outputting continuous joint action commands.

In this benchmark, tracking was measured using the linear velocity error in the horizontal plane and the yaw-rate error around the vertical axis. For step t of episode i, the commanded horizontal velocity was represented as $\mathbf{v}^{\text{cmd}}_{xy,i,t} = (v^{\text{cmd}}_{x,i,t}, v^{\text{cmd}}_{y,i,t})$, and the commanded yaw rate as $\omega^{\text{cmd}}_{z,i,t}$, with all command components sampled from the task-specific ranges in Table 2. The linear tracking error was defined as

$$e^{\text{lin}}_{i,t} = \left\| \mathbf{v}^{\text{cmd}}_{xy,i,t} - \mathbf{v}^{\text{base}}_{xy,i,t} \right\|_2, \tag{3}$$

and the angular tracking error was defined as

$$e_{i,t}^{\text{ang}} = \left|\omega_{z,i,t}^{\text{cmd}} - \omega_{z,i,t}^{\text{base}}\right|. \tag{4}$$

Here, $\mathbf{v}_{xy,i,t}^{\text{base}}$ denotes the measured horizontal base velocity in the robot base frame at step t of episode i, and $\omega_{z,i,t}^{\text{base}}$ denotes the measured yaw rate in the same frame.

These errors measure how accurately the policy follows the command. More specifically, the linear error measures the difference between the commanded horizontal velocity and the velocity that the robot actually reaches. Therefore, a small linear error suggests the robot moves in the requested direction at the requested speed. The angular error measures the difference between the commanded yaw rate and the yaw rate reached by the robot. A small angular error means that the robot turns at the requested speed.

A step t in episode i was considered successful ($\sigma_{i,t} = 1$) if the robot survived and both errors were below the respective task-specific thresholds:

$$\sigma_{i,t} = \mathbb{I}\left[\neg f_{i,t} \wedge e_{i,t}^{\text{lin}} \leq \epsilon_{\text{lin}} \wedge e_{i,t}^{\text{ang}} \leq \epsilon_{\text{ang}}\right], \tag{5}$$

where $\sigma_{i,t} \in \{0, 1\}$ is the step success indicator, $f_{i,t}$ indicates failure at step t of episode i, ϵ_{lin} is the linear velocity error threshold, and ϵ_{ang} is the yaw-rate error threshold.

A policy should minimize both errors while also keeping the robot upright. Avoiding falls is not the sole goal that the agent needs to achieve. It also has to follow the command rather than standing still or moving too slowly. Therefore, success was based on sustained step-level success, not pure survival.

For each episode i, the step indicators were first averaged to obtain the success-step ratio:

$$r_i = \frac{1}{T_i} \sum_{t=1}^{T_i} \sigma_{i,t}. \tag{6}$$

An episode was counted as successful when this ratio exceeded a minimum ρ_{min} and the episode did not end in failure:

$$s_i = \mathbb{I}\left[r_i \geq \rho_{\text{min}} \wedge \neg F_i\right], \tag{7}$$

where $s_i \in \{0, 1\}$ is the episode success indicator and F_i verifies whether episode i ended in failure (i.e., $F_i = 1$ if a failure termination occurred at any step of the episode).

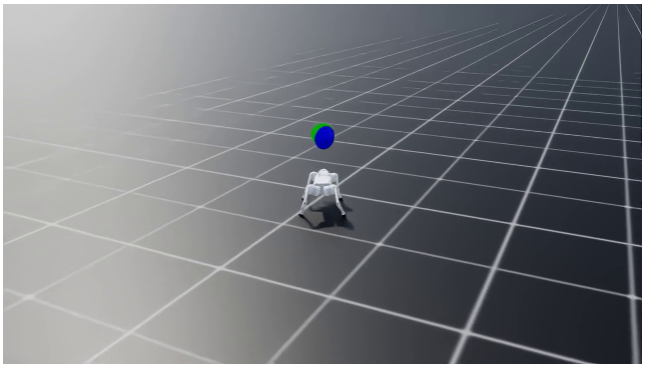
Table 3 shows the task-specific thresholds used to convert tracking errors into episode success. The thresholds are deliberately set stricter for simpler tasks and more tolerant for harder terrains. These values constitute the success criteria used in the evaluation matrices in Section 4.2.2.

Note that ϵ_{lin} and ϵ_{ang} are step-level thresholds used in Equation 5, meaning that they determine whether a single step satisfies the tracking condition. By contrast, the threshold ρ_{min} is applied to complete episodes in Equation 7. It captures what fraction of steps must satisfy this condition. The success criterion was defined this way to avoid rewarding short or unstable behavior that obeyed the command but failed to sustain tracking.

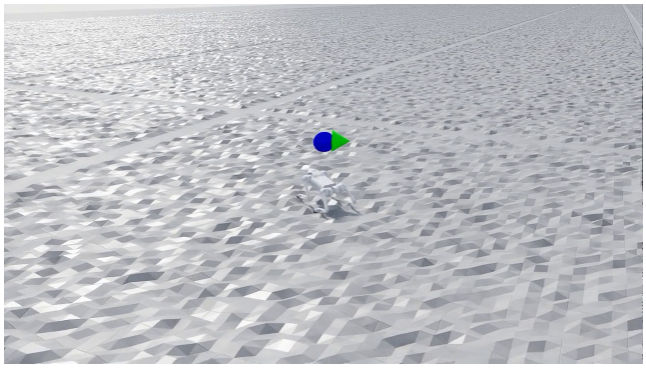
This task set defines the benchmark used for all subsequent training and evaluation experiments. Figure 1 gives a visual overview of the terrain set.

Task	ϵ_{lin}	ϵ_{ang}	ρ_{min}
A1	0.20	0.40	0.85
A2	0.25	0.50	0.80
B1	0.35	0.60	0.75
B2	0.40	0.70	0.70
C2	0.45	0.75	0.65

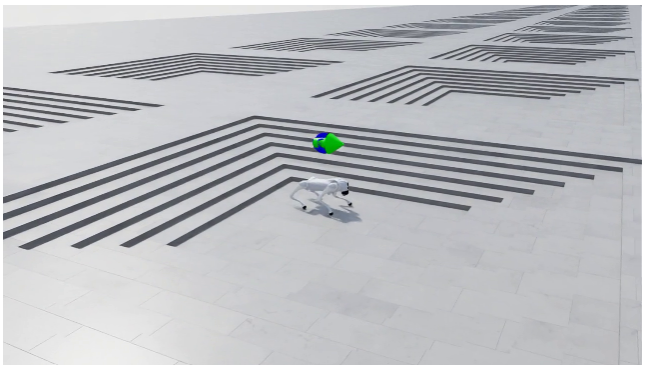
Table 3: Task-specific success thresholds. Linear error thresholds are in m/s and angular error thresholds are in rad/s.



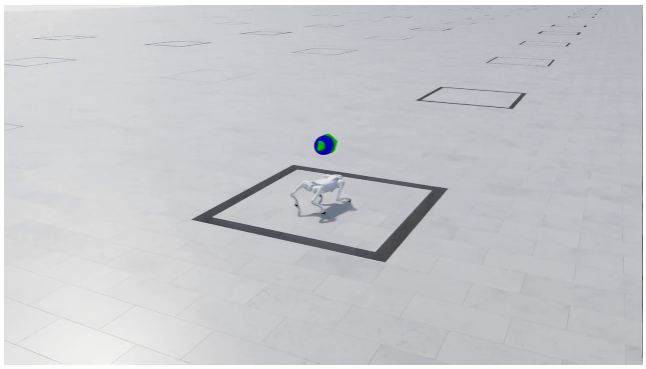
(a) A1: flat forward walking



(b) B1: random rough terrain



(c) B2: inverted pyramid stairs



(d) C2: gap crossing

Figure 1: Preview of the benchmark environments used for training and evaluation. A1 represents forward walking on a plane; B1 uses rough terrain created through height noise; B2 uses inverted pyramid stairs that encourage the agent to learn climbing; and C2 comprises discontinuous gap subterrains and is used as an agility test.

3.3 Learning Setup

Building on the framework introduced in Section 2.3, all policies were trained with Proximal Policy Optimization (PPO) using the RSL-RL integration in Isaac Lab. In this setup, the policy received the shared 235-dimensional observation vector and outputted continuous joint action commands. The conditioned MTL policy additionally received a four-dimensional terrain-condition identifier as

input.

A challenge encountered during training was that some updates changed the policy too aggressively. This was observed during early multi-task experiments, when large updates shifted the action distribution. As a consequence, previously learned behaviors were forgotten. PPO's clipped surrogate objective (Equation 2) addressed this by limiting how far the updated policy can move from the policy that generated the rollout data. Gradient clipping and a conservative learning rate were further utilized to reduce numerical instability. The full hyperparameter details are reported in Appendix C.2.

The Isaac Lab Go2 rough-terrain PPO runner was used as the starting point. It was extended with custom benchmark tasks, terrain configurations, command ranges, and reward adjustments. Based on this shared training, two learning setups were compared: single-task baselines and a multi-task setup.

3.3.1 Single-Task Baselines

The single-task setup trained a separate policy per benchmark task to evaluate performance and transferability when exposing the policy to a specific distribution (e.g., A1 trained on flat forward walking, while B2 trained on stair climbing). This meant that each policy was optimized to eventually become a specialist controller for a particular locomotion objective.

Two benchmark tasks required additional setup. The B2 specialist was trained with posture-reward annealing. The term `base_height_l2` penalizes deviation from a target base height so that the robot keeps its torso at a stable height. The term `flat_orientation_l2` penalizes base tilt, encouraging the robot to remain upright. During early training, the base-height and orientation penalties were stronger, encouraging the robot to learn to stand. Once this goal was achieved, both penalties were reduced after the first 35% of training so that climbing behavior emerged. Table 4 reports the two annealed reward terms and their initial and final weights. Step heights ranged from 0.04 to 0.16 m.

Reward term	Penalized quantity	Initial weight	Final weight
`base_height_l2`	$(z^{\text{base}} - z^{\text{target}})^2$	-10.0	-3.5
`flat_orientation_l2`	$\|\mathbf{g}_{xy}^{\text{base}}\|_2^2$	-5.0	-2.0

Table 4: Posture-reward annealing used for the B2 stair-climbing specialist. Stronger penalties were used early in training to encourage standing stability and were linearly reduced over the first 35% of training.

Here, z^{base} denotes the robot base height, and z^{target} is the target base height. The term $\mathbf{g}_{xy}^{\text{base}}$ denotes the horizontal components of the gravity vector w.r.t the robot base frame. When the robot is upright, these components are close to zero. Larger values indicate base tilt.

The C2 specialist was initialized from the corresponding A2 walking checkpoint. Because C2 was framed as an agility test, it required a stable walking gait so that the policy could learn crossing. Training C2 directly led to poor walking behavior, even though the robot achieved high rewards. The C2 baseline is therefore to be interpreted as a specialist obtained by finetuning a walking controller on the respective distribution. The resulting policy used the same cross-task protocol as the other specialists.

For evaluation, one checkpoint per specialist was selected. For A1, A2, and B1, the final checkpoint was used, since these runs converged smoothly. For B2, the checkpoint came after the posture-reward annealing phase. For C2, the selected checkpoint was finetuned from the corresponding A2 walking checkpoint. The exact files and run folders are reported in Appendix C.3.

The purpose of the baselines was to provide the reference point for judging whether the unified MTL policy lost task-specific performance. The same learning algorithm and robot were used across both learning setups to ensure fair comparison, with the main difference being the task distribution.

3.3.2 Unified Multi-Task Setup

By contrast, the multi-task setup combined multiple terrain types (flat terrain, random rough terrain, inverted stairs, gap terrain), with the share of each terrain type controlled through a sampling parameter. This setup tested whether a policy could learn a common locomotion strategy irrespective of the obstacle the agent tackles. Unlike the single-task baselines, it did not specialize in only one command distribution or terrain geometry. This created two main challenges to be addressed. First, the tasks rewarded different kinds of behaviors. Second, the tasks differed in learning difficulty. Easier tasks produced high reward signals early, while harder tasks needed more time to adapt and often suffered from poor exploration if learning parameters were not carefully configured.

In response, focused sampling phases were used to bias training toward selected terrains while retaining rehearsal on the remaining ones (e.g., B2 could be given a larger share of the overall sampling probability, while the other terrains still remained present with smaller probabilities). Here, the schedule was manually designed for intermediate diagnosis to account for weaknesses in the current policy. Starting directly from the hardest terrain instance predisposes the robot to early failure [14, 10]. This was particularly relevant to rough terrain, stairs, and gaps, where the hardest setting could lead to almost immediate falls. For this reason, difficult terrains received additional updates when they underperformed, while solved tasks remained in the training distribution to prevent forgetting. The full distribution across phases is shown in Figure 2.

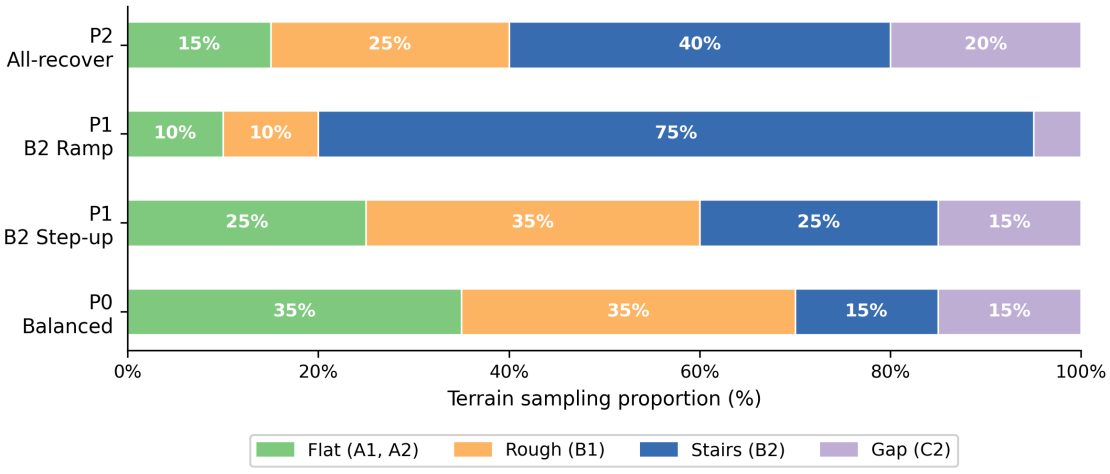


Figure 2: Terrain sampling proportions across MTL training phases. During P1 (B2 Ramp), the proportion of stairs was increased to 75% as a measure against simplicity bias. The remaining terrains were present at reduced rates to prevent forgetting.

Training began with an *initialization phase (P0)*. It used a flat- and rough-biased distribution (each at 35%), while stairs and gap were kept low (each at 15%). This produced strong A1, A2, and B1 performance but left B2 unresolved. Subsequent phases targeted stairs to compensate for the earlier imbalance. Two *P1 sub-phases* followed. *P1 B2 Step-up* increased the stair sampling share to 25% and used an easier forward command range so initial stair contact could be established. *P1 B2 Ramp* then raised B2 sampling to 75% and ramped its command velocity toward the benchmark speed. A final recovery phase (P2) then used a mixed distribution with stair emphasis (40%) to restore A2, while keeping the newly learned B2 behavior.

This is consistent with the idea that task exposure and sampling choices can serve as stability controls in multi-task training, reducing the risk of easier tasks dominating optimization [17, 18, 13]. As a result, the MTL policy could adapt to the targeted terrain and preserve previously learned skills.

A learned scheduler would be a natural extension to this work. It was left for future research as it introduces an additional optimization problem outside the thesis scope. Prior work has framed this as automatic curriculum selection, where the scheduler learns which tasks to present so that the agent continues to make progress [35, 36]. Here, such a mechanism was not necessarily required as the thesis did not aim to optimize the curriculum itself. Instead, the objective was to test whether a unified policy could retain performance given a fixed benchmark suite. The absence of such a scheduler was addressed by manually specifying sampling phases and by reporting the resulting policy through cross-task evaluation.

Lastly, the unified policy was task-conditioned, with a task identifier indicating the active terrain type (increasing the observation dimensions from 235 to 239). This meant that the controller did not infer the intended adjustments solely from terrain observations and velocity commands. Although it reduced ambiguity between tasks, this choice introduced a limitation. The resulting policy assumed that the active task label was always available, which is rarely true in real-world deployment. Here, this was acceptable because the benchmark evaluated performance under known conditions, so robustness to uncertainty was not included as an experimental variable.

Additionally, both the single-task and unified setups used a terrain curriculum to gradually increase difficulty as training advanced. Within each terrain type, a difficulty curriculum would automatically promote (or demote) robots across levels w.r.t episode outcomes. The terrain was arranged as a grid whose rows represented the difficulty levels. Robots that successfully traversed their current terrain were assigned a harder row in the following episode, and vice versa. Figure 3 illustrates this mechanism for the stair-climbing task.

For rough terrain (B1), the same structure applied, with surface irregularity scaled across levels. For gap crossing (C2), gap width varied from 0 to 0.20 m across 10 levels. The flat-terrain tasks (A1 and A2) had no curriculum enabled because the terrain difficulty was uniform by design.

In the MTL setup, each training phase additionally *capped* the maximum starting level (the highest terrain row from which a robot could be spawned at the beginning of an episode). Let ℓ^{curr} denote the terrain level assigned by the curriculum and let c_p denote the cap used in phase p. The starting level was then clipped as

$$\ell_p^{\text{start}} = \min(\ell^{\text{curr}}, c_p). \tag{8}$$

This was needed since curriculum progress from an earlier phase did not necessarily reflect competence under the current phase distribution. In a single-task run, the policy only trains on one

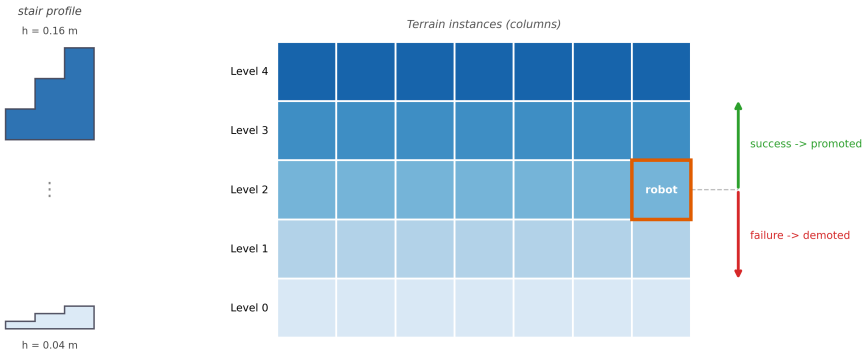


Figure 3: Terrain difficulty curriculum (stair-climbing example, simplified to 5 levels for illustration; 10 were used in practice). The terrain grid has rows of increasing step height. After each episode, a robot is promoted one level on success or demoted on failure.

terrain distribution. Therefore, reaching a higher row signals that the policy has already succeeded on easier rows of the same task.

In the MTL setting, this assumption was weaker. Phase transitions changed the sampling proportions and sometimes the phase profile. For example, after P0, the policy had already learned useful flat and rough locomotion, but B2 was still weak. If the following B2-focused phase had allowed the robot to start from the higher stair rows, the policy would have encountered those difficult stair instances before it could tackle them. Instead of determining how high the curriculum could take robots, the cap served as a reset constraint. Figure 4 illustrates this distinction.

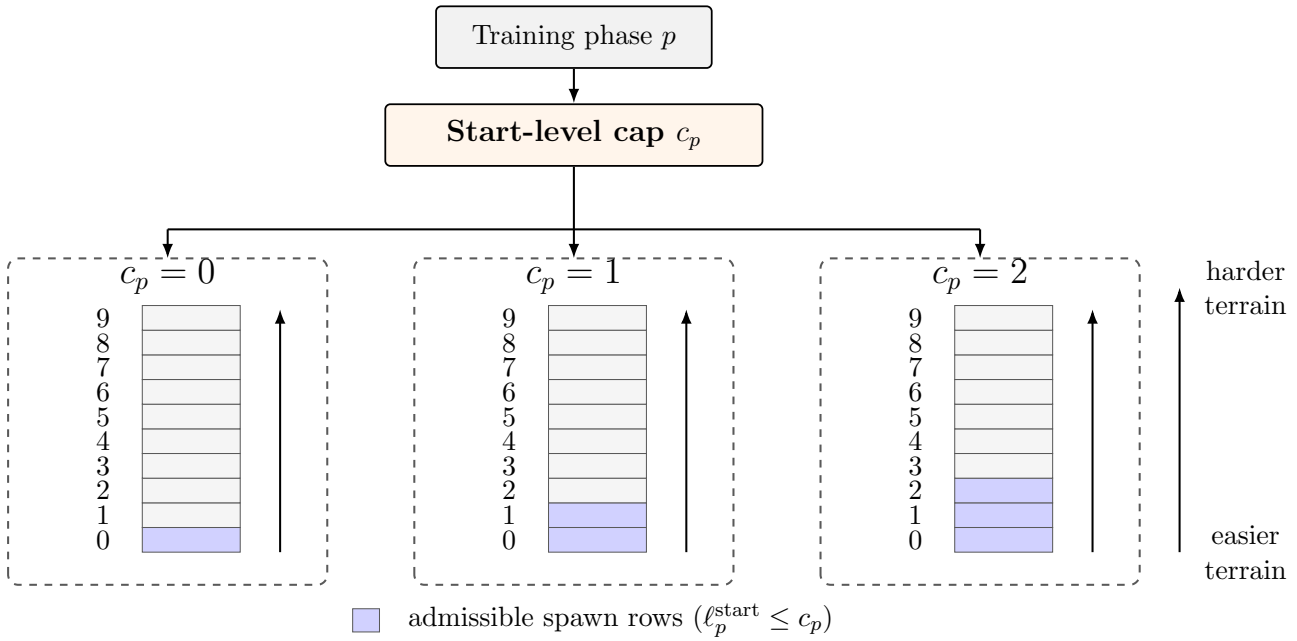


Figure 4: Illustration of phase caps used in the MTL curriculum. For phase p, the cap c_p limits the highest terrain row from which a robot may be spawned at the beginning of an episode. Shaded rows indicate admissible spawn rows (shown for $c_p = 0$, $c_p = 1$, and $c_p = 2$).

As rows featured levels 0 to 9, a $c_p = 0$ meant robots started at level 0 regardless of what the previous phase had achieved and continued advancing from there. P1 B2 Step-up used 0 (keeping stairs at the easiest entry point to establish step contact). If not adjusted, immediate falls would have followed, with stair contact being freshly introduced. P1 B2 Ramp used $c_p = 1$ to account for the policy having learned basic stepping and advancing it further. A cap of 1 meant robots could start at a slightly harder entry point, carrying over competence from the previous phase. The mixed phases (P0 and P2) used $c_p = 2$. Permitting starts from the first three rows increased terrain diversity while still avoiding the highest difficulty levels.

For the single-task baselines, the final checkpoint was used for evaluation. In the MTL setup, intermediate checkpoints were used both for evaluation and for initializing later training phases, in line with concerns about reporting and evaluation practices in deep reinforcement learning [25]. The detailed checkpoint-selection procedure is described in Section 4.1.1.

3.4 Implementation and Reproducibility

As previously noted, all policies used the same Isaac Lab Unitree Go2 stack, the RSL-RL PPO runner, as well as the same observation structure and reward/termination managers, making evaluation under a single protocol feasible. The software stack, observation/action dimensions, and compute setup are summarized in Appendix C.1. Single-task baselines used 1024 parallel environments. This number was sufficient for single-task training because samples were drawn from the same distribution. However, this budget had to be divided among the four terrain types in the MTL setting, with rehearsal terrains receiving a smaller fraction of the rollout batch. Focused sampling further amplified this effect, making retention harder as the update signal became noisier.

Because of this, all MTL training phases were run with 4096 environments. The full phase sequence is reported in Appendix C.4. Earlier phases were treated as part of the initialization process for establishing a usable gait. All training, whether single- or multi-task, was executed on the *ALICE computing cluster*. With 4096 environments and a minimum terrain sampling share of 5% (so that even the least emphasized MTL terrain still contributed ≈ 204 environments per rollout). Precisely because MTL requires greater per-task sample coverage, this change was necessary to ensure fair comparison with the single tasks.

For reproducibility, all run folders, checkpoints, and configuration files were stored under the project logging directory. The selected single-task checkpoints are listed in Appendix C.3. Each run saved the environment and agent configuration, including the PPO parameters and terrain sampling details. Cross-evaluation results were stored for every policy and exported into matrices.

Both the single-task baselines and the final MTL policy were trained across three seeds, and results were aggregated as mean $\pm$ standard deviation [25]. All three seeds used an identical training setup (the same phase structure, training sampling proportions, starting level caps, and reward configurations). Here, the phased curriculum design was developed iteratively using seed 0 as a diagnostic reference and then applied without modification to seeds 1 and 2. The reported variance, therefore, reflected variability due to random weight initialization and environment stochasticity. The curriculum was held fixed since the thesis does not claim to have found an optimal schedule, but deliberately uses a working one. As a consequence, the three seeds serve their intended purpose of establishing that the trained policy's capabilities were reproducible across different random initializations.

3.5 Analysis Plan

The analysis was built on the cross-evaluation matrix, whose rows and columns corresponded to the trained policies and the benchmark task, respectively. This matrix provided insights into the transfer across task families and retention of the unified policy. For single-task baselines, diagonal entries indicated how well tasks could be solved when trained independently, while the remaining entries showed whether learned skills transferred to other settings. The unified MTL policy was evaluated on the same benchmark tasks. Its row was compared to the specialist diagonal to measure retention loss.

The main metric was success rate, computed from completed evaluation episodes using thresholds for velocity tracking and survival. These thresholds can be found in Section 3.2, Table 3. Failure rate and mean alive time were reported as secondary metrics to distinguish policies that physically failed from those that survived but tracked commands poorly. The exact aggregation procedure and calculation of the retention difference are defined in Section 4.1.

Learning curves were plotted to interpret how performance evolved over time. For the MTL runs, per-task tracking error, terrain level, reward, and termination statistics were reported, and the extent to which focused sampling improved the targeted tasks was quantified.

RQ1 was addressed by confirming whether the benchmark tasks produced different transfer patterns. Evidence came from differences between diagonal and off-diagonal entries, asymmetries between task pairs, and performance trends per family. *RQ2* was addressed by comparing the final MTL policy against the single-task specialists and by checking how close it came to the targeted success level on each task.

4 Experiments

This section describes the study's experimental setup and reports evaluation results. Section 4.1 summarizes the evaluation procedure, including the phased curriculum design and checkpoint selection. Section 4.2 presents results for training dynamics, single-task specialist performance, and the MTL retention and cross-task transfer.

4.1 Evaluation Protocol

To compare the two training setups, a cross-evaluation protocol was used. Inspiration primarily came from benchmarks such as Meta-World and MTBench, which evaluate policies on the entire task set [1, 2]. Whether a method can scale to many tasks is assessed by using a matrix comprised of policy-task pairs. Its diagonal captures in-task performance, and the off-diagonal entries indicate generalization to tasks not used as the main training objective. In this thesis, I took a similar approach and evaluated locomotion capabilities on terrains with different physical properties. This is where the evaluation protocol connects to the environment design contribution. By changing the terrain structure and task grouping, the parts of the environment that help or limit generalization become identifiable. This connects performance to concrete design choices, so the final score is no longer the only outcome of interest. The cross-evaluation matrix reveals the differences between policies that score well only when evaluated on one terrain and fail on another, and those that perform consistently no matter the task. This ensures transparency about the overall generalization that average performance may otherwise hide.

The task-level success criterion was defined earlier in Section 3.2. This section uses those definitions to describe how the already completed evaluation episodes were aggregated to produce the cross-evaluation matrix entries.

For locomotion, it is important to report discrete success alongside continuous tracking and stability metrics. As emphasized earlier, the primary evaluation outcome is *episode success*. To compute it, step success $\sigma_{i,t}$ was first aggregated into the success-step ratio r_i (Equation 6). This ratio was then compared with the task-specific threshold $\rho_{\min}$. The episode was counted as successful if the threshold was met and it did not end in failure (Equation 7). For a policy π evaluated on benchmark task q, the reported success rate (SR) finally averaged this binary outcome (s_i) over all completed evaluation episodes:

$$\mathrm{SR}_{\pi,q} = \frac{1}{N_{\pi,q}} \sum_{i=1}^{N_{\pi,q}} s_i. \tag{9}$$

The policy-task success rate was then used to quantify retention loss. For each benchmark task q, the retention difference measured the extent to which MTL performance deviated from that of the single-task specialists. This difference was computed as

$$\Delta_q = \mathrm{SR}_{\mathrm{MTL},q} - \mathrm{SR}_{\pi_q^{\mathrm{spec}},q}. \tag{10}$$

where π_q^{spec} denotes the single-task specialist trained on task q, $\mathrm{SR}_{\pi_q^{\mathrm{spec}},q}$ is its success rate after being evaluated on the same task, and $\mathrm{SR}_{\mathrm{MTL},q}$ is the success rate of the unified MTL policy evaluated on task q. A negative Δ_q indicates that the MTL policy performed worse than the corresponding specialist, whereas a value close to zero suggests that task performance was retained.

Although success tells us whether a policy completes a given task, it does not reveal the underlying cause of failure. Unlike benchmarks focusing on manipulation, locomotion success should account for survival, balance, and tracking quality. These supporting metrics are provided in Table 5 and help determine whether the learned behavior is truly meaningful.

Metric	Calculation	Interpretation
Failure rate ($\mathrm{FR}_{\pi,q}$)	$\mathrm{FR}_{\pi,q} = \frac{1}{N_{\pi,q}} \sum_{i=1}^{N_{\pi,q}} F_i$	Share of episodes ending with failure termination.
Mean linear tracking error ($\bar{e}^{\mathrm{lin}}_{\pi,q}$)	$\bar{e}^{\mathrm{lin}}_{\pi,q} = \frac{1}{N_{\pi,q}} \sum_{i=1}^{N_{\pi,q}} \frac{1}{T_i} \sum_{t=1}^{T_i} e^{\mathrm{lin}}_{i,t}$	Mean horizontal command-tracking error, averaged first within each episode and then across episodes.
Mean angular tracking error ($\bar{e}^{\mathrm{ang}}_{\pi,q}$)	$\bar{e}^{\mathrm{ang}}_{\pi,q} = \frac{1}{N_{\pi,q}} \sum_{i=1}^{N_{\pi,q}} \frac{1}{T_i} \sum_{t=1}^{T_i} e^{\mathrm{ang}}_{i,t}$	Mean yaw-rate tracking error, averaged first within each episode and then across episodes.
Mean alive time ($\bar{\tau}_{\pi,q}$)	$\bar{\tau}_{\pi,q} = \frac{1}{N_{\pi,q}} \sum_{i=1}^{N_{\pi,q}} \tau_i$	Average episode duration until timeout or failure.

Table 5: Diagnostic evaluation metrics reported for each policy–task pair (π, q).

Here, $N_{\pi,q}$ denotes the number of completed evaluation episodes for policy π on benchmark task q (at least 1024). T_i is the length of episode i, F_i is the failure indicator, and τ_i is the alive time in

seconds. The error terms $e_{i,t}^{\text{lin}}$ and $e_{i,t}^{\text{ang}}$ follow Equations 3 and 4, evaluated at step t of episode i. Notably, these metrics are computed individually for each seed and then summarized as mean $\pm$ standard deviation.

Because legged robots can fail in qualitatively different ways, the decision criteria and the diagnostic analysis were intentionally kept separated. Instead of redefining success, these additional metrics were used to interpret failure modes. This distinction is made explicit in the cross-evaluation analysis in Section 4.2.3. Consequently, these quantities form two layers of evaluation. The success branch determines the binary episode outcome plus the final success rate. The diagnostic branch uses the same episode data to explain why performance changed. Figure 5 summarizes this relationship.

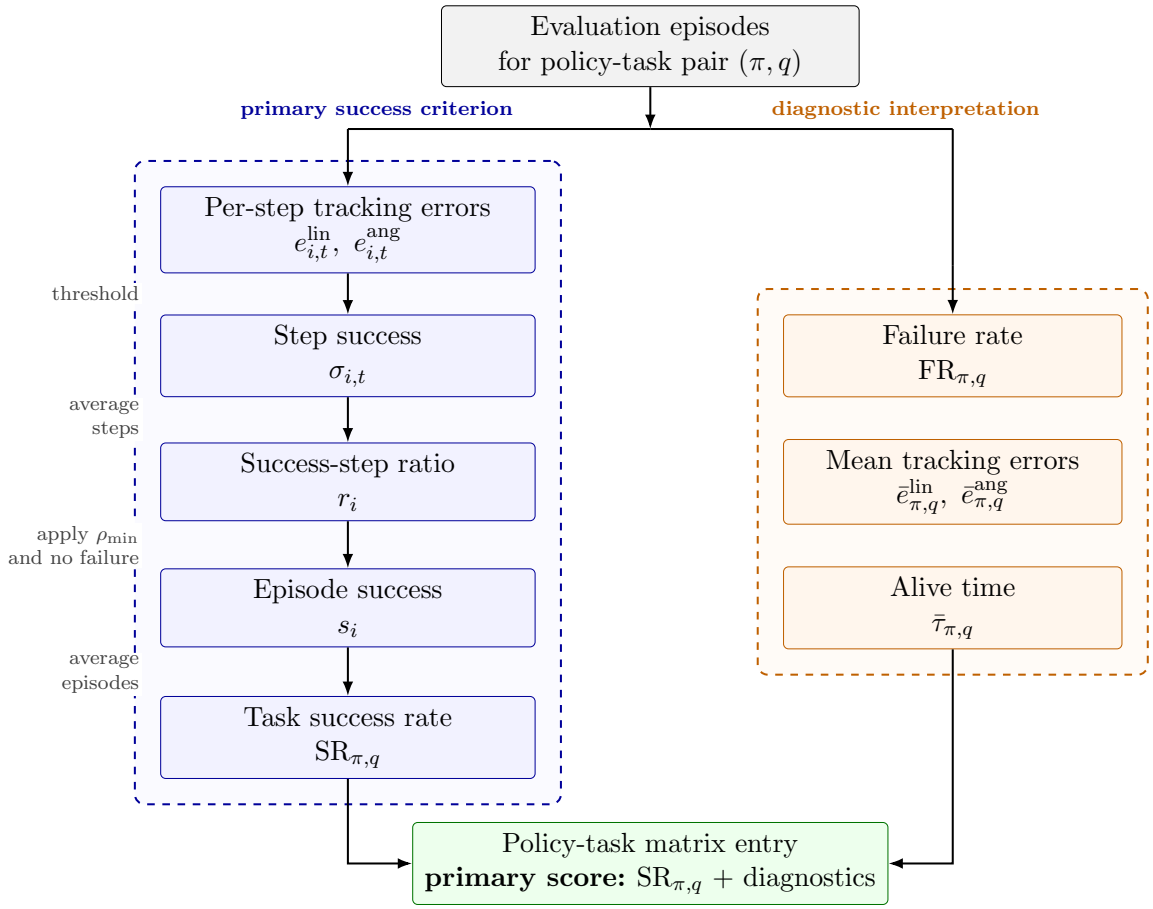


Figure 5: Evaluation structure for a policy-task pair (π, q), where q denotes the evaluation task and t denotes the timestep within episode i. The primary branch computes per-step tracking errors $e_{i,t}^{\text{lin}}$ and $e_{i,t}^{\text{ang}}$, converts them into step success $\sigma_{i,t}$, averages them into the success-step ratio r_i, and then applies the episode-level threshold $\rho_{\min}$ and failure condition to obtain episode success s_i. Averaging s_i over completed episodes gives the task success rate $\text{SR}_{\pi,q}$. The diagnostic branch reports failure rate $\text{FR}_{\pi,q}$, mean tracking errors $\bar{e}_{\pi,q}^{\text{lin}}$ and $\bar{e}_{\pi,q}^{\text{ang}}$, and mean alive time $\bar{\tau}_{\pi,q}$ from the same evaluation episodes.

4.1.1 Experimental Procedure

The experimental procedure consisted of two stages. First, single-task specialist policies were trained to establish the baselines for each individual terrain type. Then, a unified multi-task policy was trained on the combined benchmark environment to test whether one shared controller could cover the same task set. As described in Section 3.3, the multi-task training was organized into phases to control task interference. In practice, the non-focused proportions were adjusted manually because of differences in task difficulty (Table 6).

Phase	Profile	Flat / Rough / Stairs / Gap	LR	Iters
P0 balanced	`p2_b2safe`	35 / 35 / 15 / 15 %	3×10^{-4}	≈1500
P1 B2 step-up	`p1_b2_stepup_retain`	25 / 35 / 25 / 15 %	1×10^{-4}	≈500
P1 B2 ramp	`p1_b2_ramp`	10 / 10 / 75 / 5 %	3×10^{-4}	≈100
P2 all-recover	`p2_b2safe`	15 / 25 / 40 / 20 %	1×10^{-4}	≈200

Table 6: MTL phase schedule used during training. Each row reports the selected phase profile, terrain sampling proportions for flat, rough, stair, and gap terrains, learning rate, and approximate iteration budget.

The profile column refers to environment overrides corresponding to a specific phase. A phase profile could change the command ranges, terrain curriculum level, reset conditions, and selected rewards in accordance with phase demands. The P1 B2 Step-up profile used an easier stair curriculum that reduced height to make B2 non-zero. The P2 all-recover profile restored broader command ranges for non-stair terrains and kept some B2 support. Because changing the sampling distribution was insufficient for the harder stair task, these overrides were necessary to escape the trap that initially locked B2 at 0%. The policy was given a reachable stair objective, followed by a ramp phase that brought it closer to the benchmark stair setting. A more detailed summary of the phase profile overrides is provided in Appendix C.4, Table 15.

Furthermore, the focused probability increased exposure to one particular terrain type, while the rehearsal probability kept the remaining types present to avoid catastrophic forgetting. The exact run folders and selected checkpoint files for each seed are provided in Appendix C.4, Table 16.

MTL evaluations were run for both the final iteration and the saved checkpoints. Learning curves were used to better monitor training development, and individual checkpoints were cross-evaluated to verify whether reward trends matched expected behaviors.

For each phase, candidate checkpoints were evaluated on the full benchmark. Selection was performed at the policy level (i.e, whenever a checkpoint was selected, it was treated as one complete policy). The entire checkpoint was either accepted or rejected, and task scores were not assembled from different checkpoints. A higher priority was given to policies that improved the targeted terrain. For example, during the B2-focused phase (P1), checkpoints were selected when stair performance improved, matching the objective of the current phase. Some degradation on the remaining terrains was accepted, but only when the retained tasks stayed reasonably close to their performance prior to the most recent run. A P1 checkpoint could be accepted if B2 performance improved substantially while A1, A2, and B1 decreased marginally (e.g., by roughly 2-5 percentage points in success rate). Conversely, a checkpoint was rejected if the B2 improvement proved detrimental to a previously retained terrain (e.g., a drop of 20-30 percentage points in success rate).

The value of 0.70 was used as a retention target. It was chosen as a lower bound for considering a previously acquired task to be retained. The precise value was informed by the single-task specialist evaluations that preceded the multi-task experiments. The MTL policy was expected to retain a substantial fraction of the task-specific performance, and scores far below the 0.70 mark were treated as signals of forgetting. However, the threshold was not treated as a strict selection criterion. Some tasks, especially C2, never achieved it, so checkpoint selection was, for the most part, based on observed trade-offs between focused improvement and cross-task retention.

4.2 Results

The results are organized into three subsections. First, Section 4.2.1 examines how the single-task specialists and the MTL policy trained over time . Then, Section 4.2.2 reports the final evaluation performance of each specialist on its own tasks. This baseline is finally compared to the multi-task policy in Section 4.2.3.

4.2.1 Training Dynamics

The phased curriculum successfully acquired 3/5 locomotion skills trained simultaneously, but revealed dependence on task difficulty. Policies trained on flat terrain converged quickly and transferred relatively well. Performance on rough terrain also proved robust. By contrast, the two demanding tasks (stair climbing and gap crossing) were the hardest skills to retain in a single policy.

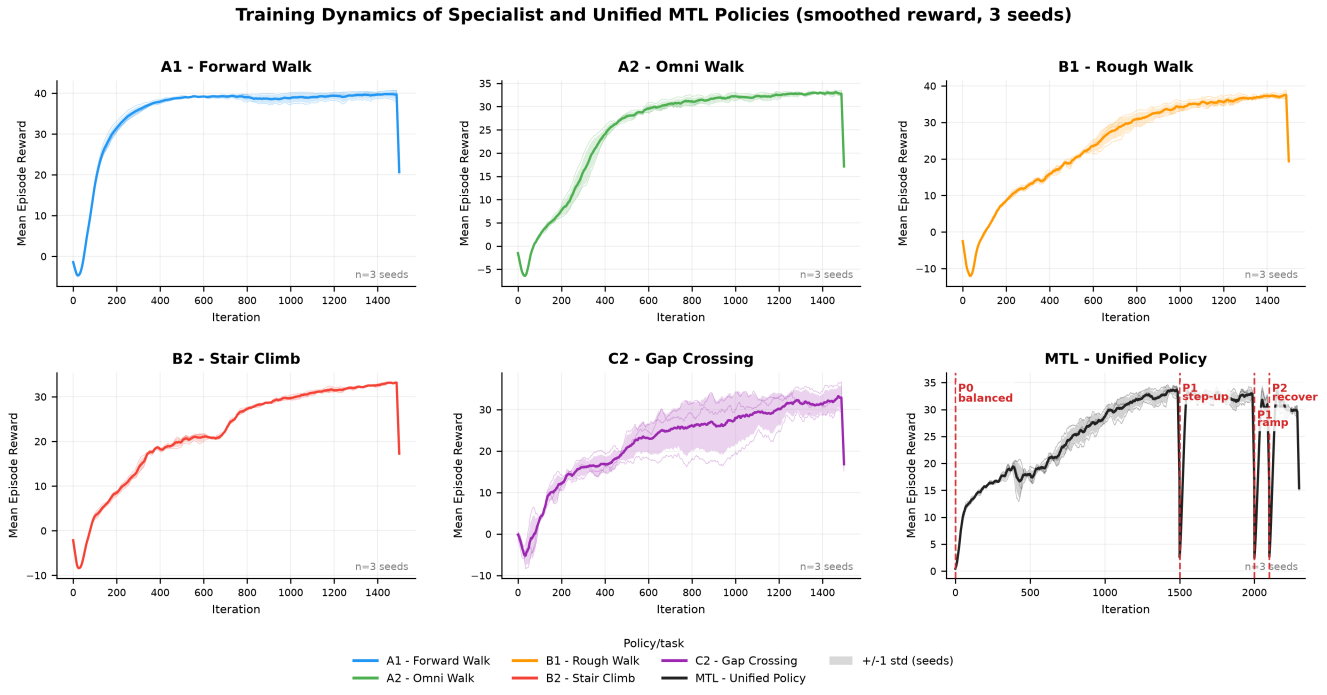


Figure 6: Training dynamics of the single-task specialists and the unified MTL policy. Each subplot shows the smoothed mean episode reward across three seeds, with shaded regions indicating variation between seeded runs. Dashed red lines in the MTL subplot indicate phase transitions.

Before turning to the MTL dynamics, it is worth noting what the single-task baselines have established. Figure 6 shows that A1, A2, and B1 converged smoothly across all three seeds, reaching high performance well before training concluded. B2 converged more slowly but achieved near-perfect success. C2 showed noticeably higher seeded variance, as the greater difficulty of gap crossing likely took a toll. Small differences in early exploration influenced how consistently successful crossing behavior was discovered, so a more precise contact strategy was needed for the policy to start receiving useful feedback.

The MTL curve shows a different pattern. Unlike the single-task specialists, which represent uninterrupted runs, the MTL subplot concatenates checkpoints from selected *successive* phases. The drops in reward correspond to phase transitions that changed the sampling distribution and training objective. In some phases, the command ranges and rewards were also adjusted (see Appendix C.4, Table 15), so reward values at phase boundaries are not directly comparable. Although this limitation concerns the interpretation of raw rewards, it did not harm the overall reproducibility or the task comparison, which was based on the protocol introduced in Section 4.1.1. Instead of being tuned manually for individual seeds, the same adjustments were applied consistently across all runs. Moreover, the setup did not include an online curriculum scheduler or a mechanism for resolving gradient conflicts to smooth these transitions. As a result, the policy was exposed to a new optimization setting that briefly reduced the average reward, making the MTL curve less smooth. This instability, however, primarily came from the policy having to re-optimize under changing conditions.

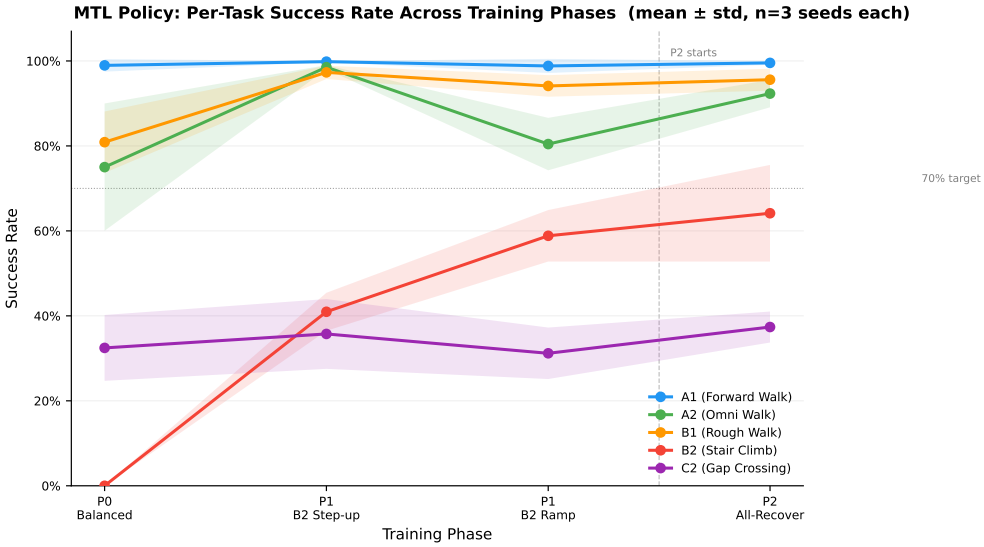


Figure 7: Success rates of the MTL policy at the end of each training phase (mean ± std, n=3 seeds per phase). B2 rises from 0% at P0 to 64% by P2 due to focused sampling. A2 temporarily interfered but recovered successfully during P2. The dotted line marks the approximate 70% target threshold. C2 remains below target. No dedicated phase was allocated to it in the current setup.

Figure 7 shows the MTL policy's per-task success at each phase endpoint. For the phase definitions and terrain sampling proportions, see Table 6. Throughout P0, B2 sat at 0% despite being included in the training (15% probability). This probably relates to the simplicity bias issue discussed earlier [13]. The policy converged easily on A1/A2 but struggled with the harder

stair-climbing objective. At 15% sampling probability, the gradient contribution from stair rollouts was very small, even though 4096 environments were used. With limited exposure, the shared parameters were pulled toward the easiest tasks, preventing B2 from achieving a non-zero score. A2 showed large variation across seeds, which suggested sensitivity to initialization.

For the phase analysis below, let π_p^{MTL} denote the MTL checkpoint selected at phase p, and $\mathrm{SR}_{\pi_p^{\mathrm{MTL}},q}$ be the success rate of the MTL checkpoint from phase p on task q. Changes between phases are reported as

$$\delta_q^{p_a \to p_b} = \mathrm{SR}_{\pi_{p_b}^{\mathrm{MTL}},q} - \mathrm{SR}_{\pi_{p_a}^{\mathrm{MTL}},q}, \tag{11}$$

where values are given in percentage points (pp). This notation is separate from the retention difference Δ_q in Equation 10 that compares the final MTL policy against the single-task specialists.

During P1, B2's sampling probability was raised to 25% and then to 75%. This made B2 non-zero, increasing success from 0% at P0 to $\approx 41\%$ after P1 Step-up ($\delta_{\mathrm{B2}}^{\mathrm{P0}\to\mathrm{P1\ Step\text{-}up}} \approx +41$ pp), then to $\approx 59\%$ after P1 Ramp ($\delta_{\mathrm{B2}}^{\mathrm{P1\ Step\text{-}up}\to\mathrm{P1\ Ramp}} \approx +18$ pp). Focused sampling can therefore be considered an effective mechanism for prioritizing underperforming tasks, but success did not come without cost. A1 and B1 consolidated above 93%. The most noticeable impact was on A2. It dropped from $\approx 99\%$ after P1 Step-up to $\approx 80\%$ after P1 Ramp ($\delta_{\mathrm{A2}}^{\mathrm{P1\ Step\text{-}up}\to\mathrm{P1\ Ramp}} \approx -18$ pp), of which gradient interference was the likely cause. Because B2 was given forward-only velocity commands to discourage the agent from backing away from stairs, its updates did not provide a gradient signal for lateral backward motion.

The all-recover phase (P2) restored A2 back to $\approx 92\%$ ($\delta_{\mathrm{A2}}^{\mathrm{P1\ Ramp}\to\mathrm{P2}} \approx +12$ pp), while B1 stabilized at $\approx 96\%$ ($\delta_{\mathrm{B1}}^{\mathrm{P1\ Ramp}\to\mathrm{P2}} \approx +2$ pp). Forgetting was only partial rather than catastrophic, and the omni-directional skills proved recoverable. B2 also increased slightly further to $\approx 64\%$ ($\delta_{\mathrm{B2}}^{\mathrm{P1\ Ramp}\to\mathrm{P2}} \approx +5$ pp).

However, this told a more nuanced story. B2 stayed below the 70% target, but the jump from zero during early training confirmed that the phased approach, although imperfect, was effective. B2 remained below target, while A2 did not rise back to its earlier P1 Step-up score. The difference between A2 and B1 is also worth discussing. A2 dropped about 18 pp, whereas B1 dropped only 3 pp. This leads to the conclusion that B2 interfered more strongly with omnidirectional command tracking than with traversing the rough terrain. Under A2, the robot is expected to track a broader command range than the forward-biased climbing behavior that B2 reinforced. The smaller drop on B1 suggested a relation between rough locomotion and stair climbing. Although B1 used an omnidirectional command range (the same as A2), it shares much of the forward locomotion and balance structure needed for stair climbing (e.g., balance recovery and forward progression over uneven terrain). These overlapping requirements likely made B1 less vulnerable to increasing B2 exposure, providing evidence that organizing tasks into families influences retention.

Importantly, this does not mean that grouping itself improves retention. The training distribution and shared updates drive changes in performance. Instead, it provides the analytical ground for interpreting retention patterns and shows that the benchmark families are not arbitrary. With the family structure, performance drops can be attributed to specific mismatches between task setups, meaning that grouping adds value by making the observed retention losses easier to explain.

C2 stayed at $31 - 37\%$, having received no dedicated phase. This is consistent with agility tasks requiring concentrated exposure [2]. Gap crossing demands the robot to place its feet over a void, which is not expected on continuous terrains. Therefore, the remaining terrains did not provide a

basis for this skill to be learned reliably, regardless of sampling pressure.

Overall, the phase-level results showed that focused sampling helped improve performance on the targeted tasks. However, temporary interference with omnidirectional tracking was also observed. The effect of these trade-offs is evaluated against the single-task specialists in Section 4.2.3.

4.2.2 Single-Task Performance

Table 7 reports the aggregated diagonal performance of the single-task baselines, with each specialist evaluated on its own task. These baselines gave a reference point against which the later MTL policy was compared.

Policy / task	Success rate (%)	Failure rate (%)	Alive time (s)	Linear error
A1 Forward	98.7 ± 1.6	1.3 ± 1.6	19.9 ± 0.1	0.038 ± 0.006
A2 Omni	96.5 ± 1.7	2.4 ± 1.3	19.8 ± 0.2	0.075 ± 0.004
B1 Rough	96.6 ± 1.6	3.2 ± 1.6	19.7 ± 0.2	0.111 ± 0.006
B2 Stairs	100.0 ± 0.1	0.0 ± 0.1	20.0 ± 0.0	0.078 ± 0.002
C2 Gap	89.9 ± 5.8	9.6 ± 5.7	18.8 ± 0.8	0.088 ± 0.010

Table 7: Aggregated single-task specialist performance on the five training tasks. Values are averaged over three seeds and reported as mean ± standard deviation.

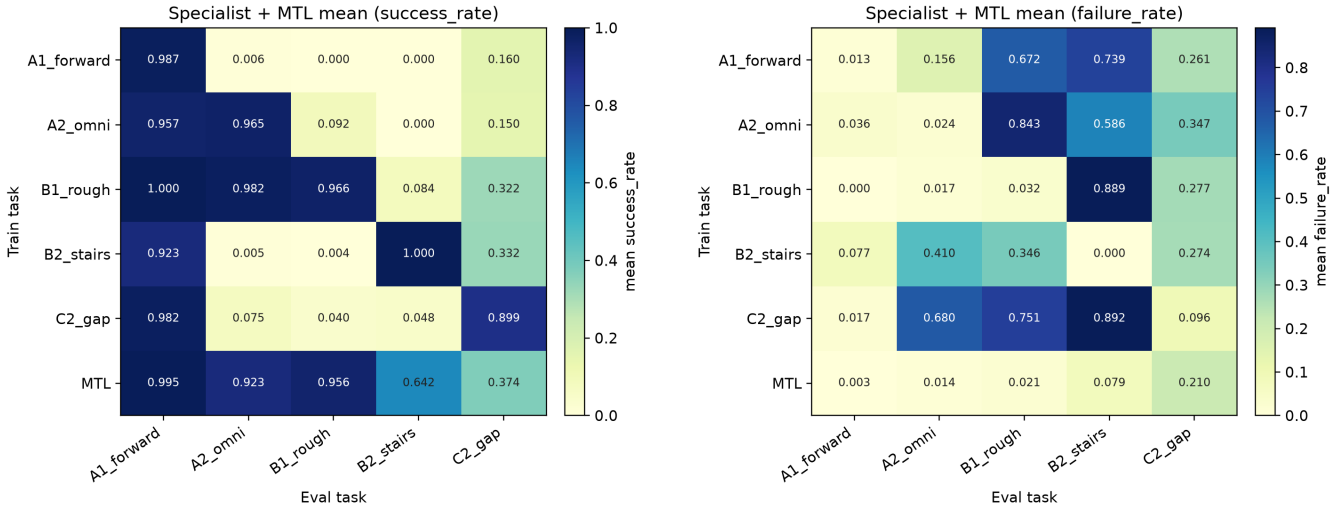
All specialist policies successfully learned their corresponding tasks, with A1, A2, B1, and B2 reaching success rates above 96%. Failure rates were also low, and alive times were close to the episode limit, indicating that the policies both satisfied the success criterion and maintained stability over full rollouts. These metrics also confirmed that the specialists provided a meaningful reference for the subsequent analysis and exposed differences in task difficulty. Despite being one of the hardest terrains, B2 achieved a near-perfect score, as its constrained forward gait and posture corrections reduced tracking demands compared to omnidirectional tasks.

C2 achieved a lower success rate of 89.9% and showed larger variance across seeds. This was expected because the task tested agility, requiring the robot to move forward at the right moment, keep its balance, and recover after landing. This meant that even small differences in timing or posture could lead to failure. The full cross-evaluation matrix is discussed in the next subsection, where off-diagonal entries are used to analyze transfer between task families.

4.2.3 Multi-Task Retention and Interference

Unlike Section 4.2.1, which covered *longitudinal dynamics* (how performance varied across training phases), this section uses the full cross-evaluation matrix to reveal which policies generalize where and why. On the one hand, the matrix showing policy success rates is used to identify transfer and retention patterns. On the other hand, failure rates reveal the cause of low success. As discussed in Section 4.1, this distinction is especially important because low success can arise for different reasons discussed in detail later in this subsection.

As a first step, single-task observations were used as probes for interpreting the later MTL strategy. They indicated which skills could be learned to support other tasks and which ones might interfere. For example, the B1 specialist transferred well to A1 and A2, suggesting that training on rough terrain taught a locomotion strategy that also worked on flat ground.



(a) Mean success rate

(b) Mean failure rate

Figure 8: Aggregated cross-evaluation matrices for the single-task specialists and the unified MTL policy. Values are averaged over three seeds.

Figure 8a first exposes the transfer pattern, which forms the backbone of this analysis. B1 proved to be the best-performing specialist, achieving 100% on A1, 98% on A2, and 32% on C2. Training on rough terrain produced a more general policy because the robot had to learn to handle varying ground heights while maintaining velocity tracking. However, the reverse was not true. With its random height variation, the rough terrain forced the policy to learn a more robust gait and exposed the weaknesses of gaits optimized for easier terrain. A robot trained on flat ground could not adapt its stepping to the uneven surface. This is a direct consequence of environment design choices, as terrain difficulty determined the kind of locomotion behaviors the policy had to develop. Taken alone, command space did not contribute much to this effect. A2 used a wider range than A1 but still failed on rough terrain. This asymmetry is important because it tells us something concrete about environment design. The choice of training terrain influences the generalization budget of the resulting policy. It also explains why A1 transferred almost nowhere outside its own task. Forward flat walking is the most restricted skill in the benchmark. It requires the narrowest range of physical behaviors, which means the policy had little to carry over to terrains that demanded more physical effort.

B2 transferred well to A1 (92%) but collapsed on A2 (0.5%) and B1 (0.4%). Because stair climbing used forward-only velocity commands, the specialist never encountered lateral and backward motion. The policy walked sufficiently well on a plane in the forward direction, but failed immediately under A2's omnidirectional commands and B1's uneven ground. C2 success was only partial (33%). This indicated that the forward locomotion learned for stairs satisfied the crossing criterion, but without a reliable crossing strategy.

C2 transferred well to A1 (98%) but poorly on all the remaining off-diagonal tasks. The policy was initialized from an A2 walking checkpoint and finetuned on gap terrain. This points to a contradiction: since C2 started from A2, one might expect it to retain its capability in this respective task. However, gap crossing used a much narrower command range (primarily forward velocity), so the lateral and backward commands that A2 defined were never reinforced. The policy effectively

forgot them, retaining only the forward walking behavior.

Regarding failure rate patterns, Figure 8b reveals two distinct causes of underperformance that success rate could not resolve: (i) a low success rate combined with a high failure rate suggested physical instability (the robot fell before the episode ended), (ii) a low success rate paired with a low failure rate meant the robot survived but did not satisfy the tracking criterion.

Figure 9 completes the failure rate matrix by making this distinction clearer. It decomposes evaluations into three episode outcomes (successful completion, survival without task completion, and failure termination). This helps distinguish cases where policies physically collapse from those that survive but fail to satisfy the task criterion.

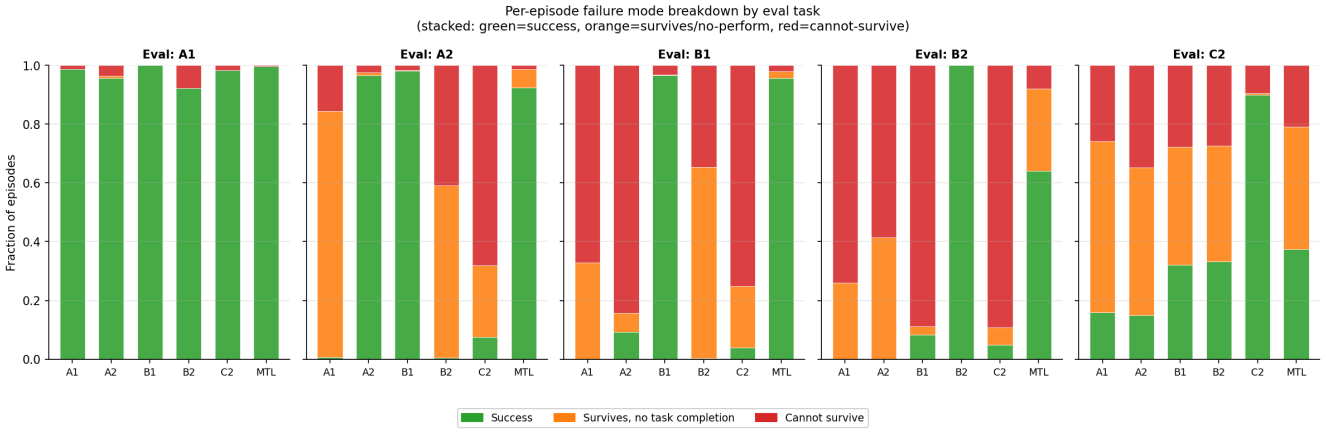


Figure 9: Per-episode evaluation outcome breakdown by evaluation task. Each policy-task pair was evaluated over at least 1024 completed episodes per seed, and bars aggregate the outcomes across the three seeds. Green indicates episodes counted as successful, orange indicates episodes where the robot survived but did not satisfy the task-success criterion, and red indicates termination due to failure.

That said, the two failure modes identified earlier call for different explanations:

- **A1 and A2 on B1 terrain:** failure rates of 67% and 84%, respectively. Both policies collapsed on rough terrain because the gait could not handle height variations, suggesting physical incompatibility with the terrain.
- **B1 on B2 stairs:** although the B1 specialist performed well on its own task (96.6% success) and transferred strongly to flat tasks (100% and 98% success on A1 and A2, respectively), it failed on stairs (89% failure rate). This was primarily a "cannot survive" case, suggesting that stair climbing required behavior that could not be inferred from rough terrain.
- **B2 on A2:** 41% failure. The B2 specialist destabilized when asked to track lateral and backward commands. However, the breakdown also shows a mixed failure mode. The policy often survived but did not complete the task. At the same time, a substantial part of the episodes still ended in failure. From this, we can conclude that the main issue was not only physical instability. B2's gait could not recover when given unfamiliar commands due to coverage mismatch.

- **C2 on A2 and B1:** 75% and 68% failure, respectively. Gap crossing preserved forward locomotion, but it was incompatible with lateral and backward commands. In both cases, the high failure rate showed a lack of robustness of the transferred gait.
- **MTL across all tasks:** failure rates of 0.3%, 1.4%, 2.1%, 7.9%, and 21% on A1 through C2. The MTL policy underperformed on C2 but did not collapse. Similarly, Figure 9 reveals a large "survives without success" component on that task. This is a different failure mode from those highlighted above. The robot remained physically stable but did not develop a crossing strategy, showing that survival does not necessarily imply task completion.

Failure rate, therefore, adds an important dimension to the analysis. A high success rate with a low failure rate marks a capable policy. A low success rate and a low failure rate indicate a policy that survives but is unable to complete the tasks (e.g., the MTL policy on C2). A low success rate with a high failure rate suggests physical incompatibility with the terrain. Therefore, reporting only the success rate would mask the last two cases and paint an incomplete picture of what future improvements are needed.

The MTL row is the only row in the matrix with non-trivial performance on all five tasks. Every specialist has at least one zero or near-zero off-diagonal cell. The MTL policy achieved the most uniform coverage (99.5%, 92.3%, 95.6%, 64.2%, and 37.4%) with no zeros and lower failure rates. This broader coverage is the most distinctive property of the MTL policy and could not be achieved by any other specialist. However, simply retraining a specialist on a new terrain at full sampling probability would likely cause it to forget the original task (e.g., the C2 specialist failed when evaluated on task A2 despite being finetuned using an A2 checkpoint as a starting point). Table 8 places this comparison in perspective by reporting diagonal performance of each specialist against the MTL policy.

Task	Specialist success (%)	MTL success (%)	Δ_q (pp)
A1 Forward	98.7 ± 1.6	99.5 ± 0.6	$+0.8$
A2 Omni	96.5 ± 1.7	92.3 ± 4.0	-4.2
B1 Rough	96.6 ± 1.6	95.6 ± 3.2	-1.0
B2 Stairs	100.0 ± 0.1	64.2 ± 13.9	-35.8
C2 Gap	89.9 ± 5.8	37.4 ± 4.5	-52.5

Table 8: Comparison between single-task specialist performance and the unified MTL policy on each benchmark task. Values are averaged over three seeds.

On A1, A2, and B1, the MTL policy remained within 5 pp of the corresponding specialist ($\Delta_{A1} = +0.8$ pp, $\Delta_{A2} = -4.2$ pp, and $\Delta_{B1} = -1.0$ pp), indicating that multi-task training caused little harm in cases when tasks shared similar locomotion demands. On B2 and C2, the deficit was more pronounced ($\Delta_{B2} = -35.8$ pp and $\Delta_{C2} = -52.5$ pp, respectively). This reflects the structural differences between these tasks and the remaining training distribution. The implications of these patterns are discussed in Section 5.

5 Conclusions and Further Research

Together, these findings provide sufficient evidence in support of the research questions. Concerning **RQ1**, the matrix showed that environment design choices (terrain difficulty and command coverage) determined which off-diagonal cells were non-trivial. Likewise, some tasks with broader command coverage transferred to simpler settings (e.g., A2 to A1 and B1 to flat tasks). Cross-family transfer was asymmetric and influenced by difficulty, with harder terrains generalizing downward and easier terrains not generalizing upward. Building on the retention patterns analysis discussed in Section 4.2.3, this further confirmed that structuring related tasks by task families and difficulty is a meaningful design choice, as it determines what the policy learns and what it does outside its training distribution. This distinction also clarifies how the benchmark could be combined with related approaches for environment variation, such as domain randomization. Here, the purpose was not to ensure robustness to arbitrary noise, but to make transfer and interference interpretable. For these to be made measurable, a practical decision had to be made, and the benchmark was organized around identifiable locomotion demands. Strategies for determining whether retention remains stable beyond the benchmark settings are discussed further in Section 5.3.

Concerning **RQ2**, MTL improved breadth but did not exceed specialist performance on all counts. On A1, A2, and B1, the policy was within 5pp of the specialist. On B2 and C2, the deficit was larger. Therefore, the shared policy covered compatible tasks almost perfectly but could not substitute for specialists on the hardest tasks.

These results also speak to the hypotheses in Section 1. **H1**, which stated that a shared observation and action interface would improve transfer between similar tasks, was partially confirmed. The shared interface revealed transfer along the family axis, but did not itself cause cross-family generalization. That was primarily determined by task difficulty, not the interface per se. **H2**, which stated that the MTL policy would retain better cross-task performance than specialists at the cost of some peak performance, was also partially confirmed. On one side, the breadth advantage is clear. The MTL policy was the only configuration that achieved non-trivial performance on every task. At the same time, the trade-off on B2 and C2 was larger than the hypothesis anticipated, suggesting that the policy reached its limits before the harder tasks were fully resolved.

To connect these results back to the thesis contributions, the study successfully provided a controlled evaluation case that demonstrated the relationship between task organization and multi-task locomotion performance. Under a shared representation and evaluation protocol, transfer proved strongest when tasks required compatible locomotion strategies. This makes the benchmark useful as a diagnostic tool, as it exposes retention patterns and maps concrete design choices to barriers in transfer. That said, the main contribution is not that the MTL policy universally outperformed specialists, but that the proposed benchmark made the trade-offs discussed easier to identify and measure.

5.1 Discussion

Reflecting on these findings, one result stood out in particular. The same challenge introduced in Section 2.2, namely task-specific updates pulling shared parameters in different directions (also known as gradient interference), was most visible during the B2-focused phase. Although the results suggest that phased training helps mitigate forgetting, the effect of this intervention could not

be fully isolated from other variables. For example, A2's performance drop during P1 coincided with reduced A2 sampling. However, its subsequent recovery in P2 showed that the forgetting was not catastrophic. The policy regained omnidirectional capability once a balanced distribution was restored, suggesting that enough plasticity was retained. This supports the conclusion that phased training with rehearsal can mitigate forgetting even though it cannot completely eliminate it on its own.

Simplicity bias also challenged balanced learning. During P0, B2 remained at 0% despite 15% sampling, which aligns with the mechanism described in Section 2.2. Phased training was a practical necessity. Without it, B2 would likely not have been learned within this setup since easier tasks dominated the gradient signal. Although not optimal, this approach achieved its goal of introducing all tasks together and was sufficient for verifying whether a unified policy could cover the benchmark. By extension, environment design, separate from the learning algorithm, determines policy generalization budget. The hard-to-easy asymmetry in the transfer matrix confirms this, suggesting that changing the optimizer or architecture would not close this gap. The benchmark structure imposes a transfer ceiling (e.g., if a task never required stair contact, the resulting policy would have little incentive to produce those behaviors at evaluation time). Therefore, the gap will persist as long as the training distribution does not expose the policy to the missing demands [3].

Task conditioning gave the shared policy explicit information about the terrain context. No comparison was made between conditioned and unconditioned MTL policies, so the results cannot formally quantify how much of the final performance was actually attributable to the presence or absence of the task identifier. The task label was defined as part of the experimental setup, separate from the thesis's contributions. It was intended to help the shared policy distinguish which task distribution it was operating under, but an ablation would be required to confirm this effect.

What the conditioned setup did provide was experimental control. It allowed the same policy architecture to be evaluated against different task distributions while making the active task context explicit. This made the comparison itself cleaner. When the MTL policy performed worse on B2 than on A2, this could be interpreted as an interference effect (not as a failure to infer whether the current episode required one locomotion mode or another). At the same time, it assumed labels were known at inference time, which is unrealistic outside controlled environments. For example, a robot deployed outdoors does not know in advance whether it is about to cross a gap or traverse uneven ground, in which case the label would have to be predicted from observations. This introduces an additional inference step not addressed in this study.

It is also worth noting that the MTL policy developed a more conservative gait compared to single-task policies. This is consistent with prior findings that shared policies trained on competing objectives tend to converge on compromise strategies [9]. In this context, the trade-off became visible through reduced aggressiveness in velocity tracking and a more cautious stepping pattern, both of which are documented consequences of multi-task training.

Regarding C2, gap crossing was weak in MTL despite previously acquiring a strong specialist (89.9%). Agility tasks reveal a boundary case that standard locomotion would have otherwise missed. Aside from difficulty, C2 failed because it required behavior that is not reinforced by any other terrain. No amount of sampling on flat or rough terrain could teach the robot to clear a gap if those terrains did not demand the skill to be learned in the first place. This means that sampling pressure cannot compensate for task neglect when the missing skill does not overlap with the rest of the training distribution.

Finally, task grouping implications were observed. In particular, A2 transferred to A1, but A1

did not transfer back to A2. B1 reached only about 8% success on B2, despite sharing a family label. This demonstrated that a family label based on terrain type does not guarantee that tasks will support each other during training. Difficulty and behavioral similarity, therefore, matter as much as terrain category [17].

5.2 Limitations

Earlier, the importance of reproducibility was discussed in Section 2.4. To acknowledge this, seeds were used for establishing consistent trends across runs. With $n = 3$ seeds and a manually designed curriculum, results provided statistical evidence, but more seeded runs would be needed to claim significance. This limitation is understood and accepted within the scope of this work. Because all three seeds produce consistent outcomes, this does not invalidate the observed patterns, but it does mean the effect sizes are approximate and should be interpreted critically.

Additionally, all training and evaluation were conducted using Isaac Lab on the Unitree Go2. A standard practice in the robotics literature is to validate policies across multiple simulators or on physical hardware [10]. Due to time constraints, a multi-platform evaluation was not feasible, so restricting it to a single simulator and platform was necessary to ensure completion. Therefore, generalization to other simulators, morphologies, or physical hardware remains untested. This was a deliberate decision to prioritize depth of analysis over coverage of alternative configurations. For the same reasons, the task set was also reduced to five terrains.

For the MTL curriculum, phase transitions and checkpoint selection were inspected manually using success curves and evaluation snapshots. However, the decision to move from one task to the next was not produced by an automatic scheduler. As such, the reported curriculum is reproducible as a fixed protocol, but the process of discovering it was largely manual. Different transition points or checkpoint choices could have led to different trade-offs between focused improvement and retention of rehearsed tasks.

Another limitation is the lack of a gradient conflict resolution mechanism to mitigate task interference. It was managed primarily through terrain sampling proportions, with gradient conflicts between tasks not being detected and optimized for. Therefore, it is difficult to determine how much of the B2 and C2 performance deficit is caused by optimization interference or curriculum design choices.

5.3 Further Research

Starting with the curriculum design limitation, automatic curriculum scheduling needs to be considered. It would replace the manual phase transitions and enable targeted recovery. Task success rate and adjusted terrain sampling dynamics would be resolved dynamically, reducing reliance on manual evaluation [35, 36]. This would also remove the seed 0 dependency and make the curriculum reproducible.

Applying PCGrad [12] or MGDA [37] is another improvement that addresses conflicting task gradients at the update step. This would isolate the effects of optimization interference, making it possible to attribute performance deficits more precisely. Along with gradient conflict resolution, the manual task ID would be replaced with a terrain classifier trained from observations. A learned context encoder would eliminate the need for predefined labels and allow for task identity to be inferred upon deployment.

The full sequence would also be run across more seeds, giving the analysis a stronger statistical grounding. Confidence in the observed asymmetries would then also be stronger, particularly for smaller effects (e.g., A2 retention deficit). In addition, the benchmark could be combined with domain randomization, varying parameters such as friction, mass, or sensor noise [34]. This would extend the present task variation to capture sim-to-real robustness before evaluating the policy on real hardware. If possible, the MTL policy would then be transferred to the physical Unitree Go2 to provide a more realistic assessment of whether the breadth advantage and conservative gait persist under real-world dynamics, as well as a measure of the sim-to-real gap for a potential follow-up study.

Finally, the task set would be extended with additional agility terrains that test whether the observed patterns (hard-to-easy transfer asymmetry, C2 as an outlier, gradient incompatibilities) are properties of multi-task locomotion learning or specific to the chosen configuration. Possible extensions include stepping stones (C1, excluded from the current set), discrete obstacle fields, wave terrain, and bridge traversal. Such terrains create a demand for different skills to be developed and would evaluate retention and cross-task transfer against a broader distribution of distinct tasks.

Multi-task locomotion learning is a study of competing pressures, as environment design, task compatibility, and gradient dynamics interact to support intelligent behavior. Without pragmatic judgment on each of these, this delicate balance would be easily disrupted by varying task difficulties and conflicting objectives. Environment design should be deliberate, not to put a ceiling on policy performance. These findings are therefore only a starting point for the targeted improvements that would enable the approach to overcome its current limits. The aforementioned directions define a clear path ahead. The immediate priority will be to make the training process less manual, replacing phase transitions with automatic curriculum scheduling and adding a mechanism to resolve conflicting tasks. Then, task coverage will be broadened to test whether the observed asymmetries persist within the updated benchmark. Having achieved robustness in simulation, the final step will be to apply domain randomization and deploy the policy on the physical Unitree Go2, aiming to confirm that the reported breadth advantage extends beyond the controlled benchmark.

This thesis provides a controlled case study of retention patterns in multi-task quadruped locomotion. Future work should turn this into a more automatic and scalable pipeline.

References

[1] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1094–1100. PMLR, 2020.

[2] Vira Joshi, Zifan Xu, Bo Liu, Peter Stone, and Amy Zhang. Benchmarking massively parallelized multi-task reinforcement learning for robotics tasks. *arXiv preprint arXiv:2507.23172*, 2025.

[3] Daniele Reda, Tianxin Tao, and Michiel van de Panne. Learning to locomote: Understanding how environment design matters for deep reinforcement learning. In *Proceedings of the 13th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pages 1–11, 2020.

[4] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.

[5] Xinyang Gu, Yen-Jen Wang, and Jianyu Chen. Humanoid-gym: Reinforcement learning for humanoid robot with zero-shot sim2real transfer. *arXiv preprint arXiv:2404.05695*, 2024.

[6] Amazon Science. Holosoma: Humanoid robotics framework for reinforcement learning training, deployment, and motion retargeting. 2025. GitHub repository.

[7] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4:eaau5872, 2019.

[8] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems XVII*, 2021.

[9] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 4497–4507, 2017.

[10] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 91–100. PMLR, 2022.

[11] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114:3521–3526, 2017.

[12] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836, 2020.

[13] M. Cho, J. Park, S. Lee, and Y. Sung. Hard tasks first: Multi-task reinforcement learning through task scheduling. In *Proceedings of the 41st International Conference on Machine Learning*, pages 8556–8577, 2024.

[14] Yoshua Bengio, Jerome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM, 2009.

[15] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

[16] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[17] Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Efficiently identifying task groupings for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 27503–27516, 2021.

[18] Hanchi Huang, Deheng Ye, Li Shen, and Wei Liu. Curriculum-based asymmetric multi-task reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7258–7269, 2023.

[19] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.

[20] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

[21] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1889–1897, 2015.

[22] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[23] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870, 2018.

[24] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320, 2021.

[25] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 3207–3214, 2018.

[26] A. Patterson, S. Neumann, M. White, and A. White. Empirical design in reinforcement learning. *Journal of Machine Learning Research*, 25:1–63, 2024.

[27] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[28] Erwin Coumans and Yunfei Bai. PyBullet, a python module for physics simulation for games, robotics and machine learning, 2016. Technical report.

[29] Nathan Koenig and Andrew Howard. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2149–2154, 2004.

[30] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu. Dart: Dynamic animation and robotics toolkit. *Journal of Open Source Software*, 3:500, 2018.

[31] J. Hwangbo, J. Lee, and M. Hutter. Per-contact iteration method for solving contact dynamics. *IEEE Robotics and Automation Letters*, 3:895–902, 2018.

[32] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. In *NeurIPS 2021 Track on Datasets and Benchmarks*, 2021.

[33] Mayank Mittal, Philipp Roth, Jordan Tigue, Anthony Richard, Owen Zhang, Paul Du, Alberto Serrano-Munoz, Xiaoyu Yao, Ruedi Zurbrugg, Nikita Rudin, et al. Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

[34] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 23–30, 2017.

[35] Alex Graves, Marc G. Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1311–1320, 2017.

[36] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3732–3740, 2019.

[37] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

A Comparison of Robotics Simulators

The appendix provides a comparison of robotics simulators. The goal is to show the practical differences between earlier CPU- and later GPU-based simulation ecosystems.

Simulator	Main orientation	Execution model	Relevance
MuJoCo [27]	Physics engine for control and reinforcement learning tasks	CPU-based	An example of earlier simulators used before GPU-based training became common.
Bullet / PyBullet [28]	Physics simulator for robotics, games, and machine learning	CPU-based	An accessible option for running robotics experiments; scaling usually requires additional engineering.
Gazebo / ODE [29]	Robotics simulator often used with ROS setups	CPU-based	Useful for realistic testing; less suitable for large-scale reinforcement learning.
DART [30]	Physics engine for rigid-body simulation and robotics	CPU-based	Focused on accurate dynamics and modelling instead of large-scale parallel policy training.
RaiSim [31]	Fast rigid-body simulator often used for legged locomotion	CPU-based (with contact handling for speed)	An intermediate step between older CPU simulators and faster training pipelines.
Isaac Gym [32]	GPU-based simulator for robot learning	GPU-based parallel simulation	Allows many environments to run in parallel on the GPU.
Isaac Sim	Robotics simulator built on NVIDIA Omniverse	GPU-accelerated simulation	The simulator used in this thesis through the *Isaac Sim/Isaac Lab* ecosystem.

Table 9: Comparison of selected robotics simulators. The table outlines properties relevant to this thesis.

The comparison motivates the selection of GPU-based simulation for this thesis. Because the experiments required repeated policy training and evaluation, being able to run multiple environments in parallel was a practical reason for choosing the *Isaac Sim/Isaac Lab* setup.

B Benchmark Reward Configuration

This appendix reports the reward weights used for each task. The main benchmark definition is provided in Section 3.2.

Table 10 adds reward details needed to reproduce the task configuration. Terms marked with a dash were not active for that task. Base height targets for `base_height_l2` were: 0.38 m (A1, B1), 0.40 m (A2), 0.42 m (B2), and 0.37 m (C2).

Reward term	A1	A2	B1	B2	C2
`track_lin_vel_xy_exp`	1.0[a]	1.0[a]	2.0	1.5	2.0
`track_ang_vel_z_exp`	0.5[a]	0.5[a]	1.0	0.75	1.0
`feet_air_time`	0.25	0.25	0.25	0.04	0.08
`flat_orientation_l2`	−2.5	−2.5	−5.0	−2.0[b]	−5.0
`base_height_l2`	−5.0	−4.0	−10.0	−3.5[b]	−10.0
`undesired_contacts`	—	—	−1.0	−0.7	−1.0
`dof_torques_l2`	−1e−5	−1e−5	−1e−5	−2e−5	−1e−5
`dof_acc_l2`	—	—	—	−2.5e−7	—
`action_rate_l2`	—	—	—	−0.003	—
`dof_pos_limits`	—	—	—	−2.0	−0.5
`stand_still`	−0.15[c]	—	—	—	—
`stand_base_height`	−2.0[c]	—	—	—	—

Table 10: Reward term weights per benchmark task. Dashes indicate the term was not active.

[a] Inherited from the Isaac Lab `UnitreeGo2RoughEnvCfg` default.
[b] B2 was trained with posture-reward annealing (Section 3.3.1); values shown are the post-annealing weights.
[c] Becomes active only when the commanded velocity magnitude falls below $0.1\,\mathrm{m\,s^{-1}}$.

C Learning Setup Reproducibility Details

This appendix reports the training details that affect the results presented in this thesis. The methodology section (Section 3.3) describes the experimental logic, while the tables below summarize the main PPO settings, selected single-task checkpoints, and multi-task sampling phases.

C.1 Software and Compute Setup

Table 11 summarizes the software stack. The observation and action dimensions define the policy interface, and the environment count indicates the amount of experience collected.

Component	Setting
Simulator	*Isaac Sim / Isaac Lab*
RL library	RSL-RL PPO runner
Code repository	https://github.com/viktoriageorrg04/bsc-thesis-multi-task-RL-env
Robot asset	Unitree Go2
Action space	12 joint-position targets
Single-task observation dimension	235
Conditioned MTL observation dimension	239 (235 + 4-dimensional terrain-condition ID)
Single-task training environment count	1024
Local MTL phase environment count	1024
ALICE MTL phase environment count	4096
Evaluation environment count	128 parallel environments
Evaluation episodes	At least 1024 completed episodes per policy-task pair
Logging root	`logs/rsl_rl/<experiment>/<run>/`
Seeded evaluation output root	`results_seeded_1024/<policy>_s<seed>/summary.json`
MTL evaluation output root	`results_seeded_4096/MTL_s<seed>/<eval_dir>/MTL_unified/summary.json`

Table 11: Software and compute setup used for training and evaluation.

C.2 PPO and Stability Settings

Table 12 lists the PPO hyperparameters that were identical across all runs. Table 13 records the settings that differed between setups. The lower learning rate, reduced entropy coefficients, and log-space noise parameterization in the MTL runs serve as stability measures to mitigate task interference.

Setting	Value
Algorithm	PPO
Network architecture	MLP [512, 256, 128], ELU activation
Steps per env per update	24
Learning epochs per update	5
Mini-batches per epoch	4
Discount γ	0.99
GAE λ	0.95
PPO clip ε	0.2
Desired KL	0.01
Gradient norm clip	1.0
Value loss coefficient	1.0

Table 12: PPO hyperparameters shared across all single-task and multi-task runs.

Setting	Single-task	MTL phases
Learning rate	1.0×10^{-3}	Phase-dependent; see Table 6
LR schedule	Adaptive	Fixed
Entropy coefficient	0.01	0.001
Init. action noise std.	1.0 (scalar)	0.5 (log)

Table 13: PPO settings that differed between single-task baseline training and the conditioned MTL phases.

C.3 Seeded Single-Task Checkpoints

Table 14 records the single-task checkpoints used for the reported cross-evaluation.

Task	Result rows	Experiment folder	Selected checkpoint	Seeds
A1	`A1_forward_s{0,1,2}`	`unitree_go2_a1_legacy_1024_seeds`	`model_1499.pt`	0, 1, 2
A2	`A2_omni_s{0,1,2}`	`unitree_go2_a2_baseline_1024_seeds`	`model_1499.pt`	0, 1, 2
B1	`B1_rough_s{0,1,2}`	`unitree_go2_b1_baseline_1024_seeds`	`model_1499.pt`	0, 1, 2
B2	`B2_stairs_s{0,1,2}`	`unitree_go2_b2_local_apr15_recheck_1024_seeds`	`model_1440.pt`	0, 1, 2
C2	`C2_gap_s{0,1,2}`	`unitree_go2_c2_from_a2_refined_1024_seeds`	`model_2998.pt`	0, 1, 2

Table 14: Seeded single-task baseline checkpoints used in the reported cross-evaluation. Each row represents three independently trained policies, one per seed. The result rows correspond to folders under `results_seeded_1024/`.

C.4 Unified Multi-Task Phases

This appendix provides implementation details for the MTL phase schedule. Table 15 summarizes the phase overrides used during training, while Table 16 lists the corresponding run folders and selected checkpoints.

Profile	Purpose	Main overrides
`p2_b2safe`	Balanced recovery while retaining some B2 support.	Broader non-stair command ranges, moderate terrain curriculum level, reduced stair-specific support.
`p1_b2_stepup_retain`	Easier B2 bridge with rehearsal on retained tasks.	Reduced stair height, forward-biased stair commands, flat/rough rehearsal, increased stair stepping support.
`p1_b2_ramp`	Transition from easier B2 stepping toward benchmark stair climbing.	Higher stair emphasis, forward-biased stair commands, increased terrain level, B2 progress support.

Table 15: Summary of the phase profile overrides used during conditioned MTL training.

Phase	Seed	Run folder
P0 balanced	0	2026-05-25_22-37-17_p0_balanced_b2safe_s0_s0
P0 balanced	1	2026-05-26_08-56-34_p0_balanced_b2safe_s1
P0 balanced	2	2026-05-26_08-56-34_p0_balanced_b2safe_s2
P1 B2 step-up	0	2026-05-26_22-26-27_p1_b2_stepup_retain_from_p0_balanced_s0_s0
P1 B2 step-up	1	2026-05-26_23-34-04_p1_b2_stepup_retain_from_p0_balanced_s1_s1
P1 B2 step-up	2	2026-05-27_00-12-39_p1_b2_stepup_retain_from_p0_balanced_s2_s2
P1 B2 ramp	0	2026-05-31_10-00-23_p1_b2_ramp_clean_s0_s0
P1 B2 ramp	1	2026-05-31_15-47-38_p1_b2_ramp_clean_s1_s1
P1 B2 ramp	2	2026-05-31_18-56-11_p1_b2_ramp_clean_s2_s2
P2 all-recover	0	2026-06-01_01-53-44_p2_allrecover_final_s0
P2 all-recover	1	2026-06-01_01-53-51_p2_allrecover_final_s1
P2 all-recover	2	2026-06-01_01-54-48_p2_allrecover_final_s2

Table 16: Execution details for the unified MTL phases. The full folders are stored under `logs/rsl_rl/unitree_go2_mtl_conditioned_minruns/`. The final reported checkpoint was `model_2247.pt` (seeds 0 and 1); `model_2296.pt` (seed 2).