DiffuseLoco: Real-Time Legged Locomotion Control with Diffusion from Offline Datasets

Xiaoyu Huang*,[1], Yufeng Chi*,[1], Ruofeng Wang*,[1], Zhongyu Li[1], Xue Bin Peng[2],
Sophia Shao[1], Borivoje Nikolic[1], Koushil Sreenath[1]
[1] UC Berkeley [2] Simon Fraser University
*Equal contribution. Email: x.h@berkeley.edu

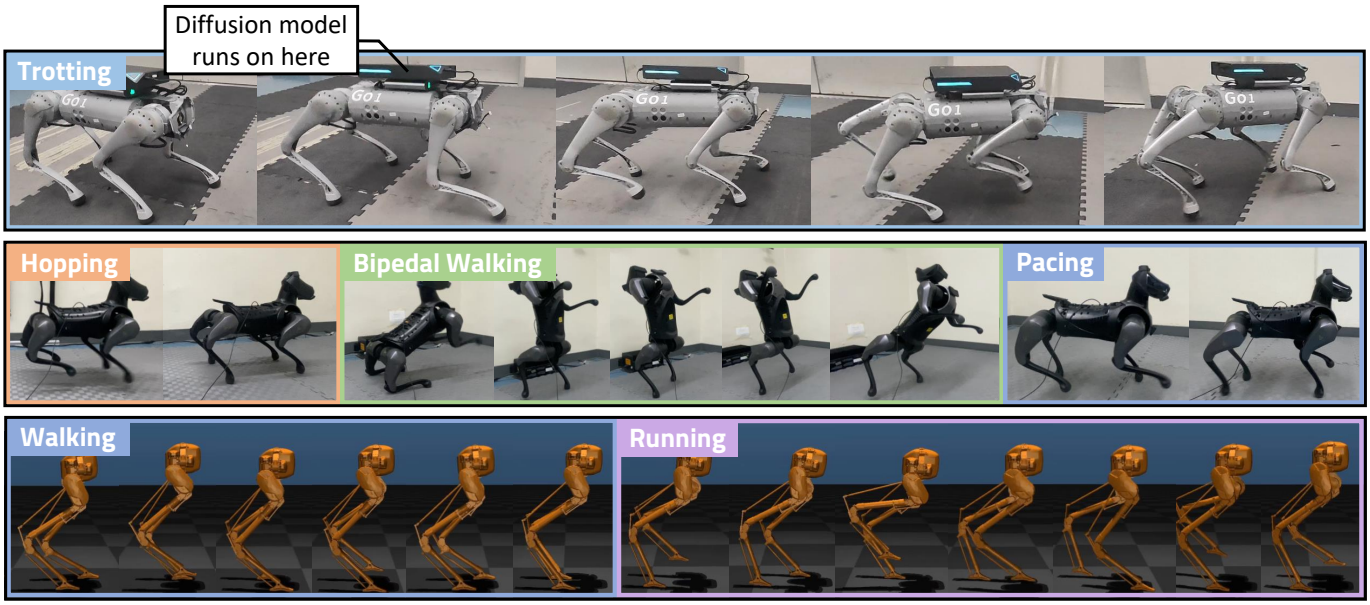


Fig. 1. Snapshots of (top) a quadrupedal robot trotting with our diffusion-based real-time locomotion control policy via onboard computing; (middle) our multi-skill offline-learned policy performing a sequence of challenging skills, including hopping, bipedal locomotion, and pacing with smooth skill transitioning; (bottom) a bipedal robot controlled by our multi-skill policy transitioning from walking to running stably. We present DiffuseLoco, a scalable framework that leverages diffusion models to learn legged locomotion control exclusively from offline datasets. DiffuseLoco learns a state-of-the-art policy that is able to perform a diverse set of agile locomotion skills with a single policy, exhibiting robustness in the real world, and is versatile to various sources of offline data. We encourage the viewers to watch supplementary videos on these runs.

***Abstract*—This work introduces DiffuseLoco, a framework for training multi-skill diffusion-based policies for dynamic legged locomotion from offline datasets, enabling real-time control of diverse skills on robots in the real world. Offline learning at scale has led to breakthroughs in computer vision, natural language processing, and robotic manipulation domains. However, scaling up learning for legged robot locomotion, especially with multiple skills in a single policy, presents significant challenges for prior online reinforcement learning methods. To address this challenge, we propose a novel, scalable framework that leverages diffusion models to directly learn from offline multimodal datasets with a diverse set of locomotion skills. With design choices tailored for real-time control in dynamical systems, including receding horizon control and delayed inputs, DiffuseLoco is capable of reproducing multimodality in performing various locomotion skills, zero-shot transfer to real quadrupedal robots, and it can be deployed on edge computing devices. Furthermore, DiffuseLoco demonstrates free transitions between skills and robustness against environmental variations. Through extensive benchmarking in real-world experiments, DiffuseLoco exhibits better stability and velocity tracking performance compared to prior reinforcement learning and non-diffusion-based behavior cloning baselines. The design choices are validated via comprehensive ablation studies. This work opens new possibilities for scaling up learning-based legged locomotion controllers through the scaling of large, expressive models and diverse offline datasets.**

I. Introduction

Learning from large-scale offline datasets has led to breakthroughs in a large variety of domains, such as computer vision and natural language processing, where scaling up both the size of models and datasets leads to improved performance and generalization [32, 74]. This has led to the development of powerful generative models, like diffusion models, which are able to model complex multi-modal data distributions [64, 67], and generate high-quality images and videos.

In robotics, learning from diverse offline datasets has also been shown to be an effective and scalable approach for developing more versatile policies for domains such as robotic manipulation [3, 4] and autonomous driving [8, 13, 54]. However, these domains typically involve agents that have

low-dimensional action spaces (*e.g.*, end-effector trajectory), with low re-planning frequency on inherently stable systems (*e.g.*, robot arms or cars). For dynamical systems featuring higher degrees of freedom and more complex dynamics, such as legged robots, data-driven approaches have largely been focused on online reinforcement learning (RL) techniques [27, 43, 56]. Unlike offline learning, it can be difficult to scale online RL to both large models and datasets due to the requirements of online rollouts. Most prior works have focused on dynamic motions with specialized single-skill models, and scaling towards a single model that can reproduce a diverse set of challenging locomotion skills remains an open problem.

To this end, we present a novel approach that emphasizes learning agile legged locomotion skills at scale by solving two aforementioned challenges: offline learning from various data sources and the ability to learn a set of diverse skills. We propose *DiffuseLoco*, a framework that leverages expressive diffusion models to effectively learn the multi-modalities that exist in the diverse offline dataset without manual skill labeling. Once trained, our controllers can execute robust locomotion skills on real-world legged robots for real-time control.

The primary contributions of this work include:

1) A state-of-the-art multi-skill controller, leveraging expressive diffusion models, that learns agile bipedal walking and various quadrupedal locomotion skills within a single policy and can be deployed zero-shot on real-world quadrupedal robots.
2) A novel framework that directly learns from a diverse offline dataset for real-time control of legged robots, showing the benefits and potentials of offline learning at scale for locomotion skills practically in a real-world scenario.
3) Extensive real-world validation showing higher stability and lower velocity tracking errors compared to baselines, while demonstrating multi-modal behaviors with skill transitioning and robustness on terrains with varying ground frictions.

This work opens up the possibility of leveraging large-scale learning to create diverse and agile multi-skill controllers for legged locomotion from offline datasets. For the first time, we show that it is feasible to zero-shot transfer such a diverse locomotion policy learned from a static dataset to real-world applications. This approach offers a scalable and versatile framework for learning-based control, allowing for continuous expansion of the dataset and integration of diverse skills from various data sources. Codebase and checkpoints will be open-sourced upon the acceptance of this work.

II. Related Work

Our proposed framework leverages diffusion models for multi-skill legged locomotion control. In this section, we review the most closely related works on learning-based legged locomotion algorithms and applications of diffusion models in robotics.

A. Multi-skill Reinforcement Learning in Locomotion

Recent advances in model-free RL have demonstrated promising results in developing agile locomotion skills for legged robots in the real world [6, 14, 40, 47, 56, 66]. Among them, prior works have demonstrated impressive performances on highly agile skills such as jumping, running and sharp turning on bipedal robots [44, 91], and walking on two feet with quadrupedal robots [20, 42, 71], which requires a high degree of agility and robustness with a floating-base robot. However, the majority of these agile locomotion skills are trained with single-skill RL and remain unscalable to large-scale learning of multiple locomotion skills.

A simple and natural idea of learning multi-skill locomotion is to train separate policies for each skill, and then coordinate through extra high-level planning [23, 26, 33, 88] or distill into a small-scale model via imitation learning such as DAgger [93]. Due to the inherent coordination difficulty and the requirements of online distillation, these methods remain unscalable to an increasing number of skills.

In comparison, learning multi-skill policies directly from scratch typically involves parameterized motions [62, 70] with limited sets of applicable motions, and more popularly, motion imitation methods through either reward shaping [11, 34, 92] or adversarial imitation learning [16, 39, 83, 89]. However, this approach still faces challenges such as needing extra model-based trajectory optimization or well-trained expert policies for acquiring reference for agile motions and the limited expressiveness of simple models in online RL frameworks in learning diverse skills.

In general, learning a diverse, agile multi-skill policy with online RL remains challenging. For example, while existing RL methods have successfully combined skills like jumping and trotting [89, 92], quadrupedal walking and standing on hind legs with wheels without bipedal walking [78], a combination of more diverse and agile skills such as stable bipedal walking and quadrupedal hopping in a single policy has not yet been demonstrated.

B. Offline Learning in Locomotion Control

Compared to online learning, offline learning offers better scalability, a simpler training scheme, and an effective way to re-use data, yet prior works in learning low-level locomotion control from offline datasets remain limited. Most prior works focus on simple simulated tasks, such as Gym locomotion tasks, with behavior cloning (BC) [76] and offline RL [38]. Among them, some leverage Q-learning on offline datasets [35, 36, 51], or supervised learning techniques [9, 28, 81, 86]. However, these tasks are oversimplified and do not adequately consider the complexities in real-world scenarios.

An alternative is the use of offline data as a foundation for online learning [50, 77]. Among them, Smith et al. develops baseline policies from offline datasets to bootstrap online learning on real robots. Yet, this approach still requires online learning.

Another recent work develops offline learning on humanoid locomotion with real robots [61], with the scope limited

to only one walking skill. In comparison, the efficacy of learning completely from offline datasets, especially at a larger scale than a few simple skills remains unproven in legged locomotion control.

C. Diffusion Models in Robotics

Recent advances have seen increasing applications of diffusion models in control and planning systems. Some prior works integrate them into the learning pipeline, including the discriminators in adversarial IL for legged control [79], and reward models for RL [52, 80]. However, the use of small multi-layer perceptron (MLP) networks as policies presents similar challenges in exploring diverse, multi-skill learning [81]. Diffusion models are also employed in high-level trajectory planning [24, 29], safe planning [85], and goal generation for low-level controllers [1, 31], as well as enhancing visuomotor planning for manipulation tasks [48, 55, 63]. However, most prior works are limited to simulation environments only.

Emerging efforts to apply diffusion models to real-world robot manipulation include using diffusion to manage a variety of manipulation tasks with visual inputs and incorporating self-supervised learning and language conditioning [10, 12, 21, 41]. Yoneda et al. leverages the reverse diffusion process for shared autonomy with a human user in end-effector planning. Additionally, hierarchical frameworks are being developed to handle tasks requiring multiple skills, pushing towards generalist policies [2, 53, 84]. However, these prior works primarily focus on high-level planning on manipulation systems, featuring a low-dimensional action space (*e.g.*, end-effector position), low-replanning frequency (*e.g.*, around 10 Hz), and inherently more stable dynamics.

In contrast, using diffusion models for high-frequency, low-level control remains limited [7]. The most relevant work uses online RL to train diffusion-based actor policies in simpler simulation settings [87], but transitioning to real-world applications with high-frequency feedback control presents significant challenges due to the instability and rapid dynamics of legged robots [82]. This work aims to leverage diffusion models in low-level control for legged locomotion and demonstrate the advantages of multimodality and scalability in the real world.

III. Versatile Framework for Diffusion Locomotion Control From Offline Dataset

In this section, we provide an overview of *DiffuseLoco*, a framework designed to generate and utilize offline datasets for training multi-skill locomotion policies scalably. DiffuseLoco is based on the diffusion model and is designed to train a low-level multi-skill locomotion policy from offline datasets containing multiple agile locomotion skills with diverse behaviors. DiffuseLoco represents the state-of-the-art performance in multi-skill locomotion, as it is able to learn both bipedal and quadrupedal locomotion skills in a unified policy, and can be deployed robustly zero-shot on hardware. DiffuseLoco addresses several challenges inherent to learning from offline data, including generating diverse skills with the same goals, handling data variability, and ensuring real-time control in real-world environments.

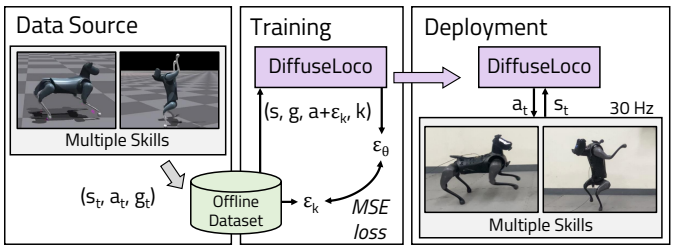


Fig. 2: Overview of the three stages of DiffuseLoco. First, we generate or utilize an offline dataset with demonstrations of a set of skills gathered with different methods (left). Then, we train DiffuseLoco policy with DDPM loss on trajectories within the dataset (middle). Finally, DiffuseLoco policy is deployed on robots in the real world and executes a diverse set of agile skills (right).

Algorithm 1 DiffuseLoco Algorithm

Initilize: N source policies $\pi^1_{\text{src}} \dots, \pi^N_{\text{src}}$, Empty or Existing Offline Dataset $\mathcal{D}_{\text{src}}$, Diffusion Model π_θ
// OBTAIN DATA FROM π_{src}
repeat
  Sample n uniformly from $\{1, \dots, N\}$
  Sample environment dynamics $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$
  Reset source-specific state $\mathbf{s}^{\text{src}}_t$
  for $t = 1$ **to** T **do**
    if $t \mod \text{random\_goal\_step} = 0$ **then**
      Sample goal $\mathbf{g}$
    end if
    Act source policy $\mathbf{a}^{\text{src}}_t = \pi^n_{\text{src}}(\mathbf{s}^{\text{src}}_t, \mathbf{g}_t)$
    Record source-agnostic state $\mathbf{s}_t$
    Step environment $\mathbf{s}^{\text{src}\prime}_t = \text{environment.step}(\mathbf{a}^{\text{src}}_t)$
    Record source-agnostic action $\mathbf{a}_t$
    Add data to offline dataset $\mathcal{D}_{\text{src}} \leftarrow (\mathbf{s}_t, \mathbf{a}_t, \mathbf{g}_t)$
  end for
until desired
// TRAIN ON OFFLINE DATASET
for each epoch **do**
  for each $(\mathbf{s}_{\text{traj}}, \mathbf{a}_{\text{traj}}, \mathbf{g}_{\text{traj}})$ in $\mathcal{D}_{\text{src}}$ **do**
    Compute the loss $L(\theta)$ with Eqn. 5
    Update model parameters θ to minimize loss $L(\theta)$
  end for
end for

A schematic illustration of the DiffuseLoco framework is shown in Figure 2. Our framework consists of three stages:

a) Data Source: We start with collecting a new or utilizing an existing offline dataset consisting of multiple skills. To generate or expand a dataset, we first obtain single-skill control policies as the source policies. These policies are conditioned on given goals $\mathbf{g}$ (commands), such as velocity commands and base heights, that are skill-agnostic. Note that

the source policies can be obtained with different methods. With the assumption of the frequency of low-level control being the same, the observation and action spaces of the *source* policies can be vastly different. Thus, we need to collect a set of *source-agnostic* state and action pairs across all source policies. For legged robots, the widely-used source-agnostic states and actions are the proprioceptive feedback $\mathbf{s}_t$ directly from the robot and the joint-level PD targets $\mathbf{a}_t$ directly to the robot, respectively. As illustrated in Algo. 1, we start an episode where the robot is controlled by the i^{th} source policy π^i. The policy acts at a source-specific state $\mathbf{s}_t^{\text{src}}$, and only the source-agnostic state-action-goal pairs $(\mathbf{s}_t,\ \mathbf{a}_t,\ \mathbf{g}_t)$ are collected during the rollout of the robot's closed-loop dynamics until it reaches the maximum episode length T. In addition, the goal $\mathbf{g}_t$ will be re-sampled within the command range after a time interval within the episode. In this work, we leverage cheap simulation data as an example, but since DiffuseLoco effectively re-uses data, we can scalably extend to more expensive data collection process, potentially from real-world hardware. The details of dataset generation in our experiments can be found in the Appendix D.

b) Training: In the second step, we train our DiffuseLoco policy from the offline dataset in an end-to-end manner. Let input state and goal history length be h and output action prediction length be n. During training, we sample a segment of state trajectory $\mathbf{s}_{\text{traj}}$ and corresponding action and goal sequences, $\mathbf{a}_{\text{traj}}$ and $\mathbf{g}_{\text{traj}}$. We sample a diffusion step k randomly from $\{1, \dots, K\}$, and sample a Gaussian noise ϵ_k to add to the action sequence. Then, a transformer-based denoising model takes the noisy action sequence along with states trajectory $\mathbf{s}_{\text{traj}}$, goal trajectory $\mathbf{g}_{\text{traj}}$, and diffusion step k as input, and predicts the added noise as ϵ_θ. The predicted noise ϵ_θ is then regressed to match the true noise ϵ_k with mean square error loss. In this way, the denoising model is learned to generate sequences of low-level actions conditioned on robot states and goals from the dataset. Details of the model architecture and training objective are introduced in Sec. IV.

c) Deployment: In the last stage, we zero-shot transfer the trained DiffuseLoco policy on the robot hardware. During deployment, the DiffuseLoco policy takes a sequence of pure noise sampled from a Gaussian distribution and denoises it conditioned on the state trajectory $\mathbf{s}_{\text{traj}}$ from the robot hardware and the given goal $\mathbf{g}_{\text{traj}}$. The denoising process is repeated for K iterations to generate a sequence of actions, but only the *immediate* action $\mathbf{a}_t$ is used as the robot's joint-level PD targets. After executing this action, the DiffuseLoco policy takes a new sequence of states from the robot and updates the immediate action from the newly generated action sequence. This is designed to align with the Receding Horizon Control (RHC) framework, instead of interpolating the action sequence at high frequency as previously used by other diffusion-based work [31, 41]. RHC enables DiffuseLoco to replan rapidly with fast-changing states of the robot to ensure up-to-date actions while keeping future steps in account. However, since the diffusion model has a large number of parameters and the diffusion process involves multiple denoising steps, we must accelerate the inference to be faster than the control frequency to achieve this RHC manner. The acceleration techniques are detailed in Appendix E, which ultimately enables running the DiffuseLoco policy on an edge-compute device that can be mounted on the robot.

IV. Diffusion Model for Real-Time Control

Having introduced the framework of DiffuseLoco, we now begin to develop its backbone: a diffusion model for locomotion control, shown in Fig. 3, with a special focus on design choices for real-time control and inference acceleration.

A. DDPM for Control

To model multi-modal behaviors from diverse datasets, we leverage Denoising Diffusion Probabilistic Models (DDPM) [22] with a transformer backbone to model different skills that can be applied to achieve a common goal. DDPM is a class of generative models in which the generative process is modeled as a denoising procedure, often referred to as Stochastic Langevin Dynamics, expressed in the following equation,

$$\mathbf{x}^{k-1} = \alpha\ \left(\mathbf{x}^k - \gamma\epsilon_\theta(\mathbf{x}^k, k) + \mathcal{N}(0, \sigma^2 I)\right) \tag{1}$$

where $\mathcal{N}(0, \sigma^2 I)$ denotes the sampled noise from a DDPM scheduler, α, γ, and σ are its hyperparameters: α regulates the rate at which noise is added at each step, γ represents the denoising strength, and σ defines the noise level. For clarity, we now use subscripts $_{t-a:t-b}$ to indicate trajectories from timestep $t-a$ to $t-b$, replacing subscripts $_{\text{traj}}$. To generate the action trajectory for control, an initial noisy action sequence, $\mathbf{a}^K_{t:t+n}$, is sampled from Gaussian noise, and the DDPM conditioned on states $\mathbf{s}_{t-h:t}$, goals $\mathbf{g}_{t-h:t}$, and previous actions $\mathbf{a}_{t-h-1:t-1}$ undergoes K iterations of denoising steps.

Unlike previous works applying DDPM in manipulation [12], the inclusion of previous actions, *i.e.*, I/O history, helps the policy to better perform system identification and state estimations for legged locomotion control, as evaluated in [44]. Furthermore, instead of concatenating state and goal into a single embedding [12, 41], we conveniently leverage the transformer's attention mechanism to assign different attention weights to separately embedded robot's I/O and static goals, enabling the policy to adjust focus between adapting to dynamic environments and achieving goals. We find that these modifications result in better command tracking performance and robustness, as shown in Sec. VII.

The denoising process yields a sequence of intermediate actions characterized by progressively decreasing noise levels: $\mathbf{a}^K, \mathbf{a}^{K-1}, \dots, \mathbf{a}^0$, until the desired noise-free output, $\mathbf{a}^0$, is attained. This process can be expressed as the following equation:

$$\begin{aligned}\mathbf{a}^{k-1}_{t:t+n} = \alpha\ \big(&\mathbf{a}^k_{t:t+n} - \gamma\epsilon_\theta(\mathbf{a}^k_{t-h-1:t+n}, \mathbf{s}_{t-h:t}, \mathbf{g}_{t-h:t}, k) \\ &+\mathcal{N}(0, \sigma^2 I)\big)\end{aligned} \tag{2}$$

where $\mathbf{a}^k_{t:t+n}$ represents the output at the k^{th} iteration, and $\epsilon_\theta(\mathbf{a}^k_{t-h-1:t+n}, \mathbf{s}_{t-h:t}, k)$ represents the predicted noise from

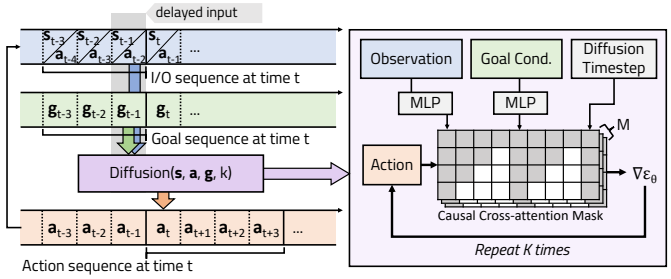


Fig. 3: The DiffuseLoco architecture. At time step t, it takes in a delayed h-step history of proprioceptive states $\mathbf{s}_{t-h-1:t-1}$, goals $\mathbf{g}_{t-h-1:t-1}$, and actions $\mathbf{a}_{t-h-2:t-2}$, and predicts a sequence of n future actions $\mathbf{a}_{t:t+n}$ for the robot's actuators. First, separate MLP encoders map state and goal into embeddings which, with a one-hot diffusion step, are queried by noisy action tokens via M transformer decoder layers for denoising. After K denoising iterations, the predicted action sequence is generated and we feed the executed action back to the model's input. The model is trained through end-to-end imitation learning.

the denoising model ϵ_θ, which is parameterized by θ, with respect to $\mathbf{a}^k_{t-h-1:t+n}$, $\mathbf{s}_{t-h-1:t}$, and iteration k.

During training, we opt to use the simplified training objective proposed by Ho et al. [22],

$$l = MSE\left(\epsilon_k, \epsilon_\theta(\mathbf{a}_{t-h-1:t+n} + \epsilon_k, \mathbf{s}_{t-h:t}, \mathbf{g}_{t-h:t}, k)\right). \quad (3)$$

where ϵ_k is the sampled noise at iteration k. The detailed model architecture can be found in Appendix B.

B. Delayed Input and Predicted Actions

To achieve real-time deployment, we introduce the technique of predicting current actions using delayed inputs, overcoming the challenge posed by the extensive inference times of large models like transformers, which exceed the common control frequency requirement of 30-50Hz.

In the DiffuseLoco policy, we use delayed inputs from the previous timestep—$\mathbf{s}_{t-h-1:t-1}$, $\mathbf{a}_{t-h-2:t-2}$, and $\mathbf{g}_{t-h-1:t-1}$—to predict actions for the current timestep $\mathbf{a}_{t:t+n}$. By initiating inference for the next action before the current state $\mathbf{s}_t$ is received, we can run the inference in parallel and ensure actions are the most up-to-date. This design choice is favored for two reasons: First, we train a sequence of predictions, instead of one-step prediction autoregressively, which is suitable for generating actions further in time than the nearest timestep. Second, these larger-scale models are adept at handling higher input delays. Since delays (zero-order-hold) exponentially deteriorate the stability of the system, this marks an advantage over small-scale MLP policies, which typically manage delays less than one control step, for instance, 25% less than DiffuseLoco as noted in [44, Table IV].

In this way, we modify the denoising process in Eqn. 2 to the following,

$$\begin{aligned}\mathbf{a}^{k-1}_{t:t+n} = \alpha\ \big(&\mathbf{a}^k_{t:t+n} \\ &-\gamma\epsilon_\theta(\mathbf{a}^k_{t-h-2:t+n}, \mathbf{s}_{t-h-1:t-1}, \mathbf{g}_{t-h-1:t-1}, k) \\ &+ \mathcal{N}(0, \sigma^2 I)\big)\end{aligned} \quad (4)$$

and the loss function in Eqn. 3 to the following,

$$\begin{aligned}l = MSE\,(\epsilon_k, \epsilon_\theta(&\mathbf{a}_{t-h-2:t+n} + \epsilon_k, \\ &\mathbf{s}_{t-h-1:t-1}, \mathbf{g}_{t-h-1:t-1}, k))\end{aligned} \quad (5)$$

Note that for more complex tasks, larger models may require more time for inference, then our method can be extended by delaying more than one timestep of inputs. However, more delayed steps will result in more challenging learning complexity because of the lack of recent state feedback.

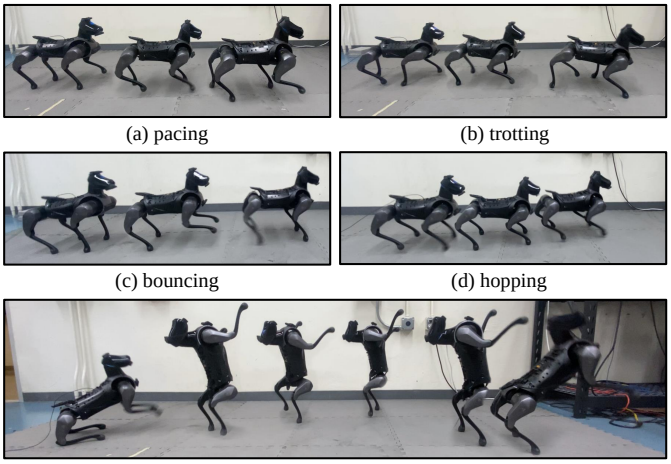
Remark 1: Another possible solution to run inference before $\mathbf{s}_t$ arrives is to predict $\mathbf{s}_t$ given h-step history of previous states, $\mathbf{s}_{t-h:t-1}$ with either a model-based Kalman Filter [59] or a learning-based prediction model [30]. However, state prediction inevitably introduces extra prediction error or bias to the policy.

C. Sampling Techniques

To accelerate diffusion models, especially during robotic deployment, prior work often uses samplers like the Denoising Diffusion Implicit Models (DDIM)[72], which employ a deterministic process to reduce sampling steps and speed up inference, albeit with some loss in sample quality. However, we find that DDIM is less suited for real-time control on legged robots due to its less accurate action outputs, which increases the compounding error of each step that leads to more frequent out-of-distribution scenarios from training to real-world deployment. Consequently, in DiffuseLoco, we continue using the Denoising Diffusion Probabilistic Model (DDPM), maintaining the same number of denoising iterations as during training. Our detailed comparisons in Sec.VII-C show that DDPM enhances robustness and performance over DDIM in real-time control scenarios.

V. Results: Model Capacities

To scale up learning locomotion skills as discussed in Sec. I, the critical questions we need to address are whether DiffuseLoco can a) be trained with various sources of demonstrations and b) incorporate a diverse set of skills present in the dataset. In this work, we answer these questions by presenting a state-of-the-art five-skill policy that combines four quadrupedal skills, and more importantly, a bipedal locomotion skill, for quadrupedal robots. As discussed in Sec. II-A, the ability to learn these diverse skills with a single model, including an agile bipedal locomotion policy along with quadrupedal skills, remains challenging and has not yet been demonstrated by prior RL frameworks.



(a) pacing (b) trotting

(c) bouncing (d) hopping

(e) bipedal walking

Fig. 4: Snapshots of five diverse agile locomotion skills with the DiffuseLoco policy. This represents a leading effort in developing a single policy that can combine an agile bipedal walking skill with other quadrupedal skills and can be deployed on real-world robots.

A. Learning from Diverse Data Sources

Fig. 4 illustrates the five skills acquired by DiffuseLoco for a quadrupedal robot, encompassing quadrupedal walking skills like trotting and pacing, jumping skills such as hopping and bouncing, and an agile bipedal walking skill. Among these, trotting and pacing are trained using AMP [16], hopping and bouncing through nominal CPG curves [70], and bipedal locomotion with symmetry augmentation [73]. After collecting demonstrations of these skills separately in simulation, we directly learn from this combined dataset and achieve robust zero-shot transfer to actual hardware. This capability surpasses previous offline learning methods that were largely confined to simulated environments with simplified dynamics, showcasing DiffuseLoco's robust real-world performance.

The ability to learn from diverse skill sources is crucial for scaling locomotion learning, especially for this five-skill controller. For instance, bipedal locomotion requires a distinct framework with specific early termination conditions and reward landscape compared to quadrupedal skills. With these specialized conditions, basic symmetry augmentation can already yield effective stepping patterns, whereas, learning the same skill with prior multi-skill RL approaches like AMP [56] or motion imitation involves complex reference motions and trajectory optimization [78], representing a significant engineering challenge. Despite the different requirements for observation, action spaces, and auxiliary signals such as phase signals across various RL methods used in this work, DiffuseLoco simplifies the process by solely relying on basic proprioceptive inputs to accomplish all skills, thanks to the powerful multimodal capability of diffusion models and receding horizon control.

B. Skill Transitioning

More importantly, we demonstrate DiffuseLoco's capacity to transition freely between skills, which are not originally present in the dataset, such as transitioning from hopping to bipedal walking and then to pacing, as shown in Fig. 1. This sequence highlights the robustness of the DiffuseLoco policy against variations in starting state and the stability required to execute these skills successfully. Out of five consecutive runs, we do not experience any failure. Additional examples of skill transitions are available in Appendix A.

In addition to transitions under *different* goals (commands), the DiffuseLoco policy also demonstrates the ability to perform both trotting and pacing under the *same* goal. In our experiment, as shown in Fig. 5, the policy begins with trotting and only switches to pacing when a sudden braking event significantly alters the contact sequence. This highlights DiffuseLoco's effectiveness in learning and adhering to different modes from the offline dataset, committing to a single mode within each rollout unless prompted by external disturbances.

C. Extension to Bipedal Robots

In addition to quadrupedal robots, we also demonstrate the effectiveness of our method on high-dimensional, highly non-linear bipedal robots with a human-sized Cassie in the MuJoCo simulation. First, we collect demonstrations evenly from two separately trained single-skill RL policies on walking and running, adapted from [44]. After training directly on this aggregated dataset, our method successfully learns both skills within a single policy. Furthermore, as shown in Fig. 1, our policy can transition from walking to running smoothly without specific transition data in the dataset, in addition to maintaining each skill's stability before and after transitions. This demonstrates one of the initial working combinations of these skills on bipedal robots.

D. Robustness

To demonstrate DiffuseLoco's robustness, we show both quadrupedal and bipedal locomotion over different ground conditions, including padded floor, bare floor, turf, and over small terrain variations, as shown in Fig. 6. We especially emphasize bipedal walking over the half-padded floor, demonstrating a high degree of robustness to the differences in ground heights, as well as friction and restitution forces on each side of the two standing legs.

Admittedly, DiffuseLoco's robustness is subpar compared to the state-of-the-art single-skill RL policy. However, while the RL policy is mostly limited by simulation randomization, one of the most desirable advantages of DiffuseLoco is its ability to learn scalably on offline real-world data without accessing expert policies in training or repeated real-world collection processes. More importantly, as we will show in Sec. VII-E, DiffuseLoco shows the potential to improve its robustness by absorbing data from various dynamics. By expanding our dataset with more diverse, potentially real-world data, we anticipate DiffuseLoco to achieve better robustness progressively.

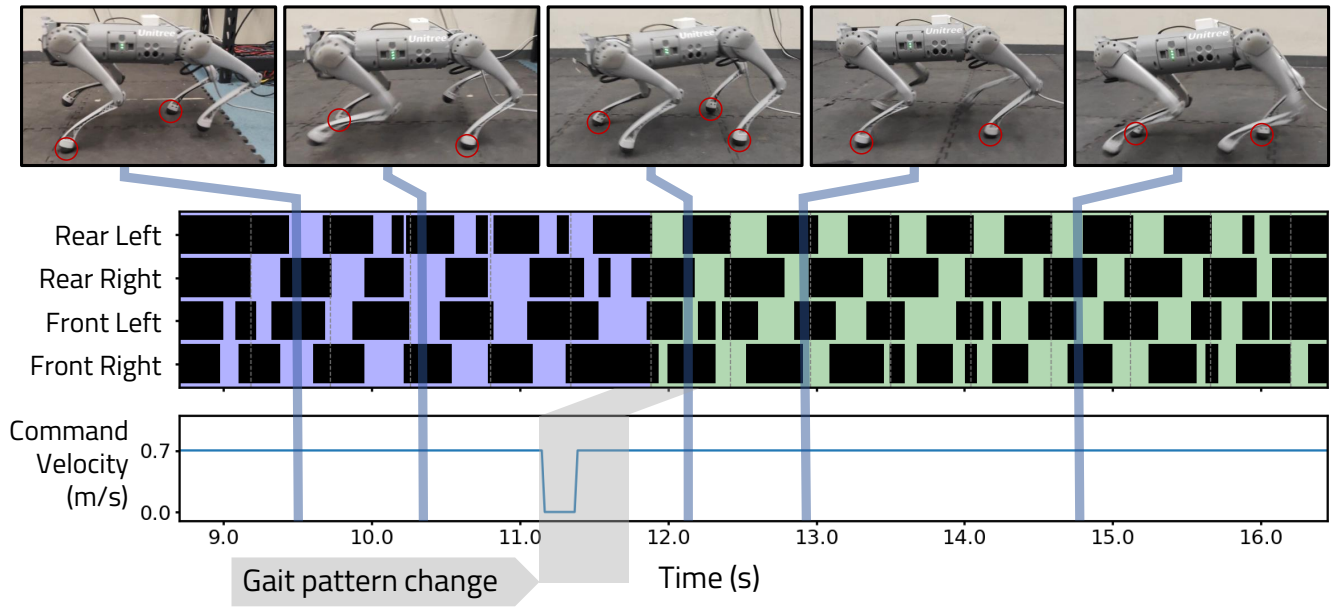


Fig. 5: Foot contact map indicating stable walking and skill switching with DiffuseLoco policy and velocity commands. The red circle denotes the legs that are in contact with the ground. The robot initially walks using trotting skill, indicated by a purple background, then switches to pacing, shown in green, following a command change that involves a sudden stop and resume. We emphasize DiffuseLoco's ability to maintain different modalities for stable walking under the same command, switching modalities only when necessary.

In conclusion, in addressing question a), we have demonstrated that DiffuseLoco is invariant to the source of offline demonstrations, and can be trained with data from multiple specialized RL frameworks. More importantly, DiffuseLoco shows better scalability in learning diverse skills that existing RL frameworks have not yet illustrated, affirming its state-of-the-art capability as an answer to question b). We believe that the potential of DiffuseLoco extends significantly beyond the current five skills showcased, suggesting promising avenues for future works.

VI. Quantitative Analysis

In this section, we seek to quantitatively compare DiffuseLoco against various existing multi-skill RL and non-diffusion behavior cloning (BC) baselines. Since there is no existing RL baseline that learns the five skills in the previous section, we refer to only quadrupedal walking skills, including pacing and trotting, as a case study for the performance of DiffuseLoco.

A. Task and Baselines

This analysis includes walking for four meters under five goals (commands) with different velocities. The goals are the following: move forward at three different speeds: 0.3 $\mathrm{m/s}$, 0.5 $\mathrm{m/s}$, and 0.7 $\mathrm{m/s}$, and make a left turn and a right turn at 0.3 $\mathrm{rad/s}$. We record the actual linear velocities via a Kalman filter state estimation [18], and the number of trials where the robot does not fall over through out the trial as the stability metrics. We repeat each experiment five times and report the mean and standard deviation across five runs.

Skill information are often unscalable or unavailable during training and deployment. For a boarder range of applicability, we limit the scope of comparisons within not-skill-conditioned multi-skill RL and non-diffusion BC baselines. Specifically, the RL baselines include,

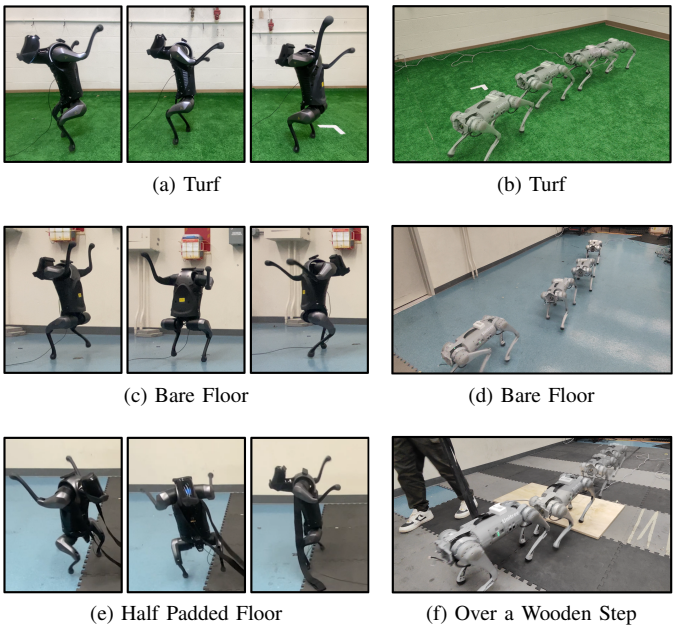
- Adversarial Motion Priors (**AMP**) [16]: An MLP policy trained using AMP with RL (PPO) and style reward of both pacing and trotting reference motions. We *directly* use the *open-sourced* checkpoint from [16]. We note that although several skill-conditioned RL policies [39, 83, 89] have been introduced since [16], yielding better sim-to-real results, progress in unconditioned multi-skill policies has been limited.
- AMP with history steps (**AMP w/ H**): To align with **DiffuseLoco**, we train an AMP policy with 8 steps of state and action history with the same setup as [16] and a similar evaluation return in simulation.

Furthermore, we compare **DiffuseLoco** with non-diffusion BC policies, which can be categorized into autoregressive token prediction [9, 25] and action sequence prediction as used in [19]. We adopt baselines for each category.

- Transformer with Autoregressive Token Prediction (**TF**): A Generative Pretrained Transformer (GPT) [5] policy similar to a decision transformer [9] without reward conditioning. This only generates one timestep action.
- Transformer with Receding Horizon Control (**TF w/ RHC**): A transformer policy with the same future step action predictions. The model's architecture is identical

Goal (Task)	Metric	AMP	AMP w/ H	TF	TF w/ RHC	DiffuseLoco (Ours)
$0.3m/s$ **Forward**	Stability (%)	**100**	**100**	80	**100**	**100**
	E_v (%)	90.44 ± 1.87	90.63 ± 4.79	75.75 ± 6.07	39.28 ± 2.34	**33.22 ± 12.48**
$0.5m/s$ **Forward**	Stability (%)	**100**	**100**	**100**	**100**	**100**
	E_v (%)	50.44 ± 1.97	46.29 ± 2.55	54.35 ± 2.66	37.46 ± 5.31	**12.91 ± 6.84**
$0.7m/s$ **Forward**	Stability (%)	0	20	0	40	**100**
	E_v (%)	~~fail 5/5~~	54.96 ± 0.00	~~fail 5/5~~	39.36 ± 5.02	**24.80 ± 8.91**
Turn Left	Stability (%)	20	**100**	0.	**100**	**100**
	E_v (%)	20.96 ± 0.00	33.39 ± 6.96	~~fail 5/5~~	13.41 ± 5.02	**12.79 ± 5.64**
Turn Right	Stability (%)	**100**	**100**	**100**	80	**100**
	E_v (%)	18.61 ± 2.40	33.39 ± 6.96	25.86 ± 1.47	8.69 ± 5.04	**2.22 ± 1.03**

TABLE I: Performance Benchmark across different baselines and our DiffuseLoco policy in the real world. Stability (the higher the better) measures the number of trials in which the robot stays stable and does not fall over. E_v (the lower the better) measures the deviation from the desired velocity in percentage. The experiments are conducted with different command settings (Left). Each command is repeated non-stop for five trials, and we report the average and standard deviation of the metrics across five trials.



(a) Turf (b) Turf

(c) Bare Floor (d) Bare Floor

(e) Half Padded Floor (f) Over a Wooden Step

Fig. 6: Depiction of DiffuseLoco's robustness on different ground conditions and terrains: bipedal walking on (a) turf terrain, (c) vinyl composite floor, and (e) half padded floor, where the ground heights, friction and restitution forces on the two standing legs are different; quadrupedal walking on (b) turf, (d) bare floor, (f) over a thick wooden board as a variation in the terrain height.

to our **DiffuseLoco** model, but it directly predicts future action sequences and the loss is replaced by the reconstruction loss $l = MSE(\pi_\theta(\mathbf{s}_t, \mathbf{g}_t), \mathbf{a}_t)$.

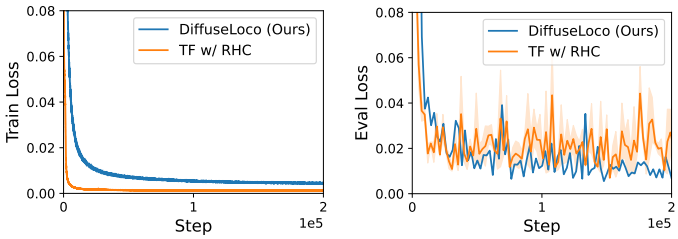
These baselines have the same parameter count of 6.8M and are trained with the same learning rate scheme and number of epochs as **DiffuseLoco**.

Remark 2: Typically, previous work uses DAgger style algorithms [65] to better cope with distribution shift, but these methods require access to the expert policy in training and online learning environments. As a more scalable and versatile framework, we limit our focus to learning entirely from offline datasets.

B. DiffuseLoco versus AMP (RL)

We first compare DiffuseLoco with RL-based multi-skill control policy **AMP**. Table I shows that **DiffuseLoco** is the only method among all of the baselines in our benchmark that is able to reliably complete all trials without falling over. Specifically, the RL-trained **AMP** and **AMP w/ H** baselines struggle with low and high speed commands. For 0.3 $\mathrm{m/s}$ forward command, these two baselines give a velocity tracking performance of more than 90% slower than the commanded velocity. For 0.7 $\mathrm{m/s}$ forward command, they achieve a stability metric of 0% and 20% respectively.

This shows the prevailing problem of mode-collapse for Generative Adversarial Network (GAN) style networks, such as a multi-skill AMP policy. Mode collapse is a significant challenge in GANs, where the generator becomes overfitted to a limited range of outputs that are often similar or identical, rather than offering a broad range of solutions [15, 46, 49]. In the context of AMP, this means the actor network excessively overfits the simulation environment, losing its ability to generalize and adapt to new environments (such as the real world), but instead reverting to early training behaviors where the discriminator is not yet converged. In real-world testing, an example of this is seen in low-speed tasks where AMP oscillates in place with correct frequency but reduced amplitude, similar to its behaviors in early training stage. In contrast, DiffuseLoco generalizes within the expert demonstrations and does not revert to infeasible actions. Note that in the simulation



(a) (b)

Fig. 7: Loss curves for training **DiffuseLoco** and **TF w/ RHC** baseline. (a): Training Loss. (b): Evaluation Loss. We report mean and standard deviation across three seeds. We find that even though **TF w/ RHC** achieves a low reconstruction loss in training, the evaluation loss stays higher than **DiffuseLoco** and increases at the end. This indicates **TF w/ RHC** tends to overfit the offline datasets while *DiffuseLoco* is not, with the same number of parameters.

environment, both **AMP** and **AMP w/ H** are able to control the robot to track different velocities without falling over.

Besides better sim-to-real transfer, **DiffuseLoco** with a diffusion model is able to efficiently learn the multi-modality presented by the different skills for the same locomotion task, and thus able to perform valid and coordinated locomotion skills without mode-collapsing, as an example given by Fig. 5. This helps **DiffuseLoco** achieve both better stability and track completion (velocity tracking) performance compared with AMP-based policy baselines.

C. DiffuseLoco versus Non-Diffusion Behavior Cloning

We further compare **DiffuseLoco** with non-diffusion BC methods. For locomotion tasks, smooth and temporally consistent actions are a necessity for stability and robustness. Looking at Table I, we find that **DiffuseLoco** outperforms **TF** and **TF w/ RHC** in both stability and robustness of the locomotion policy. Using one-step action output, we find that **TF** lacks robustness and fails the 0.7 $\mathrm{m/s}$ forward and left turn tasks completely. This is likely because single-step action prediction makes the policy less aware of future actions, leading to more jittering behavior.

With receding horizon control, **TF w/ RHC** overcomes most of the jittering problem and can complete most of the tasks. However, we note that for more agile motion such as the 0.7 $\mathrm{m/s}$ forward task, the stability metric drops drastically to merely 40%. This is likely because the reconstruction loss used in **TF w/ RHC** training tends to overfit the action trajectories in the dataset, resulting in less robust policy in the out-of-distribution scenarios (such as in the real world).

This is especially evident when looking at training curves for **TF w/ RHC** versus **DiffuseLoco** shown in Figure 7, where **TF w/ RHC** overfits significantly to the training dataset and the evaluation loss stays high. Note that the model architecture is kept identical across **TF w/ RHC** and **DiffuseLoco**, so only the loss functions are different. In addition, the evaluation loss here is calculated with samples from the same distribution as the training dataset, which means the overfitting problem becomes more pronounced when we switch to real-world experiments, as shown earlier.

In comparison, our **DiffuseLoco** shows more stable and smooth motions measured by both stability metrics and magnitudes of the body's angular velocity. On average, **DiffuseLoco** achieves 10.40% less in magnitude for the body's oscillation over all trials. As a result, the smoother locomotion skill helps **DiffuseLoco** to achieve on average 38.97% less tracking error compared to **TF w/ RHC**. Based on this observation, we suggest that DDPM-style training is more suitable for imitating locomotion tasks compared to prior Behavior Cloning methods.

VII. Ablation Study on Design Choices

In this section, we further evaluate the design choices used to build DiffuseLoco policy in simulation and the real world by extensive ablation studies. We use the same experiment setups as the previous section. For brevity of the main content, further ablation studies can be found in Appendix C.

A. Ablation Components

To validate our design choices, we ablate **DiffuseLoco** with the following critical components and compare them to our real-world benchmark.

- Without Receding Horizon Control (**DL w/o RHC**): Replace RHC with one-step prediction in an autoregressive manner and keep the diffusion model.
- Without Domain Randomization (**DL w/o Rand**): Trained on a dataset generated without domain randomization, except for the ground friction coefficient.
- DDIM Inference: We develop two DDIM baselines to investigate how training and inference steps affect performance in locomotion control.
  - 100 Training + 10 Inference (**DDIM-100/10**)
  - 10 Training + 5 Inference (**DDIM-10/5**)

  Compared with our **DiffuseLoco**, **DDIM-100/10** has the same inference steps, and **DDIM-10/5** has the same training steps.
- U-Net as the backbone (**U-Net**): Replace the Transformer with a U-Net as the backbone, adjusted to the same parameter count.

B. Single-step output versus RHC

To isolate the effects of RHC, we test a variant of DiffuseLoco without RHC (**DL w/o RHC**), finding that it struggles with faster speed goal and exhibits significant jittering behaviors, as detailed in Table II. This suggests that single-step token-prediction models like GPT are less suitable for legged locomotion control than diffusion models, which predict sequences of future actions.

C. Sampling Techniques

As discussed earlier, popular diffusion-based frameworks like DDIM often reduce sample iterations for inference acceleration, trading off output quality for speed, often with

Goal (Task)	Metric	DL w/o RHC	DL w/o Rand	DDIM-100/10	DDIM-10/5	U-Net	DiffuseLoco (Ours)
$0.3m/s$ **Forward**	Stability (%)	**100**	**100**	**100**	**100**	**100**	**100**
	E_v (%)	75.09 ± 18.98	50.45 ± 2.70	56.89 ± 2.43	47.09 ± 2.40	81.31 ± 1.90	**33.22 ± 12.48**
$0.5m/s$ **Forward**	Stability (%)	**100**	80	80	**100**	**100**	**100**
	E_v (%)	64.49 ± 1.87	41.07 ± 6.12	41.00 ± 3.18	37.92 ± 1.59	74.52 ± 2.83	**12.91 ± 6.84**
$0.7m/s$ **Forward**	Stability (%)	0	40	80	80	**100**	**100**
	E_v (%)	~~fail 5/5~~	44.30 ± 4.21	47.71 ± 6.63	42.58 ± 2.08	71.71 ± 2.93	**24.80 ± 8.91**
Turn Left	Stability (%)	**100**	**100**	**100**	**100**	20	**100**
	E_v (%)	20.96 ± 18.22	**10.17 ± 5.86**	22.22 ± 4.29	13.27 ± 2.63	18.93 ± 23.28	12.79 ± 5.64
Turn Right	Stability (%)	**100**	**100**	**100**	**100**	**100**	**100**
	E_v (%)	18.61 ± 2.40	8.18 ± 3.94	6.47 ± 2.49	7.42 ± 2.90	89.63± 3.36	**2.22 ± 1.03**

TABLE II: Performance Ablation Study across different ablations and DiffuseLoco policy in real-world experiments. Stability (the higher the better) measures the number of trials in which the robot stays stable and does not fall over. E_v (the lower the better) measures the deviation from the desired velocity in percentage. The experiments are conducted with different command settings (Left). Each command is repeated non-stop for five trials, and we report the average and standard deviation of the metrics across five trials.

ten times fewer iterations [72]. While this approach suits tasks like image generation, which tolerate some variance, it underperforms in quadrupedal locomotion control. As shown in Table II, both **DDIM-100/10** and **DDIM-10/5** exhibit worse stability and higher velocity tracking errors. Noticeably, the 100 training steps and 10 inference steps variant demonstrates limping behavior and fails two trials. Tracking errors for both variants increase by 50.69% and 42.04%, respectively, compared to **DiffuseLoco**.

Thus, we believe that noisier control signals from the DDIM pipeline likely disrupt the control of inherently unstable floating-based dynamic systems, like legged robots. An interesting future work direction could be on control-specific sampling techniques to accelerate diffusion models without compromising stability and performance.

D. Model Architecture Effects

In addition, we compare against another commonly used architecture in diffusion models, a CNN-based U-Net as the backbone of DiffuseLoco. Qualitatively, the **U-Net** policy is shaky and inconsistent, and quantitatively, it has one of the highest errors in velocity tracking, with worsened stability due to its shaky actions. We reckon that this is because CNNs are not the best fit for temporal data and also lack separate attention weights for goal conditioning. This is consistent with prior work [12] that finds **U-Net** underperforms Transformer, especially in high action-rate dynamical systems.

E. Dataset Effects

Lastly, we explore how dataset characteristics influence the robustness and performance of **DiffuseLoco** in real-world scenarios. Consistent with previous findings that diversity in training data, such as noise insertion, mitigates compounding error [37], we demonstrate that increasing the variety of dynamics parameters in simulation environments where we collect data also enhances robustness. As in Table II, training DiffuseLoco on a dataset with dynamics randomization leads to a 44.26% increase in both robustness and stability compared to **DL w/o Rand** baseline. Specifically, in the challenging 0.7 $\mathrm{m/s}$ forward task, **DL w/o Rand** falls in 3 out of 5 trials. This ablation study points to the potential of altering the dataset, by adding either more diversity and potentially real-world data or more fault-recovery behaviors, to further enhance the robustness of DiffuseLoco.

VIII. Discussion and Future Work

We have presented DiffuseLoco, a scalable learning framework to learn diverse agile legged locomotion skills from multi-modal *offline* datasets and can robustly transfer to real-world robots in real-time. Leveraging diffusion models to capture the multi-modality in the offline dataset, DiffuseLoco learns a state-of-the-art controller that combines bipedal walking and quadrupedal locomotion skills within one policy and transitions freely among the skills.

A. A Scalable Approach for Locomotion Skills

In this work, we focus on the scalability of learning legged locomotion control. Inspired by successes of large-scale learning in other robotics tasks, we leverage the most scalable approach: learning from offline datasets, with special attention to the versatility of the data sources and the multi-modality of different skills in a large-scale dataset. With an expressive diffusion model as the backbone, we are able to absorb demonstrations learned with various existing RL algorithms with potentially different observation and action spaces, and effectively execute skills, such as trotting and pacing, that represent different modalities given identical commands. With the five diverse skills presented in this work as a testimony, we show the scalability of DiffuseLoco towards a generalist policy for locomotion control tasks.

B. Benefits over Other Multi-skill Policies

As we show in our thorough real-world benchmark (Sec. VI), DiffuseLoco demonstrates smoother actions and improved stability and velocity tracking in real-world conditions compared to non-diffusion Behavior Cloning baselines that are commonly used in prior works. When not conditioned on explicit skill labels, DiffuseLoco shows better sim-to-real transfer performance for multi-skill locomotion compared to AMP policies, which often face a significant sim-to-real gap due to mode-collapsing in generator-discriminator methods. DiffuseLoco avoids these issues, providing stable, coherent, and effective control with smooth skill-switching and stable execution under consistent commands.

C. Large-scale Offline Dataset for Locomotion

In the experiments, DiffuseLoco demonstrates scalability and robustness by utilizing diverse offline datasets, as discussed in Sec. VII-E. Unlike online learning which mostly depends on simulations, it offers a practical inspiration for scalable real-world data collection and learning. Similar to [17], we also hypothesize that DiffuseLoco could adapt to datasets containing different robot morphologies, thus allowing for broader deployment and better generalization. Furthermore, as a popular direction in manipulation fields, integrating vision and language instructions into the goal-conditioning dataset could further enhance DiffuseLoco's versatility and applicability in future works.

IX. Acknowledgement

This work was supported in part by NSF 2303735 for POSE, in part byNSF 2238346 for CAREER, in part by The AI Institute and in part byInnoHK of the Government of the Hong Kong Special AdministrativeRegion via the Hong Kong Centre for Logistics Robotics.

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[2] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-shot robotic manipulation with pre-trained image-editing diffusion models. *arXiv preprint arXiv:2310.10639*, 2023.

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control, 2023.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale, 2023.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] Guillermo A Castillo, Bowen Weng, Wei Zhang, and Ayonga Hereid. Robust feedback motion policy design using reinforcement learning on a 3d digit bipedal robot. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5136–5143. IEEE, 2021.

[7] Huayu Chen, Cheng Lu, Zhengyi Wang, Hang Su, and Jun Zhu. Score Regularized Policy Optimization through Diffusion Behavior, 2023.

[8] Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2884–2890. IEEE, 2019.

[9] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement Learning via Sequence Modeling, 2021.

[10] Lili Chen, Shikhar Bahl, and Deepak Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *Conference on Robot Learning*, pages 2012–2029. PMLR, 2023.

[11] Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. *arXiv preprint arXiv:2309.14341*, 2023.

[12] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action

diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[13] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700. IEEE, 2018.

[14] Jeremy Dao, Kevin Green, Helei Duan, Alan Fern, and Jonathan Hurst. Sim-to-real learning for bipedal locomotion under unsensed dynamic loads. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10449–10455. IEEE, 2022.

[15] Ricard Durall, Avraam Chatzimichailidis, Peter Labus, and Janis Keuper. Combating mode collapse in gan training: An empirical analysis using hessian eigenvalues. *arXiv preprint arXiv:2012.09673*, 2020.

[16] Alejandro Escontrela, Xue Bin Peng, Wenhao Yu, Tingnan Zhang, Atil Iscen, Ken Goldberg, and Pieter Abbeel. Adversarial motion priors make good substitutes for complex reward functions. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 25–32. IEEE, 2022.

[17] Gilbert Feng, Hongbo Zhang, Zhongyu Li, Xue Bin Peng, Bhuvan Basireddy, Linzhu Yue, Zhitao Song, Lizhi Yang, Yunhui Liu, Koushil Sreenath, et al. Genloco: Generalized locomotion controllers for quadrupedal robots. In *Conference on Robot Learning*, pages 1893–1903. PMLR, 2023.

[18] Thomas Flayols, Andrea Del Prete, Patrick Wensing, Alexis Mifsud, Mehdi Benallegue, and Olivier Stasse. Experimental evaluation of simple estimators for humanoid robots. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 889–895. IEEE, 2017.

[19] Zipeng Fu, Tony Z Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[20] Yuni Fuchioka, Zhaoming Xie, and Michiel Van de Panne. Opt-mimic: Imitation of optimized trajectories for dynamic quadruped behaviors. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5092–5098. IEEE, 2023.

[21] Huy Ha, Pete Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Conference on Robot Learning*, pages 3766–3777. PMLR, 2023.

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models, 2020.

[23] David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.

[24] Siyuan Huang, Zan Wang, Puhao Li, Baoxiong Jia, Tengyu Liu, Yixin Zhu, Wei Liang, and Song-Chun Zhu. Diffusion-based generation, optimization, and planning in 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16750–16761, 2023.

[25] Xiaoyu Huang, Dhruv Batra, Akshara Rai, and Andrew Szot. Skill transformer: A monolithic policy for mobile manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10852–10862, 2023.

[26] Xiaoyu Huang, Zhongyu Li, Yanzhen Xiang, Yiming Ni, Yufeng Chi, Yunhao Li, Lizhi Yang, Xue Bin Peng, and Koushil Sreenath. Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2715–2722. IEEE, 2023.

[27] Jemin Hwangbo, Joonho Lee, A. Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4, 2019. doi: 10.1126/scirobotics.aau5872.

[28] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[29] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis, 2022.

[30] Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7 (2):4630–4637, 2022.

[31] Ivan Kapelyukh, Vitalis Vosylius, and Edward Johns. DALL-E-Bot: Introducing Web-Scale Diffusion Models to Robotics. *IEEE Robotics and Automation Letters*, 8 (7):3956–3963, July 2023. ISSN 2377-3774. doi: 10.1109/lra.2023.3272516. URL http://dx.doi.org/10.1109/LRA.2023.3272516.

[32] J. Kaplan, Sam McCandlish, T. Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeff Wu, and Dario Amodei. Scaling Laws for Neural Language Models. *ArXiv*, abs/2001.08361, 2020.

[33] Sunwoo Kim, Maks Sorokin, Jehee Lee, and Sehoon Ha. Humanconquad: human motion control of quadrupedal robots using deep reinforcement learning. In *SIGGRAPH Asia 2022 Emerging Technologies*, pages 1–2, 2022.

[34] Arnaud Klipfel, Nitish Sontakke, Ren Liu, and Sehoon Ha. Learning a single policy for diverse behaviors on a quadrupedal robot using scalable motion imitation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2768–2775. IEEE, 2023.

[35] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[36] Aviral Kumar, Aurick Zhou, G. Tucker, and S. Levine. Conservative Q-Learning for Offline Reinforcement Learning. *ArXiv*, abs/2006.04779, 2020.

[37] Michael Laskey, Jonathan Lee, Roy Fox, Anca Dragan, and Ken Goldberg. Dart: Noise injection for robust imitation learning. In *Conference on robot learning*, pages 143–156. PMLR, 2017.

[38] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[39] Chenhao Li, Sebastian Blaes, Pavel Kolev, Marin Vlastelica, Jonas Frey, and Georg Martius. Versatile skill control via self-supervised adversarial imitation of unlabeled mixed motions. In *2023 IEEE international conference on robotics and automation (ICRA)*, pages 2944–2950. IEEE, 2023.

[40] Chenhao Li, Marin Vlastelica, Sebastian Blaes, Jonas Frey, Felix Grimminger, and Georg Martius. Learning agile skills via adversarial imitation of rough partial demonstrations. In *Conference on Robot Learning*, pages 342–352. PMLR, 2023.

[41] Xiang Li, Varun Belagali, Jinghuan Shang, and Michael S. Ryoo. Crossway Diffusion: Improving Diffusion-based Visuomotor Policy via Self-supervised Learning, 2024.

[42] Yunfei Li, Jinhan Li, Wei Fu, and Yi Wu. Learning Agile Bipedal Motions on a Quadrupedal Robot. *arXiv preprint arXiv:2311.05818*, 2023.

[43] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2811–2817. IEEE, 2021.

[44] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement Learning for Versatile, Dynamic, and Robust Bipedal Locomotion Control, 2024.

[45] Qiayuan Liao, Zhongyu Li, Akshay Thirugnanam, Jun Zeng, and Koushil Sreenath. Walking in narrow spaces: Safety-critical locomotion control for quadrupedal robots with duality-based optimization. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2723–2730. IEEE, 2023.

[46] Kanglin Liu, Wenming Tang, Fei Zhou, and Guoping Qiu. Spectral regularization for combating mode collapse in gans. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6382–6390, 2019.

[47] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.

[48] Utkarsh A. Mishra, Shangjie Xue, Yongxin Chen, and Danfei Xu. Generative Skill Chaining: Long-Horizon Skill Planning with Diffusion Models, 2023.

[49] Vaishnavh Nagarajan and J Zico Kolter. Gradient descent GAN optimization is locally stable. *Advances in neural information processing systems*, 30, 2017.

[50] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6292–6299. IEEE, 2018.

[51] Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated Offline RL Pre-Training for Efficient Online Fine-Tuning. *arXiv preprint arXiv:2303.05479*, 2023.

[52] Felipe Nuti, Tim Franzmeyer, and João F Henriques. Extracting Reward Functions from Diffusion Models. *arXiv preprint arXiv:2306.01804*, 2023.

[53] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An Open-Source Generalist Robot Policy. https://octo-models.github.io, 2023.

[54] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. *arXiv preprint arXiv:1709.07174*, 2017.

[55] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating Human Behaviour with Diffusion Models, 2023.

[56] X. B. Peng, Erwin Coumans, Tingnan Zhang, T. Lee, Jie Tan, and S. Levine. Learning Agile Robotic Locomotion Skills by Imitating Animals. *ArXiv*, abs/2004.00784, 2020. doi: 10.15607/rss.2020.xvi.064.

[57] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[58] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)*, 40(4):1–20, 2021.

[59] Carole G. Prevost, Andre Desbiens, and Eric Gagnon. Extended Kalman Filter for State Estimation and Trajectory Prediction of a Moving Object Detected by an Unmanned Aerial Vehicle. In *2007 American Control Conference*, pages 1805–1810, 2007. doi: 10.1109/ACC.2007.4282823.

[60] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and Koushil Sreenath. Learning Humanoid Locomotion with Transformers. *arXiv preprint arXiv:2303.03381*, 2023.

[61] Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. Humanoid Locomotion as Next Token Prediction. *arXiv preprint arXiv:2402.19469*, 2024.

[62] Alexander Reske, Jan Carius, Yuntao Ma, Farbod Farshidian, and Marco Hutter. Imitation learning from mpc for quadrupedal multi-gait control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5014–5020. IEEE, 2021.

[63] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-Conditioned Imitation Learning using Score-based Diffusion Policies, 2023.

[64] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models, 2022.

[65] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, 2011.

[66] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[67] Dohoon Ryu and Jong Chul Ye. Pyramidal Denoising Diffusion Probabilistic Models, 2022.

[68] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017.

[69] Yecheng Shao, Yongbin Jin, Xianwei Liu, Weiyan He, Hongtao Wang, and Wei Yang. Learning free gait transition for quadruped robots via phase-guided controller. *IEEE Robotics and Automation Letters*, 2021.

[70] Yecheng Shao, Yongbin Jin, Xianwei Liu, Weiyan He, Hongtao Wang, and Wei Yang. Learning Free Gait Transition for Quadruped Robots Via Phase-Guided Controller. *IEEE Robotics and Automation Letters*, 7:1230–1237, 2022. doi: 10.1109/LRA.2021.3136645.

[71] Laura Smith, J Chase Kew, Tianyu Li, Linda Luu, Xue Bin Peng, Sehoon Ha, Jie Tan, and Sergey Levine. Learning and adapting agile locomotion skills by transferring experience. *arXiv preprint arXiv:2304.09834*, 2023.

[72] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models, 2022.

[73] Zhi Su, Xiaoyu Huang, Daniel Ordoñez-Apraez, Yunfei Li, Zhongyu Li, Qiayuan Liao, Giulio Turrisi, Massimiliano Pontil, Claudio Semini, Yi Wu, et al. Leveraging Symmetry in RL-based Legged Locomotion Control. *arXiv preprint arXiv:2403.17320*, 2024.

[74] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and A. Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 843–852, 2017. doi: 10.1109/ICCV.2017.97.

[75] The Linux Foundation. Open neural network exchange. https://onnx.ai/, 2024. Accessed: 2024-01-29.

[76] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[77] Francecso Vezzi, Jiatao Ding, Antonin Raffin, Jens Kober, and Cosimo Della Santina. Two-Stage Learning of Highly Dynamic Motions with Rigid and Articulated Soft Quadrupeds. *arXiv preprint arXiv:2309.09682*, 2023.

[78] Eric Vollenweider, Marko Bjelonic, Victor Klemm, Nikita Rudin, Joonho Lee, and Marco Hutter. Advanced skills through multiple adversarial motion priors in reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5120–5126. IEEE, 2023.

[79] Bingzheng Wang, Guoqiang Wu, Teng Pang, Yan Zhang, and Yilong Yin. DiffAIL: Diffusion Adversarial Imitation Learning, 2023.

[80] Hsiang-Chun Wang, Shang-Fu Chen, and Shao-Hua Sun. Diffusion Model-Augmented Behavioral Cloning. *arXiv preprint arXiv:2302.13335*, 2023.

[81] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion Policies as an Expressive Policy Class for Offline Reinforcement Learning, 2023.

[82] Eric R Westervelt, Jessy W Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback control of dynamic bipedal robot locomotion*. CRC press, 2018.

[83] Jinze Wu, Yufei Xue, and Chenkun Qi. Learning multiple gaits within latent space for quadruped robots. *arXiv preprint arXiv:2308.03014*, 2023.

[84] Zhou Xian, Nikolaos Gkanatsios, Theophile Gervet, Tsung-Wei Ke, and Katerina Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *Conference on Robot Learning*, pages 2323–2339. PMLR, 2023.

[85] Wei Xiao, Tsun-Hsuan Wang, Chuang Gan, and Daniela Rus. SafeDiffuser: Safe Planning with Diffusion Probabilistic Models. *arXiv preprint arXiv:2306.00148*, 2023.

[86] Haoran Xu, Li Jiang, Li Jianxiong, and Xianyuan Zhan. A policy-guided imitation approach for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:4085–4098, 2022.

[87] Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy Representation via Diffusion Probability Model for Reinforcement Learning, 2023.

[88] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

[89] Ruihan Yang, Zhuoqun Chen, Jianhan Ma, Chongyi Zheng, Yiyu Chen, Quan Nguyen, and Xiaolong Wang. Generalized animal imitator: Agile locomotion with versatile motion prior. *arXiv preprint arXiv:2310.01408*, 2023.

[90] Takuma Yoneda, Luzhe Sun, Bradly Stadie, Ge Yang, and Matthew Walter. To the Noise and Back: Diffusion for Shared Autonomy. *arXiv preprint arXiv:2302.12244*, 2023.

[91] Fangzhou Yu, Ryan Batke, Jeremy Dao, Jonathan Hurst,

Kevin Green, and Alan Fern. Dynamic bipedal turning through sim-to-real reinforcement learning. In *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, pages 903–910. IEEE, 2022.

[92] Chong Zhang, Jiapeng Sheng, Tingguang Li, He Zhang, Cheng Zhou, Qingxu Zhu, Rui Zhao, Yizheng Zhang, and Lei Han. Learning Highly Dynamic Behaviors for Quadrupedal Robots. *arXiv preprint arXiv:2402.13473*, 2024.

[93] Ziwen Zhuang, Zipeng Fu, Jianren Wang, Christopher Atkeson, Soeren Schwertfeger, Chelsea Finn, and Hang Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023.

APPENDIX A
MORE RESULTS ON SKILL TRANSITIONING

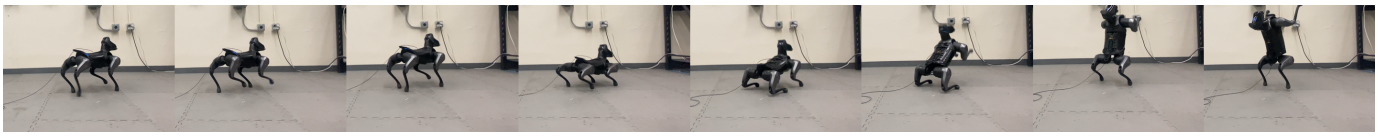


Fig. 8: Skill Transitioning: Pace to Stand

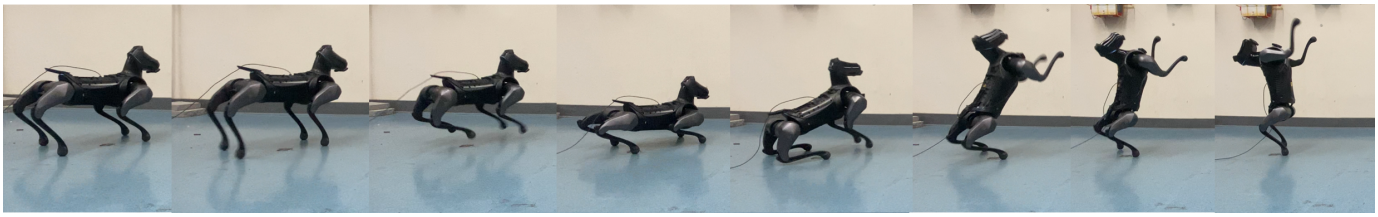


Fig. 9: Skill Transitioning: Hop to Stand on Bare Floor

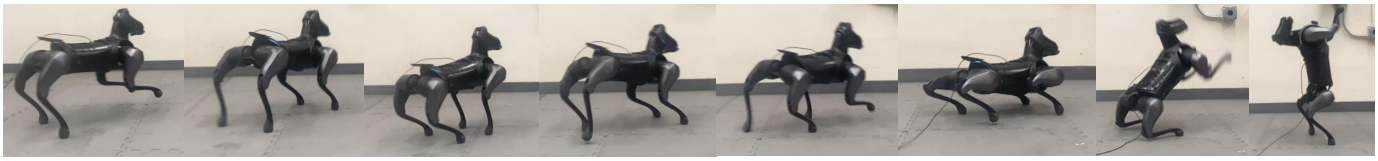


Fig. 10: Skill Transitioning: Bounce to Stand with Emergent Intermediate Pacing Skill

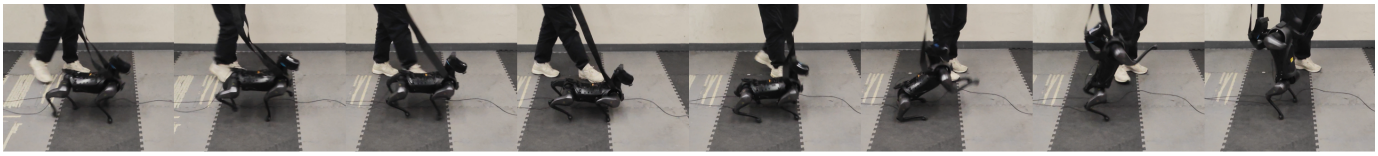


Fig. 11: Skill Transitioning: Trot to Stand

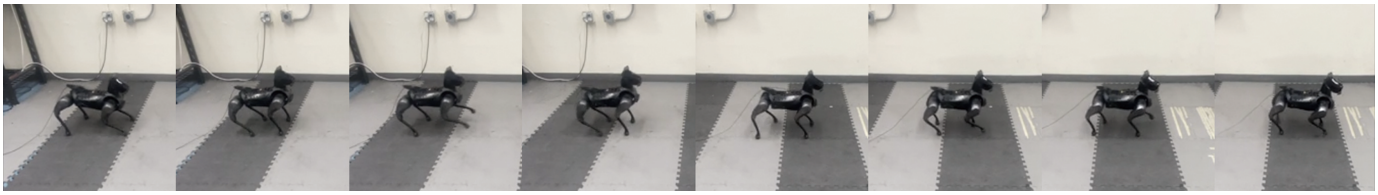


Fig. 12: Skill Transitioning: Bounce to Pace

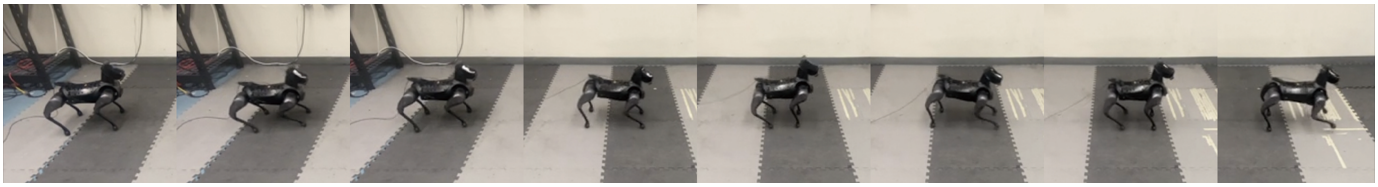


Fig. 13: Skill Transitioning: Hop to Bounce

Fig. 14: Skill Transitioning: Hop to Pace

APPENDIX B
MODEL ARCHITECTURE AND HYPERPARAMETERS

Here, we explain the details of the diffusion model's architecture, dataflow, and transformer backboone.

A. Receding Horizon Control

Learning sequences of actions instead of single-step action in training helps improving temporal consistency of the policy. However, in dynamic systems such as legged robots, error accumulates significantly after a short horizon of planned steps, and the predicted actions further ahead may no longer be useful for control. Therefore, we adopt the Receding Horizon Control (RHC) manner, where DiffuseLoco policy generates n steps of actions, but only executes the *very first step* of actions. This is in contrast to previous work which infers a sequence of actions at a lower frequency and using interpolation to get a high-frequency action [31, 41]. Such a setup allows us to replan rapidly with fast-changing states of the robot while keeping future steps in account. As we evaluated in Sec. VII-B, using RHC is critical in improving smoothness and consistency of a legged locomotion control policy.

B. Architecture

The DiffuseLoco policy leverages an encoder-decoder transformer DDPM. First, the past robot's past I/O trajectory $(\mathbf{s}_{t-h-1:t-1}, \mathbf{a}_{t-h-2:t-2})$ and given goal sequence $\mathbf{g}_{t-h-1:t-1}$ are transformed into separate I/O embedding and goal embedding by two 2-layer MLP encoders, respectively. Then, we sample noise $\epsilon(k)$ for diffusion time step k with the DDPM scheduler and add to the ground truth action $\mathbf{a}$ from the offline dataset to produce a noisy action $\mathbf{a}^k_{t:t+n} = \mathbf{a}_{t:t+n} + \epsilon_k$. The noisy action $\mathbf{a}^k_{t:t+n}$ is then passed through an MLP layer into action embedding. The noisy action tokens are then passed through 6 Transformer decoder layers, each of which is composed of an 8-head cross-attention layer. Each layer computes the attention weights for the noisy action tokens querying all the state embedding, goal embedding, and the timestep embedding reflecting the current diffusion timestep k. We apply causal attention masks to each of the state embeddings and goal embeddings separately. The predicted noise $\epsilon_\theta(\mathbf{a}_{t-h-2:t+n}, \mathbf{s}_{t-h-1:t-1}, \mathbf{g}_{t-h-1:t-1}, k)$ is then computed by each corresponding output token of the decoder stack. We then supervise the output to predict the added noise with Eqn. 5 to find optimal parameters θ of the denoising model ϵ_θ.

C. Hyperparameters

The hyperparameters are summarized in Table III,

APPENDIX C
MORE ABLATION STUDIES ON DESIGN CHOICES

A. Use of Goal-conditioning

Here, we evaluate the impact of goal-conditioning, hypothesizing enhancements tracking performance and stability. In the **DL w/o Goal** baseline, we do not add the goal-conditioning encoder. Instead, the goal is concatenated with the robot's I/O.

TABLE III: Hyperparameters for DiffuseLoco in the Experiments

	Five-Skill (Sec. V)	Walk (Sec. VI)	Cassie (Sec. V-C)
History Length	8	8	16
Predict Length	4	4	4
Token Dim	256	128	256
Attn Drop-out	0.3	0.3	0.3
Learning Rate	1e-4	1e-4	1e-4
Weight Decay	1e-3	1e-3	1e-3
Epochs	100	100	100

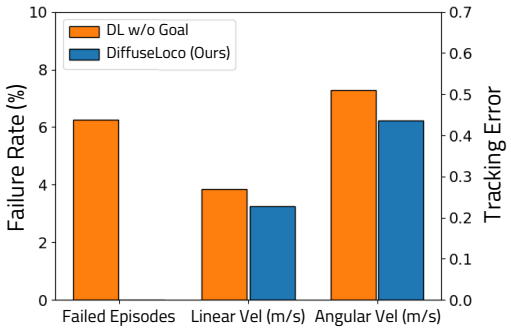


Fig. 15: Comparison of failure rates and tracking errors between **DL w/o Goal** and **DiffuseLoco** (ours) in simulation. The left y-axis is the metric for Failed Episodes. The right y-axis indicates the tracking error for linear velocity and angular velocity.

Considering the noisy base velocity estimations on real robots, we utilize simulation environments with extensive dynamics randomization to perform a large number of trials and get more systematic results. As shown in Fig.15, DiffuseLoco achieves a 15.4% reduction in linear velocity tracking error and a 14.5% reduction in angular velocity error compared to the **DL w/o Goal** baseline. Moreover, over 64 trials with identical commands, **DL w/o Goal** falls over four times, or 6.25% of all trials, whereas DiffuseLoco experiences no failures. This pattern persists in real-world testing, where **DL w/o Goal** fails one trial in a $0.7\mathrm{m/s}$ forward test.

These results underscore the importance of goal-conditioning with distinct attention weights for dynamic system control, revealing that the robot's I/O history and goals, governed by physics and arbitrary objectives respectively, should not be merged into one embedding space.

APPENDIX D
DETAILS IN OFFLINE LOCOMOTION DATASET

Here, we will introduce the details in creating the offline locomotion dataset used in this work. We will explain the state, goal, and action spaces, followed by a brief introduction to the source policies and dynamics randomization used to diversify the data. We collect a total of 4 million data of state-action-goal pairs in the offline dateset for the quadrupedal robot tasks, and 10 million transitions for the bipedal robot tasks.

A. State Space

The state space is the robot's proprioceptive feedback. In the quadrupedal locomotion control case, this consists of the measured motor positions $\mathbf{q}_m$, measured motor velocities $\dot{\mathbf{q}}_m$, base orientation $\mathbf{q}_{\psi,\theta,\phi}$, and base angular velocities $\dot{\mathbf{q}}_{\psi,\theta,\phi}$. Note that we exclude quantities from the estimation of base velocity ($\dot{\mathbf{q}}_{x,y}$) to prevent additional estimation errors.

B. Goal Space

The goal of the locomotion task is the commands given to the policy. For quadrupedal robots, the command includes desired sagittal velocity q_x^d in the range of 0 $\mathrm{m/s}$ to 1 $\mathrm{m/s}$, desired base height from 0.2 m to 0.6 m, and desired turning velocity q_ψ^d from -1 $\mathrm{rad/s}$ to 1 $\mathrm{rad/s}$.

C. Action Space

The action space is the robot's joint-level commands. In this work, we use the desired motor position $\mathbf{q}_m^d$ as the action. This is then used by joint-level PD controllers to compute motor torques τ at a higher frequency.

D. Source Policy

We obtain source policies using three different RL methods. We leverage proximal policy optimization (PPO) [68] to optimize each of the source policies, and we train the policies in simulation (Isaac Gym [66]). We evenly distribute the data generated from each of the source skill-specific policies.

1) Adversarial Motion Prior (AMP): For skills trained with AMP, we provide a reference motion retargeted from motion capture data of a dog [56], and incorporate an GAN-style discriminator to encourage the robot to imitate the reference motion without extended reward engineering. Then, the reward of this method is formulated as motion imitating term (provided by the discriminator [58]) and task term (*e.g.*, tracking error, etc).

2) Central Pattern Generator Guidance (CPG): For skills trained with CPG-guidance, we follow the formulation in [69] and provide nominal reference motions of strictly periodic motions generated by a Central Pattern Generator with phase signals. Specifically, for hopping skill, the phase selections for all legs are 0, and for bouncing skill, the phase selections are 0 for the front legs, and π for the hinder legs. The reward is composed of task term (*e.g.*, velocity tracking error, etc), motion tracking term (*e.g.*, reference motion tracking error, etc), and smoothing terms (*e.g.*, action rate, etc).

3) Symmetry Augmented RL: We train the bipedal locomotion skill for quadrupedal robots following a symmetry-augmented RL policy [73] to achieve a symmetric gait pattern that is crucial for sim-to-real transfer. Specifically, the data collection process is augmented by the addition of symmetric states and actions. The reward includes task term (*e.g.*, velocity tracking error, etc), gait pattern term (*e.g.*, feet clearance height, etc), and smoothing terms (*e.g.*, action rate, etc).

4) Other Methods: Although the source policies used to collect the dataset in this work are all RL-based policies, our framework is general and can include the data generated from model-based optimal controllers (such as from [45]) and others. The requirement is to align the state and action spaces among different source policies, and the frequency of the policy should be kept the same.

E. Dynamics Randomization

In order to diversify the training dataset for DiffuseLoco, we also include the same amount of dynamics randomization [57] during the training of the source policies and the data generation using these policies. Specifically, in each episode in simulation, the dynamics parameters are randomized. These include the motor's PD gains, the mass of the robot's base (up to the weight of the onboard compute), ground frictions, and random changes in base velocity. The randomization ranges are adapted from the source policy's original methods.

APPENDIX E
REAL-TIME INFERENCE ACCELERATION

Although the diffusion model targets real-time use, it cannot meet the real-time targets without further tuning. Compared to previous works using Transformer for locomotion control with 2M parameters [60], our model is 3 times larger in parameter count(6.8M parameters) and needs to be forwarded 10 times in each inference. Thus, an additional effort is needed to accelerate the diffusion on the edge computing device on the robot, such as the setup shown in Fig. 17. In this section, we explore several methods to accelerate the inference process of the diffusion model to enable it to run real-time onboard.

A. Acceleration Framework

Our DiffuseLoco policy has a parameter count of 6.8 million parameters, which exceeds most modern mobile processors' cache capacity. Furthermore, hardware on a typical consumer-grade central processing unit (CPU) is not optimized for the operators used in transformer networks. The graphics processing unit (GPU) is more suitable for computing the high-dimension matrix and vector operations. To ensure the portability of the setup, we use an accessible NVIDIA Mobile GPU as the deployment platform. For real-time deployment, an acceleration pipeline is built in the DiffuseLoco framework to convert and optimize our model towards the target compute platforms. The operators of the model are first extracted with ONNX [75]. Then, TensorRT is used to refine the execution graph and compile the resulting execution pipeline onto the target GPU. Through domain-specific architecture optimizations, the operations and memory access patterns are optimized to utilize the full capability of the GPU. With this approach, the speed for each denoising iteration is increased by about 7X compared to the native implementation in PyTorch, and the maximum inference (with 10 denoising iterations) frequency is increased from 17.0 Hz to 116.5 Hz. To showcase the effect of this acceleration approach, we conducted a benchmark on the inference frequency of the policy running on multiple hardware platforms we have access to, shown in Fig. 16.

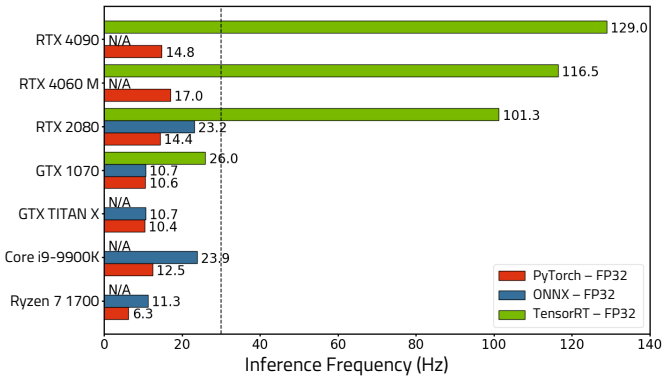


Fig. 16: Benchmark of running our DiffuseLoco policy (6.8M parameters) on different hardware platforms. The dashed line remarks the 30 Hz minimum frequency required to control the robot in real time. We utilize TensorRT to optimize the computation graph and achieve approximately 7 times speedup of inference computation time compared to the naive PyTorch implementation. The N/A entries are due to the lack of software compatibility on the corresponding platform.

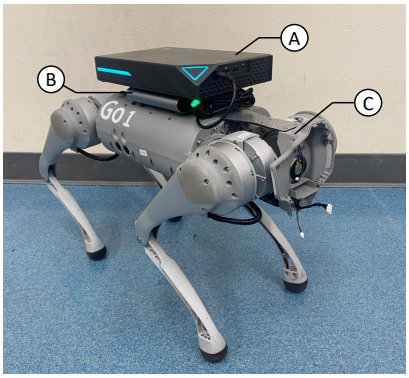


Fig. 17: Onboard compute experiment setup. A: Mini computer with Intel Core i7-13700H and NVIDIA GeForce RTX 4060 Mobile. B: Battery bank. C: Go1 quadrupedal robot. This setup is below the robot's adaptive load capacity and the robot can walk with our policy steadily. With our acceleration framework, we are able to achieve onboard and real-time deployment of our diffusion model.

B. Edge Compute for DiffuseLoco Policy on Robots

With the help of the acceleration framework, the compute platform can be deployed onboard a Go1 quadruped robot. A mini-computer equipped with Intel Core i7-13700H and NVIDIA GeForce RTX 4060 Mobile is attached to the top of the robot, as showcased in Fig. 17. This computer runs DiffuseLoco policy and is powered by a dedicated battery bank, separated from the robot's internal battery. This arrangement is capable of running the policy for up to 90 minutes. The mini-computer connects to the robot via an Ethernet cable to send action for the joint-level PD controls on the robot's computer. We note that all the experiments we present are completed on this edge computing device.

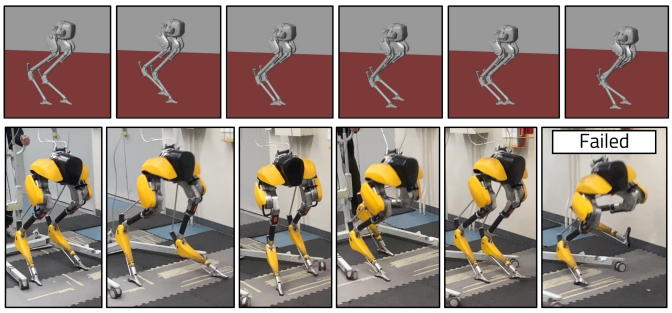


Fig. 18: Snapshots of (top) deploying DiffuseLoco policy on a high-fidelity Simulink simulation and (bottom) the Cassie hardware. Although working in sim-to-sim transfer, the DiffuseLoco policy encounters difficulty in zero-shot transferring on the real Cassie hardware due to a much larger sim-to-real gap. Extending DiffuseLoco to a higher dimensional more complex dynamic system in the real world as an important future work.

APPENDIX F
LIMITATIONS

In the experiments, our DiffuseLoco policy demonstrates good robustness against various ground conditions and small variations in terrain. However, against skill-specific RL policies, the robustness of our multi-skill DiffuseLoco policy is insufficient. For instance, a quadrupedal robot controlled by the DiffuseLoco policy struggles with recovery from large external perturbations, while skill-specific RL policies handle effectively. This limitation becomes more evident where our policy fails to overcome the significant sim-to-real gap on Cassie. Although the diffusion policy for Cassie worked in the high-fidelity simulation (Simulink), the policy only sustained a few steps in hardware experiments before Cassie lost balance and fell down as shown in Fig. 18.