Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Robust Quadrupedal Locomotion on Complex Terrains via Adaptive Entropy Learning

Jiale Chen
Xijing University
Lingyun Kong
1400100383@qq.com

Xijing University
ZhenYao Zhang
Hangzhou Dianzi University
Zhipeng Xue
Xijing University

Research Article

Keywords: Reinforcement learning, Quadrupedal robot, PPO, Dynamic entropy coefficient, VHIP model, Terrain imagination, Sim-to-real

Posted Date: April 14th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9269338/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Categories (2), (3)

Robust Quadrupedal Locomotion on Complex Terrains via Adaptive Entropy Learning

Jiale Chen[1], Lingyun Kong[1*], ZhenYao Zhang[2], Zhipeng Xue[1]

[1*]School of Electronic Information, Xijing University, Xi'an, Shaanxi, 710123, China.
[2]School of Automation, Hangzhou Dianzi University, Hangzhou, Zhejiang, 310018, China.

*Corresponding author(s). E-mail(s): 1400100383@qq.com;

Abstract

Quadruped robots are increasingly required to operate in complex, unstructured terrains for tasks such as inspection and exploration. In harsh environments, exteroceptive sensors often provide only partial or noisy observations, making proprioceptive terrain estimation and motion stability even more critical. In addition, simulation-based policy training must efficiently balance exploration and exploitation. In this paper, we make two main contributions. (1) Building on the terrain-imagination framework (CENet) of DreamWaQ, we introduce stability-oriented rewards based on the Variable Height Inverted Pendulum (VHIP) model and a stand-still pose reward, improving both static and dynamic stability on complex terrains. (2) We extend PPO with a multi-metric dynamic entropy coefficient that adapts to performance gaps (velocity tracking and terrain utilization), yielding faster convergence and improved final performance in simulation. Ablation studies in MuJoCo show that VHIP rewards significantly reduce fall rates on challenging terrain levels; comparison with DreamWaQ in Isaac Gym shows statistically significant gains in linear velocity tracking and convergence speed. We further deploy the policy on a DeepRobotics Lite3 robot; real-world tests on stairs, rough and smooth surfaces, and grassland provide qualitative evidence of deployment feasibility.

Keywords: Reinforcement learning, Quadrupedal robot, PPO, Dynamic entropy coefficient, VHIP model, Terrain imagination, Sim-to-real

1 Introduction

Due to their agile mobility and environmental adaptability, quadruped robots have become core platforms for executing high-value tasks in complex unstructured scenarios (e.g., mountain inspection, rescue operations in ruins, and field exploration). Such tasks often require the robots to carry high-precision sensors (such as LiDAR and multispectral cameras) while coping with disturbances from dynamic terrains like soft ground, slopes, and gravel. In some scenarios, sensors may only provide sparse, noisy observations due to occlusions or harsh conditions (e.g., dust, low light) [1]. Against this background, the robot's terrain estimation capability and locomotion stability under complex disturbances directly determine the reliability of task execution. Furthermore, the control policies for quadruped robots rely heavily on large-scale simulation training; thus, improving the efficiency of simulation training and shortening the policy iteration cycle are also critical bottlenecks for practical deployment.

Although significant progress has been made in recent years in quadruped robot motion control and terrain perception, robustness in complex dynamic environments still faces two core challenges.

1.1 The Coupling Challenge Between Terrain Perception and Motion Control

Traditional methods often rely on explicit terrain reconstruction (e.g., visual SLAM) to support motion planning. However, sensor noise and dynamic terrain changes (e.g., rolling stones and loose gravel) can accumulate reconstruction errors and lead to gait instability. Lee et al. [2] used proprioceptive feedback and reinforcement learning for simulation training and zero-shot sim-to-real transfer, producing locomotion policies that traverse many challenging terrains. DreamWaQ [3] further demonstrated that proprioception-only locomotion can be improved through implicit terrain imagination with CENet. In highly dynamic terrains, however, trajectory-level adaptation alone can still produce unstable body motion if no explicit dynamics-oriented stability signal is provided during policy optimization.

1.2 The Efficiency Versus Generalization Trade-off in Simulation Training

Reinforcement learning (RL)-based policy training requires a careful balance between exploration and exploitation. A static entropy coefficient can lead to insufficient exploration in early training (policies trapped in local optima) or redundant exploration in later training (slow convergence). Meanwhile, distribution discrepancies between simulation environments and real-world terrains (e.g., friction coefficients and terrain stiffness) may degrade robustness after policy transfer. Existing works often address this issue through manual parameter tuning or fixed entropy-decay schedules, but these approaches provide limited adaptation across training phases.

To address these challenges, this paper positions itself as a direct extension of the DreamWaQ pipeline under the same proprioception-centered setting. Specifically, we preserve the CENet-based framework and introduce two targeted modifications:

(1) a dynamics-informed stability reward design (VHIP and stand-still pose), and (2) a multi-metric dynamic entropy-adjustment mechanism for PPO. Our concrete contributions are as follows.

1.2.1 Reward Design Based on CENet Terrain Imagination and Dynamics Enhancement

We build on the Context-Encoded Network (CENet) terrain-imagination framework proposed by DreamWaQ to learn implicit terrain features. In addition, we introduce stability-oriented reward terms based on the Variable Height Inverted Pendulum (VHIP) dynamic model [4] and a stand-still pose reward for static stability. This reward design provides additional physics-motivated training signals for balance control under uncertain terrain conditions.

1.2.2 PPO Optimization Algorithm with Dynamic Entropy Coefficient

Inspired by the annealing idea (high exploration early, lower later), we design a performance-driven entropy coefficient mechanism for PPO. Unlike classical simulated annealing, which schedules temperature by iteration count, our mechanism adjusts the entropy coefficient based on multi-metric performance gaps, so that exploration increases when performance degrades and decreases when the policy matures. A larger entropy coefficient is used in the early training stage to encourage exploration, while a smaller coefficient is applied later to stabilize policy updates. This mechanism does not change the policy architecture and can be integrated into the original PPO training pipeline with limited additional implementation overhead.

2 Related Work

Recent quadruped locomotion studies increasingly rely on reinforcement learning with proprioceptive sensing and large-scale simulation. Lee et al. [2] demonstrated zero-shot sim-to-real locomotion on challenging terrains, and DreamWaQ [3] further improved adaptation through implicit terrain imagination. In addition, disturbance-aware internal modeling has been shown to improve robustness in complex conditions [1].

To improve stability on uneven terrains, model-informed reward design has become a practical direction. Representative works integrate VHIP, cart-table, or SLIP-style priors into the learning objective [4–8]. These approaches indicate that physics-informed reward terms can reduce unstable gait behaviors and improve robustness under disturbance.

For policy optimization, entropy regularization is commonly used to balance exploration and exploitation. Existing adaptive entropy methods suggest that fixed entropy settings can be suboptimal across training phases [9, 10]. Motivated by this line, our work introduces a multi-metric dynamic entropy mechanism tailored to quadruped locomotion training, while keeping the policy architecture unchanged.

Compared with DreamWaQ, our method keeps the same proprioception-only learning paradigm and CENet-centered architecture, but differs in two aspects. First, we introduce explicit dynamics-oriented stability rewards (VHIP and stand-still pose) to complement implicit terrain imagination. Second, we replace fixed entropy handling with a performance-driven dynamic entropy-adjustment strategy in PPO. This design allows a like-for-like comparison against DreamWaQ while isolating the effects of reward and optimization modifications.

Beyond the core baseline, recent legged-robot studies have explored broader directions in robustness and skill learning, including rapid adaptation, morphology-aware control, aggressive locomotion, and multi-stage reflexive learning [11–19]. Related advances in robust locomotion optimization and torque-level control further highlight the practical importance of stability-exploration trade-offs in highly dynamic regimes [20–22]. In the context of generalization, behavior multiplicity has been shown to improve policy robustness across diverse terrains and conditions [23].

We also note that adaptive entropy tuning has been studied in broader reinforcement learning contexts, including algorithmic optimization and application-driven control tasks [24–27]. Although these works are not all developed for quadruped locomotion, they provide useful methodological motivation for performance-driven entropy adjustment. In a related direction, reward shaping and priors in robot learning have shown that better exploration regularization can materially influence training outcomes [28, 29].

Table 1 Comparison with representative proprioception-based baselines.

Method	Prop.	CENet	Stability	Entropy	Evidence
DreamWaQ [3]	Yes	Yes	None	Fixed	Sim.+qual. RW
RMA [11]	Yes	No	None	Fixed	Sim.+transfer
EstimatorNet [30]	Yes	No	None	Fixed	State-est.+sim.
Ours	Yes	Yes	Enhanced	Dynamic	Fall-rate+conv.+qual.RW

Table 1 places our method in the context of representative proprioception-based baselines. Relative to these methods, our approach combines the DreamWaQ-style CENet setting with explicit dynamics-informed stability rewards and a performance-driven dynamic entropy-adjustment strategy.

3 Materials & Methods

3.1 Task Definition

In this study, the environment is modeled as a partially observable Markov decision process (POMDP), formally defined by the tuple $M = (S, O, A, d_0, p, r, \gamma)$. The full state, partial observation, and action are all continuous, denoted by $s \in S$, $o \in O$, and $a \in A$, respectively. The environment evolution starts from the initial state distribution

$d_0(s_0)$; state transitions follow the state transition probability $p(s_{t+1}|s_t, a_t)$; and each state transition is assigned a corresponding reward feedback via the reward function $r : S \times A \to R$. The discount factor γ ranges within $[0, 1)$.

Our overall framework is illustrated in Figure 1. We adopt the context-aided estimation framework CENet proposed by DreamWaQ [3], whose advantage lies in significantly improving the accuracy of body state estimation by modeling the interaction between latent terrain features z_t and body velocity estimates v_t using historical dimensional data from the past 5 timesteps as input. We employ an asymmetric actor-critic architecture to learn a robust locomotion policy through the interaction between the policy and value networks. This policy can implicitly imagine privileged observations given partial temporal observations. Additionally, we incorporate a dynamics model-based reward function to enhance stability assessment. Finally, we utilize an improved PPO algorithm with a dynamic entropy coefficient, which adjusts the entropy coefficient adaptively based on provided performance metrics to facilitate better policy optimization and updates. In summary, our proposed method enables the rapid training of a stable locomotion control policy for quadruped robots.

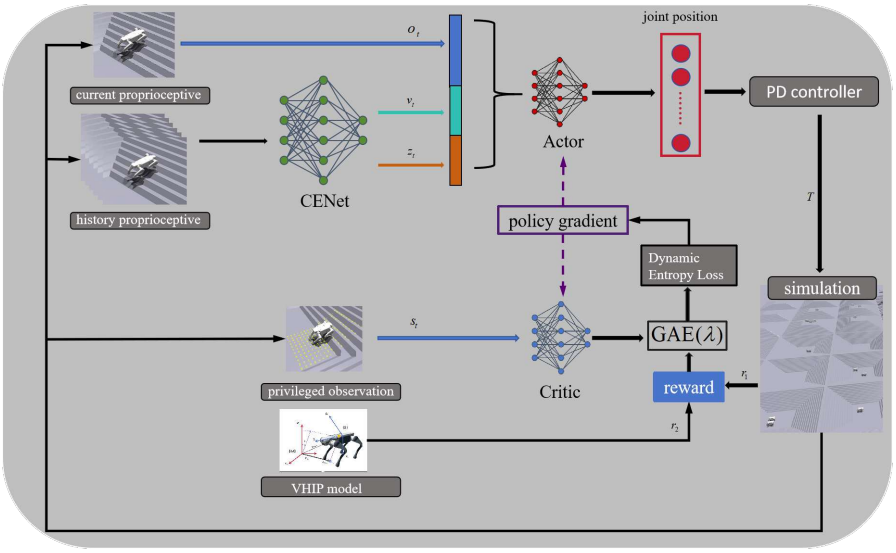


Fig. 1 Our overall framework.

In our framework, we adopt an approach that combines partial historical observations with current proprioceptive observations. First, we define the proprioceptive observation at a single time step as o_t, where o_t is an $n \times 1$ vector defined as follows:

$$o_t = [c_t, \omega_t, g_t, \theta_t, \dot{\theta}_t, a_{t-1}]^T \tag{1}$$

Furthermore, in this paper, we define the temporal observation at time t, based on the past H measurements, as

$$o_t^H = [o_t, o_{t-1}, \ldots, o_{t-H}]^T \tag{2}$$

where the superscript T denotes matrix transposition. c_t, ω_t, g_t, θ_t, $\dot{\theta}_t$, and a_{t-1} represent the 3-dimensional torso velocity command, 3-dimensional torso angular velocity, 3-dimensional gravity vector in the torso coordinate system, 12-dimensional joint angles, 12-dimensional joint angular velocities, and 12-dimensional previous action, respectively, totaling 45 dimensions. Additionally, we define the privileged observation as s_t, and the definition of s_t is as follows:

$$s_t = [o_t, v_t, d_t, h_t]^T \tag{3}$$

where d_t is the 3-dimensional external disturbance force, h_t is the 187-dimensional height scan points, totaling 238 dimensions.

The input to the CENet consists of historical observations from five time steps and the current time step's observation. The policy (Actor) takes the current observation o_t together with the CENet outputs v_t and z_t as its input. It then outputs a 12-dimensional joint action via a PD controller, which ultimately interacts with the simulation environment to obtain a reward. The value network (Critic) receives the complete privileged observation s_t and outputs a 1-dimensional value.

3.2 CENet

To enhance the environmental perception ability of reinforcement learning policies, this paper designs a variational autoencoder (VAE) as an environmental encoder component based on the terrain imagination framework CENet (Context-Encoded Network) proposed by DreamWaQ [3]. This component can extract environmental features from historical observation sequences and provide rich environmental information for the policy network.

The environmental encoder adopts an encoder-decoder architecture, as shown in Figure 2.

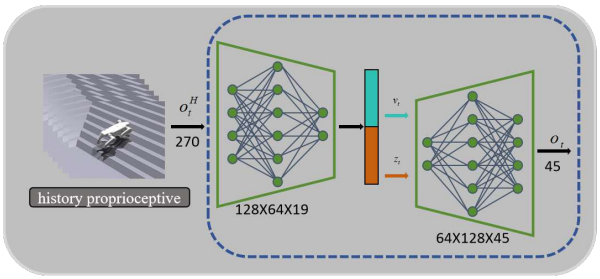


Fig. 2 CENet architecture.

The encoder network maps the sequence $o_t^H \in \mathbb{R}^{270}$ consisting of 5 time-step historical observations + 1 time-step current observation into the latent space, which contains two branches: the speed estimation branch and the latent feature branch. Specifically, the encoder network first passes through the shared hidden layer:

$$h = \text{ELU}(\text{Linear}(\text{ELU}(\text{Linear}(o_t^H)))) \tag{4}$$

Then, calculate the mean and variance of the speed branch and the potential feature branch respectively:

$$\mu_{vel} = \text{Lin}_{vel}^{\mu}(h), \quad \log \sigma_{vel}^2 = \text{Lin}_{vel}^{\sigma}(h) \tag{5}$$

$$\mu_{latent} = \text{Lin}_{latent}^{\mu}(h), \quad \log \sigma_{latent}^2 = \text{Lin}_{latent}^{\sigma}(h) \tag{6}$$

where $h \in \mathbb{R}^{64}$ is a shared hidden representation, $\mu_{vel} \in \mathbb{R}^3$ and $\mu_{latent} \in \mathbb{R}^{16}$ respectively represent the estimated speed and the mean value of the latent features.

To maintain the gradient differentiability, the reparameterization technique is employed to sample from the latent distribution:

$$z_{vel} = \mu_{vel} + \sigma_{vel} \odot \epsilon_{vel}, \quad \epsilon_{vel} \sim \mathcal{N}(0, I) \tag{7}$$

$$z_{latent} = \mu_{latent} + \sigma_{latent} \odot \epsilon_{latent}, \quad \epsilon_{latent} \sim \mathcal{N}(0, I) \tag{8}$$

where $\sigma_{vel} = \exp(\log \sigma_{vel}^2 \cdot 0.5)$, $\odot$ indicates element-wise multiplication. The final environment encoding is:

$$z = [z_{vel}, z_{latent}] \in \mathbb{R}^{19} \tag{9}$$

The decoder network reconstructs the environment encoding into a single-step observation:

$$o_t = \text{Linear}(\text{ELU}(\text{Linear}(\text{ELU}(\text{Linear}(z))))) \tag{10}$$

where $o_t \in \mathbb{R}^{45}$ is a single-step observation for reconstruction, used to verify the encoding quality.

The training of the environmental encoder adopts the loss function of the variational autoencoder, which consists of three components:

Estimated speed loss:

$$\mathcal{L}_{vel} = \|z_{vel} - v_{target}\|_2^2 \tag{11}$$

Reconstruction Loss:

$$\mathcal{L}_{recon} = \|o_t - o_{target}\|_2^2 \tag{12}$$

KL divergence loss:

$$\mathcal{L}_{KL} = -\frac{1}{2} \sum_{i=1}^{d} (1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2) \tag{13}$$

where d is the dimension of potential characteristics, v_{target} is the actual speed extracted from the privileged observation, o_{target} is the first step of the historical observation sequence.

Total loss function:

$$\mathcal{L}_{CENet} = \mathcal{L}_{vel} + \mathcal{L}_{recon} + \beta\mathcal{L}_{KL} \tag{14}$$

where β is the KL divergence weight coefficient, and in this paper, it is set to 1.0.

The output of the environment encoder is used to enhance the policy network. Specifically, the environment encoding z is blended with the current observation $o_{current}$:

$$o_{enhanced} = [z, o_{current}] \in \mathbb{R}^{64} \tag{15}$$

Then, the enhanced observations are input into the Actor network:

$$a = \text{Actor}(o_{enhanced}) \tag{16}$$

This design enables the policy to simultaneously utilize the immediate information observed at present and the historical information encoded by the environment encoder, thereby making more intelligent decisions in complex environments. Through this design of the environment encoder, the intelligent agent can learn the potential representations of the environment from historical observations, including terrain features and dynamic changes, enhancing the adaptability and robustness of the policy in complex environments.

3.3 Network Architecture

Our framework employs an asymmetric actor-critic architecture with a separate environment encoder (CENet). The detailed network architectures are as follows.

Actor Network: The actor network takes as input the concatenation of the CENet output $z \in \mathbb{R}^{19}$ (consisting of 3-dimensional velocity estimate and 16-dimensional latent features) and the current observation $o_t \in \mathbb{R}^{45}$, resulting in an input dimension of 64. The network consists of three hidden layers with dimensions [512, 256, 128], each followed by an ELU (Exponential Linear Unit) activation function. The output layer produces 12-dimensional joint angle targets, which are then tracked by a PD controller.

Critic Network: The critic network receives the privileged observation $s_t \in \mathbb{R}^{238}$ as input. It shares the same architecture as the actor network, with three hidden layers of dimensions [512, 256, 128] and ELU activations. The output layer produces a scalar value estimate.

CENet Encoder: The encoder processes the historical observation sequence $o_t^H \in \mathbb{R}^{270}$ (6 timesteps: 5 historical + 1 current, each with 45 dimensions). It first passes through two shared linear layers with dimensions [270, 128] and [128, 64], each followed by ELU activation. The shared representation $h \in \mathbb{R}^{64}$ is then split into two branches: (1) velocity estimation branch outputs mean and log-variance for a 3-dimensional velocity estimate, and (2) latent feature branch outputs mean and log-variance for a 16-dimensional latent feature vector.

CENet Decoder: The decoder reconstructs the single-step observation from the encoded representation $z \in \mathbb{R}^{19}$. It consists of three linear layers with dimensions

[19, 64], [64, 128], and [128, 45], with ELU activations between layers. The output reconstructs the original 45-dimensional observation for the VAE reconstruction loss.

All networks are initialized using standard PyTorch initialization schemes. The CENet encoder and decoder are updated together with the actor and critic in each PPO iteration. The actor receives the CENet output z as input, so gradients from the policy loss flow back into CENet; we do not use stop-gradient between CENet and the actor. Thus the entire pipeline (CENet, actor, and critic) is trained end-to-end using the PPO algorithm with the dynamic entropy coefficient mechanism.

3.4 Reward Function Engineering

Our reward function is designed to align with established methodologies while introducing components aimed at enhancing dynamic stability. We integrate the VHIP-based reward terms proposed by Xiao et al. [4] into our reward framework, together with terms for tracking accuracy, smooth joint motion, and minimal foot impact. The VHIP-based terms, derived from the Variable Height Inverted Pendulum (VHIP) model, penalize excessive torso tilt and angular acceleration, providing a physics-based incentive for balance. The overall reward tracks the torso's orientation and velocity, joint positions and torques, and foot contact states and clearances. This structure guides the policy toward stable, energy-efficient gaits while optimizing for tracking accuracy, smooth motion, minimal foot impact, and dynamic stability on complex, unstructured terrains. The specific formulation and corresponding weights ω_i for each reward component are detailed in Table 2.

The VHIP model and its dynamic modeling are shown in Figure 3:

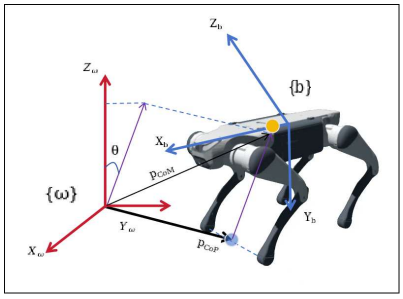


Fig. 3 VHIP model.

where p_{CoM} represents the 3D position of the center of mass, p_{CoP} represents the 3D position of the pressure center (on a plane, its Z-axis coordinate is usually 0), and $p_{\mathrm{CoM,z}}$ is the Z-axis component of p_{CoM}.

Following Xiao et al. [4], we adopt two VHIP-based reward terms: the VHIP angle reward function and the VHIP angular acceleration reward function. The pendulum angle θ can be directly used as a criterion for judging the robot's stability. Following their approach, we use the threshold $\theta_{\mathrm{threshold}} = 0.1$ rad (approximately 5.7°), which corresponds to a reasonable stability margin for quadruped robots—beyond this angle,

Table 2 Reward Function Details

Reward Term	**Expression (r_i)**	**Weight (ω_i)**	**Type**
Lin. velocity tracking	$\exp\left\{\frac{-(v_{cmd}-v_{actual})^2}{0.25}\right\}$	1.0	Reward
Ang. velocity tracking	$\exp\left\{\frac{-(\omega_{cmd}-\omega_{actual})^2}{0.25}\right\}$	0.5	Reward
Lin. velocity(z)	v_z^2	-2.0	Penalty
Ang. velocity(xy)	ω_{xy}^2	-0.05	Penalty
Orientation	$\|g\|^2$	-0.2	Penalty
Joint accelerations	$\left(\frac{v_{t-1}-v_t}{dt}\right)^2$	-2.5e-7	Penalty
Joint power	$\sum v_i \times \tau_i$	-2e-5	Penalty
Body height	$(h_{actual} - h_{target})^2$	-1.0	Penalty
Foot clearance	$\sum(h_{foot} - h_{foot_target})^2 \times v_{foot}$	-0.01	Penalty
Action rate	$(a_t - a_{t-1})^2$	-0.01	Penalty
Smoothness	$(a_t - 2a_{t-1} + a_{t-2})^2$	-0.01	Penalty
Power distribution	$\sum(\tau_i)^2$	-1e-5	Penalty
VHIP angle	$\max(0, \|\theta\| - \theta_{threshold})$	-0.1	Penalty
VHIP angular acc	$\max(0, \|\ddot{\theta}\| - \ddot{\theta}_{threshold})$	-0.001	Penalty
Stand still pose	$-\sum_{i=1}^{n}(q_i - q_{i,\text{default}})^2 \cdot \mathbf{1}(\|v_{xy}\| < 0.15)$	1.0	Reward

Type: "Reward" indicates terms that encourage desired behaviors (positive weights), while "Penalty" indicates terms that discourage undesired behaviors (negative weights). The total reward is computed as $r_t(s_t, a_t) = \sum r_i \omega_i$. exp() denotes the exponential operator, max() denotes the maximum operator, $()_{\text{cmd}}$ and $()_{\text{actual}}$ represent command and actual values, v denotes linear velocity, ω denotes angular velocity, $\mathbf{g}$ denotes the gravity vector projected onto the robot body coordinate system, h_{actual} and h_{target} denote the actual body height and target body height, τ denotes joint torque, θ denotes pendulum angle, $\ddot{\theta}$ denotes pendulum angular acceleration. q_i represents the current angle of the i-th joint, $q_{i,\text{default}}$ represents the default angle of the i-th joint, n is the total number of joints, $\mathbf{1}(\cdot)$ denotes the indicator function, v_{xy} denotes the horizontal velocity vector, and $\|\cdot\|$ denotes the Euclidean norm.

the robot's center of mass deviates significantly from the support polygon, increasing the risk of tipping over. The angle θ can be calculated using the following formula:

$$\theta = \arccos\left(\frac{|p_{\text{CoM,z}}|}{\|p_{\text{CoM}} - p_{\text{CoP}}\|}\right) \tag{17}$$

When θ increases, it indicates that the robot is tilting to the sides or forward/backward. The larger the θ angle, the more likely it is to tip over, and the worse the stability of the body becomes.

Similarly, we use the threshold $\ddot{\theta}_{\text{threshold}} = 0.01$ as proposed in [4]. This threshold captures the angular acceleration of the pendulum, representing the rate at which the robot is losing balance. A value exceeding this threshold indicates that the robot is accelerating toward an unstable configuration and requires immediate corrective action. The angular acceleration $\ddot{\theta}$ can be calculated using the formula:

$$\ddot{\theta} = \frac{g \sin\theta}{\|p_{\text{CoM}} - p_{\text{CoP}}\|} \tag{18}$$

where g represents the acceleration due to gravity. Following Xiao et al. [4], we adopt this expression for the VHIP angular acceleration; see their work for the derivation. A sudden increase in $\ddot{\theta}$ indicates that the robot is accelerating to tilt to either side or forward/backward. The larger the $\ddot{\theta}$ value, the poorer the stability of the robot body.

Furthermore, we observed that trained policies often fail to maintain a stable standing posture when initialized or when given zero velocity commands, instead continuously shifting forward or backward. We attribute this behavior to the training process: policies are consistently rewarded for maintaining balance during locomotion, but receive insufficient training signals for static stability when the commanded velocity is zero. To address this issue, we introduce a conditional reward function that penalizes deviations from the default joint configuration only when the robot's horizontal velocity is below a threshold. This ensures that the policy learns to maintain a stable standing posture during low-speed or zero-velocity scenarios without interfering with normal locomotion. The conditional reward function is formulated as follows:

$$r_{\text{stand-still}} = -\alpha \sum_{i=1}^{n} (q_i - q_{i,\text{default}})^2 \cdot \mathbf{1}(\|v_{xy}\| < v_{\text{threshold}}) \tag{19}$$

where $\mathbf{1}(\|v_{xy}\| < v_{\text{threshold}})$ is an indicator function that equals 1 when the horizontal velocity magnitude $\|v_{xy}\|$ is below the threshold $v_{\text{threshold}} = 0.15$ m/s, and 0 otherwise. This conditional formulation allows the policy to freely adjust joint angles during normal locomotion while encouraging a stable standing posture when the robot should remain stationary. The weight $\omega = 1.0$ was determined through experimental validation, balancing the need for static stability without hindering dynamic locomotion performance.

Finally, the total reward for a policy taking a specific action in each state is expressed as follows:

$$r_t(s_t, a_t) = \sum r_i \omega_i \tag{20}$$

where i is the positional index of each reward as shown in Table 2.

3.5 Dynamic Entropy Coefficient Design

To address the issue of the fixed entropy coefficient in traditional PPO being insufficient for balancing the exploration-exploitation trade-off, this section proposes a multi-metric-driven dynamic entropy coefficient mechanism. This mechanism adaptively adjusts the strength of entropy regularization by quantifying task performance and performing a weighted fusion of multi-dimensional gaps, ensuring that the policy maintains high exploration in the early stages of training and shifts towards efficient exploitation upon maturity.

In this study, the loss function of the PPO algorithm is composed of the policy clipping loss, the value function loss, the entropy regularization loss, and the CENet network update loss, formulated as:

$$\mathcal{L}_{ppo} = E_t[\mathcal{L}_{CLIP} + c_1\mathcal{L}_{VF} - c_2\mathcal{L}_{ENT} + \mathcal{L}_{CENet}] \tag{21}$$

where $\mathcal{L}_{CLIP}$ denotes the policy clipping loss, $\mathcal{L}_{VF}$ denotes the value function loss, $\mathcal{L}_{ENT}$ denotes the entropy regularization loss, $\mathcal{L}_{CENet}$ denotes the CENet network update loss, c_1 denotes the value function loss weight coefficient, and c_2 denotes the dynamic entropy regularization coefficient.

To quantify the agent's task performance and drive the adjustment of the entropy coefficient, we define three key performance indicators and their corresponding preset target thresholds: P_{lin}, P_{ang}, and $P_{terrain}$, representing linear velocity tracking performance, angular velocity tracking performance, and terrain utilization performance, respectively. Specifically, P_{lin} is the mean linear velocity tracking reward (same as the reward term in Table 2, in $[0, 1]$, higher is better); P_{ang} is the mean angular velocity tracking reward (in $[0, 1]$, higher is better); $P_{terrain}$ is the mean terrain difficulty level achieved by the policy in the current iteration, i.e., the curriculum terrain level (same scale as in Section 3.6) averaged over all parallel environments, with higher values indicating better utilization of challenging terrains. The corresponding thresholds are: $T_{lin} = 0.7$, $T_{ang} = 0.3$, and $T_{terrain} = 5$. These thresholds for the performance indicators were not preset arbitrarily but were determined through preliminary parameter sensitivity analysis. We conducted multiple rounds of small-scale training, with each round consisting of 500 episodes, systematically adjusting these thresholds and observing their impact on the convergence speed of the learning curve and the stability of the final policy. The ultimately selected thresholds were proven to provide a good balance between encouraging exploration and ensuring learning stability, demonstrating consistent effectiveness in subsequent full-scale training.

For each performance indicator, the performance gap is calculated if the target is not met; if the target is met, the gap is 0 to avoid invalid penalties:

$$G_i = \max(0, T_i - P_i) \tag{22}$$

where G_i represents the gap between the i-th indicator and its target value. To eliminate the influence of different units, the gap needs to be normalized to the interval [0,1]:

$$g_i = \frac{G_i}{T_i} \tag{23}$$

where $g_i \in [0, 1]$ indicates the degree to which the i-th indicator fails to meet the target ($g_i = 0$ indicates the target is fully met, $g_i = 1$ indicates complete failure to meet the target). Furthermore, to achieve synergistic driving of the entropy coefficient by multiple indicators, we employ the Softmax function to weight the normalized gaps. Compared to simple methods like taking the average ($\omega_i = 1/3$) or manually setting fixed weights, this weighting method possesses self-adaptive capability. During different stages of training, the robot's performance bottlenecks change dynamically (e.g., velocity tracking might be the issue initially, while stability on complex terrain becomes critical later). The Softmax weighting can automatically capture this change, ensuring that the adjustment of the entropy coefficient always addresses the most critical bottleneck problem, thereby making the exploration process more targeted.

The normalized gap column vector is constructed as:

$$g = [g_{lin}; g_{ang}; g_{terrain}] \in \mathbb{R}^3 \tag{24}$$

A temperature parameter $\tau > 0$ is introduced to control the sharpness of the weight distribution. The temperature parameter τ governs the concentration degree of the weight allocation. Through preliminary testing, we found that when the value of τ is too small (e.g., $\tau = 0.1$), the weights become overly concentrated, leading to drastic changes in the entropy coefficient and consequently unstable training. Conversely, when the value of τ is too large (e.g., $\tau = 10$), the weights become excessively averaged, losing the short-board-driven effect. Ultimately, we selected $\tau = 2$, as this value provides a robust balance between focusing on the primary performance shortcoming and appropriately considering other metrics. Its effectiveness has been verified in the main experiments.

The overall form of the weight calculation is:

$$\omega = \text{softmax}\left(\frac{g}{\tau}\right) = \frac{\exp\left(\frac{g}{\tau}\right)}{\sum_{k=1}^{3} \exp\left(\frac{g_k}{\tau}\right)} \tag{25}$$

The expanded component form is:

$$\omega_i = \text{softmax}\left(\frac{g_i}{\tau}\right) = \frac{\exp\left(\frac{g_i}{\tau}\right)}{\sum_{k=1}^{3} \exp\left(\frac{g_k}{\tau}\right)} \tag{26}$$

where $\omega_i \in [0, 1]$ represents the influence weight of the i-th category of indicator on the entropy coefficient, satisfying $\sum_{i=1}^{3} \omega_i = 1$.

Finally, based on the weighted average gap $\overline{g}$, the dynamic entropy coefficient is calculated as:

$$c_2(t) = c_{low} + \overline{g} \cdot (c_{high} - c_{low}) \tag{27}$$

where $c_{low} = 0.01$ and $c_{high} = 0.02$ define the lower and upper bounds of the entropy coefficient range, respectively, and $\overline{g}$ is calculated as follows:

$$\overline{g} = (\omega_{lin} g_{lin} + \omega_{ang} g_{ang} + \omega_{terrain} g_{terrain}) \tag{28}$$

The selection of the entropy coefficient range $c_2 \in [0.01, 0.02]$ is based on empirical observations and the stability requirement of quadrupedal locomotion. In standard

PPO implementations, the entropy coefficient typically ranges from 0.01 to 0.1. We adopt a narrower range to avoid excessive action randomness while preserving adaptive exploration through the proposed metric-driven mechanism.

Relation to Existing Adaptive Entropy Methods: A representative line of work in maximum-entropy reinforcement learning is the automatic temperature tuning used in Soft Actor-Critic (SAC), where the entropy temperature is adjusted to match a target entropy level. Such methods are effective for balancing global exploration-exploitation behavior with a single entropy objective. In contrast, our target task is quadrupedal locomotion under multi-factor constraints, where training bottlenecks may shift across different dimensions (e.g., velocity tracking versus terrain adaptation). Therefore, we adopt a multi-metric mechanism that adjusts entropy according to fused performance gaps from multiple task indicators, allowing the policy to react to the currently dominant bottleneck rather than relying on a single entropy target signal.

Implementation Details for Reproducibility: The dynamic entropy coefficient is updated once per PPO iteration. For each metric, we first compute the raw iteration-level performance $P_i(t)$ and then smooth it using exponential moving average (EMA):

$$\tilde{P}_i(t) = \alpha \tilde{P}_i(t-1) + (1-\alpha) P_i(t), \quad \alpha = 0.9 \tag{29}$$

The smoothed values $\tilde{P}_{lin}(t)$, $\tilde{P}_{ang}(t)$, and $\tilde{P}_{terrain}(t)$ are used in the equations above to compute g_i, ω_i, and $\overline{g}$. Then, $c_2(t)$ is computed, clipped into $[c_{low}, c_{high}]$, and applied in the next PPO iteration. To prevent abrupt oscillations, we additionally limit per-iteration change by:

$$\Delta c_2(t) = c_2(t) - c_2(t-1), \quad |\Delta c_2(t)| \leq 0.002 \tag{30}$$

This implementation keeps the entropy adjustment smooth and straightforward to reproduce in standard PPO codebases.

3.6 Experimental Setup

We utilized the open-source implementation of the Isaac Gym simulator [31] for training. The simulation runs 4096 parallel robot environments simultaneously on a single NVIDIA RTX 4070Ti GPU (16GB VRAM). Each environment is an independent instance of the quadruped robot simulation, enabling massive parallel data collection. The policy network, value network, and CENet were trained synchronously for 4000 iterations. We applied domain randomization across all 4096 environments. Detailed information on the randomized parameters is listed in Table 4. The code for this project is available at https://github.com/defineyumi/rl_training_dwaq.git.

Simulation Configuration: The simulation time step is set to $dt = 0.005$ seconds, resulting in a simulation frequency of 200 Hz. The control frequency is 50 Hz, achieved through a decimation factor of 4 (i.e., the policy is updated every 4 simulation steps). Each episode has a maximum length of 20 seconds, corresponding to 4000 simulation steps per episode. The Isaac Gym simulator is configured with PhysX physics engine using 10 threads for parallel computation, with position iterations set to 4 and velocity iterations set to 0 for efficient simulation.

Training Budget: During each training iteration, we collect 100 environment steps from each of the 4096 parallel environments, resulting in 409,600 samples per iteration. Over 4000 training iterations, this yields a total of approximately 1.64 billion environment samples. The policy achieves successful locomotion capability after approximately 500 iterations (corresponding to approximately 200 million samples), and converges to stable performance after the full 4000 iterations. The complete training process requires approximately 4 hours on a single NVIDIA RTX 4070Ti GPU.

PPO Hyperparameters: The complete hyperparameter configuration is detailed in Table 3. For all algorithms, the policy network was trained using PPO with a clip range of 0.2, generalized advantage estimate (GAE) factor $\lambda = 0.95$, and discount factor $\gamma = 0.99$. The networks were optimized using the Adam optimizer with a learning rate of 1×10^{-3} and default momentum parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$). Gradient clipping is applied with a maximum gradient norm of 1.0 to ensure training stability.

Table 3 PPO Training Hyperparameters

Hyperparameter	**Value**	**Description**
Clip range (ϵ)	0.2	PPO clip threshold
GAE factor (λ)	0.95	Advantage estimation factor
Discount factor (γ)	0.99	Reward discount
Learning rate	1×10^{-3}	Adam learning rate
Value loss coefficient (c_1)	1.0	Value loss weight
Entropy coefficient range (c_2)	[0.01, 0.02]	Entropy coefficient bounds
Softmax temperature (τ)	2.0	Multi-metric weighting temperature
PPO epochs per update	5	Epochs per iteration
Mini-batch size	102,400	Samples per mini-batch
Batch size (per iteration)	409,600	Samples per iteration
Max gradient norm	1.0	Gradient clip threshold
Adam β_1	0.9	First moment decay
Adam β_2	0.999	Second moment decay
Adam ϵ	10^{-8}	Numerical stability

Domain Randomization: To bridge the sim-to-real gap and improve robustness to real-world uncertainty, we apply comprehensive domain randomization across all 4096 parallel environments during training. The randomization covers multiple aspects of the simulation setup, including robot dynamics, ground properties, and external disturbances. Specifically, we randomize body mass (0.8–1.2× nominal), link

Table 4 Domain Randomization Range

Parameters	**Range**$[Min, Max]$	**Unit**
Body Mass	[0.8, 1.2] × nominal value	kg
Link Mass	[0.9, 1.1] × nominal value	kg
CoM	$[-0.05, 0.05]$ (each axis)	m
Payload Mass	[-1, 2]	kg[1]
Ground Friction	[0.2, 1.25]	–
Ground Restitution	[0.0, 1.0]	–
Motor Strength	[0.9, 1.1] × motor torque	Nm
Joint Kp	[0.9, 1.1] × 40	–
Joint Kd	[0.9, 1.1] × 1	–
Initial Joint Positions	[0.5, 1.5] × nominal value	rad
System Delay	[0, 3Δt]	s
External Force	$[-30, 30]$ (each axis)	N

[1]Negative payload mass values represent reduced robot mass (simulating scenarios with reduced load capacity).

masses (0.9–1.1× nominal), center-of-mass position (±0.05 m per axis), and payload mass (-1 to 2 kg, where negative values simulate reduced load-capacity scenarios). Ground properties are randomized with friction coefficients in [0.2, 1.25] and restitution coefficients in [0.0, 1.0], enabling adaptation to surfaces ranging from slippery to high-friction. Motor characteristics are randomized through strength variations (0.9–1.1× nominal torque) and joint-controller gains (Kp: 0.9–1.1× 40, Kd: 0.9–1.1× 1). We also randomize initial joint positions (0.5–1.5× nominal), system delays (0–3Δt), and apply random external forces (±30 N per axis) to emulate disturbances and modeling uncertainty. This strategy improves robustness to parameter variation and supports generalization to real-world deployment. Detailed parameter ranges are summarized in Table 4.

Curriculum Learning: For policy training, we adopted a curriculum learning approach, facilitating progressive learning from simple to complex scenarios across two dimensions: terrain and command velocity.

The command velocity curriculum learning aims to enable the policy to gradually adapt to locomotion at higher speeds. The initial ranges for both linear and angular velocities are set to [-1.0, 1.0]. The key mechanism for updating the velocity curriculum involves monitoring the linear velocity tracking reward: when this reward reaches 80% of its maximum possible value consistently over multiple episodes, the system automatically expands the velocity range boundaries by 0.2 m/s until the preset upper limit of 3.0 m/s is reached. Concurrently, we employ an environment grouping strategy: 20% of the environments consistently receive higher velocity commands to encourage

exploration, while the remaining 80% are assigned velocities according to the current curriculum level, thus balancing training stability and challenge.

The terrain is generated using triangular meshes with the following core parameters: horizontal scaling of 0.1 m/pixel, vertical scaling of 0.005 m/pixel, and a single environment size of 8m × 8m. The terrain curriculum comprises 10 difficulty levels, with the progressively increasing difficulty strategy quantitatively controlled by the following formulas:

$$\begin{aligned}
&\text{Slope} = \text{difficulty} \times 0.4 \\
&\text{Step Height} = 0.05 + 0.14 \times \text{difficulty} \\
&\text{Discrete Obstacle Height} = 0.05 + \text{difficulty} \times 0.1
\end{aligned}$$

where difficulty represents the difficulty level. The terrain types are distributed by proportion as follows: smooth slopes (10%), rough slopes (20%), ascending stairs (30%), descending stairs (30%), and discrete terrain (10%) to ensure training diversity. The trigger condition for updating the curriculum is based on the robot's traveled distance: the terrain level increases when the distance traveled exceeds half the environment length, and decreases when the distance traveled falls below half the product of the command velocity and the maximum episode length, thereby achieving adaptive difficulty adjustment.

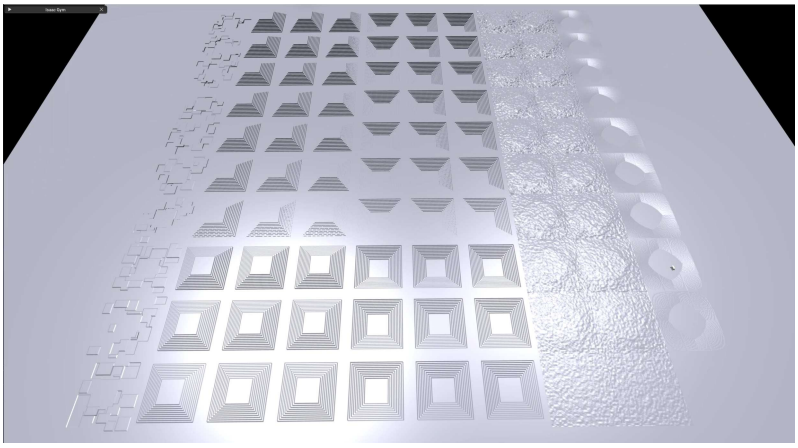


Fig. 4 Simulation environment overview.

4 Results

4.1 Comparative Experiments

To systematically validate the effectiveness of each component in our proposed method, we conduct comprehensive ablation studies and comparative experiments. All experiments are conducted under identical training conditions (same random seeds, training

iterations, and hyperparameters) to ensure fair comparison. For the training-based comparisons, we report mean performance with standard deviation across repeated runs.

4.1.1 Stability Reward Validation via MuJoCo Simulation

All main policy training and the DreamWaQ baseline comparison (Section 4.1.3) are conducted in Isaac Gym. To isolate the effect of the stability reward terms under a different physics engine and to align with the evaluation setup of Xiao et al. [4], we conduct the stability reward ablation in this subsection using the MuJoCo physics engine. We compare four method variants: (1) our complete method with both stand-still pose and VHIP rewards, (2) our method without stand-still pose reward, (3) our method without VHIP rewards, and (4) our method without both reward components.

The evaluation is conducted in two scenarios: (1) **Static stability test**: The robot receives zero-velocity commands and is initialized at the default pose. We record the body orientation (roll and pitch angles) over a 10-second simulation period to assess static stability. (2) **Dynamic stability test**: The robot performs locomotion on complex terrains with varying difficulty levels. We record the body orientation trajectories and VHIP metrics (angle θ and angular acceleration $\ddot{\theta}$) during locomotion to assess dynamic stability.



Fig. 5 Body orientation (roll and pitch angles) trajectories during the static-stability test (zero-velocity command). The complete method maintains stable body orientation, whereas methods without the stand-still pose reward show larger drift.

Figure 5 shows the body-orientation trajectories during the static-stability test. The complete method maintains stable orientation, with roll-angle variance of 0.001 ± 0.001 rad and pitch-angle variance of 0.001 ± 0.001 rad, while the method without the stand-still pose reward exhibits substantially larger drift (roll: 0.024 ± 0.006 rad, pitch: 0.031 ± 0.008 rad). These results suggest that the stand-still pose reward helps reduce forward/backward body shift under zero-velocity commands, partially addressing the static-stability issue identified in Section 3.4.

For dynamic-stability validation, Table 5 reports fall rates across terrain difficulty levels 0–6 based on 100 independent simulation trials. Both methods achieve 0% fall rate on levels 0–4. Starting from level 5, the method without VHIP rewards shows higher fall rates (5% at level 5, 15% at level 6), while the complete method remains lower (2% at level 5, 6% at level 6). A two-sample proportion test at terrain level 6 (6% vs. 15%, $n = 100$ per group) yields a p-value of approximately 0.03 (p ¡ 0.05), supporting a statistically significant difference at this level. At terrain level 5 (2% vs. 5%), the p-value is approximately 0.25 (p ¿ 0.05), suggesting a non-significant trend. Taken together, these observations suggest that explicit dynamic-stability constraints can help reduce failure risk on harder terrains. Through the reward terms in Table 2, the VHIP model provides a physics-informed stability signal by penalizing unstable center-of-mass motion.

Table 5 Dynamic Stability Comparison Across Terrain Difficulty Levels (100 simulation trials)

Metric / Level	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Fall Rate (With VHIP)	0%	0%	0%	0%	0%	2%	6%
Fall Rate (Without VHIP)	0%	0%	0%	0%	0%	5%	15%

4.1.2 Dynamic Entropy Coefficient Experiments

To further illustrate the behavior of the adaptive entropy mechanism, we visualize the entropy coefficient trajectory during policy optimization, as shown in Figure 6. The curve is stage-dependent rather than purely monotonic. In the early stage, a relatively high coefficient maintains sufficient exploration and accelerates policy discovery. In the middle stage, as task performance improves and tracking gaps shrink, the coefficient gradually decreases, corresponding to a transition toward policy refinement. Importantly, the figure also shows local rebounds: around iteration 2000, the model learned a poor policy that led to significant performance degradation. At this point, the entropy coefficient adaptively increases to re-introduce exploration pressure. After approximately 700 iterations, the model recovers from these local degradations, and the coefficient decreases again and stabilizes, indicating a return to exploitation-dominant optimization.

Building on the adaptive behavior illustrated in Figure 6, we further evaluate the selected entropy-coefficient range $c_2 \in [0.01, 0.02]$ (Section 3.5) through small-scale training (400 iterations per run). We compare our dynamic entropy-coefficient method with several baselines: (1) dynamic range $[0.01, 0.02]$, (2) dynamic range $[0.01, 0.03]$, (3) dynamic range $[0.01, 0.015]$, (4) dynamic range $[0.005, 0.01]$, (5) fixed value $c_2 = 0.02$, and (6) fixed value $c_2 = 0.01$. All methods use the same reward function and network architecture. Each configuration is run with 5 independent random seeds (each run trains from scratch for 400 iterations). We define convergence as the first iteration at

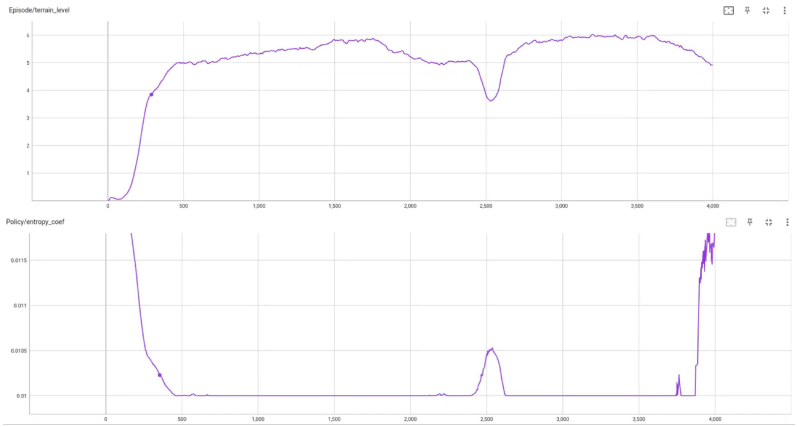


Fig. 6 Adaptive entropy response during training. When policy performance degrades due to unfavorable updates, the entropy coefficient increases automatically to promote re-exploration and support recovery; after performance improves, the coefficient decreases again to stabilize optimization.

which the mean terrain level (averaged over the last 50 iterations) reaches or exceeds 5.0; configurations that do not reach this threshold within 400 iterations are reported as ">400".

The evaluation metrics include: (1) convergence speed (iterations to reach the convergence criterion above), and (2) standard deviation of the mean terrain level across the 5 runs, which evaluates the training stability of different parameter configurations.

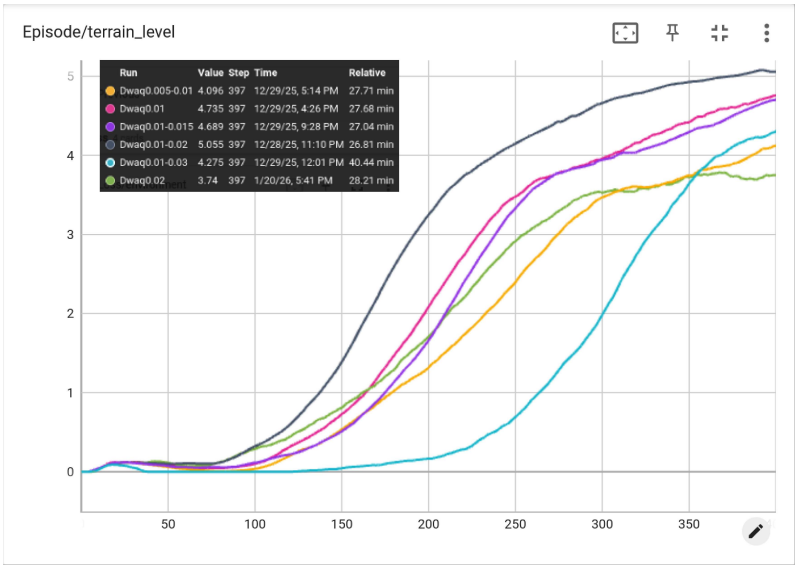


Fig. 7 Impact of entropy coefficient range on early-stage training acceleration. The figure shows that a reasonable entropy coefficient range helps improve training efficiency in the early stage.

Experimental results (Table 6) show that, among the tested settings, the dynamic entropy-coefficient range $[0.01, 0.02]$ provides the most favorable trade-off, requiring the fewest iterations and the shortest convergence time. The other tested configurations do not reach the convergence criterion within 400 iterations, and wider ranges converge more slowly in this setup. In addition, fixed entropy coefficients cannot adapt to changing training conditions. Figure 7 illustrates the adaptive behavior of the proposed mechanism, with gradual decay during performance improvement and rebounds when performance degrades.

Table 6 Comparison of Different Entropy Coefficient Configurations (5 independent training runs per configuration). Convergence: first iteration at which mean terrain level ≥ 5.0 (over the last 50 iterations); ">400" indicates not converged within 400 iterations.

Method	**Convergence Iterations**	**Mean Terrain Level**	**Level Std**
$[0.01, 0.02]$ (Ours)	<400	5.06	0.10
$[0.01, 0.03]$	>400	4.28	0.15
$[0.01, 0.015]$	>400	4.69	0.12
$[0.005, 0.01]$	>400	4.10	0.18
Fixed $c_2 = 0.02$	>400	3.74	0.20
Fixed $c_2 = 0.01$	>400	4.74	0.14

4.1.3 Overall Performance Comparison

To evaluate the overall effectiveness of our complete method, we compare it with the baseline method DreamWaQ [3] used in this study. Both methods are evaluated on the same terrain curriculum learning task in Isaac Gym, trained for 4000 iterations. Each method was run with 5 independent random seeds; we report mean ± standard deviation across these 5 runs. Convergence is defined as the first iteration at which the smoothed mean terrain level reaches or exceeds 5.9.

The comparison results show that our complete method performs slightly better than DreamWaQ in final terrain level (6.0 ± 0.03 vs. 5.8 ± 0.1) and linear velocity tracking accuracy (0.8 ± 0.015 vs. 0.75 ± 0.01), while maintaining comparable angular velocity tracking accuracy. Statistical testing shows that the final-terrain-level difference (6.0 vs. 5.8, $n = 5$ per method) yields a p-value of approximately 0.08, which does not meet the conventional significance threshold (p ¡ 0.05) but indicates a positive trend. For linear-velocity tracking accuracy (0.8 vs. 0.75), the p-value is approximately 0.02 (p ¡ 0.05), supporting a statistically significant difference. In addition, our method converges faster (3000 ± 23 vs. 3500 ± 38 iterations to reach terrain level 6), with p ¡ 0.001 for the convergence-speed difference. This faster convergence is consistent with the intended role of the dynamic entropy-coefficient mechanism in improving the exploration-exploitation balance during training.

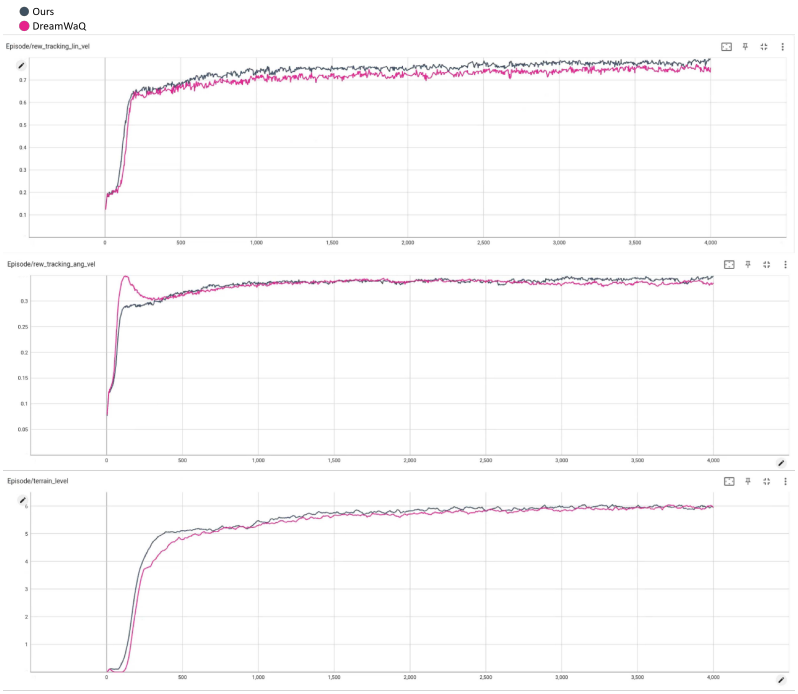


Fig. 8 Compared with the baseline method, the figure shows the results of linear velocity and angular velocity tracking rewards and terrain curriculum learning levels after 4000 iterations.

4.2 Model Deployment

4.2.1 MuJoCo Simulation Deployment

A MuJoCo simulation test video is available at https://www.bilibili.com/video/BV1aFypBoE3N/?share_source=copy_web&vd_source=5915f7024dc31b4dba8b43a1be23a213. The video demonstrates locomotion performance on flat ground (straight motion and turning), stairs, and sloped terrain. The simulation environment is provided by Deep Robotics, and the related code is available on the official website.

4.2.2 Real-World Deployment

We conducted real-world experiments using the Deep Robotics Lite3 robot. All control and inference processes were executed on the robot's onboard computer. Control commands were issued via a handheld controller, and policy inference was performed at 50 Hz. The joint-angle commands produced by the policy were tracked by a PD controller with gains $k_p = 40$ and $k_d = 1$. In this manuscript, the real-world evaluation is qualitative and is intended primarily to demonstrate deployment feasibility on representative terrains.

The physical deployment results (Figures 9–13) indicate that the learned policy executes stably in the tested scenarios. We attribute this behavior mainly to two

design choices. First, the proprioceptive-only policy reduces dependence on difficult-to-simulate visual attributes. Second, domain randomization exposes the policy to broad dynamic-parameter variation during training, improving adaptability to real-world uncertainty. For example, on rough woven-bag surfaces (Figure 10) and smooth marble surfaces (Figure 11), the policy adjusts its gait to handle friction changes. The policy also remains executable on terrains not explicitly represented in simulation, such as those in Figures 12 and 13. These observations should be interpreted as qualitative deployment evidence rather than a full quantitative benchmark.

As shown in Figure 9, the Lite3 robot executes our policy and traverses a 16 cm step.

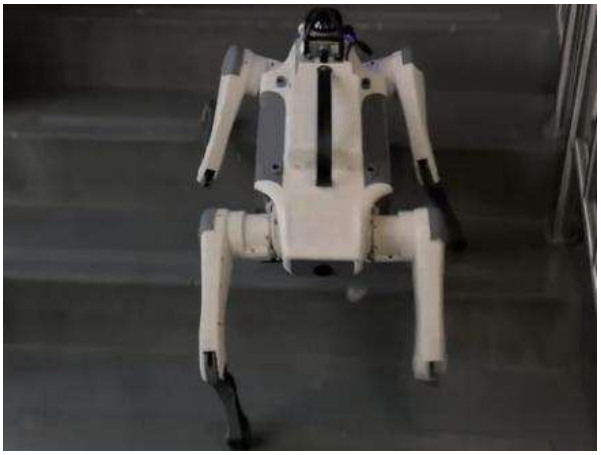


Fig. 9 The Lite3 robot climbing stairs.

Figure 10 shows the Lite3 robot walking on rough ground covered with woven bags.



Fig. 10 The Lite3 robot walking on rough ground.

Figure 11 shows the Lite3 robot walking on a smooth marble floor.
Figure 12 shows the Lite3 robot walking through bushes.
Figure 13 shows the Lite3 robot walking in deeper grassland.



Fig. 11 The Lite3 robot walking on smooth ground.



Fig. 12 The Lite3 robot walking through small shrubs.



Fig. 13 The Lite3 robot walking in deep grassland.

5 Discussion

The results suggest that the proposed method provides a practical balance between exploration and exploitation, with improved locomotion performance in the evaluated complex-terrain settings. The dynamic entropy-coefficient mechanism supports efficient early-stage exploration and is associated with statistically significant gains in convergence speed and linear velocity tracking (Section 4.1.3). The VHIP-based reward provides an explicit stability-oriented signal that helps reduce dynamically unstable gaits; the MuJoCo ablation (Section 4.1.1) shows a statistically significant reduction in fall rate at the highest terrain level.

The ablation studies suggest that both components are beneficial in our setup. Removing the VHIP reward weakens stability on harder terrains, while removing the dynamic entropy mechanism leads to slower convergence and lower final performance in the tested training pipeline. The stability-reward ablation was conducted in MuJoCo for direct comparability with Xiao et al. [4]; the main training and DreamWaQ comparison were conducted in Isaac Gym. Therefore, the benefit of our reward design in the main pipeline is supported jointly by the overall comparison and the MuJoCo ablation.

The real-world deployment results provide qualitative evidence that the policy remains deployable across the tested scenarios, including rough woven-bag surfaces, smooth marble floors, bushes, and deep grassland. We attribute this behavior mainly to two factors: (1) proprioceptive-only perception reduces reliance on hard-to-simulate visual appearance, and (2) domain randomization exposes the policy to broad dynamic variation during training.

The overall comparison with DreamWaQ uses five random seeds per method; the non-significant result for final terrain level ($p \approx 0.08$) may reflect limited statistical power for this metric, whereas linear velocity tracking and convergence speed show significant gains. This proprioceptive-only control method has an inherent limitation: the robot typically needs physical contact before adapting its gait, which can reduce performance under sudden terrain transitions. In future work, we plan to incorporate exteroceptive inputs (e.g., elevation maps) through multi-modal fusion for anticipatory adjustment, and to explore compliant motor control (e.g., virtual spring-based joint control) for more adaptive motion behaviors.

6 Conclusions

In this work, we present a DreamWaQ-aligned extension that augments the baseline with dynamics-informed reward terms (VHIP and stand-still pose) and a multi-metric dynamic entropy-adjustment mechanism for PPO. Under the evaluated simulation settings, this extension improves training behavior and locomotion performance relative to the DreamWaQ baseline, with statistically significant gains in linear velocity tracking and convergence speed and a positive trend in final terrain level. The real-world section provides qualitative deployment evidence on stairs, rough and smooth surfaces, and grassland. These results suggest that, within a DreamWaQ-consistent pipeline, combining stability-oriented reward design with adaptive entropy control is a practical direction for improving proprioception-based quadruped locomotion.

Acknowledgements

The authors thank Hangzhou Kuangxing Technology Co., Ltd. for providing the DeepRobotics Lite3 Explorer quadruped robot.

References

[1] Long, J., Wang, Z., Li, Q., Gao, J., Cao, L., Pang, J.: Hybrid internal model: Learning agile legged locomotion with simulated robot response. arXiv preprint arXiv:2312.11460 (2023)

[2] Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., Hutter, M.: Learning quadrupedal locomotion over challenging terrain. Sci. Robot. **5**, 5986 (2020)

[3] Nahrendra, I.M.A., Yu, B., Myung, H.: Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. In: Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, pp. 5078–5084 (2023)

[4] Xiao, E., Dong, Y., Ma, J., Lu, P.: Stable imitation of multigait and bipedal motions for quadrupedal robots over uneven terrains. Adv. Robot. Res. **202500036** (2025)

[5] Kang, D., Cheng, J., Zamora, M., Zargarbashi, S.H., Coros, S.: Rl+model-based control: Using on-demand optimal control to learn versatile legged locomotion. IEEE Robot. Autom. Lett. **8**, 6619–6626 (2023)

[6] Winkler, A.W., Farshidian, F., Neunert, M., Pardo, D., Buchli, J.: Online walking motion and foothold optimization for quadruped locomotion. In: Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, pp. 5308–5313 (2017)

[7] Kim, B., Lim, H., Park, S.: Spring-loaded inverted pendulum modeling improves neural network estimation of ground reaction forces. J. Biomech. **113**, 110069 (2020)

[8] Luo, Z., Xiao, E., Lu, P.: Ft-net: Learning failure recovery and fault-tolerant locomotion for quadruped robots. IEEE Robot. Autom. Lett. **8**, 8414–8421 (2023)

[9] Lee, K.: Adaptive tsallis entropy regularization for efficient reinforcement learning. In: Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), Jeju, South Korea, pp. 668–673 (2022)

[10] Zhang, Y., Yue, M., Wang, J.: Off-policy deep reinforcement learning with automatic entropy adjustment for adaptive online grid emergency control. Electr. Power Syst. Res. **217**, 108912 (2023)

[11] Kumar, A., Fu, Z., Pathak, D., Malik, J.: Rma: Rapid motor adaptation for legged robots. arXiv preprint arXiv:2107.04034 (2021)

[12] Kumar, A., Li, Z., Zeng, J., Pathak, D., Sreenath, K., Malik, J.: Adapting rapid motor adaptation for bipedal robots. In: Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, pp. 1161–1168 (2022)

[13] Luo, Z., Dong, Y., Li, X., *et al.*: Moral: Learning morphologically adaptive locomotion controller for quadrupedal robots on challenging terrains. IEEE Robot. Autom. Lett. **9**, 4019–4026 (2024)

[14] Zhao, L., Luo, Z., Han, Y., *et al.*: Learning aggressive animal locomotion skills for quadrupedal robots solely from monocular videos. npj Robotics **3**, 32 (2025)

[15] Xiao, E., Dong, Y., Lam, J., *et al.*: Learning stable bipedal locomotion skills for quadrupedal robots on challenging terrains with automatic fall recovery. npj Robotics **3**, 22 (2025)

[16] Seto, R., Li, G., Kutsuzawa, K., Owaki, D., Hayashibe, M.: Two-stage learning of cpg and postural reflex toward quadruped locomotion on uneven terrain with simple reward. IEEE Access **13**, 106103–106114 (2025)

[17] Zhuang, Z., Fu, Z., Wang, J., Atkeson, C., Schwertfeger, S., Finn, C., Zhao, H.: Robot parkour learning. In: Proceedings of the Conference on Robot Learning (CoRL), Atlanta, GA, USA (2023)

[18] Luo, S., Li, S., Yu, R., Wang, Z., Wu, J., Zhu, Q.: Pie: Parkour with implicit-explicit learning framework for legged robots. IEEE Robot. Autom. Lett. **9**, 9986–9993 (2024) https://doi.org/10.1109/LRA.2024.3459797

[19] Cheng, X., Shi, K., Agarwal, A., Pathak, D.: Extreme parkour with legged robots. arXiv preprint arXiv:2309.14341 (2023)

[20] Long, J., Yu, W., Li, Q., Wang, Z., Lin, D., Pang, J.: Learning h-infinity locomotion control. arXiv preprint arXiv:2404.14405 (2024)

[21] Xue, H., Pan, C., Yi, Z., Qu, G., Shi, G.: Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing. In: Proceedings of the 2025 IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, USA, pp. 4974–4981 (2025)

[22] Guan, K., Yamamoto, K., Nakamura, Y.: Virtual-mass-ellipsoid inverted pendulum model and its applications to 3d bipedal locomotion on uneven terrains. In: Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, pp. 1401–1406 (2019)

[23] Margolis, G.B., Agrawal, P.: Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In: Proceedings of the Conference on Robot Learning (CoRL), Auckland, New Zealand (2022)

[24] Chen, B.: Optimization of soft actor-critic algorithm based on model performance and sample bias. PhD thesis, South China University of Technology, Guangzhou, China (2023)

[25] Rao, N., Xu, H., Qi, Z., *et al.*: Communication jamming resource allocation method based on maximum policy entropy deep reinforcement learning. J. Northwestern Polytech. Univ. **39**, 1077–1086 (2021)

[26] Liu, Q., Zhu, F., Ling, X.H., Lu, J.Y.: Meta-reinforcement learning algorithm based on automating policy entropy. Comput. Sci., 168–174 (2021)

[27] Wang, J.Y., Wang, Z., Li, H.X., Chen, C.L.: Adaptive noise-based evolutionary reinforcement learning with maximum entropy. Acta Autom. Sinica **49**, 54–66 (2023)

[28] Escontrela, A., *et al.*: Adversarial motion priors make good substitutes for complex reward functions. In: Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, pp. 25–32 (2022)

[29] Luo, J., Xu, C., Wu, J., Levine, S.: Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. arXiv preprint arXiv:2410.21845 (2024)

[30] Ji, G., Mun, J., Kim, H., Hwangbo, J.: Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. IEEE Robot. Autom. Lett. **7**(2), 4630–4637 (2022) https://doi.org/10.1109/LRA.2022.3152244

[31] Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., *et al.*: Isaac gym: High performance gpu-based physics simulation for robot learning. In: Advances in Neural Information Processing Systems, vol. 34. Virtual, pp. 24508–24520 (2021)

Statements and Declarations

Funding. The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Competing interests. The authors have no relevant financial or non-financial interests to disclose.

Author contributions. All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Jiale Chen, ZhenYao Zhang and Lingyun Kong. The first draft was written by Jiale Chen

and all authors commented on previous versions. All authors read and approved the final manuscript.

Ethics approval. Not applicable.

Consent to participate. Not applicable.

Consent for publication. Not applicable.

Data availability. The source code and key training configurations are available at https://github.com/defineyumi/rl_training_dwaq.git. Training logs and raw run data can be provided by Lingyun Kong upon reasonable request.