2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

U2E: Uncertainty-Aware Modeling and Uncertainty-Guided Exploration with Deep Ensemble for Quadrupedal Robot

Zitong Bai, Yince Gao, Naiyuan Liu, Yiming Huang, Xiaolong Yu and Wei Wang

Abstract— **Reinforcement learning has facilitated agile locomotion in quadrupedal robots. However, most works remain highly dependent on the accuracy of simulation models in describing real-world robot dynamics. Consequently, policy transfer from simulation to hardware is still hindered by the well-known sim-to-real gap, which typically arises from modeling errors and the challenges of efficiently obtaining informative data in large state-action spaces. To address these challenges, this work proposes an innovative framework *U2E* that integrates *Uncertainty-aware actuator modeling* with an *Uncertainty-guided Exploration policy*. The actuator model leverages a deep ensemble of neural networks to provide both precise predictions and uncertainty estimates, allowing for the assessment of model confidence and the identification of regions with inadequate data coverage. The exploration strategy then actively guides data collection to autonomously acquire informative real-world samples and refine actuator models, thereby enhancing compensation for simulation discrepancies. Experiments on the quadrupedal locomotion tasks, including jumping and trajectory tracking, demonstrate that our approach reduces the sim-to-real gap and improves performance without the dependence on manually designed trajectories.**

I. Introduction

Reinforcement learning (RL) has achieved remarkable progress in legged robots, powering agile locomotion on quadrupedal and humanoid platforms. Most approaches follow a simulation-to-real (sim-to-real) paradigm, where policies are trained in physics simulators before deployment on hardware. However, discrepancies between simulated and real-world dynamics remain a significant challenge, including inaccuracies in friction, contact mechanics, and actuator characteristics. Policies may exploit these discrepancies, resulting in suboptimal or unsafe behaviors when transferred to real robots.

Domain randomization (DR) is a widely used strategy to mitigate the sim-to-real gap by varying physical parameters during training, thereby exposing policies to diverse dynamics [1]. Its simplicity has contributed to its broad adoption [2], [3]. However, DR typically requires extensive manual tuning of parameter ranges. Some works attempt to adapt the randomization during the learning process [4], [5], yet these approaches do not fundamentally resolve model inaccuracies. As a result, DR often sacrifices performance for robustness, producing overly conservative policies when deployed on real hardware [6].

An alternative approach is system identification (SysID), which aims to estimate the dynamics model or parameters from pre-collected data to improve simulation fidelity. Classical methods formulate rigid-body or actuator dynamics analytically and identify physical parameters through optimization techniques such as least squares [7], [8]. These approaches are well established in robotics, but rely on carefully designed excitation trajectories and struggle to capture certain complex nonlinearities, such as motor friction or joint elasticity. More recent methods employ a data-driven approach, viewing the system identification problem as a regression problem and using supervised learning to fit dynamics models by black-box models such as Gaussian processes regression (GPR) [9] or neural network (NN) [10]. These methods improve expressiveness and can capture complex dynamics beyond analytical models.

This work focuses on data-driven SysID to mitigate the sim-to-real gap in quadrupedal robot RL. Directly applying NN often requires large datasets and extensive manual tuning. Yet, data collected from random or expert-designed trajectories may fail to sufficiently excite all dynamic modes, resulting in suboptimal models. Moreover, most existing methods lack uncertainty quantification, making it difficult to assess whether the learned model adequately captures the true system dynamics.

To address these challenges, we propose **U2E**, an innovative framework that integrates **U**ncertainty-aware actuator modeling with **U**ncertainty-guided **E**xploration, enabling accurate system identification and efficient data collection for sim-to-real transfer in quadrupedal robots. We adopt probabilistic regression, where the networks are used to model one or several parameters of the distribution [10]–[12]. Specifically, the quadrupedal robot's actuator dynamics are modeled as a Gaussian distribution, with its mean and covariance parameterized by a neural network conditioned on the current state and control input. This probabilistic

This work was supported in part by the National Natural Science Foundation of China (62373019), the Hangzhou High-tech Zone (Binjiang) Platform-based Open Competition Project (2025JBGS-PT006), and the Fundamental Research Funds for the Central Universities (501XSKC2025103001).

Z. Bai, Y. Gao, and N. Liu are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: {baizitong, yince, archliu}@buaa.edu.cn).

Y. Huang is with the School of Mechanical Engineering, Zhejiang Sci-Tech University, Hangzhou 310018, China (e-mail: 2024220503053@mails.zstu.edu.cn).

X. Yu is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China, and also with Zhejiang Key Laboratory of Industrial Big Data and Robot Intelligent Systems, Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China (e-mail: yxiaolong@buaa.edu.cn).

W. Wang is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China, with the Hangzhou Innovation Institute, Beihang University, Hangzhou 310051, China, and also with Zhongguancun Laboratory, Beijing 100194, China (e-mail: w.wang@buaa.edu.cn).

Corresponding author: Wei Wang.

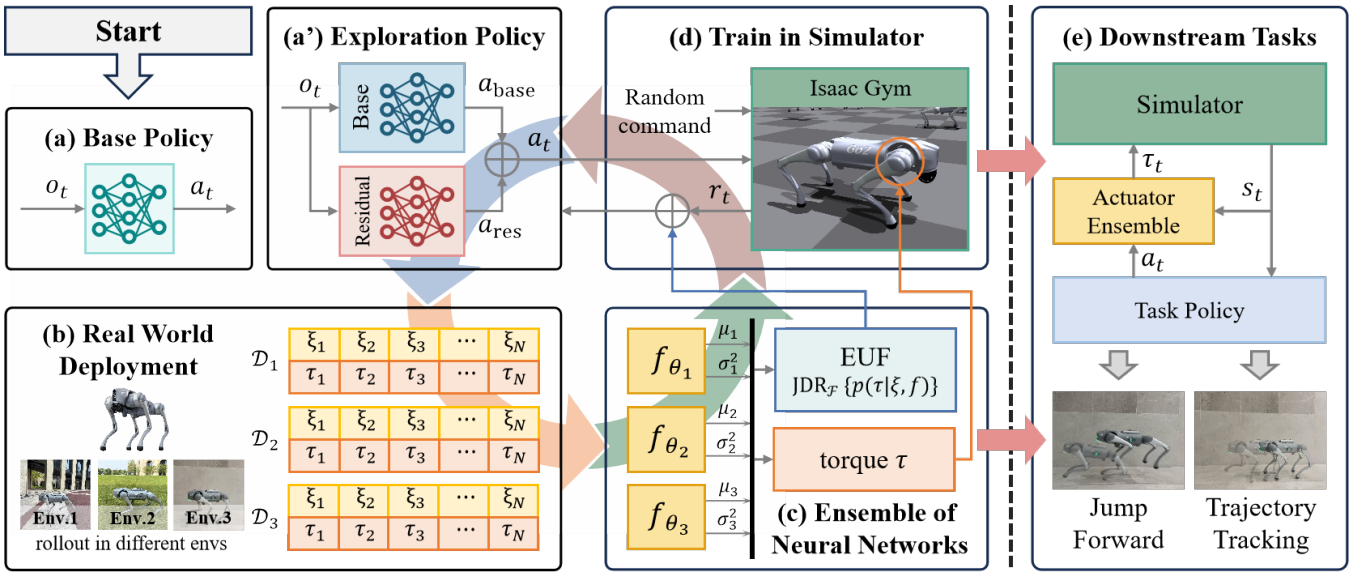


Fig. 1. The overall framework of our approach. (a) A base policy is trained in simulation with ideal actuators. (b) An initial dataset is collected on the real robot using the base policy. (c) A deep ensemble model is then trained to identify the actuator dynamics and quantify model uncertainty. (d) Integrating the trained model into simulator, we train an exploration policy (a') guided by the uncertainty quantification, and deploy it on the robot to gather more informative data. The actuator model is retrained with the newly collected data, and the exploration-retraining cycle is repeated until the model converges. (e) Finally, the actuator ensemble is integrated into simulation to facilitate downstream tasks' training.

formulation provides a flexible way to capture uncertainty in the model's predictions.

Among different types of uncertainty in SysID, epistemic uncertainty reflects the model's lack of knowledge about unexplored regions of the data space [13], which can provide guidance in the collection of informative data. Bayesian inference is a principled approach to capture epistemic uncertainty, which theoretically provides a complete characterization of uncertainty by inferring a posterior distribution over model parameters given the data [14]. However, exact Bayesian inference is often intractable, necessitating approximations such as Monte Carlo method or variational inference [15]. In contrast, deep ensembles offer an even simpler yet highly effective alternative: they require no additional hyperparameter tuning beyond the base model, scale naturally with modern training pipelines, and have been shown to deliver robust and well-calibrated uncertainty estimates [11]. In this work, a deep ensemble of NNs is employed to model the actuator dynamics, allowing uncertainty estimation and laying the foundation for leveraging uncertainty to guide exploration for informative data.

Our work is inspired by [16], which focuses on actively exploring the state space to improve reinforcement learning in simple simulated environments. However, their approach does not address the sim-to-real gap, and we concentrate on efficiently exploring the real world to collect informative data for system identification.

Our main contributions are summarized as follows.

- We introduce **U2E**, a framework that bridges the sim-to-real gap by combining deep-ensemble-based actuator modeling with uncertainty-guided exploration, enabling accurate system identification and efficient real-world data collection for quadrupedal robots.
- Through extensive real-world experiments across a diverse set of representative locomotion tasks on the quadrupedal platform, we show that **U2E** not only reduces modeling errors but also provides a more reliable basis for downstream task training, offering significant practical value for quadrupedal robot learning.

II. Problem Formulation

The goal of this work is to mitigate the sim-to-real gap in quadrupedal robot RL by exploring the real world to collect informative data for accurately identifying actuator dynamics.

We focus on SysID for actuator rather than the full system dynamics because inaccuracies in actuator modeling have a significant impact on robot performance and are especially difficult to capture. In contrast, other components, such as inertial parameters, can often be reliably estimated with existing techniques [17]. A similar motivation underlies the approach in [2], where an MLP was specifically trained to approximate actuator behavior. Building on this insight, our work concentrates on identifying actuator dynamics to improve sim-to-real transfer.

The problem is formalized in four parts: (i) the RL background and challenges arising from model inaccuracies, (ii) uncertainty-aware actuator modeling, (iii) uncertainty-guided exploration for informative trajectory collection, and (iv) leveraging the updated actuator ensemble in downstream RL to improve sim-to-real transfer.

A. Reinforcement Learning

As the full physical states of practical robotic systems are not always directly observable, the considered RL problem

is formulated as a Partially Observable Markov Decision Process (POMDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \gamma, \mathcal{P}, \pi) \tag{1}$$

where $\mathcal{S}$ denotes the state space, $\mathcal{O}$ the observation space, $\mathcal{A}$ the action space, $\mathcal{R}$ the reward function, γ the discount factor, $\mathcal{P}(s_{t+1} \mid s_t, a_t)$ the state transition, and $\pi(a_t \mid o_t)$ the policy. At each step t, the agent receives an observation $o_t \in \mathcal{O}$ rather than the full state s_t, selects an action a_t according to its policy $\pi(a \mid o_t)$, and receives reward r_t. The objective of RL is to learn a policy π that maximizes the expected discounted return:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]. \tag{2}$$

The prevailing pipeline for quadrupedal robot RL involves training policies in simulator, where the state transition $\mathcal{P}$ is governed by the physics engine. However, unmodeled or inaccurate actuator dynamics introduce errors in $\mathcal{P}$, which degrade policy performance and amplify the sim-to-real gap. In the sim-to-real setting considered in this work, the transition can be more explicitly written as $\mathcal{P}(s' \mid s, a, f_\theta)$, where f_θ denotes the actuator model, to highlight the impact of actuator modeling on the simulated dynamics.

B. Uncertainty-Aware Actuator Modeling

Analytical modeling of actuator dynamics is often difficult due to complex nonlinearities and numerous parameters, as highlighted by Gehring et al. [18]. This motivates a data-driven approach, where a key challenge is ensuring that collected data sufficiently excites the system. Deterministic regression alone cannot reliably indicate whether the data covers the relevant dynamics, leading to potential bias in out-of-distribution (OOD) regions. To address this, we adopt a probabilistic formulation that outputs both predictions and uncertainty estimates. In particular, epistemic uncertainty quantifies the model's lack of knowledge in poorly explored regions and can be used as a reward signal to guide exploration toward informative trajectories.

Let χ_t and ν_t denote the observable states and control input of an actuator at time step t, and τ_t the output torque. The objective is to learn the conditional distribution:

$$p(\tau_t \mid \chi_t, \nu_t), \tag{3}$$

which captures both the deterministic mapping and stochastic variations in actuator dynamics. For practical applications, we adopt a Gaussian formulation:

$$\tau_t \sim \mathcal{N}(\mu_\theta(\chi_t, \nu_t), \sigma_\theta^2(\chi_t, \nu_t)), \tag{4}$$

where μ_θ and σ_θ^2 are parameterized by neural network with parameters θ. For the selection of χ_t and ν_t in the context of quadrupedal robots, we follow [2] and assume that the network can be trained to infer the actuator's internal states (e.g., unmeasured dynamics) from temporal features of the joint tracking error. Specifically, we use the joint position error $\Delta q_t = q_t^{\text{cmd}} - q_t$ and the velocity error $\Delta \dot{q}_t = \dot{q}_t^{\text{cmd}} - \dot{q}_t$, where the superscript cmd denotes the desired command. In RL-based locomotion, the desired joint position is typically provided by the policy output, while the desired joint velocity is commonly set to zero. Thus model input (χ_t, ν_t) can be rearranged as $\xi_t = [\Delta q_t, \Delta \dot{q}_t]$, with the input-output data pair represented as $d_t = (\xi_t, \tau_t)$. The actuator model is then expressed as $\tau_t \sim f_\theta(\xi_t)$, and the space of all possible models is denoted by $\mathcal{F}$.

For systems with homogeneous actuators like quadrupedal robots, we assume that all actuators share the same underlying dynamics. Accordingly, data collected from different actuators are pooled together to train a single dynamics model, which is then applied uniformly across all actuators.

C. Uncertainty-Guided Exploration for Informative Trajectories

To reduce epistemic uncertainty in underexplored regions, we design an exploration policy $\pi_{\text{exp}}(a_t \mid o_t)$ that actively guides the robot to execute informative trajectories. By targeting states and actions where the actuator model is uncertain, the policy autonomously collects diverse hardware data, improving the accuracy and reliability of the learned actuator model and providing better simulation for RL.

D. Downstream RL with Updated Actuator Model

After collecting informative trajectories and updating the probabilistic actuator model, the learned distribution is integrated into the simulator. During the training process of the policy $\pi_{\text{task}}(a_t \mid o_t)$ for downstream tasks, torque at each joint is sampled from the actuator distribution, exposing the RL agent to plausible actuator variations. The refined actuator model thus helps to bridge the sim-to-real gap, leading to improved policy performance when deployed on real hardware.

III. Method

The overall framework is illustrated in Fig. 1. It begins with collecting an initial dataset using a preliminary policy [19] executed on the real robot. Based on this dataset, a deep ensemble model is trained to identify actuator dynamics while quantifying uncertainty. Using the uncertainty, an exploration policy is learned and deployed on the robot to collect informative data, which is then used to update the actuator model again. This iterative process continues until convergence, yielding a refined actuator model that is ultimately integrated into simulation to support the training of downstream tasks. The following sections provide a detailed description of each component.

A. Actuator Modeling with Deep Ensemble

Complex nonlinear systems can be effectively captured by deep state-space models, which combine expressive neural parameterizations with recurrent structures to represent temporal dependencies beyond static mappings. Within this family, we employ long short-term memory (LSTM) networks to model actuator dynamics. LSTMs provide a structured way to encode fading-memory effects and long-term dependencies inherent in actuators, such as frictional hysteresis

and elastic couplings, making them well suited for accurate nonlinear system identification.

We view the torque τ_t in the collected data as a sample from the Gaussian distribution, whose mean $\mu(\xi_t)$ and variance $\sigma^2(\xi_t)$ are approximated with a LSTM-based neural network and vary with ξ_t. The model is trained by minimizing the negative log-likelihood (NLL) loss:

$$\mathcal{L}_{NLL} = \frac{1}{2}\log\sigma_\theta^2(\xi_t) + \frac{(\tau_t - \mu_\theta(\xi_t))^2}{2\sigma_\theta^2(\xi_t)}. \tag{5}$$

However, a single model learned via discriminative training cannot capture epistemic uncertainty [12]. Therefore, we employ an ensemble of models. Given a dataset $\mathcal{D} = \{(\xi_t, \tau_t)\}_{t=1}^N$ containing N samples, we train an ensemble of M models $\{f_{\theta_i}\}_{i=1}^M$ independently on the same dataset $\mathcal{D}$, each with different random initialization and data shuffling. This allows the ensemble members to learn diverse representations and provide robust uncertainty estimates, even though they are trained on identical data.

While this deep ensemble produces multiple means and variances, the key challenge is how to exploit these outputs to drive efficient data collection. This involves, first, designing a principled uncertainty quantification to serve as a reward signal, and second, leveraging it to train an exploration policy that actively acquires informative trajectories. The following subsections detail these two components.

B. Uncertainty Quantification

Building on the ensemble predictions, a principled uncertainty quantification is constructed to guide efficient data collection. Each network in the ensemble outputs both a torque prediction and an associated variance, capturing predictive uncertainty at the single-model level. By aggregating predictions across the ensemble, the variation among networks reflects model disagreement, which can be interpreted as epistemic uncertainty in regions of the data space that are poorly explored [20]. This ensemble-derived uncertainty is then used to define an **Epistemic-Uncertainty Function (EUF)**, serving as a reward signal for training an exploration policy that actively seeks out informative trajectories.

To formalize this intuition, we use information gain (IG) of the ensemble about the true dynamics after observing a new sample $d = (\xi, \tau)$:

$$\mathrm{IG}(d) = D_{KL}(P(\mathcal{F}|d)||P(\mathcal{F}))\,. \tag{6}$$

where $\mathcal{F}$ is the space of all possible models, with a prior distribution $P(f)$ over it. D_{KL} is the Kullback-Leibler (KL) divergence measuring the difference between two distributions. Intuitively, IG quantifies how much the ensemble's belief about the true dynamics changes after observing the new data point d. A higher IG indicates that d is more informative for reducing epistemic uncertainty. The expected IG for policy π can be computed as:

$$\begin{aligned}\mathrm{IG}(\pi) &= \int_\Xi \int_\mathcal{T} \mathrm{IG}(\xi,\tau) p(\xi,\tau|\pi)\mathrm{d}\tau\mathrm{d}\xi \\ &= \mathbb{E}_{f\sim P(\mathcal{F})}\left[\mathbb{E}_{\xi\sim P(\Xi|\pi,f)}\left[u(\xi)\right]\right]\end{aligned} \tag{7}$$

where $u(\xi) = \int_\mathcal{T} \mathrm{IG}(\xi,\tau)p(\tau|\xi)\mathrm{d}\tau$ is defined as the EUF which quantifies the informativeness of a candidate input data ξ. Further expanding the EUF by KL-divergence definition and applying Bayes' theorem, we have:

$$\begin{aligned}u(\xi) &= \int_\mathcal{T}\int_\mathcal{F} p(\tau|\xi)p(f|\xi,\tau)\log\frac{p(f|\xi,\tau)}{p(f)}\mathrm{d}f\mathrm{d}\tau \\ &= -\mathbb{E}_{f\sim P(\mathcal{F})}\left[H(p(\tau|\xi,f))\right] + H(\mathbb{E}_{f\sim P(\mathcal{F})}[p(\tau|\xi,f)])\end{aligned}$$

It turns out that the EUF is the Jensen-Shannon Divergence (JSD) which measures the disagreement among the model ensemble:

$$u(\xi) = \mathrm{JSD}_\mathcal{F}\{p(\tau|\xi,f)\} \tag{8}$$

However, computing the JSD is intractable for Gaussian distributions. Therefore, we resort to the Jensen-Rényi Divergence (JRD) as an alternative metric [16], which replaces the Shannon entropy in JSD with Rényi entropy. Specifically, we choose the JRD with order 2 because it yields a tractable closed-form solution for a mixture of M 1-D Gaussian distributions $\mathcal{N}_i(\mu_i,\sigma_i^2), i = 1,\ldots,M$ [21]:

$$\mathrm{JRD} = -\log\left(\sum_{i,j}^M w_i w_j Z_{ij}\right) + \sum_i^M w_i \log Z_{ii} \tag{9}$$

where $w_i = \frac{1}{M}$ are the mixture weights, and $Z_{ij}(\mu_i,\sigma_i^2,\mu_j,\sigma_j^2)$ is defined as:

$$Z_{ij} = \frac{1}{\sqrt{2\pi(\sigma_i^2+\sigma_j^2)}}\exp\left(-\frac{(\mu_i-\mu_j)^2}{2(\sigma_i^2+\sigma_j^2)}\right) \tag{10}$$

The EUF for a candidate data point ξ is then computed as the JRD of the ensemble's predicted Gaussian distributions:

$$u(\xi) = \mathrm{JRD}_\mathcal{F}\{p(\tau|\xi,f)\} \tag{11}$$

By construction JRD ≥ 0 and equals zero if and only if all component distributions are identical. Higher JRD or EUF indicate states where the ensemble strongly disagrees, implying greater potential to reduce epistemic uncertainty.

C. Exploration Policy Based on EUF

The training framework follows an iterative procedure that alternates between policy optimization, real-world data collection, and actuator network refinement. The overall pipeline is summarized in Fig. 1.

Initialization. A locomotion policy is first trained in simulation without the actuator network. On the one side, this policy serves as a initial controller π_{init} to collect preliminary data on the real robot. On the other side, it provides a base policy π_{base} for subsequent residual policy learning. The policy is deployed on hardware to collect joint-level data, which is then used to train the initial actuator network.

Exploration Policy via Residual RL. With the learned actuator network integrated into the simulation, an exploration policy π_{exp} is trained to collect informative data. Residual reinforcement learning [22] is adopted to ensure that exploration remains within safe and stable behaviors. The executed action is defined as:

$$a_t = \pi_{\text{base}} + \pi_{\text{res}} \tag{12}$$

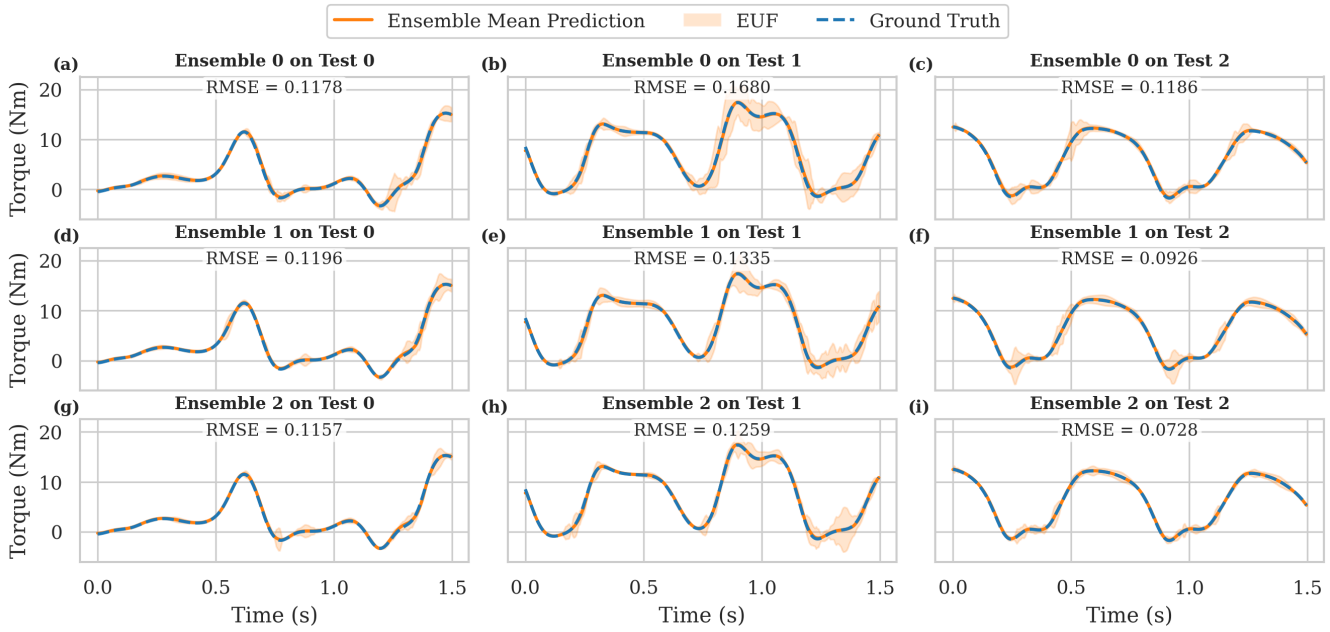


Fig. 2. Cross-evaluation of ensembles trained on different exploration stages (rows) and tested on corresponding datasets (columns). Each cell shows measured torque (blue), ensemble mean (orange), EUF band, and RMSE. Ensembles fit in-distribution and less-explored data well, while larger EUF appears on more-explored test sets, indicating epistemic uncertainty and OOD behavior.

where the frozen base policy provides nominal locomotion, while the residual module, conditioned on the same observation vector o_t, learns corrective actions that probe actuator dynamics. The residual policy can be optimized with prevailing RL algorithm like PPO [23], using a reward shaped by the estimated EUF $u(\xi_t)$ and task reward r_t^{task}:

$$r_t = \alpha \cdot u(\xi_t) + \beta \cdot r_t^{task} \tag{13}$$

The EUF reflects ensemble disagreement and thus the expected information gain of each data point. Safety is enforced through penalties on joint limit violations, torque overuse, and destabilizing behaviors, ensuring that exploration remains bounded within feasible motions.

Iterative Loop. The exploration policy is deployed on hardware to collect new trajectory data, which is then used to retrain the actuator network. The updated network provides a refined EUF landscape, which in turn guides the next round of exploration policy optimization. Through this closed loop, the actuator network progressively improves its accuracy, while the exploration policy becomes increasingly effective at targeting high-uncertainty regions for data acquisition. This iterative data collection loop is terminated when the relative decrease of the EUF on newly collected trajectories falls below a predefined threshold.

Moreover, during policy training in simulation, at each timestep a single actuator model is randomly sampled from the ensemble and used to simulate the actuator dynamics. This stochastic selection exposes the policy to different plausible actuator behaviors, encouraging it to learn corrective actions that are robust to model uncertainty and preventing overfitting to any single model in the ensemble.

IV. Experiment

In this section, we design experiments to systematically evaluate the effectiveness of the proposed framework in modeling actuator dynamics and reducing the sim-to-real gap. Our experiments aim to answer the following key questions:

- **Q1:** Can the proposed actuator modeling accurately capture actuator dynamics, and does the proposed EUF reliably expose regions of epistemic uncertainty?
- **Q2:** Does the full pipeline reduce the sim-to-real gap more effectively than other baselines?

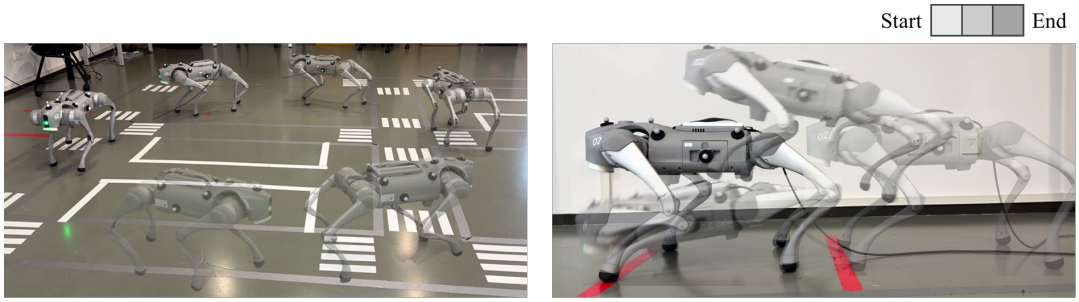
A. Experimental Setup

The settings of quadrupedal robot locomotion tasks for base policy and exploration policy follow [19] with some modifications. The observation consists of:

$$o_t = [\omega_t, g_t, c_t, q_t, \dot{q}_t, a_{t-1}] \tag{14}$$

where $\omega_t \in \mathbb{R}^3$ denotes the base angular velocity, $g_t \in \mathbb{R}^3$ the gravity vector in the base frame, $c_t \in \mathbb{R}^3$ the velocity command, $q_t \in \mathbb{R}^{12}$ and $\dot{q}_t \in \mathbb{R}^{12}$ the joint positions and velocities, and $a_{t-1} \in \mathbb{R}^{12}$ the previous action. The action $a_t \in \mathbb{R}^{12}$ represents the target joint positions. The policy is performed at 50 Hz with control decimation of 4, which means the actuator network is queried at 200 Hz if enabled.

All policies are trained in Isaac Gym [24] on a single NVIDIA A100 GPU. For real-world evaluation, we use a Unitree Go2 robot equipped with 12 motors, each capable of a maximum torque of approximately 45 N · m. The trained policy is exported to ONNX for efficient inference. For convenience, the policy is deployed on a laptop with an



(a) Trajectory Tracking **(b) Forward Jumping**

Fig. 3. Illustration of the two downstream tasks on the Unitree Go2. **(a) Trajectory tracking:** the robot follows a sequence of waypoints defined in a Motion Capture frame. The right part shows the forward jumping task, **(b) Forward jumping:** the robot performs a forward jump, with performance evaluated by the horizontal distance between landing and desired position

Intel Core i7-13700H CPU, and communication with the robot is established via Ethernet. Note that such a high-performance CPU is not strictly necessary; the policy can also run onboard the robot. For the trajectory tracking task, a CHINGMU Motion Capture system is employed to record the robot's ground-truth position during real-world experiments.

B. Validating Actuator Ensembles

To address **Q1**, we try to validate that (i) the deep ensemble of actuator networks accurately fits in-distribution actuator dynamics and (ii) ensemble disagreement (measured by EUF) reliably highlights out-of-distribution (OOD) regions induced by successive exploration iterations.

To capture temporal dependencies, each network in the ensemble employs a recurrent encoder based on LSTM to process the input sequence of joint position and velocity errors, generating a hidden representation that summarizes the actuator dynamics. This representation is then fed into separate multilayer perceptron (MLP) heads: a *mean head* $\mu_\theta(\xi_t)$, which predicts the torque, and a *variance head* $\sigma^2_\theta(\xi_t)$, which estimates the predictive variance. The variance is constrained to be positive using the softplus function $\log(1+\exp(\cdot))$, with a small constant (10^{-6}) added for numerical stability.

The experiment is designed to train 3 independent ensembles; each ensemble contains 6 networks with identical architecture. The three ensembles are trained on datasets collected on the real Unitree Go2 under different exploration stages:

- **Init**: data collected by the initial base policy (zero exploration iterations);
- **Exp-1**: data collected after one iteration of the exploration policy;
- **Exp-2**: data collected after two iterations of the exploration policy.

For each ensemble we reserve an independent test set drawn from the same collection run. Cross-evaluation is conducted by testing each ensemble on all three test sets (Init, Exp-1, Exp-2). Evaluation uses recorded time-series sequences rather than isolated timesteps, such that temporal prediction quality is measured on realistic trajectories. The raw sequence is filtered by a cubic smoothing spline to reduce sensor noise. For each test sequence we compute:

1) Ensemble mean prediction of torque at each timestep;
2) EUF at each timestep, measuring ensemble disagreement (see Sec. III-B);
3) Root-mean-square error (RMSE) between measured torque and predictive mean.

We expect good RMSE on in-distribution evaluations (ensemble trained on the same data) and on test sets from *less-explored* policies, while evaluations on test sets from *more-explored* policies will contain temporal segments with markedly larger EUF values, indicating epistemic uncertainty and OOD behavior. The results are presented in a 3×3 grid illustrated in Fig. 2. Its rows correspond to the ensembles and columns correspond to the test datasets. Each cell shows a representative time window (several seconds) of: measured torque (blue), ensemble mean prediction of torque (orange), and a shaded band of EUF. The RMSE values are annotated above each cell. It can be seen that an ensemble trained on data with exploration iterations k will (i) fit test sets of iterations $\leq k$ with lower RMSE and EUF, and (ii) produce relatively larger EUF on parts of test sequences from iterations $> k$. Such asymmetric cross-behavior validates the EUF's ability to expose epistemic uncertainty and motivates its use for guiding exploration.

C. Evaluation of Sim-to-Real Transfer

To address **Q2**, we compare 3 different methods:

- **Vanilla RL with domain randomization (Vanilla)** where policies are trained in Isaac Gym with broad dynamics randomization but no actuator model;
- **Single-actuator (Single)** where a single actuator network (same architecture as Sec. IV-B) is trained from randomly collected on-robot data and incorporated into simulation during policy training;
- **Ours**, the iterative ensemble pipeline with EUF-driven exploration and residual RL described in Sec. III.

We evaluated these methods on two downstream tasks, trajectory tracking and forward jumping illustrated in Fig. 3, both executed on the Unitree Go2.

TABLE I

PERFORMANCE COMPARISON ON DOWNSTREAM TASKS.

Methods	Trajectory Tracking	Forward Jumping	
	L_{track}	Landing Dist. (cm)	L_{jump}
Vanilla	0.5498	78.4 ± 2.30	0.184
Single	0.4769	53.8 ± 1.79	0.062
Ours	0.2696	59.2 ± 1.64	0.008

1) Trajectory Tracking: In this task, the robot is commanded to follow a sequence of waypoints defined in a Motion Capture (MoCap) frame. Performance is measured by the average tracking error to the reference trajectory:

$$L_{\text{track}} = \frac{1}{T}\sum_{t=1}^{T}\left(\|x_t - x_t^{\text{ref}}\|_2 + \|y_t - y_t^{\text{ref}}\|_2 + \|\theta_t - \theta_t^{\text{ref}}\|_2\right), \tag{15}$$

where (x_t, y_t, θ_t) and $(x_t^{\text{ref}}, y_t^{\text{ref}}, \theta_t^{\text{ref}})$ are the robot's position (m) and yaw angle (rad) at time t and their corresponding reference values, respectively. The robot is initialized at the starting point of the trajectory and the policy is executed for prescribed time to follow the waypoints. The training pipeline for this task is similar to the base policy training in Sec. III, except that the reward function is modified to encourage accurate tracking of the reference trajectory.

2) Forward Jumping: In this task, the robot is tasked with jumping forward to a desired position from a standing start, and the performance is measured by the horizontal distance between the landing and desired position:

$$L_{\text{jump}} = |x_{\text{land}} - x_{\text{desired}}|, \tag{16}$$

where x_{land} and x_{desired} are the horizontal positions (m) of the robot at landing and target, respectively. We set $x_{\text{desired}} = 0.6$ m, approximately the robot's body length. The policy observation extends the base policy inputs with a phase variable $\phi \in [0,1]$ to indicate the jumping stage. Certain reward terms are specifically designed to encourage precise jumps while penalizing unstable landings, as detailed in the appendix. A curriculum strategy is employed, gradually increasing the required jump distance as the policy improves.

Table I summarizes the results across the two tasks. For trajectory tracking, our method achieves the lowest L_{track} on the real robot, and more importantly, the smallest discrepancy between simulation and reality, indicating a clear reduction in the sim-to-real gap. Vanilla RL with DR produces policies that perform well in simulation but degrade on hardware, while the single-actuator baseline partially alleviates this gap but remains limited by insufficient data exploration. The executed trajectory is visualized in Fig. 4.

For forward jumping, a similar trend is observed: vanilla RL tend to underestimate the distance that real robot can jump, resulting in a large sim-to-real mismatch. The single-actuator baseline reduces this discrepancy but still exhibits a noticeable bias. In contrast, our method consistently performs accurate jumping in simulation and reliable execution on hardware, achieving the smallest gap between simulated and actual jump distances. The corresponding result is presented in Fig. 5.

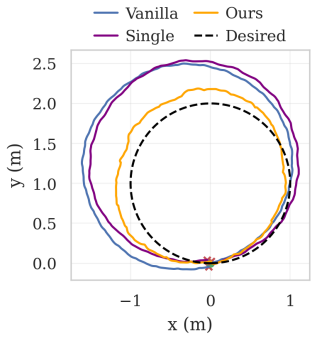


Fig. 4. Real-world experiment of the trajectory tracking task, where the robot is commanded to follow a circular path. Our method achieves the closest match between the executed trajectory (orange) and the reference path (dashed line), outperforming vanilla RL (blue) and the single-actuator (purple) baseline in reducing the sim-to-real gap.

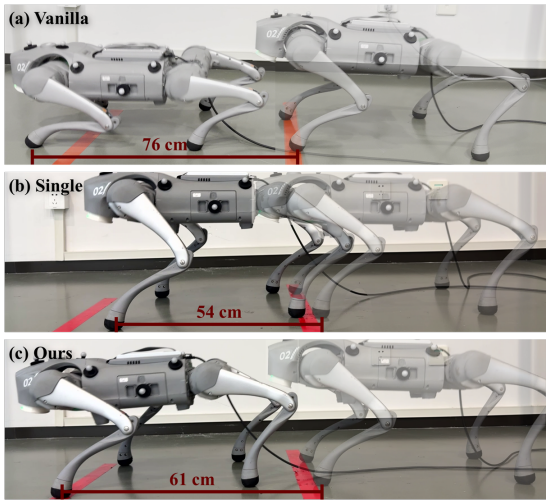


Fig. 5. Real-world experiment of the forward jumping task, where the robot aims to achieve a prescribed target distance. Our method yields the closest match between simulated predictions and actual performance, outperforming vanilla RL and the single-actuator baseline.

These results confirm that actuator ensembles trained with uncertainty-guided exploration provide not only better in-distribution accuracy but also improved coverage of hard-to-model dynamics, enabling downstream policies to transfer more faithfully to real hardware. An accompanying video of the real-world experiments is available at: https://youtu.be/F4rqJZ7nGeY.

V. CONCLUSIONS

In this work, we present **U2E**, a unified framework that addresses the sim-to-real gap in quadrupedal robot learning by jointly leveraging uncertainty-aware modeling and uncertainty-guided exploration. By employing a deep ensemble of neural networks, **U2E** captures both accurate actuator dynamics and the model's uncertainty, enabling the identification of poorly explored regions in the actuator's

data space. Guided by these uncertainty estimates, the exploration strategy efficiently collects informative real-world data, which in turn refines the actuator models and reduces discrepancies between simulation and hardware. Extensive experiments on tasks such as jumping and trajectory tracking validate that **U2E** not only improves the fidelity of system modeling but also provides a stronger foundation for training downstream policies, demonstrating its practical effectiveness for real-world quadrupedal locomotion.

APPENDIX

A. Rewards of Forward Jumping Task

Table II summarizes the rewards used for the forward jumping task. These rewards are designed to elicit and stabilize jumping behaviors. Each reward is computed from the robot's state using the notations defined in Table III and combined into a weighted sum.

TABLE II
REWARD TERMS FOR FORWARD JUMPING

Term	Equation	Weight
landing position	$\mathbf{1}_{\text{landing}} \cdot \rho(\|\Delta\mathbf{p}_{xy}\|, 0.5)$	2.0
success	$\mathbf{1}_{\text{landing}} \cdot \mathbf{1}\{\|\Delta\mathbf{p}_{xy}\| < 0.05\}$	50.0
In air time	$\mathbf{1}_{\text{in air}}$	6.0
xy take off vel.	$\mathbf{1}_{\text{before taking off}} \cdot \rho(\Delta\mathbf{v}_{xy}, 1.0)$	1.0
z take off vel.	$\mathbf{1}_{\text{before taking off}} \cdot \rho(\Delta\mathbf{v}_{z}, 2.5)$	2.0
feet relative z	$\rho(\|\mathbf{p}_z^{\text{feet}} - \mathbf{p}_z^{\text{feet,des}}\|, 0.75)$	2.0
symmetry	$\|q_{\text{left}} - q_{\text{right}}\|_2$	-0.5

TABLE III
NOTATION FOR REWARD FORMULAS

Notation	Description
$\rho(e, \sigma)$	$\exp(-e/\sigma)$ encourages less error e
$\Delta\mathbf{p}_{xy}$	Horizontal distance between base and target
$\mathbf{p}_z^{\text{feet}}$	Feet positions relative to base in z direction
$\mathbf{p}_z^{\text{feet,des}}$	Desired feet positions relative to base in z direction
$\Delta\mathbf{v}_{xy}$	Horizontal take-off velocity error: $\mathbf{v}_{xy}^{\text{base}} - \mathbf{v}_{xy}^{\text{takeoff}}$
$\Delta\mathbf{v}_{z}$	Vertical take-off velocity error: $v_z^{\text{base}} - v_z^{\text{takeoff}}$
q_{left}	Left leg joint positions
q_{right}	Right leg joint positions
$\mathbf{1}_{\text{landing}}$	Indicator function for landing event
$\mathbf{1}_{\text{in air}}$	Indicator function for in-air phase
$\mathbf{1}_{\text{before taking off}}$	Indicator for phase before take-off

REFERENCES

[1] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 23–30.

[2] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[3] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control," *The International Journal of Robotics Research*, vol. 44, no. 5, pp. 840–888, 2025.

[4] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Conference on Robot Learning*. PMLR, 2020, pp. 1162–1176.

[5] F. Muratore, C. Eilers, M. Gienger, and J. Peters, "Data-efficient domain randomization with bayesian optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 911–918, 2021.

[6] N. Fey, G. B. Margolis, M. Peticco, and P. Agrawal, "Bridging the sim-to-real gap for athletic loco-manipulation," *arXiv preprint arXiv:2502.10894*, 2025.

[7] L. Ljung, "System identification," in *Signal Analysis and Prediction*. Springer, 1998, pp. 163–173.

[8] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to non-linear system identification," *International Journal of Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[9] G. Giacomuzzo, R. Carli, D. Romeres, and A. Dalla Libera, "A black-box physics-informed estimator based on gaussian process regression for robot inverse dynamics identification," *IEEE Transactions on Robotics*, 2024.

[10] G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. B. Schön, "Deep networks for system identification: A survey," *Automatica*, vol. 171, p. 111907, 2025.

[11] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[12] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[13] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, 2021.

[14] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[15] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt, "Advances in variational inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 2008–2026, 2018.

[16] P. Shyam, W. Jaśkowski, and F. Gomez, "Model-based active exploration," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5779–5788.

[17] T. Lee, B. D. Lee, and F. C. Park, "Optimal excitation trajectories for mechanical systems identification," *Automatica*, vol. 131, p. 109773, 2021.

[18] C. Gehring, S. Coros, M. Hutter, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, *et al.*, "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics & Automation Magazine*, vol. 23, no. 1, pp. 34–43, 2016.

[19] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 91–100.

[20] D. Pathak, D. Gandhi, and A. Gupta, "Self-supervised exploration via disagreement," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5062–5071.

[21] F. Wang, T. Syeda-Mahmood, B. C. Vemuri, D. Beymer, and A. Rangarajan, "Closed-form Jensen-Renyi divergence for mixture of gaussians and applications to group-wise shape registration," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2009, pp. 648–655.

[22] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

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

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