IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

# GRAM: Generalization in Deep RL With a Robust Adaptation Module

James Queeney, Xiaoyi Cai, Alexander Schperberg, *Member, IEEE*, Radu Corcodel, *Member, IEEE*, Mouhacine Benosman, *Senior Member, IEEE*, and Jonathan P. How, *Fellow, IEEE*

***Abstract*—The reliable deployment of deep reinforcement learning in real-world settings requires the ability to generalize across a variety of conditions, including both in-distribution scenarios seen during training as well as novel out-of-distribution scenarios. In this work, we present a framework for dynamics generalization in deep reinforcement learning that unifies these two distinct types of generalization within a single architecture. We introduce a robust adaptation module that provides a mechanism for identifying and reacting to both in-distribution and out-of-distribution environment dynamics, along with a joint training pipeline that combines the goals of in-distribution adaptation and out-of-distribution robustness. Our algorithm GRAM achieves strong generalization performance across in-distribution and out-of-distribution scenarios upon deployment, which we demonstrate through extensive simulation and hardware locomotion experiments on a quadruped robot.**

***Index Terms*—Reinforcement learning (RL), machine learning for robot control.**

## I. Introduction

DUE to the diverse and uncertain nature of real-world settings, generalization is an important capability for the reliable deployment of data-driven, learning-based frameworks such as deep reinforcement learning (RL). Policies trained with deep RL must be capable of generalizing to a variety of different environment dynamics at deployment time, including both familiar training conditions and novel unseen scenarios, as the complex nature of real-world environments makes it difficult to capture all possible variations during training.

Received 14 August 2025; accepted 20 November 2025. Date of publication 8 December 2025; date of current version 6 January 2026. This article was recommended for publication by Associate Editor S. Ha and Editor J. Kober upon evaluation of the reviewers' comments. The work of James Queeney, Alexander Schperberg, and Radu Corcodel was supported by MERL. The work of Xiaoyi Cai and Jonathan P. How was supported in part by ARL under Grant W911NF-21-2-0150 and in part by ONR under Grant N00014-18-1-2832. *(Mouhacine Benosman and Jonathan P. How contributed equally to this work.) (Corresponding author: James Queeney.)*

James Queeney is with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139 USA, and also with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: queeney@merl.com).

Xiaoyi Cai and Jonathan P. How are with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: xyc@mit.edu; jhow@mit.edu).

Alexander Schperberg and Radu Corcodel are with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139 USA (e-mail: schperberg@merl.com; corcodel@merl.com).

Mouhacine Benosman is with Amazon Robotics, North Reading, MA 01864 USA (e-mail: m_benosman@ieee.org).

Code is publicly available at https://github.com/merlresearch/gram.

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3641155, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3641155

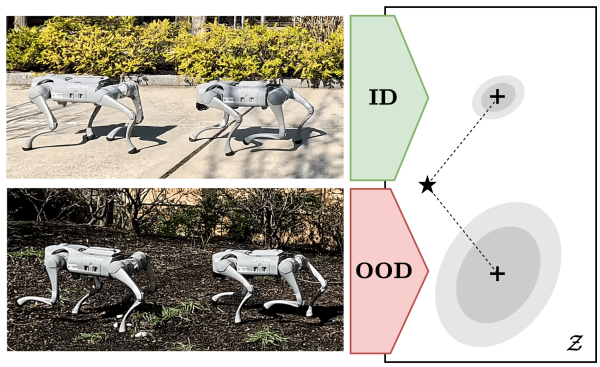


Fig. 1. GRAM generalizes to both ID and OOD environment dynamics at deployment time with a single unified architecture. GRAM introduces a robust adaptation module that quantifies uncertainty about the deployment environment using a recent history of observations, and biases latent context estimates towards a special robust latent feature ($\star$) when uncertainty is high.

For complex robotics applications such as legged locomotion, existing deep RL approaches to zero-shot dynamics generalization have focused on two complementary concepts: adaptation and robustness. Contextual RL techniques [1] such as teacher-student training [2], [3], [4] learn to identify and adapt to the current environment dynamics to achieve the best performance, but this adaptation is only reliable for the range of in-distribution (ID) scenarios seen during training. Robust RL methods [5], [6] such as adversarial training [7], on the other hand, maximize the worst-case performance across a range of possible environment dynamics, providing generalization to out-of-distribution (OOD) scenarios at the cost of conservative performance in ID environments.

This work shows how to extract the benefits of these complementary approaches in a unified framework called GRAM: Generalization in deep RL with a Robust Adaptation Module. GRAM achieves both adaptive ID and robust OOD dynamics generalization at deployment time within a single architecture. Our main contributions are as follows:

1) We introduce a *robust adaptation module* in Section IV that provides a mechanism for identifying both ID and OOD environment dynamics within the same architecture. We extend existing contextual RL approaches by using an epistemic neural network [8] to incorporate a measure of uncertainty about the environment at deployment time, as illustrated in Fig. 1.
2) We propose a joint training pipeline in Section V that combines a teacher-student architecture for learning adaptive

ID performance with adversarial RL training for robust OOD performance, resulting in a single unified policy that can achieve both ID and OOD dynamics generalization.

3) We demonstrate the strong ID and OOD performance of GRAM through comprehensive locomotion experiments on a Unitree Go2 quadruped robot, including both simulation analysis in Section VII and hardware results in Section VIII.

## II. Related Work

Deep RL has demonstrated success on complex robotics tasks in recent years, where policies are trained in simulation and deployed in the real world. In robotic control applications such as legged locomotion, it is critical for these learned policies to achieve zero-shot generalization across a range of scenarios at deployment time [9]. In order to accomplish this, deep RL methods for robotic control have applied techniques from contextual RL [1] and robust RL [5], [6].

Contextual RL focuses on adapting across a range of ID environments seen during training, and many studies have demonstrated the importance of leveraging contextual information in deep RL to improve generalization (e.g., [10]). In robotic control applications, the context is typically available in simulation during training, but unknown at deployment time where it must be inferred from past observations. The most common implementation of contextual RL for legged locomotion leverages privileged context information during training, and applies a teacher-student architecture to train a policy that can be deployed using only the history of past observations [2], [3], [4]. Self-supervised techniques have also been applied to infer context from history in legged locomotion through the use of variational inference [11] and contrastive learning objectives [12], and have been explored more broadly in the deep RL literature [13], [14], [15], [16], [17], [18], [19], [20]. All of these methods are designed to adapt across the range of ID contexts seen during training, but are not specifically trained to handle OOD contexts with different environment dynamics. As a result, their ability to generalize is sensitive to the distribution of ID training contexts, and they may not generalize well to OOD contexts not considered during training.

Robust RL focuses on generalizing to OOD environments at deployment time by maximizing worst-case performance over a set of transition models [5], [6]. Deep RL methods for robotic control most commonly incorporate robustness through the use of adversarial training, which applies worst-case perturbations during training to provide robustness to unknown dynamics or disturbances at deployment time. Adversarial interventions in legged locomotion have primarily applied external forces to the robot [7], [21], while more general adversarial RL methods have considered perturbations to actions [22], observations [23], and transitions [24], [25], [26]. These approaches provide robust generalization to environment dynamics that were not explicitly seen during training, but often sacrifice ID performance to achieve robustness.

In this work, we are interested in both adaptive ID and robust OOD generalization. The possibility of different modes at deployment is related to deep RL methods that train a collection of policies to select from at deployment time [27], [28], [29], [30], [31], [32]. This has been applied to legged locomotion by switching between a task-based policy and a recovery policy in order to guarantee safety [31], as well as selecting from a finite collection of different behaviors based on the deployment environment [30], [32]. Contextual RL can also be viewed as learning a collection of policies for ID adaptation, and our robust adaptation module extends this approach to incorporate a mode for robust OOD generalization as well.

## III. Problem Formulation

*a) Contextual RL:* We model the problem of dynamics generalization in deep RL as a Contextual Markov Decision Process (CMDP) [1]. A CMDP considers a set of contexts $\mathcal{C}$ that define a collection of MDPs $\{\mathcal{M}_c\}_{c\in\mathcal{C}}$. For each $c \in \mathcal{C}$, we have an MDP given by the tuple $\mathcal{M}_c = (\mathcal{S}, \mathcal{A}, p_c, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p_c : \mathcal{S} \times \mathcal{A} \times \mathcal{C} \to P(\mathcal{S})$ is the context-dependent transition model where $P(\mathcal{S})$ represents the space of probability measures over $\mathcal{S}$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0$ is the initial state distribution, and $\gamma$ is the discount rate. We focus on the setting where the transition model $p_c$ depends on the context $c \in \mathcal{C}$ (i.e., varying dynamics), while the reward function $r$ remains the same across contexts (i.e., same task). For a policy $\pi$ and context $c \in \mathcal{C}$, performance is given by the expected total discounted returns

$$J(\pi, c) = \mathop{\mathbb{E}}_{\tau\sim(\pi,c)}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right], \tag{1}$$

where $\tau = (s_0, a_0, s_1, \ldots)$ and $\tau \sim (\pi, c)$ represents a trajectory sampled by deploying the policy $\pi$ in the MDP $\mathcal{M}_c$.

*b) Problem statement:* We assume that the context is available as privileged information during training, but is not available for deployment. This is often the case when a policy is trained in simulation and deployed in the real world. We train across a range of contexts to achieve generalization, but it is typically not possible to consider all possible contexts due to unknown factors at deployment time. Instead, we assume access to a subset of ID training contexts $\mathcal{C}_{\text{ID}} \subset \mathcal{C}$, and we write $c \sim \mathcal{C}_{\text{ID}}$ to represent a sample from a training distribution over ID contexts. We define $\mathcal{C}_{\text{OOD}} = \mathcal{C} \setminus \mathcal{C}_{\text{ID}}$ as the set of OOD contexts that are not seen during training. *Our goal is to train a single policy that performs well in both ID and OOD contexts at deployment time:*

$$\max_{\pi} J(\pi, c) \quad \forall c \in \mathcal{C} = \mathcal{C}_{\text{ID}} \cup \mathcal{C}_{\text{OOD}}. \tag{2}$$

Because we have access to ID contexts during training, we can achieve adaptive ID generalization by directly maximizing performance for each $c \in \mathcal{C}_{\text{ID}}$. On the other hand, we do not have access to OOD contexts during training, so we instead seek to achieve robust OOD generalization for $c \in \mathcal{C}_{\text{OOD}}$ by applying techniques from robust RL. Note that adaptive ID performance and robust OOD performance represent two distinct types of generalization with different objectives, making it challenging to achieve both with a single policy.

*c) Teacher-student training:* The teacher-student approach to generalization in deep RL assumes access to $c_t \in \mathcal{C}_{\text{ID}}$ at

every timestep throughout training, and leverages this privileged context information to train a teacher policy that can adapt to different contexts. The teacher policy applies a context encoder $f : \mathcal{C} \to \mathcal{Z}$ that maps the context to a latent feature $z_t = f(c_t)$, which is then provided as an input to the policy $\pi : \mathcal{S} \times \mathcal{Z} \to P(\mathcal{A})$ and critic $V^\pi : \mathcal{S} \times \mathcal{Z} \to \mathbb{R}$. In this work, we consider the latent feature space $\mathcal{Z} = \mathbb{R}^d$. The context encoding $z_t = f(c_t)$, policy $\pi(a_t \mid s_t, z_t)$, and critic $V^\pi(s_t, z_t)$ are trained to minimize the average actor-critic RL loss over ID contexts given by

$$\mathcal{L}_{\text{RL}} = \mathop{\mathbb{E}}_{c \sim \mathcal{C}_{\text{ID}}} \left[ \mathcal{L}_\pi(c) + \mathcal{L}_V(c) \right], \tag{3}$$

where $\mathcal{L}_\pi(c)$ and $\mathcal{L}_V(c)$ represent the policy loss and critic loss, respectively, of a given RL algorithm for the context $c \sim \mathcal{C}_{\text{ID}}$. In our experiments, we apply Proximal Policy Optimization (PPO) [33] as the RL algorithm.

Note that the teacher policy cannot be applied at deployment time because it requires privileged information about the context $c_t$ in order to compute the latent feature $z_t$. For this reason, RL training is followed by a supervised learning phase where a student policy is trained to imitate the teacher policy using only the recent history of states and actions from the last $H$ timesteps $h_t = (s_{t-H}, a_{t-H}, \ldots, s_t) \in \mathcal{H}$. In particular, an adaptation module $\phi : \mathcal{H} \to \mathcal{Z}$ that maps recent history to a latent feature $\hat{z}_t = \phi(h_t)$ is trained to minimize the loss

$$\mathcal{L}_{\text{enc}} = \mathop{\mathbb{E}}_{c \sim \mathcal{C}_{\text{ID}}} \left[ \mathop{\mathbb{E}}_{\tau \sim (\pi, c)} \left[ \| f(c_t) - \phi(h_t) \|^2 \right] \right], \tag{4}$$

where expectation is taken with respect to trajectories sampled using the student policy in ID contexts $c \sim \mathcal{C}_{\text{ID}}$ during training. This training represents a form of implicit system identification across ID contexts. Using the history encoding $\hat{z}_t = \phi(h_t)$, the policy $\pi(a_t \mid s_t, \hat{z}_t)$ can be applied at deployment time because it does not require privileged information as input.

## IV. Robust Adaptation Module

We build upon the teacher-student architecture for adaptation in deep RL, which has demonstrated strong performance in complex robotics applications such as legged locomotion [2], [3], [4]. However, because this approach focuses only on adaptation across ID contexts seen during training, its OOD generalization capabilities depend strongly on the relationship between $\mathcal{C}_{\text{ID}}$ and $\mathcal{C}_{\text{OOD}}$. The learned adaptation module $\phi$ is trained to identify ID contexts from history, so its output $\hat{z}_t = \phi(h_t)$ and resulting policy $\pi(a_t \mid s_t, \hat{z}_t)$ are only reliable for the distribution of ID history inputs $h_t$ that were observed during training. This leads to strong performance across $c \in \mathcal{C}_{\text{ID}}$, but the standard adaptation module estimate $\hat{z}_t = \phi(h_t)$ may not be useful for achieving generalization in OOD contexts $c \in \mathcal{C}_{\text{OOD}}$ with different environment dynamics not seen during training.

In order to generalize to both ID and OOD contexts at deployment time within the same architecture, we introduce a *robust adaptation module* that explicitly incorporates a mechanism for identifying and reacting to OOD contexts. We accomplish this by quantifying the level of uncertainty present in the latent feature estimate $\hat{z}_t$. We represent the adaptation network $\phi$ as an

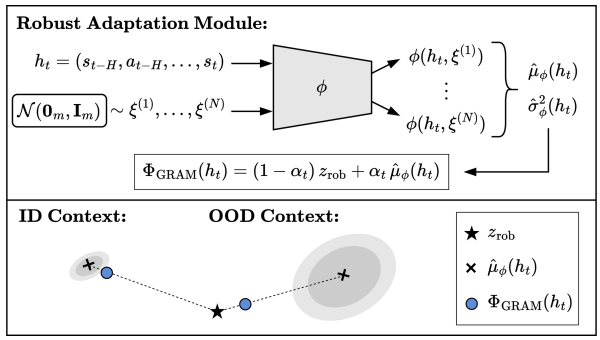


Fig. 2. Robust adaptation module used by GRAM at deployment time. Top: Epistemic neural network $\phi$ outputs a sample mean and variance of latent feature estimates for a history $h_t$, which are used to calculate $\Phi_{\text{GRAM}}$ in (9). Bottom: In ID contexts, variance of latent feature estimates will be low and $\Phi_{\text{GRAM}}$ will be close to the mean estimate. In OOD contexts with different environment dynamics, variance will be high and $\Phi_{\text{GRAM}}$ will output an estimate close to $z_{\text{rob}}$.

epistemic neural network [8] with the form

$$\phi(h_t, \xi) = \phi_{\text{base}}(h_t) + \phi_{\text{epi}}(\tilde{h}_t, \xi), \tag{5}$$

where $\xi \in \mathbb{R}^m$ is a random input that is sampled from a multivariate standard Gaussian distribution $q = \mathcal{N}(\mathbf{0}_m, \mathbf{I}_m)$, and $\tilde{h}_t$ is the concatenation of $h_t$ and the output from the last hidden layer of $\phi_{\text{base}}$ with gradients stopped as in [8]. By incorporating a random input in the second component of (5) (i.e., the "epinet"), this architecture provides a distribution of latent feature estimates for a history input $h_t$ rather than a single point estimate, as illustrated on the top of Fig. 2. For a history $h_t$ and $N$ random input samples $\xi^{(1)}, \ldots, \xi^{(N)} \sim q$, we can write the sample mean and variance of the latent feature estimates as

$$\hat{\mu}_\phi(h_t) = \frac{1}{N} \sum_{i=1}^{N} \phi(h_t, \xi^{(i)}), \tag{6}$$

$$\hat{\sigma}^2_\phi(h_t) = \frac{1}{N-1} \sum_{i=1}^{N} (\phi(h_t, \xi^{(i)}) - \hat{\mu}_\phi(h_t))^2, \tag{7}$$

where all operations are performed per-dimension. We train (5) to minimize the encoder loss across random input samples $\xi \sim q$, resulting in the modified encoder loss given by

$$\mathcal{L}^{\text{GRAM}}_{\text{enc}} = \mathop{\mathbb{E}}_{\xi \sim q} \left[ \mathop{\mathbb{E}}_{c \sim \mathcal{C}_{\text{ID}}} \left[ \mathop{\mathbb{E}}_{\tau \sim (\pi, c)} \left[ \| f(c_t) - \phi(h_t, \xi) \|^2 \right] \right] \right]. \tag{8}$$

By doing so, the total variance of the latent feature estimates $\|\hat{\sigma}_\phi(h_t)\|^2$ will be small over the distribution of history inputs $h_t$ that were seen during training (i.e., trajectories sampled from ID contexts), but not for histories in OOD contexts with different dynamics not seen during training.

Using the epistemic neural network architecture in (5), we introduce a *robust adaptation module* to generalize to both ID and OOD contexts at deployment time. When uncertainty of the latent feature estimates is low, we output the mean estimate $\hat{\mu}_\phi(h_t)$ to allow for adaptation in ID contexts. When uncertainty of the latent feature estimates is high, we bias the mean estimate towards a special robust latent feature $z_{\text{rob}} = \mathbf{0}_d$ to identify that

OOD dynamics have been detected. For a given history $h_t$, our robust adaptation module $\Phi_{\text{GRAM}} : \mathcal{H} \to \mathcal{Z}$ outputs a latent feature $\hat{z}_t = \Phi_{\text{GRAM}}(h_t)$ according to

$$\Phi_{\text{GRAM}}(h_t) = (1 - \alpha_t)\, z_{\text{rob}} + \alpha_t\, \hat{\mu}_\phi(h_t),$$
$$\alpha_t = \exp\left(-\beta(\|\hat{\sigma}_\phi(h_t)\|^2 - \delta)_+\right), \tag{9}$$

where $(\cdot)_+ = \max(\cdot, 0)$ and $\alpha_t \in [0, 1]$, with $\alpha_t \to 1$ when uncertainty is low and $\alpha_t \to 0$ when uncertainty is high. This formulation is motivated by the posterior predictive mean in evidential deep learning [34], where $z_{\text{rob}}$ represents a robust prior, $\hat{\mu}_\phi(h_t)$ is a data-driven estimate, and $\alpha_t$ is the normalized evidence. We include a scale parameter $\beta$ and shift parameter $\delta$ to allow for easy finetuning of $\alpha_t$ based on the output magnitude of the trained epinet in ID contexts, which can be calculated using a validation set of $\|\hat{\sigma}_\phi(h_t)\|^2$ values collected at the end of training.

By defining a special robust latent feature $z_{\text{rob}}$, we incorporate the failure mode of OOD dynamics directly into the existing adaptation framework as a single instance in latent feature space. This allows us to leverage the existing teacher-student training procedure for adaptation in ID contexts, while applying tools from robust RL to encode robust behavior into $\pi(a_t \mid s_t, z_{\text{rob}})$ for OOD generalization *within the same architecture*. Note that the privileged context encoding $z_t = f(c_t)$ begins training near $z_{\text{rob}} = \mathbf{0}_d$ for a randomly initialized context encoder, and the RL loss $\mathcal{L}_{\text{RL}}$ in (3) incentivizes $z_t$ to move away from $z_{\text{rob}}$ as needed to achieve adaptive performance. At deployment time, we bias the latent feature estimate back towards the robust anchor point $z_{\text{rob}}$ if the estimate is unreliable due to OOD environment dynamics. See the bottom of Fig. 2 for an illustration.

## V. Training for Robust Adaptation

Our robust adaptation module provides an intuitive structure for achieving both ID and OOD dynamics generalization within a single architecture. In order to accomplish this goal, we jointly train our policy $\pi(a_t \mid s_t, z_t)$ for adaptive performance in ID environments and robust performance in OOD environments (i.e., when $z_t = z_{\text{rob}}$). We consider parallel training environments as in [35], and assign each training environment to either *adaptive training* or *robust training*. This assignment determines how the latent feature vector is calculated during training, as well as how data collection occurs in the environment. See Algorithm 1 and Fig. 3 for an overview of the joint RL training pipeline, which is followed by a supervised learning phase to train the adaptation network $\phi(h_t, \xi)$. Note that the robust adaptation module $\Phi_{\text{GRAM}}(h_t)$ is only applied at deployment time and not during training.

*a) RL training:* Within each iteration of RL training, all data for a given training environment is collected according to either standard ID data collection or adversarial data collection, as described in the following. This provides temporal consistency when training the policy for adaptive or robust performance, respectively. We alternate these assignments between iterations, which allows full trajectories to contain a mixture of both forms of data. As shown in our experiments, this mixed data collection design provides additional robustness benefits compared to using the same training assignment for entire trajectories.

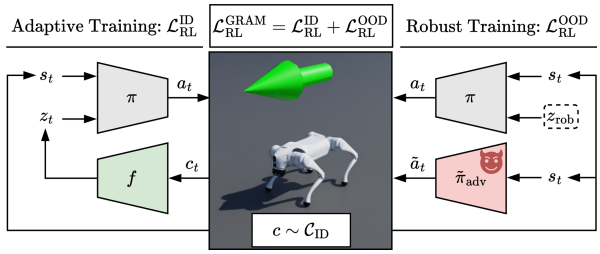


Fig. 3. Joint RL training pipeline used by GRAM, which combines standard ID data collection and adversarial data collection for every RL update. Training environments are assigned to adaptive training or robust training at each iteration, and assignments alternate between iterations. RL training is followed by supervised learning to train the adaptation network $\phi(h_t, \xi)$.

**Algorithm 1:** Joint RL training pipeline.

**for** $K$ updates **do**
  In *adaptive training* environments, collect standard ID data with $\pi(a_t \mid s_t, z_t)$, $z_t = f(c_t)$. Calculate $\mathcal{L}_{\text{RL}}^{\text{ID}}$.
  In *robust training* environments, collect adversarial data with $\pi(a_t \mid s_t, z_{\text{rob}})$ and $\tilde{\pi}_{\text{adv}}(\tilde{a}_t \mid s_t)$. Calculate $\mathcal{L}_{\text{RL}}^{\text{OOD}}$.
  Optimize the RL loss $\mathcal{L}_{\text{RL}}^{\text{GRAM}} = \mathcal{L}_{\text{RL}}^{\text{ID}} + \mathcal{L}_{\text{RL}}^{\text{OOD}}$.
  Alternate training environment assignments.
**end**

For environments assigned to *adaptive training*, we follow the same data collection and training updates as the standard teacher-student architecture. We calculate the privileged latent feature vector as $z_t = f(c_t)$, and perform standard data collection with $\pi(a_t \mid s_t, z_t)$. We update the policy $\pi$, critic $V^\pi$, and context encoder $f$ according to (3), and we denote this loss by $\mathcal{L}_{\text{RL}}^{\text{ID}}$.

For environments assigned to *robust training*, we use the robust latent feature vector $z_{\text{rob}}$ and apply an adversarial RL training pipeline to provide robustness to worst-case environment dynamics at deployment time. We perform data collection with $\pi(a_t \mid s_t, z_{\text{rob}})$, and we introduce an adversary policy $\tilde{\pi}_{\text{adv}} : \mathcal{S} \to P(\tilde{\mathcal{A}}_{\text{adv}})$ that is trained with RL to minimize the returns of $\pi(a_t \mid s_t, z_{\text{rob}})$. In our experiments on a Unitree Go2 quadruped robot, the adversary applies forward and lateral external forces to the robot's body 5% of the time during data collection, where the direction of the external force is learned by the adversary. Using this adversarially collected data, we update the policy $\pi$, critic $V^\pi$, and adversary policy $\tilde{\pi}_{\text{adv}}$ according to (3) with the robust latent feature vector $z_{\text{rob}}$ as input. We denote this loss by $\mathcal{L}_{\text{RL}}^{\text{OOD}}$. Note that adversarial RL provides a method for sampling worst-case trajectories during training, which leads to robust generalization in unseen OOD contexts at deployment time when the robust latent feature $z_{\text{rob}}$ is provided to the policy as input.

By combining *adaptive training* and *robust training*, the RL loss used by GRAM is given by

$$\mathcal{L}_{\text{RL}}^{\text{GRAM}} = \mathcal{L}_{\text{RL}}^{\text{ID}} + \mathcal{L}_{\text{RL}}^{\text{OOD}}. \tag{10}$$

*b) Adaptation module training:* As in the standard teacher-student architecture, RL training is followed by a supervised

TABLE I
IMPLEMENTATION DETAILS

| Hyperparameter | Value |
|---|---|
| Network architectures: MLP hidden layers | |
| Policy ($\pi$), critic ($V^\pi$), adversary policy ($\tilde{\pi}_{\text{adv}}$) | 512, 256, 128 |
| Context encoder ($f$) | 64, 64 |
| Base adaptation network ($\phi_{\text{base}}$) | 512, 256, 128 |
| Epinet ($\phi_{\text{epi}}$) | 16, 16 |
| Robust adaptation module hyperparameters | |
| Context encoder latent feature size ($d$) | 8 |
| History length ($H$) | 16 |
| Epinet random input dimension ($m$) | 8 |
| Epinet random input samples per data point ($N$) | 8 |

learning phase to train the adaptation module for deployment using data collected with the student policy. We accomplish this by training the epistemic neural network architecture $\phi(h_t, \xi)$ in (5) on the modified encoder loss $\mathcal{L}_{\text{enc}}^{\text{GRAM}}$ in (8). Data collection uses the student policy $\pi(a_t \mid s_t, \hat{z}_t)$ with $\hat{z}_t = \hat{\mu}_\phi(h_t)$ or $\hat{z}_t = z_{\text{rob}}$, depending on the training assignment. We do not apply an adversary during the supervised learning phase, as the goal is to train the epinet in (5) to output estimates with low variance in ID contexts.

## VI. GRAM ALGORITHM

Together, the robust adaptation module in (9) and the training procedure in Section V form our algorithm GRAM. GRAM combines standard ID data collection and adversarial data collection during training to optimize the RL loss $\mathcal{L}_{\text{RL}}^{\text{GRAM}}$, followed by a supervised learning phase to optimize the encoder loss $\mathcal{L}_{\text{enc}}^{\text{GRAM}}$. Finally, by applying the robust adaptation module $\Phi_{\text{GRAM}}$ at deployment time, our policy achieves both adaptive ID and robust OOD dynamics generalization within a single unified architecture.

*a) Implementation details:* In our experiments, we apply PPO [33] as the base RL algorithm with the parallel training scheme from [35]. We perform 10,000 updates during RL training, followed by a supervised learning phase where we train the adaptation network for 5,000 updates. We consider feedforward neural networks for all trainable components as in [4], and we follow the design choices proposed in [8] to model the epinet. The adversary policy is trained to apply worst-case external forces that cause abrupt changes to the robot's forward and lateral linear velocity. The direction of the external force is learned by the adversary, and the impact of the force on linear velocity increases from 0.0 to 1.0 meters per second over the course of training. See Table I and our code for additional implementation details.

GRAM uses the epinet architecture to quantify uncertainty *relative* to the uncertainty estimates in ID contexts. We accomplish this by including a scale parameter $\beta$ and shift parameter $\delta$ in the calculation of $\alpha_t$ in (9), which we finetune at the end of training using a validation set of $\|\hat{\sigma}_\phi(h_t)\|^2$ values collected from ID contexts. We set $\delta$ to be the $u_{\text{min}} = 0.90$ quantile of the validation set (note that $\alpha_t = 1.00$ when $\|\hat{\sigma}_\phi(h_t)\|^2 \leq \delta$), and we set $\beta$ such that $\alpha_t = 0.01$ at the $u_{\text{max}} = 0.99$ quantile of the validation set.

TABLE II
ID CONTEXT SET FOR TRAINING

| Parameter | Dim. | Nominal | Range |
|---|---|---|---|
| Friction multiple | 1 | 1.00 | $[0.05, 4.50]$ |
| Added base mass (kg) | 1 | 0.00 | $[-1.00, 3.00]$ |
| Motor strength multiple | 12 | 1.00 | $[0.80, 1.20]$ |
| Joint angle bias (rad) | 12 | 0.00 | $[-0.10, 0.10]$ |

As we show in the following sections, we found that the use of a teacher-student architecture for ID adaptation and adversarial RL for OOD robustness result in strong performance on the quadruped robot locomotion tasks we consider in our experiments. However, note that it is also possible to apply GRAM with different choices of contextual RL methods for ID adaptation and robust RL methods for OOD generalization, which represents an interesting avenue for future work.

## VII. SIMULATION EXPERIMENTS

First, we evaluate the performance of GRAM on realistic simulated locomotion tasks with the Unitree Go2 quadruped robot in Isaac Lab [36]. The goal of the robot is to track a velocity command $\mathbf{v}_t^{\text{cmd}} = [v_x^{\text{cmd}}, v_y^{\text{cmd}}, \omega_z^{\text{cmd}}] \in \mathbb{R}^3$ provided as input, where $v_x^{\text{cmd}}, v_y^{\text{cmd}}$ represent target forward and lateral linear velocities, respectively, and $\omega_z^{\text{cmd}}$ represents a target yaw angular velocity. For each episode, we uniformly sample linear velocity commands between $-1.0$ and $1.0$ meters per second (i.e., $v_x^{\text{cmd}}, v_y^{\text{cmd}} \sim \mathcal{U}([-1.0, 1.0])$) and calculate the yaw angular velocity command $\omega_z^{\text{cmd}}$ throughout the episode based on a target heading direction. The simulated quadruped robot can be seen in Fig. 3, with the velocity command represented by a green arrow.

The policy has access to noisy proprioceptive observations available from standard onboard sensors at every timestep (joint angles, joint velocities, projected gravity, and base angular velocities), and outputs target joint angles $a_t \in \mathbb{R}^{12}$ for each of the robot's 12 degrees of freedom that are converted to torques by a PD controller operating at 200 Hz with proportional gain $K_p = 25$ and derivative gain $K_d = 0.5$. The maximum episode length is 20 seconds with target joint angles processed at 50 Hz, which corresponds to 1,000 timesteps per episode. We consider the reward function used in [4], which includes rewards for tracking the velocity command $\mathbf{v}_t^{\text{cmd}}$ and regularization terms to promote smooth and stable gaits. See [4] for details.

*a) Benchmark comparison:* We compare the performance of GRAM against popular deep RL methods for generalization in legged locomotion: robust RL using adversarial training [7], contextual RL using a teacher-student architecture [2], [3], [4], and domain randomization [37]. Our goal is to investigate the generalization behavior of deep RL methods when faced with deployment environments that differ from those used in training. We conduct this analysis by training all methods on flat, solid ground across the ID context set $\mathcal{C}_{\text{ID}}$ described in Table II, which represents moderate dynamics variations that are commonly used to promote sim-to-real transfer. We evaluate the average velocity-tracking task returns normalized to [0,1] for all algorithms across a variety of deployment settings in Table III. We consider deployment in ID, near OOD, and far

TABLE III
SIMULATION RESULTS

| Algorithm | Average Normalized Task Returns ↑ ID | Near OOD | Far OOD |
|---|---|---|---|
| GRAM | $0.92 \pm 0.01$ | $\mathbf{0.79 \pm 0.02}$ | $\mathbf{0.58 \pm 0.01}$ |
| Robust RL | $0.87 \pm 0.01$ | $0.74 \pm 0.02$ | $\mathbf{0.59 \pm 0.01}$ |
| Contextual RL | $\mathbf{0.94 \pm 0.00}$ | $0.75 \pm 0.02$ | $0.42 \pm 0.03$ |
| Domain Rand. | $0.93 \pm 0.01$ | $0.74 \pm 0.03$ | $0.43 \pm 0.03$ |
| GRAM Ablations | | | |
| No $\Phi_{\text{GRAM}}$ | $0.93 \pm 0.00$ | $\mathbf{0.80 \pm 0.01}$ | $0.55 \pm 0.02$ |
| Separate Data | $0.92 \pm 0.00$ | $0.74 \pm 0.02$ | $0.46 \pm 0.03$ |
| Modular | $\mathbf{0.94 \pm 0.00}$ | $0.74 \pm 0.02$ | $0.41 \pm 0.02$ |

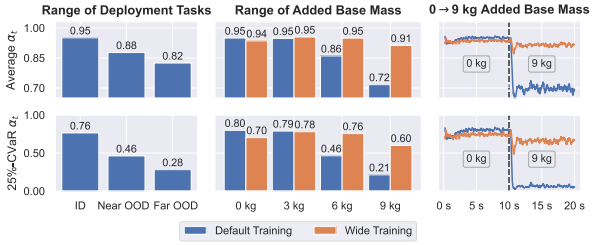


Fig. 4. GRAM average $\alpha_t$ (top) and 25%-CVaR $\alpha_t$ (bottom). Left: Deployment environments from Table III with default ID context set for training. Middle: Range of added base mass deployment scenarios (default vs. wide ID context sets for training). Right: 9 kg base mass added 10 seconds into deployment (default vs. wide ID context sets for training).

OOD contexts. Near OOD contexts consider moderate OOD variations to contexts sampled from $\mathcal{C}_{\text{ID}}$ in Table III: (i) 6 kg added base mass, (ii) 5° incline, (iii) 5 cm rough terrain, and (iv) frozen back hip joint. Far OOD contexts consider more difficult OOD variations: (i) 9 kg added base mass, (ii) 10° incline, (iii) 10 cm rough terrain, and (iv) random frozen joint.

We see in Table III that contextual RL achieves the best ID performance as expected. GRAM achieves performance that is similar to contextual RL and domain randomization in ID contexts, while robust RL is overly conservative in this setting. Conservative ID performance is the main drawback of robust RL methods, which can be meaningful in practical real-world settings where we expect to encounter ID conditions the majority of the time. Importantly, GRAM does not encounter the same ID performance issues as robust RL. Instead, GRAM demonstrates adaptive performance by detecting low uncertainty in ID environments.

As we consider generalization beyond the ID context set seen during training, GRAM automatically identifies and reacts to the increased uncertainty of OOD deployment environments by decreasing the coefficient $\alpha_t$ in (9). See Fig. 4 for the average and 25% conditional value at risk (CVaR) of $\alpha_t$ at deployment time. Note that OOD environments lead to high uncertainty at certain timesteps during deployment (resulting in low 25%-CVaR $\alpha_t$), but for many timesteps the adaptive locomotion skills learned during training can still be applied (resulting in moderate $\alpha_t$ on average). GRAM leverages its adaptation capabilities to maintain strong performance in near OOD contexts without being overly conservative, outperforming all other methods in this setting. As deployment environments become more difficult in the far OOD setting, GRAM achieves

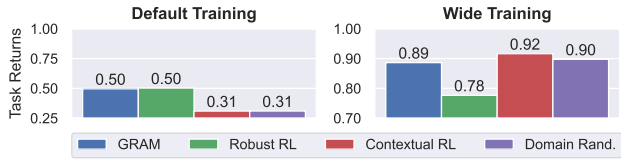


Fig. 5. Average normalized task returns for 9 kg added base mass scenario. Left: Training with default ID context set from Table II. Right: Training across wide added base mass range of $[-1.00, 9.00]$.

robust performance comparable to robust RL. The non-robust baselines contextual RL and domain randomization, on the other hand, experience a more dramatic performance decline. Overall, existing baseline methods demonstrate trade-offs between ID and OOD performance, while GRAM can achieve strong ID and OOD generalization with a single unified policy.

*b) Impact of ID context set for training:* Next, we analyze how the set of ID training contexts $\mathcal{C}_{\text{ID}}$ impacts the performance of each algorithm. In addition to training with the default ID context set in Table II, we also significantly expanded the range of added base masses seen during training to $[-1.00, 9.00]$. For both training setups, we compare performance of the learned policies at deployment time for a 9 kg added base mass in Fig. 5, which represents a far OOD scenario under the default training setup but an ID scenario when trained with the wide mass range.

We see in Fig. 5 that the generalization capabilities of the baseline algorithms strongly depend on the set of ID contexts seen during training. When trained on the default range of added base masses, a 9 kg added base mass is far OOD and robust RL outperforms the non-robust baselines. When trained on a wider range of added base masses that includes 9 kg, contextual RL achieves the best performance by adapting to the ID deployment environment while robust RL becomes overly conservative. Unlike the baseline algorithms, GRAM identifies ID contexts from OOD contexts at deployment time in a way that automatically adjusts for different choices of $\mathcal{C}_{\text{ID}}$. As shown in Fig. 4, GRAM detects low uncertainty (high average and 25%-CVaR $\alpha_t$) across a range of added base masses when trained on the wide ID context set because these scenarios were all seen during training, resulting in adaptive performance similar to contextual RL. When trained on the default ID context set, GRAM decreases $\alpha_t$ for large added base masses to capture OOD environment uncertainty, achieving robust performance similar to robust RL.

*c) Ablation analysis:* Finally, we conduct an ablation study in Table III to analyze the impact of GRAM's unified architecture and joint training pipeline. We compare GRAM to an ablation that applies our joint RL training pipeline without the robust adaptation module, a variant that applies separate ID and adversarial data collection to train a unified policy, and a modular approach that combines separate robust and adaptive policies at deployment time using an uncertainty threshold for switching. We see that these approaches can lead to comparable or slightly improved ID performance, but this comes at the cost of robustness in OOD scenarios. GRAM's single unified policy with its robust adaptation module and joint training pipeline provide both adaptive ID and robust OOD performance, a combination that is difficult to achieve with other approaches.

TABLE IV
HARDWARE RESULTS: MODERATE TASKS

| Scenario | Time-to-Goal (s) ↓ GRAM | Robust RL | Contextual RL |
|---|---|---|---|
| Nominal | $\mathbf{7.95 \pm 0.05}$ | $8.07 \pm 0.18$ | $8.62 \pm 0.04$ |
| Frozen Back Hip | $\mathbf{7.70 \pm 0.16}$ | $8.79 \pm 0.20$ | $9.33 \pm 0.30$ |
| 5° Decline | $\mathbf{6.97 \pm 0.17}$ | $\mathbf{7.05 \pm 0.16}$ | $\mathbf{7.05 \pm 0.15}$ |
| 5° Incline | $\mathbf{10.03 \pm 0.23}$ | $10.92 \pm 0.50$ | $12.51 \pm 0.35$ |
| Average | **8.17 s** | 8.71 s | 9.38 s |
| Success Rate | **20/20** | **20/20** | **20/20** |

TABLE V
HARDWARE RESULTS: DIFFICULT TASKS

| Scenario | Success Rate ↑ GRAM | Robust RL | Contextual RL |
|---|---|---|---|
| 9 kg Payload | **5/5** | **5/5** | 0/5 |
| Slippery | **5/5** | 3/5 | 2/5 |
| Ramp to Foam | **5/5** | 4/5 | 0/5 |
| Total | **15/15** | 12/15 | 2/15 |

## VIII. HARDWARE EXPERIMENTS

In addition to simulation experiments, we evaluate the real-world performance of GRAM by deploying our trained policies onboard a Unitree Go2 quadruped robot. We consider zero-shot sim-to-real transfer of the policies trained in simulation. All policies were trained on flat, solid ground with the moderate variations described by $\mathcal{C}_{\text{ID}}$ in Table II, which differs from prior works on legged locomotion that evaluate generalization after training on a broad set of diverse environments (e.g., [3]). As in simulation, the policy outputs target joint angles at 50 Hz, which are converted to torques by the built-in PD controller at 200 Hz with $K_p = 25$ and $K_d = 0.5$. Observations are available from standard onboard sensors, and all computation is performed onboard.

*a) Controlled experiments:* First, we considered controlled hardware experiments across 7 different scenarios, with the task of forward locomotion for 6 meters from start to goal using a forward velocity command of 1.0 meters per second (i.e., $v_x^{\text{cmd}} = 1.0, v_y^{\text{cmd}} = 0.0$). We measured the success rate and time-to-goal of GRAM, contextual RL, and robust RL across each scenario. We collected 5 trials per scenario for each algorithm, resulting in a total of 105 controlled hardware trials.

In 4 of the 7 scenarios, we considered moderate environment changes where all algorithms successfully reached the goal in all trials. We summarize the results from these scenarios in Table IV. GRAM achieves the fastest time-to-goal in each of these scenarios by balancing adaptive and robust behavior at deployment. Robust RL can be overly conservative, which leads to slower time-to-goal in these moderate scenarios. Contextual RL suffers from a sim-to-real gap due to its lack of robustness when only trained on the moderate variations described by $\mathcal{C}_{\text{ID}}$ in Table II, resulting in the slowest time-to-goal across all scenarios.

In the remaining 3 out of 7 scenarios, we considered more difficult environment variations that differ significantly from those seen during training (see Fig. 6 and Table V). Because these scenarios represent far OOD environment dynamics relative to the set of ID training environments, contextual RL only reached the goal in 2 out of 15 trials. Many of these failures occurred when dynamics changed within a trial, such as when the robot first stepped onto the foam or slippery whiteboard. Robust RL successfully reached the goal in 12 out of 15 trials, with failures caused by the robot becoming stuck due to the conservative wide gait learned from robust training. Unlike the baseline algorithms, GRAM achieves 100% success rate in these difficult scenarios, demonstrating the robust generalization capabilities of our algorithm.

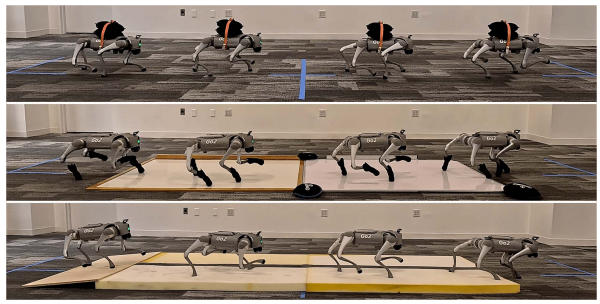

Fig. 6. Difficult controlled hardware experiments where at least one baseline algorithm fails. GRAM achieves 100% success rate on these tasks. From top to bottom: 9 kg Payload, Slippery, Ramp to Foam.

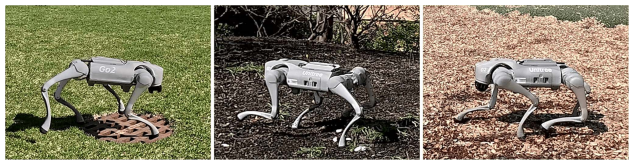

Fig. 7. Qualitative outdoor experiments. GRAM is trained in simulation on flat, solid ground, and achieves robust zero-shot sim-to-real transfer across a variety of outdoor terrains not seen during training. From left to right: grass with storm drain, uneven dirt, wood chips.

*b) Qualitative results:* In addition to controlled trials, we tested the generalization capabilities of GRAM through a variety of qualitative outdoor experiments (see Fig. 7). Despite only training on flat, solid ground in simulation, GRAM successfully transfers zero-shot to a variety of real-world terrains including pavement, grass, uneven dirt, and wood chips. In addition, GRAM is robust to transitions across terrains, inclines, and other unfamiliar surfaces such as the storm drain shown in Fig. 7. These experiments further demonstrate GRAM's ability to achieve robust locomotion in both ID and OOD environments at deployment time. See the Supplementary Material for videos of all experiments.

## IX. CONCLUSION

In this work, we have presented a deep RL framework that achieves both ID and OOD dynamics generalization at deployment time within a single architecture. Our algorithm GRAM leverages a robust adaptation module that allows for adaptation in ID contexts, while also identifying OOD environments with a special robust latent feature $z_{\text{rob}}$. We presented a training pipeline that jointly trains for adaptive ID performance and robust OOD performance, resulting in strong generalization capabilities across a range of simulation and hardware locomotion experiments with a Unitree Go2 quadruped robot. The ability to achieve ID and OOD

generalization within a unified framework is critical for the reliable deployment of deep RL in real-world settings, and GRAM represents a principled step towards this goal.

*a) Limitations and future work:* Because OOD contexts are unknown during training by definition, the OOD generalization of GRAM depends on how well the robust RL training pipeline captures worst-case OOD dynamics, which may lead to failures in extreme OOD scenarios. We applied a standard choice of adversary that worked well in our controlled hardware trials and qualitative outdoor experiments on a quadruped robot, but it would be interesting to combine GRAM with other robust RL techniques and explore its generalization capabilities across a broader range of unstructured environments. Beyond quadruped locomotion, there are also opportunities to apply GRAM across other robotic control applications where generalization is important. For example, GRAM is a flexible deep RL framework that can be extended to other robots and tasks, where domain-specific knowledge can be incorporated through the choice of ID context set for adaptive training and the choice of adversary for robust training.

## Acknowledgment

The work of James Queeney and Mouhacine Benosman was completed prior to joining Amazon Robotics.

## References

[1] A. Hallak, D. D. Castro, and S. Mannor, "Contextual Markov decision processes," 2015, *arXiv:1502.02259*.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Sci. Robot.*, vol. 5, no. 47, 2020, Art. no. eabc5986.

[3] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Proc. Robot. Sci. Syst.*, 2021. doi: 10.15607/RSS.2021.XVII.011

[4] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," *Int. J. Robot. Res.*, vol. 43, no. 4, pp. 572–587, 2024.

[5] A. Nilim and L. E. Ghaoui, "Robust control of Markov decision processes with uncertain transition matrices," *Oper. Res.*, vol. 53, no. 5, pp. 780–798, 2005.

[6] G. N. Iyengar, "Robust dynamic programming," *Math. Oper. Res.*, vol. 30, no. 2, pp. 257–280, 2005.

[7] F. Shi, C. Zhang, T. Miki, J. Lee, M. Hutter, and S. Coros, "Rethinking robustness assessment: Adversarial attacks on learning-based quadrupedal locomotion controllers," in *Proc. Robot. Sci. Syst.*, Jul. 2024. doi: 10.15607/RSS.2024.XX.057

[8] I. Osband et al., "Epistemic neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 2795–2823.

[9] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of zero-shot generalisation in deep reinforcement learning," *J. Artif. Intell. Res.*, vol. 76, pp. 201–264, 2023.

[10] C. Benjamins et al., "Contextualize me the case for context in reinforcement learning," in *Proc. Trans. Mach. Learn. Res.*, 2023. doi: 10.15607/RSS.2024.XX.060

[11] S. Lyu, X. Lang, H. Zhao, H. Zhang, P. Ding, and D. Wang, "RL2AC: Reinforcement learning-based rapid online adaptive control for legged robot robust locomotion," *Robot. Sci. Syst.*, 2024.

[12] J. Long, Z. Wang, Q. Li, L. Cao, J. Gao, and J. Pang, "Hybrid internal model: Learning agile legged locomotion with simulated robot response," in *Proc. Int. Conf. Learn. Representations*, 2024.

[13] J. Yang, B. Petersen, H. Zha, and D. Faissol, "Single episode policy transfer in reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[14] X. Chen et al., "An adaptive deep RL method for non-stationary environments with piecewise stable context," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 35449–35461, 2022.

[15] H. Ren, A. Sootla, T. Jafferjee, J. Shen, J. Wang, and H. B. Ammar, "Reinforcement learning in presence of discrete Markovian context evolution," in *Proc. Int. Conf. Learn. Representations*, 2022.

[16] K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin, "Context-aware dynamics model for generalization in model-based reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5757–5766.

[17] F.-M. Luo, S. Jiang, Y. Yu, Z. Zhang, and Y.-F. Zhang, "Adapt to environment sudden changes by learning a context sensitive policy," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 7637–7646.

[18] A. Nagabandi et al., "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2019.

[19] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5331–5340.

[20] L. Zintgraf et al., "VariBAD: A very good method for bayes-adaptive deep RL via meta-learning," in *Proc. Int. Conf. Learn. Representations*, 2020.

[21] Z. Xiao, X. Zhang, X. Zhou, and Q. Zhang, "PA-LOCO: Learning perturbation-adaptive locomotion for quadruped robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 9110–9115.

[22] C. Tessler, Y. Efroni, and S. Mannor, "Action robust reinforcement learning and applications in continuous control," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6215–6224.

[23] H. Zhang et al., "Robust deep reinforcement learning against adversarial perturbations on state observations," *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 21024–21037.

[24] J. Queeney and M. Benosman, "Risk-averse model uncertainty for distributionally robust safe reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 1659–1680.

[25] J. Zhang et al., "Robust situational reinforcement learning in face of context disturbances," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 41973–41989.

[26] J. Queeney, E. C. Ozcan, I. C. Paschalidis, and C. G. Cassandras, "Optimal transport perturbations for safe reinforcement learning with robustness guarantees," *Trans. Mach. Learn. Res.*, 2024. [Online]. Available: https://openreview.net/forum?id=cgSXpAR4Gl

[27] B. Thananjeyan et al., "Recovery RL: Safe reinforcement learning with learned recovery zones," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 4915–4922, Jul. 2021.

[28] N. C. Wagener, B. Boots, and C.-A. Cheng, "Safe reinforcement learning using advantage-based intervention," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 10630–10640.

[29] A. Ajay, A. Gupta, D. Ghosh, S. Levine, and P. Agrawal, "Distributionally adaptive meta reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 25856–25869.

[30] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Proc. Conf. Robot Learn.*, 2023, pp. 22–31.

[31] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," in *Proc. Robot. Sci. Syst.*, 2024. doi: 10.15607/RSS.2024.XX.059

[32] A. S. Chen et al., "Adapt on-the-go: Behavior modulation for single-life robot deployment," in *Proc. Conf. Lifelong Learn. Agents*, 2025.

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

[34] D. T. Ulmer, C. Hardmeier, and J. Frellsen, "Prior and posterior networks: A survey on evidential deep learning methods for uncertainty estimation," *Trans. Mach. Learn. Res.*, 2023. [Online]. Available: https://openreview.net/forum?id=xqS8k9E75c

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

[36] M. Mittal et al., "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robot. Automat. Lett.*, vol. 8, no. 6, pp. 3740–3747, Jun. 2023.

[37] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 3803–3810.