HACL: History-Aware Curriculum Learning for Fast Locomotion

Prakhar Mishra[1], Amir Hossain Raj[2], Xuesu Xiao[2], Dinesh Manocha[1]
[1] University of Maryland, College Park
[2]George Mason University
Videos and code: URL: https://prakharmishra27.github.io/HACL-2025/

***Abstract*—We address the problem of agile and rapid locomotion, a key characteristic of quadrupedal and bipedal robots. We present a new algorithm that maintains stability and generates high-speed trajectories by considering the temporal aspect of locomotion. Our formulation takes into account past information based on a novel history-aware curriculum Learning (HACL) algorithm. We model the history of joint velocity commands with respect to the observed linear and angular rewards using a recurrent neural net (RNN). The hidden state helps the curriculum learn the relationship between the forward linear velocity and angular velocity commands and the rewards over a given time-step. We validate our approach on the MIT Mini Cheetah, Unitree Go1, and Go2 robots in a simulated environment and on a Unitree Go1 robot in real-world scenarios. In practice, HACL achieves peak forward velocity of 6.7 m/s for a given command velocity of 7m/s and outperforms prior locomotion algorithms by nearly 20%.**

I. Introduction

One key capability of bipedal and quadrupedal robots is to maneuver rapidly and with agility in different scenarios. Moving fast allows the robot to traverse large distances quickly and navigate diverse terrains effectively and efficiently. However, performing fast locomotion on diverse terrains gives rise to many challenges because we do not have accurate information about the real world characteristics around the robot, e.g., dynamics parameters like friction, obstacles, uneven terrain, slippery or inclined surfaces, etc. One possibility is to model the environment parameters [47], [11]. However, designing robust techniques based on model-based control requires a significant amount of expertise and creativity on the part of the human designer.

Another possibility is to infer these parameters [22], [16]. The current paradigm is to train the robot in simulation and use state-of-the-art techniques [38] to deploy it on the real robot in the physical world. These methods achieve either high speed or stability, but not both. Although rapid locomotion is highly desirable, it should not compromise the stability and efficiency of the robot. Operation in the real world can result in many challenges in the form of terrain diversity, such as sudden slopes, slippery surfaces, icy patches, or obstacles. The major issues in terms of using learning methods include sparse rewards distribution, the exploration-exploitation dilemma, the sim-to-real gap, and the non-Markovian nature of locomotion [6], [46].

One way to overcome such challenges is to use a curriculum [28], [22], [29], [18], [16]. These methods first train the robot on smaller velocity ranges. In the process, they gradually increase the difficulty if the robot succeeds in performing easier tasks, decrease difficulty if the robot performs worse, and go to a random difficulty level if the agent performs well on even the most difficult tasks. The objective of curriculum learning is to improve training efficiency and final performance, but it can result in overfitting[5] or suboptimal exploration [22], [12].

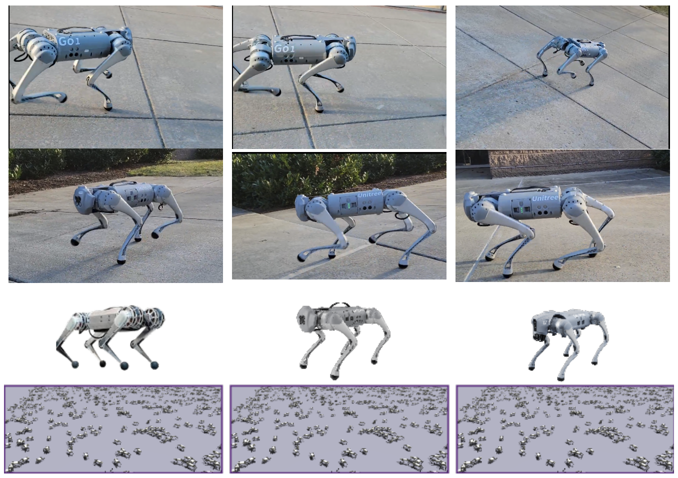


Fig. 1: **Testing and parallelized training:** Application of HACL in the real-world on the Go1 robot (above 2 rows). Parallelized training of MIT Mini Cheetah, Go1, and Go2 quadrupeds with HACL in Nvidia IsaacGym using 4000 environments (bottom row).

Main Results: We present a new approach for fast and stable locomotion that takes into account temporal dependencies by incorporating history in the curriculum. Many approaches for locomotion perform sequential decision-making and have inherent temporal dependencies due to their non-Markovian nature [6], [46], [44]. We present a novel History-Aware Curriculum Learning (HACL) algorithm, which addresses issues of temporal dependencies and suboptimal exploration, and improves the forward linear velocity computation and stability. The novel components of our approach include:

- We present history-aware curriculum learning (HACL), a curriculum approach that schedules the next episode bins based on past observed training rewards for linear and angular velocities (r_{lin}, r_{ang}). Our formulation also models temporal dependencies.

- We show that by capturing historical information, we can bridge the gap between non-Markovian dynamics and curriculum learning by incorporating past state information (h_{t-1}). This leads to improved convergence and learning.
- We have conducted an extensive evaluation of our method in simulated environments on MIT Mini Cheeta, Unitree Go1 and Go2 robots in NVIDIA IsaacGym. We also highlight the benefitson a Unitree Go1 robot operating in the real world. Our results show that HACL achieves higher speed, stability, and improved efficiency. HACL achieves velocity of 6.7m/s with a standard deviation of 1.73 m/s, which is significantly better than [22]. The stability score is around 2000 and energy efficiency is -4000, which is much better compared to the current RL methods. HACL consumes less energy for the higher velocity output compared to fixed curricula, bandit-based task sampling including UCB [3], Thompson [7]), and non-history neural networks such as MLP [30] and CNN [17].

II. Related Work

A. Curriculum learning in Robotics

Bengio et al. [4] propose a method similar to continuation methods where training happens in a progression, starting with easier examples and gradually increasing the difficulty. Wang et al. [39] discuss how to implement/design a predefined curriculum or an automatic curriculum. Their work categorizes automatic curriculum learning (CL) into 4 categories: Self-paced Learning, Transfer Teacher, RL Teacher Matiisen et al. [23], and Other Automatic CL. While [2] employs a multi-terrain curriculum approach and similarly environment progression based by curriculum [16] for learning locomotion skills. And [8] ,[19] discuss hindsight curriculum learning, Wang et al. [40] arm wheel curriculum, and Tidd et al. [37] guided curriculum for terrains. While [22], employed a fixed rule based curriculum update method for learning the locomotion control policy, absence of history information was common across all these works.

B. Model-based control and Reinforcement Learning for Locomotion

Some of the earliest attempt to implement model-based control can be dated back to DARPA's learning research in locomotion [48], [15], [24]. However, A key limitations of these methods regarding fast locomotion is sim-to-real gap. To reduce the gap between sim-to-real, several methods have been proposed like domain randomization [38]. Domain randomization gives robust behavior, but it can result in a conservative policy [20]. Alternatively, one can design simulation environment as close as possible to the real world scenarios [11], [33]. But they would require extensive amount of real world data and need to be setup for different configurations. One way to avoid these issues is to infer the data during deployment [16], [22].

Symbol(s)	**Description**
v_x , ω_z	The linear velocity, and angular velocity of the robot base
v_x^{cmd} , ω_z^{cmd}	The forward command linear velocity, and angular velocity of the robot base
$\hat{r}_{\text{lin}}$, $\hat{r}_{\text{ang}}$	RNN predicted linear velocity rewards and angular velocity rewards
r_{lin} , r_{ang}	Observed linear velocity rewards and angular velocity rewards
τ	The torque applied by the robot joint
g	The projected gravity of the robot base
$b_1, b_2, \ldots, b_n$	The discretized design space bins totaling 4000 bins
$w_1, w_2, \ldots, w_n$	The normalized probability weights distribution for bins
$\hat{\mu}(b)$	The predictions made by RNN
h_{t-1}	The RNN hidden state for the time-step $t-1$
x_t	The one hot encoded bin index input to the RNN
w_n	The probability distribution of each bins
$f_{\text{RNN}}(h_{t-1}, x_t)$	The RNN function with the h_{t-1} and x_t as inputs
$R(T)$	The cumulative regret over time step T
$L(\theta, \phi)$	The loss function with variable θ, ϕ
q_t , $\dot{q}_t$, q_t^{des}	The robot's joint position, joint velocity and task parameter joint command at time t
η, S	Energy efficiency and the stability score of the robots
DOF	Robot's degrees of freedom

TABLE I: Symbols used and their descriptions.

C. Non-Markovian nature of locomotion

As [6, 46, 44] have pointed out, locomotion has a non-Markovian nature, where the state is not only dependent on the immediate control inputs but also the previous states [46]. These researchers have used state space re-planning strategies and robust bundles, implicitly acknowledging the importance of history. While Byl [6] suggests a non-Markovian or discrete Markov chain, DeFazio et al. [9] suggests Markovian challenges. Li et al. [19] hindsight curriculum learning also addresses Markov property and Thuruthel et al. [36] implies markovian nature of locomotion.

D. Neural network architectures and static reward strategies

Zhao and Gu [46] have used a neural net for legged locomotion while Rodriguez and Behnke [27] used neural net for capturing hidden state information. Some methods like UCB [3], Thompson sampling [35] are good for balancing exploration and exploitation. Kasaei et al. [13] explores into exploration and exploitation. However these methods have not been used to learn the hidden locomotion pattern.

III. History-Aware Curriculum Learning (HACL)

In this section, we present our novel approach that takes into account temporal aspect of locomotion. Our goal is to shape our curriculum such that it takes the observed linear velocity (r_{lin}) and angular velocity rewards (r_{ang}) and, based on our HACL algorithm, outputs the task parameters (v_x^{cmd} , ω_z^{cmd}). We also want to learn a policy (π_θ) that takes sensory observation (o_t) inputs like joint position (q_t) and joint velocity ($\dot{q}_t$) and outputs the parameterized joint position commands (q_t^{des}), as shown in Fig. 2.

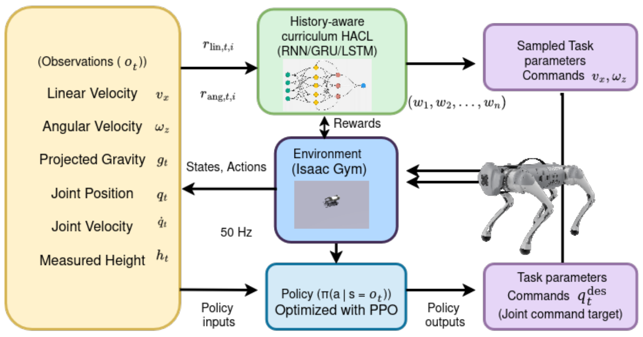

(Observations (o_t))
Linear Velocity v_x
Angular Velocity ω_z
Projected Gravity g_t
Joint Position q_t
Joint Velocity $\dot{q}_t$
Measured Height h_t
$r_{\text{lin},t,i}$
$r_{\text{ang},t,i}$
History-aware curriculum HACL (RNN/GRU/LSTM)
Sampled Task parameters Commands v_x, ω_z
$(w_1, w_2, \ldots, w_n)$
Rewards
States, Actions
Environment (Isaac Gym)
50 Hz
Policy inputs
Policy (π(a | s = o_t)) Optimized with PPO
Policy outputs
Task parameters Commands q_t^{des} (Joint command target)

Fig. 2: **HACL overview.** Our HACL module receives the observations and rewards from the IsaacGym environment. HACL learns the hidden pattern between these reward distributions and the high-level sampled task parameters. The policy model is optimized using **PPO** and generates the low-level task parameter commands, which the robot executes in the environment.

A. Design space

Curriculum learning is a popular method for learning locomotion skills [22], [18], [45]. In a simulated environment, the design space is divided into a discrete set of bins, represented using B (1). The curriculum design space is divided into the bins (b_i):

$$B = \{b_1, b_2, \ldots, b_n\}. \tag{1}$$

Each individual bin carries command task parameters and target linear (v_x^{cmd}), and angular velocities (ω_z^{cmd}); HACL aims to optimize the bins that can give the best rewards to the agent over a period of time, capturing this temporal information using RNN hidden state h_{t-1}. Therefore, the agent learns locomotion conditioned on a command linear and angular velocity.

Based on this, our curriculum design method is similar to [28], [22], [29], [18]. Moreover, it will: 1) increase difficulty if the policy performs better; 2) decrease difficulty if the policy performs worse; 3) go to a random difficulty if the policy performs well on the highest difficulty.

As [22], [16] , [11] have observed, robots learn better when the velocities are selected from small probability distribution in low range of $[-1.0, 1.0]$ and fails to learn when velocities are selected from the large distribution, e.g., $[-6, 6]$. Learning fails when velocity commands $((v_x^{cmd}), (\omega_z^{cmd}))$ are sampled from large distribution, which could be caused by a lack of initial curriculum or sparse rewards [22], a very large exploration space and possible dynamics constraints [28]. We have observed that this is the case regardless of the robot type (e.g. MIT Mini Cheetah, GO1, or GO2) used in our simulated environments.

B. Non-Markovian nature of legged locomotion

There are many challenges in locomotion like sparse rewards, poor learning, and human design errors [22], [28], [42]. We have considered integrating the temporal aspect of locomotion in curriculum learning to overcome these challenges and have highlighted its generalization in our results section. Our approach also overcomes limitation of exploration and exploitation faced by traditional curriculum approaches [22], [16].

We frame our approach based on q-learning [41],which has been proposed by others [34] for long-horizon, open-world tasks. In q-learning, the agent learns a value function $Q(s_t, a_t)$ given by equation (2), where s_t, a_t are the state and action at time-step t, r and $E_{a'}$ are the immediate rewards and expectation over future actions from state s_t and γ is the discount factor.

$$Q(s_t, a_t) = r + \gamma E_{a'}[Q(s_{t+1}, a')]. \tag{2}$$

Equation (2) assumes the Markovian nature of state-action dynamics [26], where the next state depends on the current state s_t and action a_t. However, as [46], [44] , [6] have observed, modeling past history can be crucial for rapid and agile maneuvers and unmodeled history parameters can affect the robot's stability, speed, and energy efficiency. One way to solve this is to model these history parameters. So the above q-learning equation can be modified in order to capture the non-Markovian dynamics of locomotion by slightly modifying it (2) into (3) to incorporate the history or "hidden" state h_{t-1} given by (3).

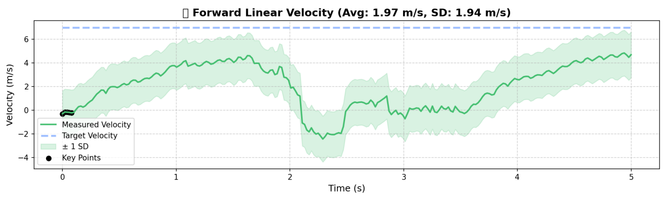
$$Q\left(S_t, A_t \mid h_{t-1}\right) = r + \gamma E_{A'}\left[Q\left(S_{t+1}, A' \mid h_t\right)\right]. \tag{3}$$

The hidden state (h_{t-1}) captures the information from previous time-steps and allows the learned policy to capture the long-term effects of the actions on the joints and joint velocities based on the rewards received (r_{lin}, r_{ang}), which the fixed update curriculum or other similar curriculum methodology are not able to capture [22], [28]. To model (h_{t-1}), we have incorporated the Recurrent Neural Network [30], using (4) in our curriculum learning:

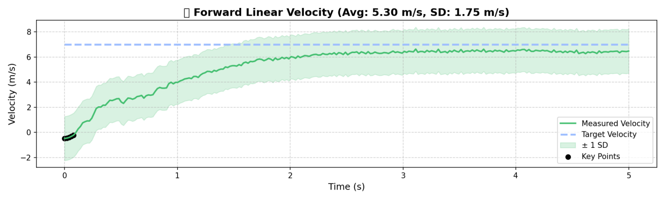
$$h_t = f(h_{t-1}, x_t; \theta). \tag{4}$$

C. RNN-based h_{t-1} updates

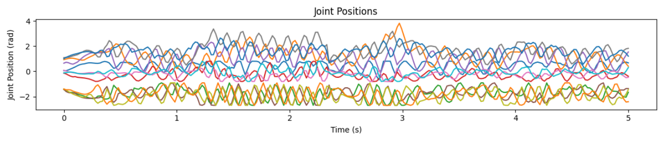
While neural nets have been used in locomotion for various reasons such as training efficiency[32], the importance of history in locomotion has not been explored, particularly not along with a curriculum. While [22], [29], [11], [45], etc. use curriculum, they do not use any method to track or model history. Therefore, while such methods achieve the desired outputs, they lack the skills to achieve a very high velocity or good stability. To address this and model the history or "hidden" state h_{t-1} of the equation (3), we propose a Recurrent Neural Net (RNN) based History-Aware Curriculum Learning (HACL).



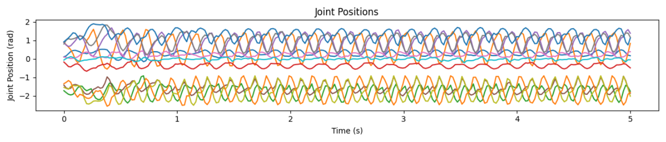
(a) Margolis et al. [22] Velocity graph for 7m/s (Baseline)



(b) HACL Velocity graph 7m/s (Ours)



(c) Margolis et al. [22] Joints position graph (Baseline)



(d) HACL Joints position graph (Ours)

Fig. 3: **Comparison of prior methods with HACL**. As per (a), Margolis et al. [22] grid-adaptive curriculum reaches a maximum velocity of 5.8 m/s for command velocity of 7 m/s and standard deviation of 1.94 m/s, while (b) HACL achieves a higher velocity of 6.7m/s with a standard deviation of 1.73 m/s. In (c) we observ the chaotic nature of the joint position graph with respect to time, indicating poor learning at higher command velocities. (d) The other HACL's smooth joint position graph indicates better control at a higher velocity along with a stable gait. Overall, HACL results in stable and faster locomotion as compared to prior methods.

Metric	**UCB**	**Thompson Sampling**	**Hwangbo et al. [11]**	**Kumar et al. [16]**	**Margolis et al. [22]**	**Aractingi et al. [2] +Margolis et al. [22]**	**Aractingi et al. [2] + HACL**	**HACL**
v_x m/s (v_x^{cmd}= 6)	0.23	0.16	1.5	1.8	5.45	5.3	5.5	6.0
v_x m/s (v_x^{cmd}= 7)	0.2	0.11	1.5	1.8	5.70	5.48	5.8	6.7
ω_z rad/s	0	0	N/a	N/a	1.8	0.9	0.8	1.25
r_{lin}	0.0745	0.0046	-	-	0.02	0.014	0.07	0.087
History-aware	Yes*	Yes*	No	No	No	No	Yes	Yes
Energy Efficiency (η)	-6500	-7300	-	-	-6200	-6800	-5400	-4000
Stability (S)	749	640	-	-	1200	1100	1150	2000
Robots Tested (Sim)	Go1	Go1	AnyMal	A1	Mini cheetah	Go1	Go1	**Mini cheetah, Go1, Go2**
Task Success Rate	20%	0%	-	-	70%	70%	80%	90%

TABLE II: **Comparison of HACL with SOTA:** HACL achieves the highest velocity for all the methods for a command velocity of 7 m/s, and it reaches 6.7 m/s. While previous works such as Margolis et al. [22], [2] + Margolis et al. [22], [2] + HACL achieve velocities of around 5.3-5.8 and Hwangbo et al. [11], Kumar et al. [16] are around 1.5-1.8, none of them surpasses HACL. HACL is also very energy efficient (-4000) compared to other works, indicating they consume more energy during locomotion while UCB and Thompson Sampling performs worse for locomotion tasks. Stability score (S) is highest for HACL and lowest for the static reward strategies but Margolis et al. [22] and Aractingi et al. [2] follows HACL in terms of stability.

RNNs (whether LSTMs, GRUs, or regular RNN) are better suited for capturing long-term dependencies because they maintain a hidden stateh_{t-1}, which captures the past information. The curriculum can then adapt online to the robot's evolving performance over a period of time. In our discrete curriculum design bins, each bin has specific ID, and we define this input x_t (5) as one-hot encoding of bins at time-step t:

$$x_t = \text{one-hot}(\text{BinID}_t) \in R^{4000}, \tag{5}$$

where 4000 represents the total number of discretized bins. The RNN predicts (6) the expected linear and angular velocity for the encoded bins.

$$\hat{\mu}(b) = f_{\text{RNN}}(h_{t-1}, x_t) = \begin{bmatrix} \hat{r}_{\text{lin}}(b_i) \\ \hat{r}_{\text{ang}}(b_i) \end{bmatrix}, \tag{6}$$

where h_{t-1} is the hidden state from the previous time-step and $\hat{r}_{\text{lin}}(b_i)$, $\hat{r}_{\text{ang}}(b_i)$ represents the predicted linear and angular rewards for the bin b_i.

$$h_t = \text{LSTM}(h_{t-1}, [x_t, r_t(b_t)], \theta). \tag{7}$$

We feed the observed linear (r_{lin}) and angular (r_{ang}) reward ($r_t(b_t)$; each bin contains both the bin IDs BinID_t at time-step t to the RNN (LSTM, GRU, or regular RNN).

D. Predicted returns and curriculum updates

As [46], [44] suggested, locomotion is non-Markovian in nature, i.e., the current state is not only dependent on the previous state but also the other states preceding it. Each bin (b_i)in equation (1) is asssigned a respective probability weight (w_i) distribution given in equation (8).

$$W = \{w_1, w_2, \ldots, w_n\}. \tag{8}$$

The weights of bins which lead to better rewards are increased using RNN predictions (4) and is given by the equation (9):

$$w_{t+1}(b) = w_t(b) + \alpha \left(\hat{r}_{\text{lin}}(b) + \hat{r}_{\text{ang}}(b)\right), \tag{9}$$

As the training progresses, we collect more observed linear (r_{lin}) and angular (r_{ang}) rewards, which in turn improve the predictions and minimize our loss function. This in turn also affect the command sampling for the next timestep for the given episode, given by (10), indicating the sampling of command from that particular bin:

$$P_\theta(c \in b) = \frac{w_b}{\sum_{b' \in B} w_{b'}}, \tag{10}$$

Let the hidden state h_t of our curriculum captures the non-Markovian dynamics of locomotion by learning the hidden pattern between sampled command and observed rewards.

E. Training Objective

The RNN specifically retains the information across time using hidden layers, it is the ideal choice for making our curriculum history-aware. We have compared our approach with other approaches like fixed rule- based methods, static reward methods, and other non-history neural networks. We have defined our loss function (11) as sum of minimizing the difference between observed linear velocity r_{lin} and predicted linear velocity $\hat{r}_{\text{lin}}$ plus the difference between observed angular velocity r_{ang} and predicted angular velocity $\hat{r}_{\text{ang}}$ over a a period of timestep T.

$$L(\theta, \phi) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{N} \Big((r_{\text{lin}} - \hat{r}_{\text{lin}})^2 + (r_{\text{ang}} - \hat{r}_{\text{ang}})^2 \Big). \tag{11}$$

F. Comparison of HACL with Static Reward strategies and non-history Neural Nets

Thompson Sampling: Thompson Sampling [7] tries to balance exploration (potential rewards for the arms) and exploitation (final selection of an arm that gives you the highest reward). It increases the training time (550 minutes) significantly, as the curriculum is stuck balancing exploration and exploitation and might not converge as fast as other methods. One possible reason is it over explore without any concrete convergence and it requires too much hyper-parameter tuning without significantly improving the curriculum. In this case equation (9) becomes (12):

$$w_{t+1}(b) = \text{Beta}(\alpha_b, \beta_b). \tag{12}$$

UCB (Upper Confidence Bound): Strategies like UCB and Thompson Sampling don't consider history in a strict sense like HACL, but they do assume a static nature of reward distribution. The problem with UCB and Thompson sampling is that they ignore the continuous rewards over time and only count the success/failure rates. Since UCB counts only success or failure, it might also over-penalize bins that might give some good rewards later as the robot improves its gait and masters balance. The bin weights in case of UCB is given by (13):

$$w_{t+1}(b) = \hat{\mu}_b + \sqrt{\frac{2 \ln t}{n_b}}. \tag{13}$$

Non-History Neural Nets: We have analyzed the fixed schedule, static reward strategies, and history-aware methods. Out of curiosity we also explored the effect of non-history neural nets like Convolutional Neural Net (CNN) and Multilayer perceptron (MLP). UCB and Thompson Sampling perform worse than non-history methods like CNN or MLP. While non-history methods like CNN or MLP perform better than UCB and Thompson Sampling because, even though they do not consider the history of the rewards or assume static distribution, they are still better because they are good in capturing the relevant information at each timestep and maybe gradient updates leads to achieving better rewards (Table II).

IV. Experimental Validation

A. Simulation environment:

We model the robot, with a total of 12 Degrees of Freedom (DoF), as an URDF file in the simulated environment (IsaacGym Simulator [21]). Each RL episode lasts for $20s$ with a total of 4000 parallel environments with a time-step of $dt = 0.005$. The robot then receives a sampled linear command velocity of $v_x^{cmd} \in [-1, 1]$ and an angular command velocity of $\omega_x^{cmd} \in [-1, 1]$.

We have adapted our code mostly from the open-source repositories [22], [29]. Similar to [22], we have also trained 400 million timesteps in simulation using 4000 environments of Go1, Go2, and MIT mini cheetah robots for validating our history-aware curriculum. For training, we have used an Nvidia RTX 4090 laptop-based GPU, which takes less than 2 hours to finish the whole simulation.

The design space of the IsaacGym simulator has dimensions (X, Y, Z), where the X-axis is split into $x = (-1, 1)$ with 20 bins; the Y-axis is split into $y = (-1, 1)$ with 10 bins; and the Z-axis is split into $z = (-1, 1)$ with 20 bins, thereby creating a total of 4000 bins. In our HACL method, we have kept our velocity ranges as follows: linear velocities as $v_x \in [-1.0, 1.0]$ m/s during the initial phase and angular velocities as $\omega_x \in [-1.0, 1.0]$ m/s.

B. Domain randomization and reward functions

We randomized joint friction , motor delays, and sensor noise and achieved a total velocity of 5.3 m/s, which is slower than the 5.45 m/s [22] achieved for command velocity of 6m/s. Due to over randomization of parameters the policy became conservative and hence this reduction in velocity [22][33] [43]. We therefore reverted to the original parameters with minor modifications to their ranges. Reward functions are derived from [22] [29], with minor tweaking.

C. Teacher-student training and policy optimization

Similar to [22], [29], we also deploy the student policy $\pi_S(x_t, x_{[t-h:t-1]})$ on the Go1 robot and let the policy infer the dt parameter using the state history of h time stamp, which performs the online system identification [16][18]. Our history-aware curriculum helps dynamically adjust the difficulty based on the rewards signal received and thereby avoids the pitfall of over-fitting. For policy optimization, we have used Proximal Policy Optimization (PPO) [31]. Our HACL is incorporated within this framework, so the policy is refined based on the return for linear and angular velocities and the curriculum adjusts the difficulty in order to maximize the cumulative rewards over a period of time.

D. Hardware:

To validate our experiment, we have used a Unitree Go1 EDU robot. The robot has a weight of around 12 kg, a payload of 5 kg, and a height of 40 cm. The robot has a total of 12 Degrees of Freedom (DOF), meaning it's essentially composed of 12 servo motors. The sensor suite of Go1 EDU consists of raspberry pi, an inertial measurement unit (IMU), a joint position endorse, and an Nvidia jetson, on which our trained policy incorporating HACL runs.

V. Results and Evaluation

A. Evaluation metrics

One of the key metrics for our evaluation was the energy efficiency (η) of the Go1 robot. While training the policy in simulation and while doing sim-to-real, we want to make sure that the robot does not consume too much power. The energy or power consumed by each joint at each timestep is given by the torque required for each joint at the timestep, i.e., $\tau_j(i)$ multiplied by $\dot{q}_j(i)$, the joint velocity for each joint at timestep i. The distance traveled per timestep is given by $\mathbf{v}_x$ time $\mathbf{dt}$, using (14)

$$\eta = \frac{\sum_{j=1}^{M} \tau_j(i) \cdot \dot{q}_j(i)}{\|\mathbf{v}_x(i)\| \cdot \Delta t}. \tag{14}$$

We have defined the **overall stability** score (S) as the cumulative sum of key indicators that are important for stable locomotion. In the given formula (15), r_i maintains a desired orientation, heights and safety constraints at any given timestep.

$$S = \sum_{i=1}^{N} \Big(r_{\text{orient}}(i) + r_{\text{base height}}(i) + r_{\text{ang vel xy}}(i) + r_{\text{lin vel z}}(i) + r_{\text{dof pos limits}}(i) + r_{\text{dof vel limits}}(i) + r_{\text{dof vel}}(i) + r_{\text{collision}}(i) + r_{\text{torque limits}}(i) \Big)$$

We have also defined the task success rate as the percentage of successful runs divided by the total runs (unsuccessful runs include any in which the robot crashes, tips, or touches the ground) when the trained weights are run for 10 timesteps after a fully trained policy. And most importantly the generalization across multiple robots, in this case, we demonstrate the performance on MIT Mini Cheetah [14], Unitree Go1 [1], and Unitree Go2 [1] robots.

B. History-aware curriculum lead to a very fast locomotion

HACL leads to a very rapid locomotion, achieving 6.7m/s for a command velocity of 7m/s. While previous works such as [22], [2], achieve velocities within range [5.3, 5.8] m/s and [11], [16] are around 1.5 and 1.8, none of them surpasses HACL. We also demonstrate how HACL can be generalized to quadrupeds like MIT Mini Cheetah [14], Unitree Go1 [1], and Unitree Go2 [1]. HACL enables adaptive learning by utilizing history (h_{t1}) and significantly improving velocity tracking, energy efficiency, stability and task success rate.

Metric	**RNN**	**LSTM**	**GRU**	**CNN**	**MLP**
v_x m/s ($v_x^{cmd} = 6$)	6.0	6.0	6.0	0.6	0.56
v_x m/s ($v_x^{cmd} = 7$)	6.62	6.58	6.72	0.62	0.5
ω_z rad/s	1.25	1.2	1.28	0	0
r_{lin}	0.0823	0.0873	0.085	0.08	0.07
History-Aware	Yes	Yes	Yes	No	No
Energy Efficiency (%)	-2600	-4000	-5200	-200,000	-250,000
Stability (S)	1400	2000	1900	2000	1100
Task Success Rate	80%	90%	90%	0%	0%

TABLE III: **History-Aware Curriculum Learning (HACL) vs. non-history methods:** HACL enables adaptive learning by utilizing history (h_{t-1}) and significantly improving velocity tracking, energy efficiency, stability and task success rate. History-aware network (RNN, LSTM, GRU) outperform non-history networks (CNN, MLP) achieving 5-10 times better velocity, energy efficiecy and task success rate. HACL has been validated across various robots like MIT mini cheetah, Go1, Go2 in simulation.

C. Testing in the real world on a Unitree Go1 robot

Currently, We are testing our trained model weights on the Unitree Go1 robot and testing various use case scenarios like for $v_x^{cmd} \in [3, 6]$ and $\omega_z^{cmd} \in [2, 6]$ rad/s. And we are also testing on various terrains like grass, concrete and smooth surface. We have limited our v_x^{cmd} within 6m/s due to hardware constraints. Our results have shown that Unitree can achieve a real-world velocity in range of 3.2-4.4 m/s.

VI. Conclusion, Limitations and Future Work

We have presented a novel approach (HACL) for fast and stable locomotion of quadruped and bipedal robots. Our approach exploits the temporal aspects of locomotion. The main benefits of HACL include: (i) improved linear velocity; (ii) better stability than the current methods [22], [16], [2]; (iii) higher task success rate; (iv) improved learning (higher r_{lin} rewards); (v) better energy efficieny and less consumption even at higher velocities. We demonstrate that considering a history-based curriculum approach for legged locomotion improves the overall performance of the robot in terms of the velocity, efficiency, and stability over a series of time-steps. Most importantly, our methodology generalizes to different robots and can also be combined with other methods like [2], [22] to improve their performance. The main limitation of our work is that we have not considered the the robot morphologies [25], [10], which also affects the overall learning process. We plan to test the performance of our methods on different robots operating in complex terrains with obstacles and varying characteristics. As part of future work, we would like to extend these history-aware methods to reinforcement learning (RL) based locomotion algorithms.

References

[1] Unitree. https://www.unitree.com/. [Accessed 01-03-2025].

[2] M. Aractingi, P. Léziart, T. Flayols, J. Perez, T. Silander, and P. Souères. Controlling the solo12 quadruped robot with deep reinforcement learning. *scientific Reports*, 13 (1):11945, 2023.

[3] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[4] Y. Bengio, J. Louradour, R. Collobert, and Ja. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[5] D. Blessing, O. Celik, X. Jia, M. Reuss, M. Li, R. Lioutikov, and G. Neumann. Information maximizing curriculum: A curriculum-based approach for learning versatile skills. *Advances in Neural Information Processing Systems*, 36:51536–51561, 2023.

[6] K. Byl. *Metastable legged-robot locomotion*. PhD thesis, Massachusetts Institute of Technology, 2008.

[7] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24, 2011.

[8] Y. Chen, H. Bui, and M. Posa. Reinforcement learning for reduced-order models of legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5801–5807. IEEE, 2024.

[9] D. DeFazio, Y. Hayamizu, and S. Zhang. Learning quadruped locomotion policies using logical rules. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pages 142–150, 2024.

[10] Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pages 4455–4464. PMLR, 2020.

[11] 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(26):eaau5872, 2019.

[12] M. Jiang, M. Dennis, J. Parker-Holder, A. Lupu, H. Küttler, E. Grefenstette, T. Rocktäschel, and J. Foerster. Grounding aleatoric uncertainty for unsupervised environment design. *Advances in Neural Information Processing Systems*, 35:32868–32881, 2022.

[13] M. Kasaei, M. Abreu, N. Lau, A. Pereira, L. Reis, and Z. Li. Learning hybrid locomotion skills–learn to exploit residual dynamics and modulate model-based gait control. *arXiv preprint arXiv:2011.13798*, 2020.

[14] B. Katz, Jared Di C., and S. Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 international conference on robotics and automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[15] D. Kim, C. Di, B. Katz, G. Bledt, and S. Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*, 2019.

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

[17] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[18] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[19] S. Li, G. Wang, Y. Pang, P. Bai, S. Hu, Z. Liu, L. Wang, and J. Li. Learning agility and adaptive legged locomotion via curricular hindsight reinforcement learning. *Scientific Reports*, 14(1):28089, 2024.

[20] J. Luo and K. Hauser. Robust trajectory optimization under frictional contact with iterative learning. *Autonomous Robots*, 41:1447–1461, 2017.

[21] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[22] G. B Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43 (4):572–587, 2024.

[23] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[24] H. Park, P. Wensing, and S. Kim. High-speed bounding with the mit cheetah 2: Control design and experiments. *The International Journal of Robotics Research*, 36(2): 167–192, 2017.

[25] Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: a study of generalization via modularity. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] M. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[27] D. Rodriguez and S. Behnke. Deepwalk: Omnidirectional bipedal gait by deep reinforcement learning. In *2021 IEEE international conference on robotics and automation (ICRA)*, pages 3033–3039. IEEE, 2021.

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

[29] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep rein-

forcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[30] D. Rumelhart, G. Hinton, R. Williams, et al. Learning internal representations by error propagation, 1985.

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

[32] A. Srisuchinnawong and P. Manoonpong. An interpretable neural control network with adaptable online learning for sample efficient robot locomotion learning. *arXiv preprint arXiv:2501.10698*, 2025.

[33] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[34] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 8, 2024.

[35] W R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

[36] G. Thuruthel, G Picardi, F Iida, C Laschi, and M. Calisti. Learning to stop: a unifying principle for legged locomotion in varying environments. *Royal Society Open Science*, 8(4):210223, 2021.

[37] B. Tidd, N. Hudson, and A. Cosgun. Guided curriculum learning for walking over complex terrain. *arXiv preprint arXiv:2010.03848*, 2020.

[38] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[39] X. Wang, Y. Chen, and W. Zhu. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576, 2021.

[40] Y. Wang, Z.and Jia, L. Shi, H. Wang, H. Zhao, X. Li, J. Zhou, J. Ma, and G. Zhou. Arm-constrained curriculum learning for loco-manipulation of a wheel-legged robot. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10770–10776. IEEE, 2024.

[41] C. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[42] Z. Xie, H. Ling, N. Kim, and M. van de Panne. Allsteps: curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum*, volume 39, pages 213–224. Wiley Online Library, 2020.

[43] Z. Xie, X. Da, M. Van de Panne, B. Babich, and A. Garg. Dynamics randomization revisited: A case study for quadrupedal locomotion. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4955–4961. IEEE, 2021.

[44] W. Yu, V. Kumar, G. Turk, and C K. Liu. Sim-to-real transfer for biped locomotion. In *2019 ieee/rsj international conference on intelligent robots and systems (iros)*, pages 3503–3510. IEEE, 2019.

[45] C. Zhang, N. Rudin, D. Hoeller, and M. Hutter. Learning agile locomotion on risky terrains. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11864–11871. IEEE, 2024.

[46] Y. Zhao and Y. Gu. A non-periodic planning and control framework of dynamic legged locomotion. *International Journal of Intelligent Robotics and Applications*, 4(1): 95–108, 2020.

[47] M. Zucker, J A. Bagnell, C. Atkeson, and J. Kuffner. An optimization approach to rough terrain locomotion. In *2010 IEEE International Conference on Robotics and Automation*, pages 3589–3595. IEEE, 2010.

[48] M. Zucker, N. Ratliff, M. Stolle, J. Chestnutt, A. Bagnell, C. Atkeson, and J. Kuffner. Optimization and learning for rough terrain legged locomotion. *The International Journal of Robotics Research*, 30(2):175–191, 2011.