LocoFormer: Generalist Locomotion via Long-context Adaptation

Min Liu **Deepak Pathak** **Ananye Agarwal**
Skild AI

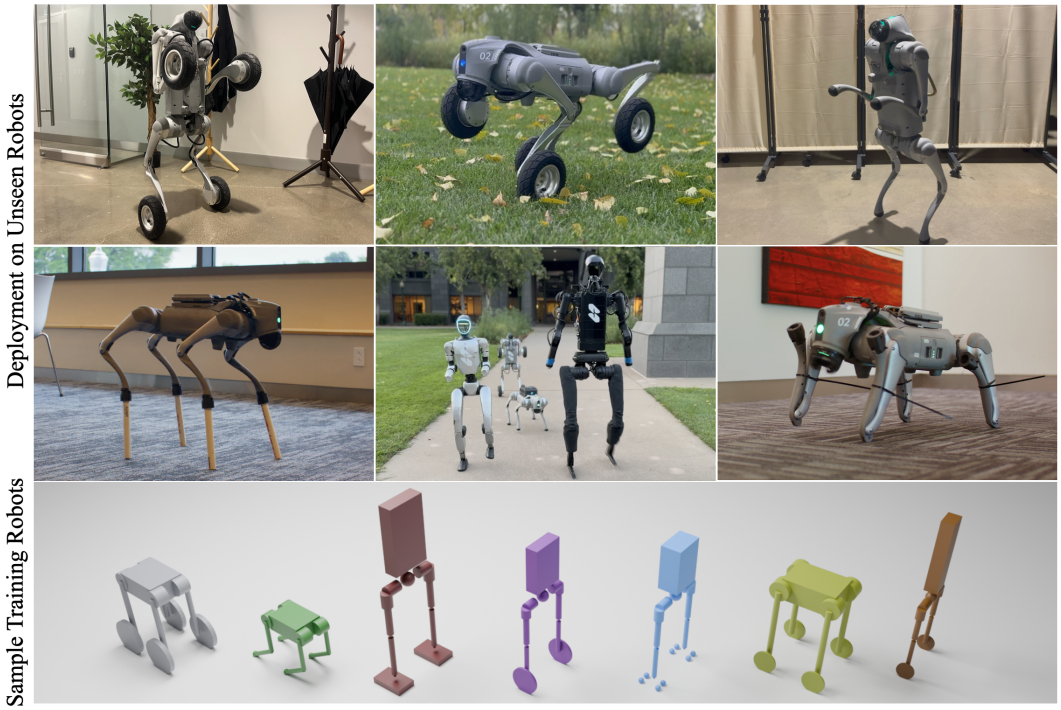


Figure 1: We propose LocoFormer: ***a generalist omni-bodied locomotion model*** that transfers *zero-shot* to any kind of robot body, whether completely different form-factor or bodily changes due to damaged limbs, without even training on them (top two rows). The key innovation is *long-term multi-episodic adaptation* trained with massively large-scale reinforcement learning – this allows generalization to a variety of real robots by just training on randomly generated procedural robots (bottom row). We see emergent adaptation behavior across very long contexts and even few-shot multi-episodic adaptation across a handful of trials where LocoFormer uses suboptimal initial rollouts to improve in subsequent ones. Videos are at generalist-locomotion.github.io.

Abstract: Modern locomotion controllers are manually tuned for specific embodiments. We present *LocoFormer*, a generalist omni-bodied locomotion model that can control previously unseen legged and wheeled robots, even without precise knowledge of their kinematics. LocoFormer is able to adapt to changes in morphology and dynamics at test time. We find that two key choices enable adaptation. First, we train massive scale RL on procedurally generated robots with aggressive domain randomization. Second, in contrast to previous policies that are myopic with short context lengths, we extend context by orders of magnitude to span episode boundaries. We deploy the same LocoFormer to varied robots and show robust control even with large disturbances such as weight change and motor failures. In extreme scenarios, we see emergent adaptation across episodes, LocoFormer learns from falls in early episodes to improve control strategies in later ones. We believe that this simple, yet general recipe can be used to train foundation models for other robotic skills in the future. Videos at generalist-locomotion.github.io.

Keywords: Cross-Embodied Learning, Legged Locomotion, Online Adaptation

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

1 Introduction

Humans and animals are remarkably flexible at control. In the case of locomotion, they exhibit highly adaptive behaviors, such as learning to walk shortly after being born [2], or adapting to limb amputations [3]. Even humans at birth possess reflexes such as alternating stepping that promote the learning of robust locomotion [4]. This is in stark contrast to robotic locomotion today, which is typically learned *tabula rasa* with reinforcement learning and tuned for a specific embodiment [5, 6, 7, 8, 9, 10]. Although such policies adapt to some randomization, this is very narrow and myopic over a timescale of a few hundred milliseconds [11, 12, 5]. This design choice favors in-distribution performance over generality, since myopic policies fail to adapt to wider task distributions, leading to lower performance. These resulting controllers therefore do not generalize to new embodiments and fail catastrophically in the case of large changes such as burnt out motors, broken legs, or modeling errors. This often means that successful sim2real transfer requires painstaking system identification for each robot. Further, a narrow task distribution is unlikely to promote the learning of general and re-usable strategies.

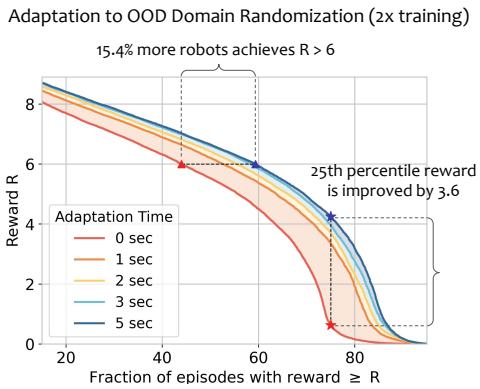


Figure 2: We test on OOD scenario by doubling the ranges of domain randomization. We see that LocoFormer uses multiple seconds of history to improve performance. In contrast, previous policies are myopic and fail to improve beyond few hundreds of milliseconds. Figure design from [1]

In this paper, We present *LocoFormer*, a generalist omni-bodied locomotion model that can control any kind of robot body, even ones that are completely out-of-distribution. It can also adapt to large changes such as locked or damaged limbs even without training on them. We hypothesize that the key to learning such controllers is to train on a wide task distribution with extreme domain randomization. When faced with large task diversity during training, policies must learn general strategies to efficiently perform task identification and then adapt control to achieve good task performance [1]. A similar phenomenon is observed in LLMs where web-scale pretraining leads to the ability to adapt in-context to new tasks at test time [13]. We design a large space of procedurally generated robots containing bipeds and quadrupeds of the legged and wheeled variety and aggressively randomize the parameters of these robots including large variations in inertia, control gains and joint limits.

Unfortunately, existing approaches are unable to learn effective policies in this task space. These only maintain a few hundred milliseconds of environment interaction in context. While this is sufficient when training specialist controllers on single embodiments, this short history doesn't contain enough information to adapt on the large task space. To increase the information available to the policy, we extend the context length by several orders of magnitude. We find that these simple changes, coupled with large-scale training, leads to emergent adaptation strategies.

We test LocoFormer on unseen real-world robots including the Go2 (wheeled and legged variants) in both biped and quadruped mode, G1 and H1 robots among others. We find that in out-of-distribution (OOD) scenarios, LocoFormer can improve performance across multiple episodes (Fig. 2). In particular, when testing with double the amount of domain randomization the policy was trained with we see that LocoFormer can continuously adapt behavior across seconds of time (at least two orders of magnitude larger than prior work). This ability has at least two practical implications. First, LocoFormer is more likely to bridge the sim2real gap better since it can withstand larger domain randomization at training time and adapt to OOD dynamics at test time. Second, since LocoFormer can adapt to large changes in morphology, robots equipped with such a policy are likely to continue to function even in the face of adverse circumstances such as the loss of a limb or worse.

Although we only present applications to locomotion in this paper, our method is simple and generally applicable in any scenario where large-scale data can be generated, such as in simulation. We envision this being useful for training generalist controllers for other low-level robotic skills as well.

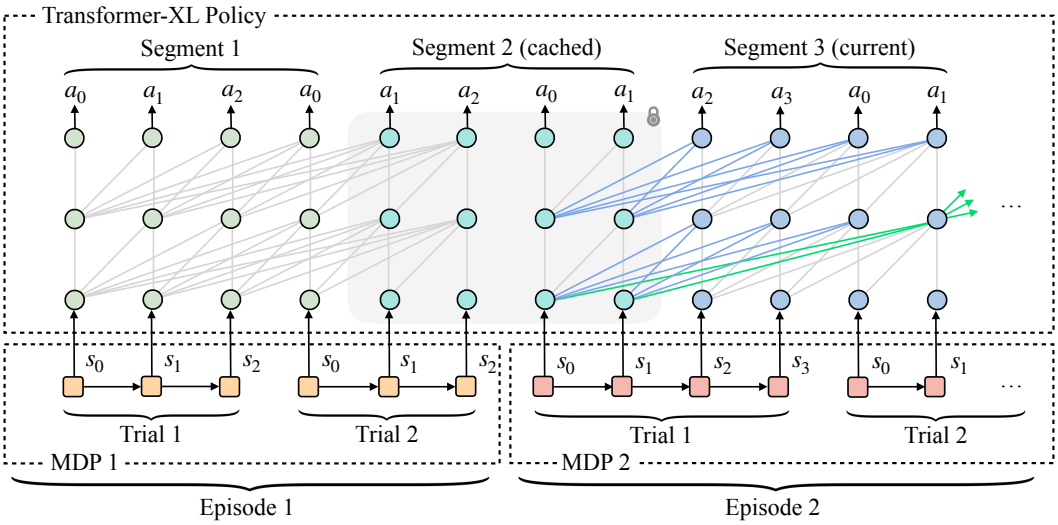


Figure 3: To enable long-context adaptation, we allow LocoFormer to attend to states from prior trials. We use a long-context Transformer backbone, that divides context into fixed-length segments. When processing the current segment, keys and values are also computed from states in the previous (cached) segment as well, but gradients are not propagated (blue lines). As the number of layers increases, this allows LocoFormer to use information even segments prior to the cached one (green lines).

2 Method

In this paper, our aim is to train a generalist locomotion policy that can robustly control any robot and adapt to with drastic changes. Given an unseen robot, the policy can be quickly deployed within seconds of adaptation. Our LocoFormer, achieves the goal with a pre-training procedure that allows efficient in-context learning during test time (Sec. 2.1), and a policy architecture that enables long-term memory for adaptation (Sec. 2.2). An overview of LocoFormer is illustrated in Fig. 3.

2.1 Pre-training to Learn in Context

Legged robots exhibit significant variations in their kinematics and dynamics, necessitating drastically different control strategies for them. One way to train a unified policy across diverse robots is to condition the policy on detailed kinematics and dynamics information [14]. However, this method typically demands extensive system identification and remains susceptible to inaccuracies caused by wear and tear over time. In contrast, LocoFormer takes a different approach by enabling in-context learning during test time.

However, the naive approach of taking existing locomotion policy architectures and applying them to our procedurally generated task space doesn't work well. This is because most locomotion controllers are myopic and only use a few hundred milliseconds of history to adapt. This works well for a narrow task space, but with a large task space, this short history doesn't contain enough information to adapt effectively and leads to failure. Therefore, we need to train with much longer context lengths that sometimes may span trial boundaries [15]. Specifically, we let the policy run multiple trials in an episode. Memory persists across trials, and the objective is to maximize the expected cumulative reward across the entire episode.

Formulation Let $M = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \rho_0, \gamma, T)$ represent a Markov Decision Process (MDP), with state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, initial state distribution ρ_0, discount factor $\gamma \in [0, 1)$, and horizon T. A *trial* is a single trajectory $\tau^M = [s_0, a_0, \dots]$ generated from initial state $s_0 \sim \rho_0(s_0)$. An *episode* consists of k trials, $\mathcal{E}^{M,k} = [\tau_1^M, \dots, \tau_k^M]$, where k is sampled from $\rho_K = \text{Uniform}(1, \dots, K)$. A history-dependent policy $\pi_\theta(a_t \mid s_{0:t}, a_{0:t-1})$ selects actions based on the entire previous state and action history within

an episode. Let H_i denote the cumulative trajectory length up to trial i, with the convention $H_0 = 0$. Consider a distribution $\rho_{\mathcal{M}}$ over a set of MDPs $\mathcal{M}$. The objective is to maximize the expected discounted return:

$$J(\pi_\theta) = \mathop{\mathbb{E}}_{\substack{M\sim\rho_{\mathcal{M}} \\ k\sim\rho_K \\ \mathcal{E}^{M,k}\sim\pi_\theta}} \left[\sum_{i=1}^{k} \sum_{t=0}^{T} \gamma^{t+H_{i-1}} \mathcal{R}^M(s_t, a_t) \right]. \tag{1}$$

Task Generation The above formulation assumes access to a distribution over MDPs. In our setting, we procedurally generate various legged robots and randomize their physical parameters to create diverse MDP instances. Specifically, we mainly focus on common morphologies such as bipeds, quadrupeds, and their wheeled counterparts without actually incorporating any exact robot parameters available on the market. For each morphology, we incorporate common design principles of existing legged robots. We randomize usual kinematic parameters, including joints, sequence of joints, etc. Dynamics parameters are randomized as well, including mass, center of mass, among other things. See Fig. 1 for a sample of generated robots and the appendix for more information.

Unified Observation, Action and Reward Space To train a policy shared across different legged robots, we use a simple *unified joint space* for consistent observation and action representation by concatenating a superset of joints that subsumes the number of motors in the majority of contemporary legged robots. The policy outputs target joint positions within the unified joint space, from which relevant joint states are extracted for each specific robot. For rewards, we adopt a design scheme inspired by [12], consisting of a command tracking reward and penalty terms to reduce torque usage, encourage smooth actions, and prevent hardware damage.

Large-Scale RL in Simulation Locomotion tasks typically span long trials. To emphasize the adaptation learning process, we modify our formulation slightly: instead of directly sampling a fixed number of trials, we sample an adaptation time budget of u seconds, $u \sim \text{Uniform}(0, U)$, allowing the policy to conduct an arbitrary number of trials within this allocated time before a final trial. Under the modified setting, we optimize the policy with Proximal Policy Optimization (PPO) [16] at scale in physical simulation. Training proceeds in two phases: initially, we use shorter trial durations and smaller values of U to prioritize the acquisition of adaptive behaviors. Subsequently, we extend trial lengths and adaptation time to prepare the policy for real-world deployment.

Deployment A pre-trained policy can perform in-context learning in both zero-shot and few-shot manner. In zero-shot setting ($u = 0$), the policy quickly adapts to an unseen embodiment within seconds, often achieving stable locomotion immediately. However, initial exploration may occasionally result in failures. In such cases, the policy benefits from few-shot learning, progressively improving performance by leveraging experiences from previous trials.

2.2 Policy Architecture with Long-Term Memory

A Transformer-based policy has strong representational capacity and can attend to arbitrary positions in its history, essential for effective large-scale cross-embodied learning. However, high-frequency control tasks quickly generate extensive contexts—even short experiences lasting around 5 seconds can translate into hundreds of tokens, posing significant computational challenges for a vanilla Transformer where cost scales quadratically with sequence length. For computational tractability, we leverage the Transformer-XL (TXL) architecture [17] as the backbone of our history-dependent policy, which incorporates a segment-level recurrence mechanism to enable modeling of long-range dependencies. We encode observations at timestep t through an MLP into an embedding x_t. These embeddings serve as inputs to the TXL policy, whose outputs are subsequently decoded into actions and value estimates by separate actor and critic heads (each an MLP). Below, we detail the Transformer-XL backbone (Fig. 3) and describe our approach to accelerating inference.

Method	Biped					Quadruped			Wheeled		Average
	G1	H1	GR1	TRON1	BK-H	A1	Spot	AnyMal C	TRON1-W	Go2-W	
Ours (zero-shot)	0.96±0.16	0.98±0.12	0.95±0.21	0.87±0.31	0.98±0.11	0.92±0.26	1.00±0.06	0.95±0.22	0.99±0.07	0.97±0.16	0.96±0.19
Ours (few-shot)	0.98±0.11	0.99±0.08	0.99±0.09	0.96±0.18	0.99±0.09	0.94±0.23	1.00±0.05	0.96±0.19	1.00±0.04	0.98±0.10	0.98±0.13
GRU	0.09±0.06	0.08±0.11	0.09±0.11	0.03±0.03	0.47±0.40	0.46±0.44	0.68±0.37	0.42±0.45	0.66±0.34	0.74±0.36	0.37±0.41
Conditioning	0.94±0.17	0.94±0.16	0.81±0.33	0.07±0.15	0.62±0.38	0.93±0.24	0.99±0.09	0.89±0.28	0.72±0.40	0.92±0.22	0.78±0.37
Expert Policy	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	1.00±0.00	0.97±0.18	0.99±0.08	1.00±0.04	1.00±0.00	0.98±0.12	0.99±0.07

Table 1: We evaluate each method across 10 previously unseen robots spanning three categories in 1,000 simulation environments per robot, with rough terrains and extensive domain randomization. Each entry reports the average displacement towards a randomly sampled goal normalized to $[0, 1]$. LocoFormer performs competitively in zero-shot settings and shows further improvement with a brief 5-second adaptation window (10% improvement on TRON1). On average, it approaches the performance of expert policy, outperforming both GRU and Conditioning baselines.

Transformer-XL Policy Suppose $\mathbf{o}_t$ are the observations at each timestep encoded into tokens $\mathbf{x}_t$. Transformer-XL (TXL) divides tokens into segments of length L, consider two such consecutive segments: $\sigma_z = [\mathbf{x}_{z,1}, \ldots, \mathbf{x}_{z,L}]$ and $\sigma_{z+1} = [\mathbf{x}_{z+1,1}, \ldots, \mathbf{x}_{z+1,L}]$. Let $\mathbf{h}_z^n \in \mathbb{R}^{L \times d}$ denote the hidden states of the n-th Transformer layer for segment σ_z. TXL uses cached hidden states from previous segments to compute current attention keys and values but doesn't backpropate gradients to them to save compute and memory. Formally, given previous hidden states $\mathbf{h}_z^{n-1}$, TXL computes:

$$\tilde{\mathbf{h}}_{z+1}^{n-1} = \left[\text{SG}(\mathbf{h}_z^{n-1}) \circ \mathbf{h}_{z+1}^{n-1}\right], \tag{2}$$

$$\mathbf{q}_{z+1}^n,\ \mathbf{k}_{z+1}^n,\ \mathbf{v}_{z+1}^n = \mathbf{h}_{z+1}^{n-1}\mathbf{W}_q,\ \tilde{\mathbf{h}}_{z+1}^{n-1}\mathbf{W}_k,\ \tilde{\mathbf{h}}_{z+1}^{n-1}\mathbf{W}_v, \tag{3}$$

$$\mathbf{h}_{z+1}^n = \text{Transformer-Layer}(\mathbf{q}_{z+1}^n,\ \mathbf{k}_{z+1}^n,\ \mathbf{v}_{z+1}^n) \tag{4}$$

where $\text{SG}(\cdot)$ denotes a stop-gradient operation, $\circ$ indicates concatenation along the sequence length dimension, and $\mathbf{W}_q$, $\mathbf{W}_k$, $\mathbf{W}_v$ represent learnable model parameters. We use the standard Transformer architecture [18] for TXL.

In RL training, each segment is cached to condition the subsequent iteration. This segment-level recurrence results in an effective context length growing as $O(NL)$ where N is number of layers and L is the segment length. For instance, a Transformer with 6 layers and segment length of 128 achieves a largest possible memory of 896 timesteps, corresponding to approximately 18 seconds at a control frequency of 50 Hz. Note that segments may span episode boundaries, in which case the attention mask is modified such that attention doesn't span episode boundaries.

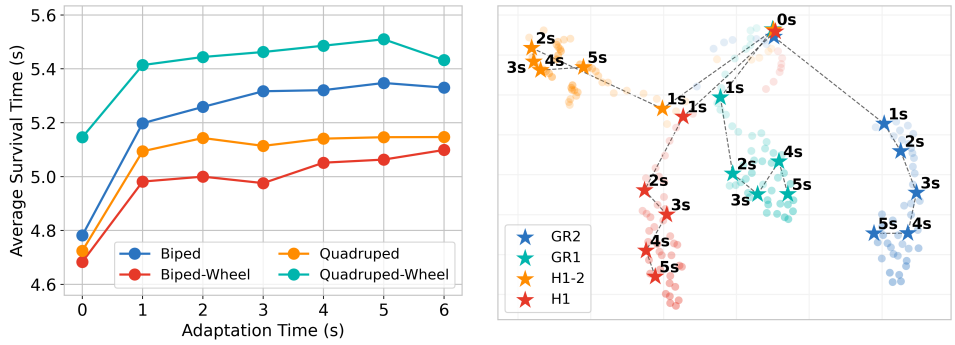
Inference Acceleration The inference complexity of Transformer scales quadratically with segment length, which can significant slow down inference speed. To alleviate the computational bottleneck, we leverage KV-cache [19] to reduce inference complexity from quadratic to linear dependence on timestep t. During training-time rollouts, we initialize the KV-cache using keys and values derived from the previous segment's hidden states, and then append new timesteps incrementally. For real-time deployment, the KV-cache maintains only the most recent $2 \times L - 1$ timesteps, which corresponds to the maximum directly conditioned history length encountered during training.

3 Experimental Results

3.1 Simulation Evaluation

We compare the performance of LocoFormer with several baselines in simulation. We additionally assess how the policy improves with growing adaptation time and examine the dynamics of its internal representations during adaptation.

Baselines We consider the following baselines (1) *GRU Policy* A GRU-based policy trained under the same setting as LocoFormer to evaluate the effect of architectural choice; (2) *Conditioning* A Transformer policy conditioned on privileged kinematics information but with restricted memory for adaptation (context length 64); (3) *Expert Policy* A Transformer policy trained directly on the target embodiment, serving as an upper-bound with full access to the deployment domain. LocoFormer is evaluated in both zero-shot and few-shot settings. Under the few-shot setting, the policy is allowed a 5-second adaptation window prior to evaluation.



(a) (b)

Figure 4: **(a)** Performance of LocoFormer with different adaptation time. We deploy the policy on procedurally generated robots across four morphology types under intensified domain randomization, and report average survival over 6-second rollouts across varying adaptation time budgets. Our policy consistently benefits from longer adaptation. **(b)** Temporal evolution of policy representations. We visualize the mean output of the second Transformer layer over time across 4,096 zero-shot rollouts of four humanoid variants. Representations begin similarly but diverge progressively, forming distinct clusters by 5 seconds. This indicates that LocoFormer dynamically builds stable, embodiment-specific representations online, without explicit morphology specification.

Evaluation on Unseen Robots We test all methods on a set of previously unseen robots: five bipeds (Unitree G1, H1, Fourier GR1, LimX Dynamics TRON1, Berkeley Humanoid BK-H), three quadrupeds (Unitree A1, Boston Dynamics Spot, ETH AnyMal C), and two wheeled robots (TRON1-W, Untiree Go2-W). Evaluation is conducted across 1,000 randomized simulation environments with rough terrains and intensified domain randomization. We randomly sample a goal in each environment an measure the average displacement towards the goal. Results are presented in Tab. 1. Despite having no prior access to these robots, our method achieves performance close to the expert policies in both zero-shot and few-shot settings. While it adapts effectively in a zero-shot manner, extended adaptation proves beneficial in challenging scenarios. For example, on TRON1—a biped without ankle joints—5 seconds of pre-evaluation adaptation yields a 10% performance gain. By contrast, the GRU baseline performs poorly with an average score 0.37, highlighting the importance of architecture choice for cross-embodiment generalization. The Conditioning baseline benefits from morphology inputs but is constrained by its short context window, resulting in moderate performance.

Adaptation Performance To assess adaptation capability of LocoFormer, we double domain randomization intensity and deploy procedurally generated robots on rough terrain. We record the average time to fall over 6-second rollouts across varying adaptation time budgets (50K episodes per setting). As shown in Fig. 6c, LocoFormer can adapt to challenging domains effectively even zero-shot, with substantial gains from a short adaptation period. All morphologies exhibit improved survival with increased adaptation time, highlighting the advantage of long-context learning.

Representation Dynamics We further analyze adaptation by tracking the temporal evolution of internal representations. Specifically, we monitor the average output of the second Transformer layer over 4,096 zero-shot rollouts of four humanoid variants (Fig. 6d). Initially similar embeddings (within the first second) diverge over time, reflecting the policy's growing specialization on different embodiments. By 5 seconds, distinct representation clusters emerge, indicating the policy's ability to form stable, embodiment-specific internal models without explicit robot descriptors.

3.2 Real-World Evaluation

We first evaluate how LocoFormer transfers to unseen robots in real world. Our test set includes four Unitree robots: G1, H1, Go2, and Go2-W, as illustrated in Fig. 1. Notably, we also evaluate Go2 and Go2-W in a bipedal mode by activating only their rear legs. Our policy is deployed zero-shot on these robots and adapts within seconds, without any prior tuning or retraining.

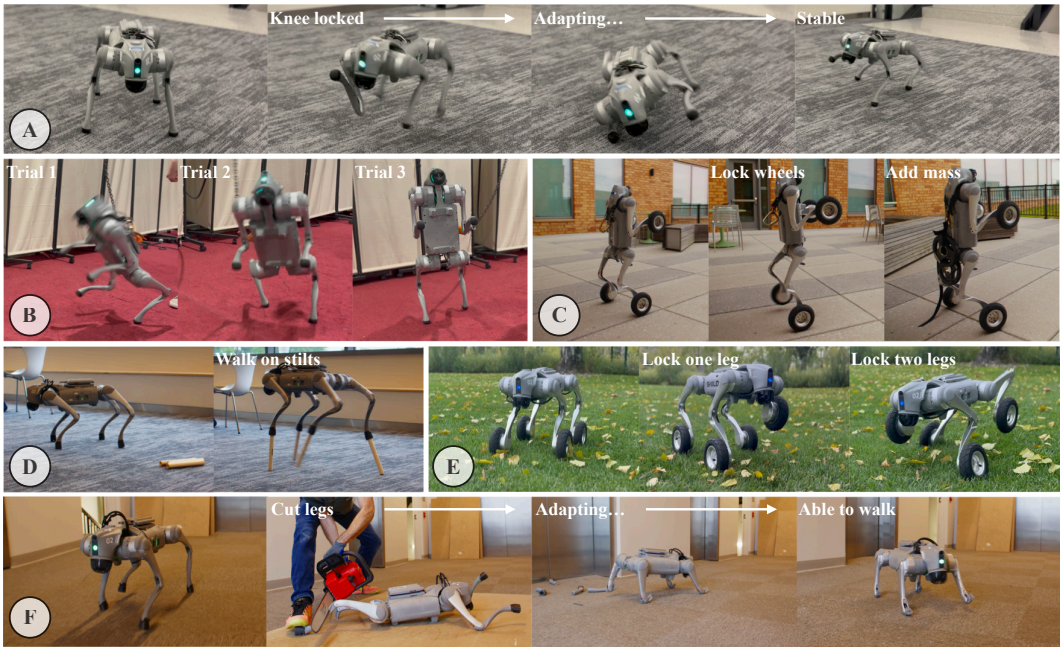


Figure 5: We showcase emergent adaptation behaviors, all from the same LocoFormer model. **A**: A knee is locked, and LocoFormer adapts to walk with three remaining legs. This is not a part of training. **B**: If LocoFormer is unable to adapt within a trial, it retains memory of previous trials and improves performance. **C**: The wheels are locked and LocoFormer detects the sudden change, and shifting to a walking gait. LocoFormer also adapts when additional mass is attached. **D**: Stilts are attached, and LocoFormer adapts to the extended leg length to continue walking. **E**: With one or two legs locked, LocoFormer adapts its motion to maintain balance. **F**: After lower legs are cut, the robot adapts after tens of seconds to walk on its knees despite not seeing it at training time.

LocoFormer is trained at scale across diverse morphologies under heavy online perturbations , and it incorporates memory across trials. We hypothesize that this enables the policy to handle large real-world disturbances such as sudden morphology changes and added payloads, and to leverage prior experience for efficient control. We showcase these abilities below (see Fig. 5).

A, E: Leg locking While Go2 is walking normally, a knee is locked by overriding the PD setpoint to a hardcoded value. This converts the quadruped into a three legged robot, destabilizing it. To recover, the policy must adopt a gait that keeps the center of mass within the triangular support polygon—an unseen scenario since three-legged robots are outside our design space. LocoFormer initially tips forwards but learns to shift its weight backward onto three legs and is even able to walk after 2-3s of adaptation. Similar behavior occurs on a wheeled quadruped: when one or two legs are locked, LocoFormer adapts its gait to redistribute load, preserving balance and mobility despite reduced actuation.

B: Adaptation across trials In highly unstable cases, adaptation within a single attempt may be impossible. We run LocoFormer as a biped with no ankle motor and a single point of support. This is a highly unstable platform and affords very little time to stabilize before failure occurs. Since LocoFormer has no idea about the morphology it fails in the first trial. However, we can retain the failed trajectory information in the TransformerXL cache and attend to it leading to improved performance in subsequent trials. On the third trial, the robot walks stably in biped mode and is even robust to pushes and weights.

C: Wheel locking and payload change While Go2-W is initially rolling, we programmatically lock the wheels during execution by setting the gains to zero in software. This instantly alters the robot's dynamics, as it can no longer use rolling to locomote. LocoFormer detects this discrepancy, as sending commands to the wheel no longer has the same effect. It quickly adapts to this change and switches to a walking gait instead like in a standard legged biped. At some point the wheels are again unlocked. LocoFormer detects this and switches to a lower energy rolling gait. Similarly, when external mass is added, LocoFormer adjusts its gait to maintain balance.

D: Walking on stilts We attach stilts to the robot's legs, increasing the effective leg-to-body length ratio far beyond those seen during training. This alters both kinematics and dynamics of locomotion by

raising the center of mass and changing swing/stance leverage. Initially, the robot takes a few unstable steps, but LocoFormer quickly adapts its coordination—adjusting step timing and foot placement—to account for the extended morphology and is able to walk forward with stable, repeatable gaits.

F: Cutting lower legs We cut the calf of the robot to its thigh. This essentially removes 4 degrees of freedom and lowers the limb length. This configuration is unseen at training. Initially, the robot is unable to locomote effectively and merely steps in place. However, after 7-8s of adaptation, it discovers that large amplitude swings are required at the thigh joint and is able to locomote effectively thereafter.

4 Related Work

4.1 Legged Locomotion

In recent years, sim2real transfer has emerged as an effective paradigm for training locomotion. For blind robots, this is done by training an adaptive policy in simulation with domain randomization and then transferring zero-shot to the real world, both for quadrupeds [5, 6, 7, 8, 9, 10] and humanoids [11, 20, 21, 22, 23, 24]. Subsequent works couple locomotion with vision to adapt walking behavior based on upcoming terrain geometry using either implicit [12, 25, 26, 27, 28, 29] or explicit mapping [30, 31, 32]. In all these works, the controller is tuned for a specific robot with narrow dynamics randomization and a short context length (hundred millisecond) for adaptation. In contrast, we train a single, generalist policy that adapts over long contexts to vastly different embodiments and dynamics.

4.2 Cross-embodiment Learning

The success of large-scale pretraining in language and vision has spurred interest in developing similar models for robots that can absorb data from multiple embodiments. For locomotion, many methods focus on adding structural biases to the policy architecture based on the robot morphology in the form of message passing on graph neural networks [33] or through attention masks in a transformer [34]. Gupta et al. [35] co-evolve locomotion controllers with robot morphologies. Other methods focus on parameterizing the action and observation spaces of the policy to promote knowledge transfer. Doshi et al. [36] train a transformer with imitation learning that uses different action heads to decode readout tokens from a transformer to different morphologies. Bohlinger et al. [14] trains with RL on a few chosen robots with a custom latent observation space that encodes joint observations as well as precise kinematic structure of the robot. In contrast, Reed et al. [37] simply trains with next token prediction on data from multiple embodiments. Closely related to our work, Feng et al. [38] train a single controller across procedurally generated quadrupeds. The key difference from these works is that we train at massive scale with long contexts across a much larger task space consisting of wheeled and legged quadrupeds and humanoids. We show that this leads to novel and highly flexible adaptation strategies.

4.3 Adaptation

Adaptation, meta-learning [39] and in-context learning [13] are related concepts which deal with the problem of achieving high task performance from a few examples. Meta-RL or fine-tuning can be used for rapidly adapting specialist locomotion policies to changes in dynamics [40, 41, 42]. In game-like environments, [15] show that training with contexts spanning episodes leads to adaptive behavior where policies learn to improve from early mistakes. Team et al. [1] shows that large-scale training in an open-ended task space leads to agents that learn to adapt on human-timescales by reasoning across long contexts. In our paper, we study adaptation in the high-frequency closed loop control setting, and present a single policy that adapts to varied embodiments and dynamics.

5 Conclusion

In this paper, we present LocoFormer, a generalist locomotion policy that can robustly control a variety of legged and wheeled robots in the real world. This is enabled by emergent adaptive behavior where the policy can use long contexts as information to improve performance. Although we only discuss locomotion in this paper, we believe this recipe is general and can be used in the future to train generalist controllers for other skills, especially in simulation where such large-scale RL is possible.

6 Limitations

We identify several key limitations of our approach:

- Training LocoFormer is highly resource-intensive compared to a specialist policy. While this is not surprising given the much wider task distribution and context, there may be algorithmic improvements possible in the massively parallel RL setting [43].
- LocoFormer relies on a procedurally generated task space which is hand-crafted. This might be hard to design in general. For future work, automated task generation from LLMs or other web-scale sources might be a more scalable approach.

Acknowledgment

We thank Kevin Gmelin, Jayesh Singla, Jared Meija, Shikhar Bahl for fruitful discussions.

References

[1] A. A. Team, J. Bauer, K. Baumli, S. Baveja, F. Behbahani, A. Bhoopchand, N. Bradley-Schmieg, M. Chang, N. Clay, A. Collister, et al. Human-timescale adaptation in an open-ended task space. *arXiv preprint arXiv:2301.07608*, 2023.

[2] G. D. Muir. Early ontogeny of locomotor behaviour: a comparison between altricial and precocial animals. *Brain research bulletin*, 53(5):719–726, 2000.

[3] B. Goldner, A. Fuchs, I. Nolte, and N. Schilling. Kinematic adaptations to tripedal locomotion in dogs. *The Veterinary Journal*, 204(2):192–200, 2015. ISSN 1090-0233. doi:https://doi.org/10.1016/j.tvjl.2015.03.003. URL https://www.sciencedirect.com/science/article/pii/S1090023315001008.

[4] E. Thelen and D. M. Fisher. Newborn stepping: An explanation for a" disappearing" reflex. *Developmental psychology*, 18(5):760, 1982.

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

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

[7] I. M. A. Nahrendra, B. Yu, and H. Myung. Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5078–5084. IEEE, 2023.

[8] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.

[9] Y. Kim, H. Oh, J. Lee, J. Choi, G. Ji, M. Jung, D. Youm, and J. Hwangbo. Not only rewards but also constraints: Applications on legged robot locomotion. *IEEE Transactions on Robotics*, 40: 2984–3003, 2024. doi:10.1109/TRO.2024.3400935.

[10] L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine. Learning and adapting agile locomotion skills by transferring experience. *arXiv preprint arXiv:2304.09834*, 2023.

[11] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

[12] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on robot learning*, pages 403–415. PMLR, 2023.

[13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[14] N. Bohlinger, G. Czechmanowski, M. Krupka, P. Kicki, K. Walas, J. Peters, and D. Tateo. One policy to run them all: an end-to-end learning approach to multi-embodiment locomotion. *arXiv preprint arXiv:2409.06366*, 2024.

[15] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

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

[17] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[19] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5:606–624, 2023.

[20] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang. Expressive whole-body control for humanoid robots. *arXiv preprint arXiv:2402.16796*, 2024.

[21] T. He, W. Xiao, T. Lin, Z. Luo, Z. Xu, Z. Jiang, J. Kautz, C. Liu, G. Shi, X. Wang, et al. Hover: Versatile neural whole-body controller for humanoid robots. *arXiv preprint arXiv:2410.21229*, 2024.

[22] Z. Chen, X. He, Y.-J. Wang, Q. Liao, Y. Ze, Z. Li, S. S. Sastry, J. Wu, K. Sreenath, S. Gupta, et al. Learning smooth humanoid locomotion through lipschitz-constrained policies. *arXiv preprint arXiv:2410.11825*, 2024.

[23] Y. Xue, W. Dong, M. Liu, W. Zhang, and J. Pang. A unified and general humanoid whole-body controller for fine-grained locomotion. *arXiv preprint arXiv:2502.03206*, 2025.

[24] J. Siekmann, Y. Godse, A. Fern, and J. Hurst. Sim-to-real learning of all common bipedal gaits via periodic reward composition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7309–7315. IEEE, 2021.

[25] H. Duan, B. Pandit, M. S. Gadde, B. Van Marum, J. Dao, C. Kim, and A. Fern. Learning vision-based bipedal locomotion for challenging terrain. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 56–62. IEEE, 2024.

[26] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450. IEEE, 2024.

[27] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023.

[28] S. Luo, S. Li, R. Yu, Z. Wang, J. Wu, and Q. Zhu. Pie: Parkour with implicit-explicit learning framework for legged robots. *IEEE Robotics and Automation Letters*, 2024.

[29] R. Yang, G. Yang, and X. Wang. Neural volumetric memory for visual locomotion control. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1430–1440, 2023.

[30] W. Sun, B. Cao, L. Chen, Y. Su, Y. Liu, Z. Xie, and H. Liu. Learning perceptive humanoid locomotion over challenging terrain. *arXiv preprint arXiv:2503.00692*, 2025.

[31] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science robotics*, 7(62):eabk2822, 2022.

[32] H. Wang, Z. Wang, J. Ren, Q. Ben, T. Huang, W. Zhang, and J. Pang. Beamdojo: Learning agile humanoid locomotion on sparse footholds. *arXiv preprint arXiv:2502.10363*, 2025.

[33] W. Huang, I. Mordatch, and D. 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.

[34] C. Sferrazza, D.-M. Huang, F. Liu, J. Lee, and P. Abbeel. Body transformer: Leveraging robot embodiment for policy learning. *arXiv preprint arXiv:2408.06316*, 2024.

[35] A. Gupta, S. Savarese, S. Ganguli, and L. Fei-Fei. Embodied intelligence via learning and evolution. *Nature communications*, 12(1):5721, 2021.

[36] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. *arXiv preprint arXiv:2408.11812*, 2024.

[37] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[38] G. Feng, H. Zhang, Z. Li, X. B. Peng, B. Basireddy, L. Yue, Z. Song, L. Yang, Y. Liu, K. Sreenath, et al. Genloco: Generalized locomotion controllers for quadrupedal robots. In *Conference on Robot Learning*, pages 1893–1903. PMLR, 2023.

[39] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[40] K. Bao, C. Li, Y. As, A. Krause, and M. Hutter. Toward task generalization via memory augmentation in meta-reinforcement learning. *arXiv preprint arXiv:2502.01521*, 2025.

[41] X. Song, Y. Yang, K. Choromanski, K. Caluwaerts, W. Gao, C. Finn, and J. Tan. Rapidly adaptable legged robots via evolutionary meta-learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3776. IEEE, 2020.

[42] L. Smith, I. Kostrikov, and S. Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

[43] J. Singla, A. Agarwal, and D. Pathak. Sapg: split and aggregate policy gradients. *arXiv preprint arXiv:2407.20230*, 2024.

Appendix

A Implementation Details

This section outlines the procedures for generating diverse robot morphologies, randomizing environment dynamics, designing rewards, and training the LocoFormer policy.

A.1 Procedural Robot Generation

We generate four categories of morphologies: quadrupeds, bipeds, and their respective wheeled counterparts. Random samples are illustrated in Fig. 6. While our procedural generation covers a wide morphological space, it is not exhaustive. Our aim is to train a policy capable of generalizing to out-of-distribution robots. In practice, unseen robots often deviate from our design assumptions. For instance, Berkeley-Humanoid exhibits anhedral angles in both hip roll and yaw joints—our bipeds include at most one. Additionally, robots like G1 have joint axes offset along the x-axis, a feature not captured in our parameterization. As for dynamics randomization, we randomized the standard parameters [12] but over a much wider range so that the model learns to adapt.

A.2 Reward Design

We design a unified reward structure combining morphology-agnostic and morphology-specific terms. For clarity, the time subscript t is omitted in some expressions.

Shared Reward Components

- **Linear velocity command tracking** $\exp(\|v_{xy} - v_{xy}^{cmd}\|^2/s_1)$
- **Angular velocity command tracking** $\exp(\|w_z - w_z^{cmd}\|^2/s_2)$
- **Base linear velocity penalty** $\|v_z\|^2$
- **Base angular velocity penalty** $\|w_{xy}\|^2$
- **Base height penalty** $\|x_z - x_z^{nominal}\|^2$
- **Joint acceleration penalty** $\|\ddot{q}\|^2$
- **Torque penalty** $-\|\tau\|^2$
- **Alive reward**: 1

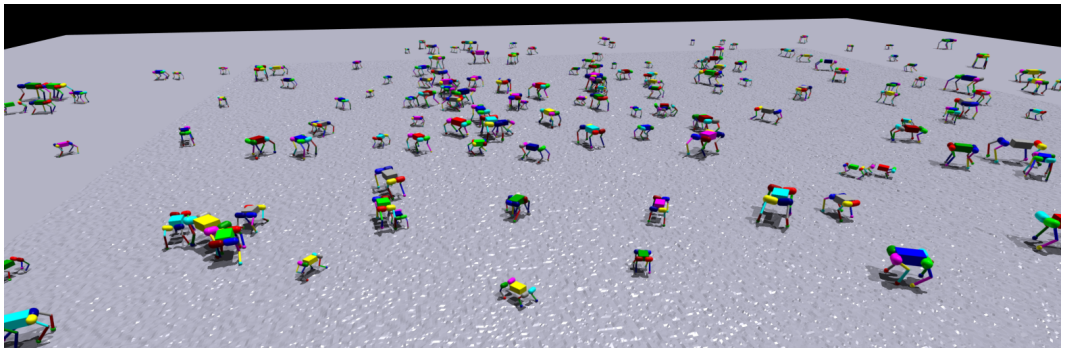
B Experiment Details

Fig. 7 shows the 10 simulated robots used for comparison against baseline methods. Among them, we observe a notable sign of cross-trial adaptation in the TRON1 robot, which achieves a 10% performance improvement through 5-seconds pre-evaluation adaptation. We find the robot initially fails in early trials but gradually learns to stabilize and control itself effectively. An illustrative example of this adaptation process is shown in Fig. 8.

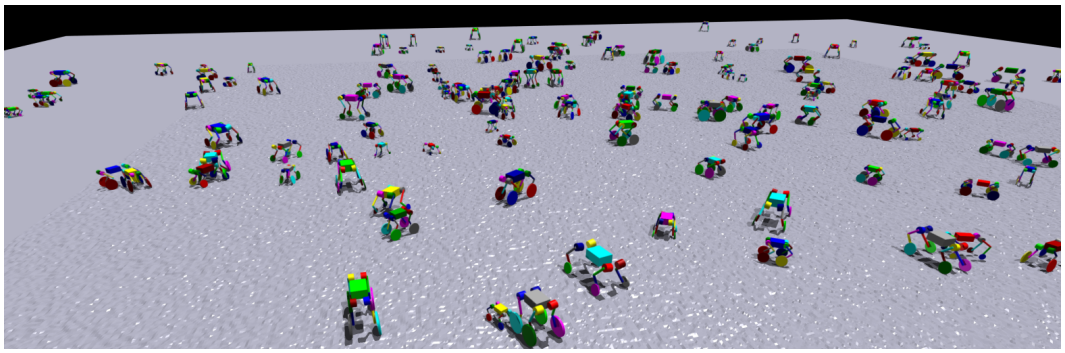
We provide videos of real world evaluation on generalist-locomotion.github.io.

C Computation Cost

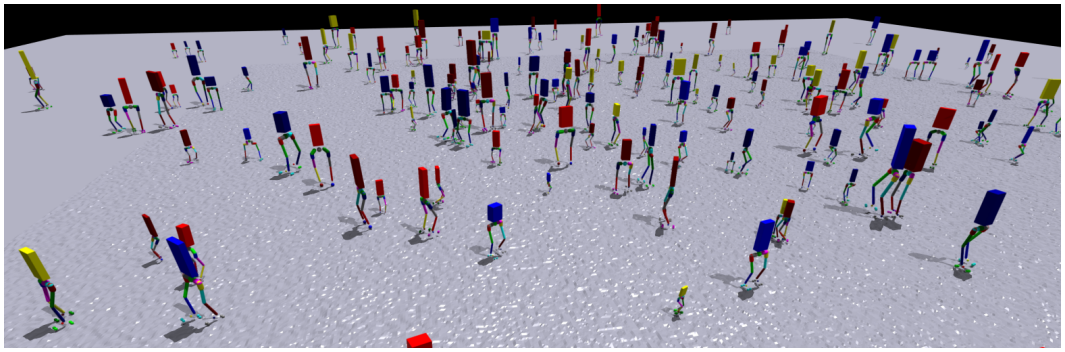
LocoFormer is a highly performant generalist locomotion model trained with a high capacity Transformer model across 100k robots. Compared to a specialist policy, LocoFormer takes 500 times more compute for 100000 more robots – this is a significant reduction in amortized compute cost per robot (1 vs. 0.005 day).



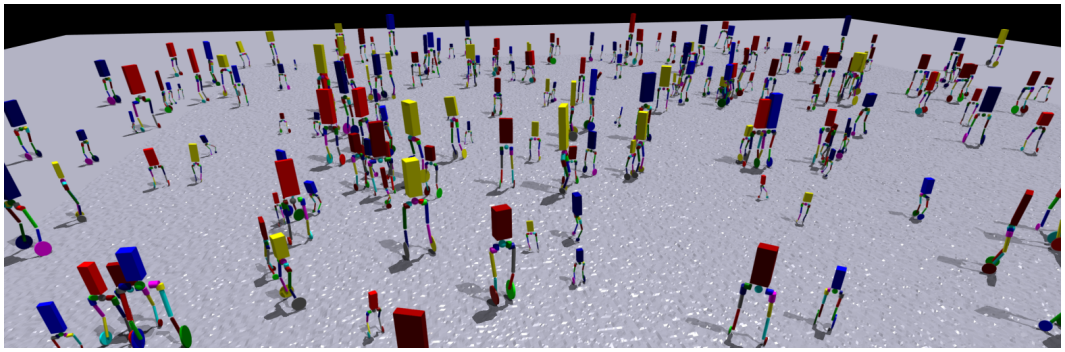
(a)



(b)



(c)



(d)

Figure 6: Visualization of sample procedurally generated robots. **(a)** Quadrupeds. **(b)** Quadrupeds with wheels. **(c)** Bipeds. **(d)** Bipeds with wheels.

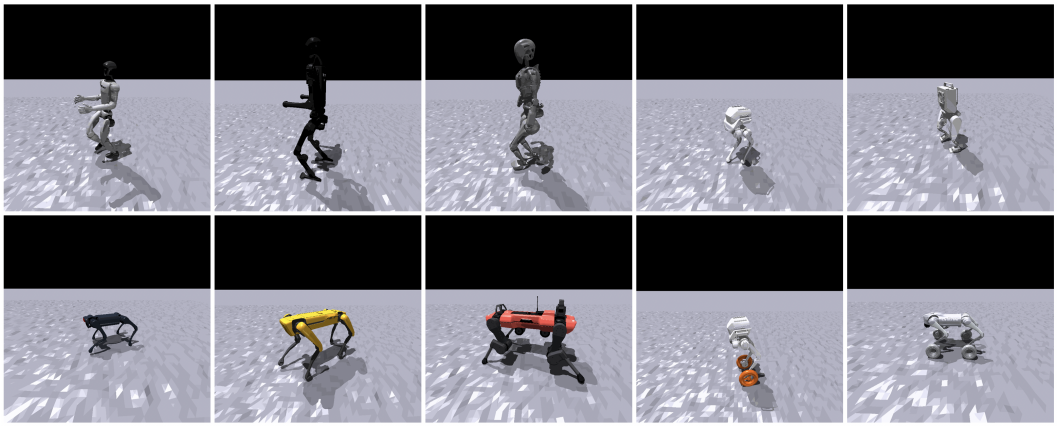


Figure 7: Robots where we compare with the baseline methods in simulation. First row: Unitree G1, H1, Fourier GR1, LimX Dynamics TRON1, Berkeley Humanoid. Second row: Unitree A1, Boston Dynamics Spot, ETH AnyMal C, LimX Dynamics TRON1-WF, Unitree Go2-W.

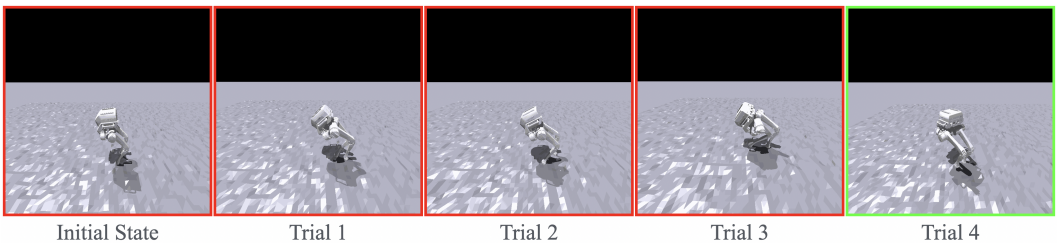


Figure 8: Cross-trial adaptation of TRON1 in simulation. The robot fails in early trials but progressively learns to stabilize and locomote effectively by Trial 4.

Optimizing compute further: We trained LocoFormer at scale to get the best possible model. However, there is a favorable trade-off between performance and compute – even a variant trained with much lesser compute shows clear signs of adaptation and cross-embodiment generalization (Fig. 9). Since we use a Transformer backbone, techniques from LLM literature can be directly used to reduce compute requirement further, such as, mixed precision, ZeRO memory optimization.

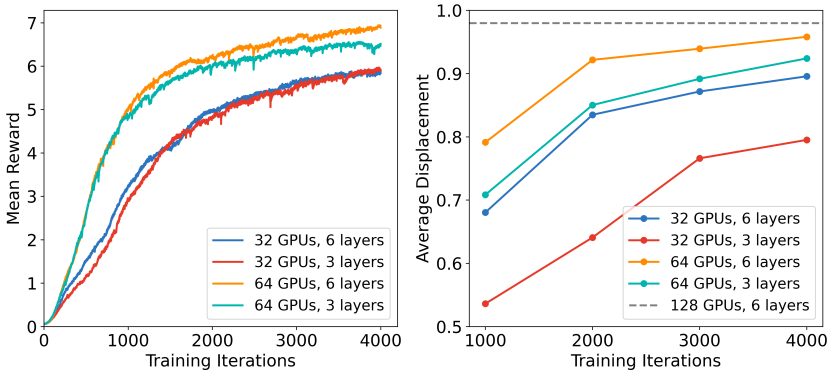


Figure 9: We train with different numbers of GPUs and TXL layers for 4K iterations (28 hours for 6-layer TXL), and evaluate on 10 robots. The left plot shows reward on robots in the train set, while the right reports avg. displacement of selected checkpoints on 10 eval robots (including OOD cases as described in Tab. 1). While more GPUs yields the best model, there is a favorable tradeoff between performance and compute.