CARoL: Context-Aware Adaptation for Robot Learning

Zechen Hu, Tong Xu, *Graduate Student Member, IEEE*, Xuesu Xiao, and Xuan Wang, *Member, IEEE*

***Abstract*—Using Reinforcement Learning (RL) to learn new robotic tasks from scratch is often inefficient. Leveraging prior knowledge has the potential to significantly enhance learning efficiency, which, however, raises two critical challenges: how to determine the relevance of existing knowledge and how to adaptively integrate it into learning new task variants with differing dynamics. In this letter, we propose *Context-aware Adaptation for Robot Learning* (CARoL), a novel framework to efficiently learn a similar but distinct new task from prior knowledge. CARoL incorporates context awareness by analyzing state transitions in system dynamics to identify similarities between the new task and prior knowledge. It then utilizes these identified similarities to prioritize and adapt specific knowledge pieces for the new task. Additionally, CARoL has a broad applicability spanning policy-based, value-based, and actor-critic RL algorithms. We validate the efficiency and generalizability of CARoL on both simulated robotic platforms and physical ground vehicles. Simulations include CarRacing and LunarLander environments, where CARoL demonstrates faster convergence and higher rewards when learning policies for new tasks. In real-world experiments, we show that CARoL enables a ground vehicle to quickly and efficiently adapt policies learned in simulation to smoothly traverse real-world off-road terrain.**

***Index Terms*—Reinforcement learning, transfer learning, autonomous agents.**

I. Introduction

In RECENT years, Reinforcement Learning (RL) approaches have achieved remarkable success in advanced robotic control and complex task learning in dynamic environments, enabling applications across various domains [1], [2], [3].

Received 24 May 2025; accepted 16 September 2025. Date of publication 9 October 2025; date of current version 16 October 2025. This article was recommended for publication by Associate Editor S. Ha and Editor J. Kober upon evaluation of the reviewers' comments. This work was supported in part by MICO and RobotiXX labs, in part by MICO through Army Research Office under Grant W911NF2220242, in part by the National Science Foundation under Grant NSF EPCN-2332210, in part by RobotiXX through NSF under Grant 2350352, in part by ARO under Grant W911NF2220242, Grant W911NF2320004, Grant W911NF2420027, and Grant W911NF2520011, in part by Air Force Research Laboratory (AFRL), in part by US Air Forces Central through AFCENT under Grant GS00Q14OADU309, in part by Google DeepMind, in part by Clearpath Robotics, and in part by Raytheon Technologies. *(Corresponding author: Xuan Wang.)*

Zechen Hu and Xuan Wang are with the Department of Electrical and Computer Engineering, College of Engineering and Computing, George Mason University, Fairfax, VA 22030 USA (e-mail: xwang64@gmu.edu).

Tong Xu and Xuesu Xiao are with the Department of Computer Science, College of Engineering and Computing, George Mason University, Fairfax, VA 22030 USA.

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2025.3619779, provided by the authors.

Digital Object Identifier 10.1109/LRA.2025.3619779

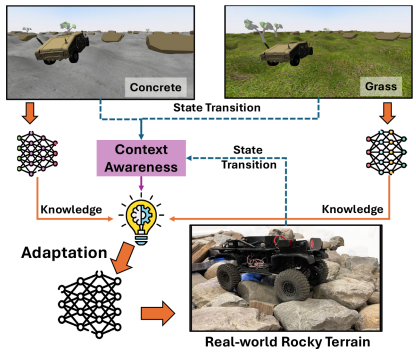

Fig. 1. Autonomous off-road navigation: a vehicle extensively trained in (simulated) grass and concrete environments. However, its next task requires it to navigate on (real-world) unseen rocky terrain. The vehicle must effectively leverage the prior knowledge to solve this new challenge.

Despite these advancements, RL methods are typically computationally demanding, as they rely on repeated trial-and-error exploration to discover high-reward outcomes. Knowledge fusion and adaptation [4] provide promising approaches to address the inefficiency of RL. They leverage knowledge (such as a learned control policy, value function, etc.) from previously explored tasks to accelerate training on new tasks, eliminating the need to train from scratch for every scenario. For example, consider a vehicle navigating highly complex off-road terrain as shown in Fig. 1. Suppose the vehicle has undergone extensive training in several existing environments, it should ideally be capable of adapting to a new type of terrain by utilizing previously learned knowledge.

Our goal is to leverage prior knowledge to enhance learning efficiency of new task variants with differing dynamics. Existing indiscriminate knowledge fusion methods [5] do not consider (and exploit) the relationship between previous and new environments, i.e., being unaware of the environment context, even though many skills are context-specific. Instead, robots should selectively prioritize and utilize knowledge that is relevant to the new task. This poses two key challenges: how to effectively quantify the relevance of existing knowledge and how to adaptively integrate it into the learning process.

In this work, we propose **C***ontext-aware* **A***daptation for* **Ro***bot* **L***earning* (CARoL), a framework that leverages context awareness to enable efficient adaptation to task variants with differing dynamics using prior knowledge. CARoL introduces a novel approach to quantify task similarities by using *state transition* (dynamic awareness) representations as context, which is different from other context-awareness works that commonly use environment [6] or robot [7] representations. The rationale

2377-3766 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: George Mason University. Downloaded on October 26,2025 at 05:42:33 UTC from IEEE Xplore. Restrictions apply.

behind this choice is that the state transition is the core of a Markov Decision Process (MDP) upon which reinforcement learning is established. State transition inherently captures the combined impact of the environment and the robot, thus, is a straightforward and comprehensive measure of task similarity, without the need to carefully choose features from environments or robots. For instance, steep inclines and partially icy surfaces may differ significantly in their environmental characteristics but can produce similar state transitions such as wheel traction loss. By leveraging this shared context, control policies can be adapted in a more general and robust manner. The contributions of CARoL:

- CARoL explicitly leverages state transition to identify task-specific contextual similarity, allowing targeted learning rather than indiscriminate adaptation from prior knowledge.
- CARoL has broad applicability that can be integrated into both policy-based and value-based (or combined actor-critic) RL algorithms to achieve knowledge adaptation.
- Our experimental study demonstrates the effectiveness of CARoL, not only in simulation environments but also through validation on physical off-road navigation tasks.

II. Related Work

1) Learning Adaptation: By leveraging shared knowledge, multi-task learning improves sample efficiency and generalization in robotics and RL. MT-Opt [8] demonstrated how robots can learn a broad spectrum of skills. For quadrotor control tasks, the study [9] leveraged shared physical dynamics to enhance sample efficiency and task performance. Although parameter sharing across tasks can intuitively improve data efficiency, during the training process, gradients from different tasks can interfere negatively with each other. Methods such as policy distillation [10] and actor-mimic [11] train a single policy by using the guidance of several expert teachers [12]. These methods perform well on already encountered tasks or tasks with known structures. However, in most applications, robots often face unknown tasks, making it difficult to ensure the performance of fused control strategies in these unfamiliar scenarios. To guarantee performance on new tasks, knowledge fusion typically requires an additional adaptation process. For instance, the work [13] leveraged lifelong learning techniques to learn a navigation policy complementary to classical motion planners to adapt to new navigation scenarios without forgetting previous ones. The approach in [14] adopted dynamic generation and adjustment of network structures to adapt prior knowledge, leveraging unsupervised knowledge integration for new task adaptation. Lifelong learning is storage-efficient, as it retains only one model that solves all scenarios over the learning lifespan. However, as the model tries to avoid forgetting existing knowledge, it may negatively impact the efficiency of acquiring a new one due to contradicting gradients.

Few-shot reinforcement learning addresses the challenge of rapidly adapting to new tasks with minimal experience by leveraging structured prior knowledge. RL^2 [15] approaches this through a recurrent neural network that learns a learning algorithm itself, enabling fast adaptation to new tasks within the same distribution. PEARL [16] uses probabilistic embeddings to encode task-specific information, allowing an actor-critic agent to quickly adapt to new tasks by inferring latent task representations from limited experience. VariBAD [17] extends this concept by maintaining a belief over possible task parameters and using Bayesian updates to refine task understanding during adaptation. For these methods, the accuracy of latent representation has a critical impact on performance. Meta-learning represents another prominent paradigm within few-shot learning, where algorithms like MAML [18] learn initialization parameters that enable rapid convergence on new tasks through gradient-based adaptation. Modular meta-learning approaches [19] extend this by learning to compose reusable modules for different tasks. However, they suffer from optimization instability due to their reliance on second-order gradients and incur significant computational overhead.

2) Context-Awareness Learning for Robotics: Context-aware learning considers a set of Markov Decision Processes (MDPs) that share the same state and action spaces but differ in transition probabilities and rewards based on contextual variations. Thus, a learning model can be made adaptive, by using context-awareness as an augmented input [20], [21], [22]. For example, the work presented in [20] proposed contextual MDP to model how human decision-making varies depending on their surrounding environment. Context-Aware Policy reuSe (CAPS) [21] leverages contexts as identifiers to determine when and which source policies should be reused. Another notable approach is Successor Features [23], which decomposes value functions into reusable successor features and task-specific reward weights, using the reward weights as context to enable rapid adaptation to new reward structures while preserving learned environmental dynamics. For robotic applications, the method in [22] used a latent context vector to capture robot-specific structural features, enabling the prediction of future states. For quadruped robot control on diverse terrains, the framework in [6] treated environmental encodings as contexts, which are then used as an extra argument to a base policy for quick adaptation. A similar idea has also been extended to other domains, such as bipedal locomotion [24] and in-hand object rotation [25]. For these existing works, context is typically represented as latent encodings of either environments or robots, generated during end-to-end training processes. In contrast, our work directly uses state transitions as contexts. Moreover, while existing methods primarily rely on data-driven mechanisms to map the impact of context on model outputs, our approach explicitly integrates context to guide knowledge adaptation, which offers a more interpretable and structured way to enhance robot learning efficiency.

III. Preliminaries and Problem Formulation

A. Markov Decision Processes and Knowledge

An RL task can be modeled by a Markov Decision Process defined as a tuple $\mathbf{T} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}\}$, where each element represents state space, action space, transition probability measure, and reward function, respectively. The objective of an RL agent is to learn a decision-making strategy that selects actions from the action space $\mathcal{A}$ based on current states in state space $\mathcal{S}$, in order to maximize the cumulative reward $\mathcal{R}$. In this letter, we use a general notation $\mathcal{K}$ to represent the critical knowledge required for such decision-making. Depending on the RL methods employed, $\mathcal{K}$ can take various forms: For policy-based RL, it represents a policy that directly maps states to actions. For value-based RL, it represents the value function that estimates the cost-to-go which informs the choice of action; For actor-critic RL, it combines both the policy (actor) and value function (critic).

Authorized licensed use limited to: George Mason University. Downloaded on October 26,2025 at 05:42:33 UTC from IEEE Xplore. Restrictions apply.

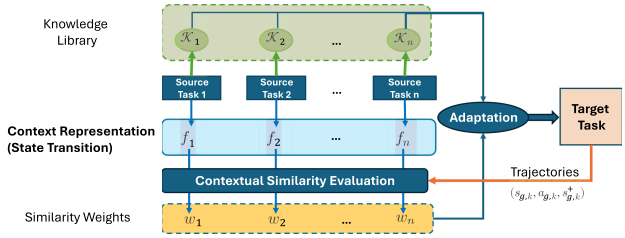

Fig. 2. CARoL consists of three key steps. (i) Acquiring both the context representation and prior knowledge for each source task. (ii) Evaluating the contextual similarities between the target task and source tasks. (iii) Performing contextual knowledge adaptation, which can be applied to policy-based, value-based, or actor-critic RL approaches.

B. Problem Formulation

Consider a set of n source tasks $\mathbb{T} = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_n\}$, where each source task $\mathbf{T}_i = \{\mathcal{S}, \mathcal{A}, \mathcal{P}_i, \mathcal{R}\}$ for $i \in \{1, \ldots, n\}$ is defined by shared state and action spaces $(\mathcal{S}, \mathcal{A})$ and a common reward function $(\mathcal{R})$, but distinct transition probability measures $(\mathcal{P}_i)$. A new unseen target task variant with differing dynamics is represented as $\mathbf{T}_g = \{\mathcal{S}, \mathcal{A}, \mathcal{P}_g, \mathcal{R}\}$. The target task shares the same state space, action space, and reward function as the source tasks, but its transition probability measure $\mathcal{P}_g$ is unknown.

The formulation applies to scenarios with similar objectives, such as robot navigation tasks with a consistent success metric. We can use the same state and action spaces for all scenarios, but variations in robot configurations (e.g., chassis, wheels, tires, etc.) and deployment across various terrain (e.g., geometry, surface material, etc.) will lead to differences in transition probability measures.

We assume the source tasks can be sufficiently trained to acquire a knowledge set $\mathbb{K} = \{\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_n\}$. The problem of interest is to develop an efficient algorithm that derives the knowledge $\mathcal{K}_g$ for the target task $\mathbf{T}_g$, by leveraging the prior knowledge $\mathbb{K}$ to accelerate learning and improve performance.

IV. Context-Aware Adaptation for Robot Learning (CARoL)

To explain the rationale behind CARoL, note that in a given MDP, the transition probability $\mathcal{P}_i$ plays a central role in determining the knowledge $\mathcal{K}_i$ required to solve the problem. This makes $\mathcal{P}_i$ a natural choice as the contextual marker for knowledge adaptation: greater similarities in state transitions indicate stronger relevance between the knowledge associated with a source task and that needed for a new target task. We explicitly use state transitions as the context, i.e., *dynamic awareness* for adaptation. This provides a more interpretable and structured method to enhance learning efficiency. We present details of the proposed CARoL algorithm. As illustrated in Fig. 2 with components summarized in the caption.

Context Representation: For each source task $\mathbf{T}_i$, along with the standard use of RL that acquires the knowledge $\mathcal{K}_i$, we also train an approximation of the transition function to represent the context, denoted by $f_i(s_i, a_i\,|\phi_i) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ for each source task. This function predicts the next state based on the current state-action pair and is parameterized by ϕ_i. The parameters are updated via gradient descent for the i-th task:

$$\phi_i' = \phi_i - \alpha \nabla_{\phi_i} \mathcal{L}_{\mathbf{T}_i}(\phi_i), \tag{1}$$

where α is the learning rate and the task collects state-action-next state triplets $\{(s_{i,k}, a_{i,k}, s_{i,k}^+)\}$ sampled from robot trajectories, with $k \in \{1, \ldots, m_i\}$ being the indicator of samples. The loss function is defined as the difference between the next true state from samples and the predicted next state: $\mathcal{L}_{\mathbf{T}_i}(\phi_i) = \sum_{k=1}^{m_i} \|f_i(s_{i,k}, a_{i,k}\,|\phi_i) - s_{i,k}^+\|^2$.

Evaluation of Contextual Similarity: We use state transition functions $f_i(\cdot)$ to evaluate the contextual similarity between source task i and the new target task, which quantifies the relevance of their knowledge. A straightforward approach is to learn a transition function $f_g(\cdot)$ for the new task, then compare it with $f_i(\cdot)$ from source tasks. However, learning $f_g(\cdot)$ for a new task can be computationally expensive.

To address this, our method uses sampled trajectories $\{(s_{g,k}, a_{g,k}, s_{g,k}^+)\}, k \in \{1, \ldots, m_g\}$ from the target task. These samples are applied to each state transition function $f_i(\cdot)$, $i \in \{1, \ldots, n\}$ learned from the source tasks to compute their similarity, that is, for source task i:

$$\mathcal{Y}_i = \sum_{k=1}^{m_g} \|f_i(s_{g,k}, a_{g,k}) - s_{g,k}^+\|^2. \tag{2}$$

We regulate these measures into weights as $i \in \{1, \ldots, n\}$,

$$w_i = \frac{\exp(-\mathcal{Y}_i)}{\sum_{j=1}^{n} \exp(-\mathcal{Y}_j)}. \tag{3}$$

Remark 1: We assume the transition functions are well-tuned, then $\mathcal{Y}i$ measures the difference between the predicted next state (from the sampled state-action pair) and the true next state from the exploration data. A smaller $\mathcal{Y}i$ indicates higher contextual similarity between the i-th source task and the target task in (3). The resulting w_i prioritize relevant knowledge for the target task, thereby guiding knowledge transfer when source tasks are sufficiently related (i.e. a subset of source tasks contains useful behavioral priors under similar local contextual dynamics). However, a limitation of the algorithm is that if none of the source tasks are relevant, the target policy may perform poorly due to the absence of effective guidance. This issue will be preliminarily addressed by CARoL+ (Section IV-A4).

Context-aware Adaptation: The similarity weights w_i enable the robot to prioritize knowledge from the corresponding source tasks.

1) CARoL for Policy-Based RL: Suppose the knowledge $\mathcal{K}_i$ of each source task is a pre-trained policy, denoted by $\pi_i(s) : \mathcal{S} \to \mathcal{A}$. The contextual similarity weights are w_i. Let the target policy $\pi_g(s|\theta_g)$ be parameterized by θ_g and assume it interacts with the target environment on-policy to generate an action distribution $\pi_g(s|\theta_g)$ based on the current state $s \in \mathcal{S}$ and parameter θ_g. Let ξ_g represent the set of trajectories explored by the policy in the target task and each trajectory $\tau \in \xi_g$ is a sequence of state-action pairs.

We define a learning loss based on the divergence between the action distribution of the target policy and a weighted combination of the action distributions of the source policies:

$$\mathcal{L}_P = \sum_{\tau \in \xi_g} \sum_{s \in \tau} \sum_{i=1}^{n} w_i \mathcal{D}(\pi_i(s), \pi_g(s|\theta_g)), \tag{4}$$

Authorized licensed use limited to: George Mason University. Downloaded on October 26,2025 at 05:42:33 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: CARoL for Policy-Based RL.

input Knowledge as pre-trained source policies $\{\pi_1, \ldots, \pi_n\}$; contextual similarity weights $\{w_1, \cdots, w_n\}$; the target task $\mathbf{T}_g$;
initialize target policy $\pi_g(s|\theta_g)$; a learning rate η;
for *iteration = 1 to K* **do**
  Collect trajectories ξ_g by interacting with the target task $\mathbf{T}_g$ using $\pi_g(s|\theta_g)$;
  for *each minibatch of data from* ξ_g **do**
    Compute $\mathcal{L}_P$ with equation (4);
    Update: $\theta'_g = \theta_g - \eta \cdot \nabla_{\theta_g} \mathcal{L}_P$;
  end
end
return *target policy* π_g.

Algorithm 2: CARoL for Value-Based RL.

input Knowledge as pre-trained source Q networks $\{Q_1, Q_2, \ldots, Q_n\}$; contextual similarity weights $\{w_1, \cdots, w_n\}$; the target task $\mathbf{T}_g$;
initialize target value function network $Q_g(s, a\,|\psi_g)$; a replay buffer $\mathcal{M}$; a learning rate η
for *episode = 1 to K* **do**
  Greedy method to choose action a with explorations or $\arg\max_a(Q_g(s, a\,|\psi_g))$ in target task $\mathbf{T}_g$ to get next state s' and reward r;
  Store (s, a, r, s') in $\mathcal{M}$;
  if *the size of* $\mathcal{M}$ *is sufficient* **then**
    Randomly sample a minibatch $\mathcal{B}$ from $\mathcal{M}$;
    Compute $\mathcal{L}_Q$ with equation (7);
    Update: $\psi'_g = \psi_g - \eta \cdot \nabla_{\psi_g} \mathcal{L}_Q$;
  end
end
return *target value function* Q_g.

where $\mathcal{D}(\cdot,\cdot)$ is the Kullback–Leibler (KL) divergence:

$$\mathcal{D}(\pi_i(s), \pi_g(s|\theta_g)) = \Phi\left(\frac{\pi_i(s)}{T}\right) \ln \frac{\Phi(\frac{\pi_i(s)}{T})}{\Phi(\pi_g(s|\theta_g))}. \quad (5)$$

Here, $\Phi(\cdot)$ is the softmax function and T is the temperature parameter. This process selectively incorporates knowledge from relevant source policies to optimize the target task, as summarized in Algorithm 1.

2) CARoL for Value-Based RL: Value-based RL algorithms use value functions to estimate the expected cumulative reward starting from a given state-action pair (or from a state), then guide action selection. In the following, we consider the knowledge $\mathcal{K}_i$ of each source task to be a pre-trained state-action value function $Q_i(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (a similar mechanism can be easily generalized to state value function $V_i(s)$.). The contextual similarity weights are w_i.

Let the target value function be a neural network $Q_g(s, a\,|\psi_g)$ parameterized by ψ_g. Actions are greedily selected for the target task using an epsilon-greedy strategy, transitioning from early exploration to later exploitation. During exploitation, we select the action that yields the highest Q-value output from the current state in the Q-network.

Once the replay buffer reaches a sufficient size, we randomly extract a minibatch $\mathcal{B}$ of data. For each data pair in this minibatch, the target value is computed. However, unlike traditional value-based RL algorithms, we do not rely on the inadequately trained target Q_g-network for updates. Instead, we leverage the knowledge from the source $Q_i(s, a)$ functions, reweighted by contextual similarities, as follows:

$$Q_{\text{next}} = \sum_{i=1}^{n} w_i Q_i(s^+, \arg\max_{a^+}(Q_i(s^+, a^+))), \quad (6)$$

where s^+ is the next state, and $\arg\max$ selects the action by maximizing each source Q_i value function.

Using the Bellman equation with a discount factor γ, we define the following learning loss based on the difference between the target value function and the value using reweighted source Q_i networks as prior knowledge:

$$\mathcal{L}_Q = \sum_{s\in\mathcal{B}} \left[\|Q_g(s, a\,|\psi_g) - r - \gamma Q_{\text{next}}\|^2\right], \quad (7)$$

where $\mathcal{B}$ is the minibatch extracted from the replay buffer, and r is the reward for the corresponding state-action pair.

The process, summarized in Algorithm 2, selectively incorporates knowledge from the relevant source value functions to approximate the value function for the target task. To test the learned $Q_g(s, a\,|\psi_g)$ in $\mathbf{T}_g$, the greedy method can be used to select actions.

3) CARoL for Actor-Critic RL: If the prior knowledge $\mathcal{K}_i$ is in the form of an actor-critic structure, involving both a policy and a value function, then both of them can be leveraged to enhance learning in a new task. Instead of directly combining the methods introduced earlier to simultaneously learn a policy and a value function for the target task, we focus on learning a target actor (i.e., policy) $\pi_g(s\mid\theta_g)$, while adopting a static fusion method to leverage the source critic (value) functions without using a parameterized function approximation. Otherwise, if the actor and critic are updated simultaneously, the method might suffer from issues such as instability and overestimation of Q-values, which are common in actor-critic RL [26].

Similar to the policy-based approach outlined in Algorithm 1, we define part of the learning loss to be $\mathcal{L}_P$, which measures the weighted combination of divergences between the target policy's action distribution and source action distributions.

Additionally, we define a loss from critic as follows:

$$\mathcal{L}_C = -\sum_{\tau\in\xi_g}\sum_{s\in\tau}\left[\sum_{i=1}^{n} w_i \cdot Q_i(s, \pi_g(s|\theta_g))\right], \quad (8)$$

where the action for each state s is determined by the current actor π_g. Thus, $\mathcal{L}_C$ evaluates the performance of the target actor using the weighted combination of source critics, prioritizing those that are more contextually similar to the target task. Minimizing $\mathcal{L}_C$ helps to iteratively improve the target actor by leveraging the guidance of source critics.

The combined losses in (4) and (8) yield:

$$\mathcal{L}_{AC} = \mathcal{L}_P + \beta\mathcal{L}_C. \quad (9)$$

By minimizing $\mathcal{L}_{AC}$, the target actor π_g is informed by both the source actors $\pi_i(s)$ through $\mathcal{L}_P$ as well as the source critics $Q_i(s, a)$ through $\mathcal{L}_C$. The value of $\beta \in \mathbb{R}_+$ is adjustable and

Authorized licensed use limited to: George Mason University. Downloaded on October 26,2025 at 05:42:33 UTC from IEEE Xplore. Restrictions apply.

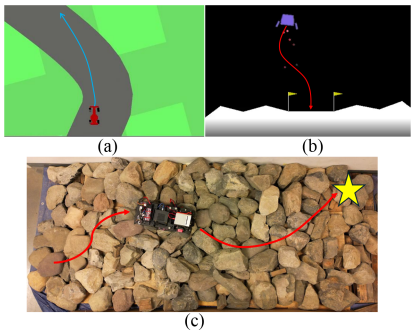

Fig. 3. Two OpenAI simulation environments for validating our algorithm: (a) CarRacing environment; (b) LunarLander environment. One physical experiment environment: (c) Ground vehicle off-road navigation.

Algorithm 3: CAROL for Actor-Critic RL.

input Knowledge as pre-trained source actors $\Pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ and critics $\{Q_1, Q_2, \ldots, Q_n\}$; contextual similarity weights $\{w_1, \cdots, w_n\}$; the target task $\mathbf{T}_g$;
initialize target actor $\pi_g(s|\theta_g)$;
for *iteration = 1 to K* **do**
  Collect trajectories ξ_g by interacting with the target task $\mathbf{T}_g$ using $\pi_g(s|\theta_g)$;
  for *each minibatch of data from* ξ_g **do**
    Compute $\mathcal{L}_{AC}$ with equation (4) and (8);
    Update: $\theta_g' = \theta_g - \eta \cdot \nabla_{\theta_g} \mathcal{L}_{AC}$;
  end
end
return *target actor* π_g.

determines the importance of each. Algorithm 3 summarizes how to use CAROL for actor-critic RL.

4) CAROL+: In the above subsections, the introduced CAROL focuses on leveraging prior knowledge to solve a new target task. It omits the aspect of using standard RL approaches to acquire new knowledge from the task. This is because additional learning may not always be necessary when prior knowledge is already sufficient for solving the task. However, if learning from the target task is required, this can be achieved by adding standard RL loss terms to the functions (4), (7), and (9) for policy-based, value-based, and actor-critic methods. This extension enables the model to not only adapt prior knowledge but also refine it through direct learning from the target task. An implementation of this, called CAROL+, will be included in the experiments section for comparison.

V. Experiments

In this section, we present both simulated and physical experiments to evaluate the effectiveness of CAROL. To ensure reproducibility, the simulated experiments are conducted using OpenAI's CarRacing and LunarLander tasks, as shown in Fig. 3(a), (b). To demonstrate CAROL's capabilities in real-world scenarios (Fig. 3(c)), we deploy it on a ground vehicle, enabling the rapid and efficient adaptation of policies learned from source tasks to smoothly navigate real-world off-road terrain. In these setups, CAROL is integrated with two types of RL algorithms: PPO (Proximal policy optimization, focusing on *policy gradient* optimization using a clipped surrogate objective) and TD3 (Twin Delayed Deep Deterministic, with the actor updated to maximize the Q-value estimated by the critic, following a *value-based update mechanism*) for knowledge adaptation [26]. The performance of CAROL is compared against multiple baseline approaches.

TABLE I
Contextual Similarities - CarRacing

μ	a: 0.2	b: 0.5	c: 0.7	d: 1.0	e: 1.3	f: 1.7	g: 2.0	h: 2.3
$\mathbf{T}_1$	1.00	0.98	0.83	0.05	0.10	0.06	0.06	0.88
$\mathbf{T}_2$	0.00	0.02	0.17	0.94	0.64	0.15	0.11	0.13
$\mathbf{T}_3$	0.00	0.00	0.00	0.01	0.26	0.79	0.83	0.79

A. Simulated Experiment: CarRacing

1) Experiment Setup: As shown in Fig. 3(a), CarRacing is a continuous control task where an autonomous mobile robot must navigate a procedurally generated racetrack to maximize cumulative rewards. The task involves balancing speed and control to avoid going off-track while efficiently moving forward on the racetrack. The dynamics of the car depend on track friction. We design three task environments with low $\mu = 0.5$, medium $\mu = 1$, and high friction $\mu = 2$, as source tasks. To implement CAROL, we obtain all source knowledge (SK) using PPO (Proximal policy optimization) [27]. While PPO has a value function, it is primarily an actor-focused method. The source knowledge is represented as pre-trained source policies. Therefore, our CAROL framework utilizes Algorithm 1 for adaptation. Implementation of Baselines:

- **Policy Distillation (PD)** [28]: PD is implemented using source policies as teachers to train a target policy, which fuses all source policies equally in an indiscriminate manner.
- **Model-Agnostic Meta-Learning (MAML)** [18]: We include MAML as a representative method for meta-learning. The meta-model is trained using all source tasks. Then the meta-model is adapted to the new environment using PPO.
- **Learning from Scratch (LfS)**: The same method that is used in source knowledge training but adjust the network structure to match that of CAROL.
- **Source Knowledge (SK)**: We directly apply source knowledge to different target tasks. The cumulative reward values are evaluated by averaging over 20 episodes.

For the purpose of fair comparison between CAROL and other baseline methods, we used neural networks of similar sizes; then, the number of episodes would serve as the measure for both convergence and computational efficiency.

2) Results and Analysis: Table I presents the contextual similarity weights learned for the three source tasks in different target tasks under the CarRacing simulated experiment. We observe that the contextual similarity weight assigned to a source task for a given target task is generally negatively correlated with their friction difference. In particular, the tasks with $\mu = 0.5$, $\mu = 1$, and $\mu = 2$ are scenarios where the target task is the same as one of the source tasks. The results confirm the effectiveness of our similarity weight, as the corresponding source task consistently receives the highest values.

Fig. 4 presents the reward curves, where the x-axis represents the number of training steps, and the y-axis represents the reward

Authorized licensed use limited to: George Mason University. Downloaded on October 26,2025 at 05:42:33 UTC from IEEE Xplore. Restrictions apply.

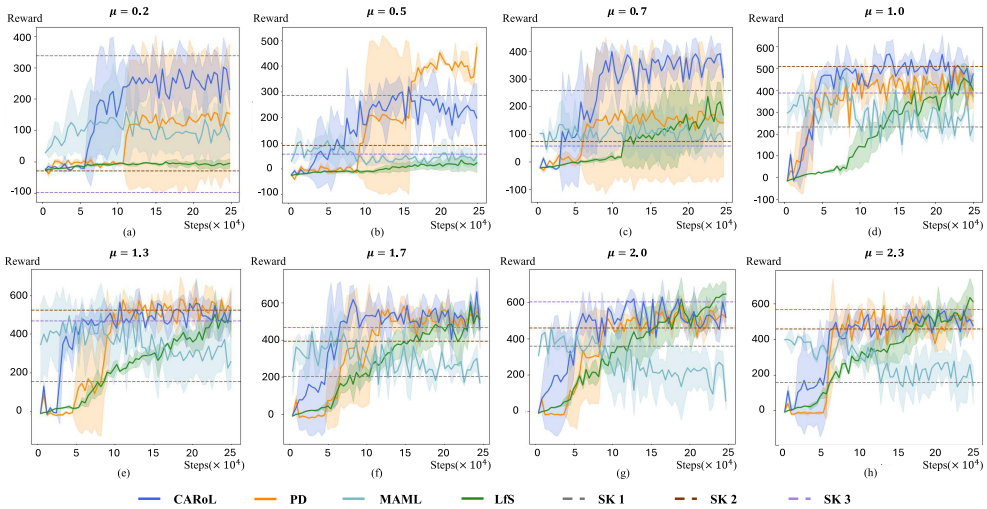

Fig. 4. Comparison of cumulative reward for CARoL and baselines under different CarRacing tasks. The x-axis represents the running steps and the y-axis represents the reward. The height of dash lines demonstrates the average cumulative reward of source knowledge (SK) under the task.

value. We observe that each SK performs well on its corresponding task, indicating that the SK has been well-trained. In all cases, CARoL demonstrates a faster convergence speed compared to other methods. This verifies CARoL's capability to efficiently leverage existing knowledge and quickly adapt to the target task. In terms of the final reward, CARoL generally achieves the best performance. Specifically, when the target environment matches one of the source environments, as in subfigures (b, d, g), CARoL produces results comparable to the well-trained SK policies. When compared to PD, which indiscriminately fuses SK policies, CARoL can prioritize relevant SKs, leading to better performances in (a, c). Additionally, CARoL exhibits a more stable training process than PD.

The LfS uses the same parameter settings as CARoL. LfS struggles to converge in most tasks within the given training steps, whereas CARoL adapts significantly faster. Regarding MAML, despite extensive testing meta-training and adaptation, its performance remains poor. This may be due to an undesirable meta-model that is difficult to generalize effectively to new environments.

B. Simulated Experiment: LunarLander

1) Experiment Setup: As shown in Fig. 3(b) LunarLander is a control task where an autonomous lander robot must safely land on a designated landing pad in a simulated lunar environment. The robot must balance thrust and control to achieve a smooth landing while minimizing fuel consumption. The LunarLander environment allows adjustments of gravity and wind power. For source tasks, we set the two parameters as $(-5, 0)$; $(-10, 0)$; $(-5, 10)$; $(-10, 10)$, respectively. To implement CARoL, we obtain all source knowledge using TD3 (Twin Delayed Deep Deterministic) [26]. The source knowledge is represented as pre-trained source policies and state-action value functions. Therefore, our CARoL utilizes Algorithm 3 for adaptation with $\beta = 1$. In addition to the baselines used in the CarRacing, we also include two further variants in our evaluation:

- **CARoL+**: This variant incorporates CARoL's original adaptation loss in (9), while adding an additional TD3 learning objective on the target task.

TABLE II
CONTEXTUAL SIMILARITIES - LUNARLANDER

	a: (-5, 0)	b: (-10, 0)	c: (-5, 10)	d: (-10, 10)	e: (-3, 2)	f: (-6, 9)	g: (-9, 5)	h: (-7, 0)	i: (-11, 9)	j: (-8, 12)
$\mathbf{T}_1$	0.95	0.01	0.06	0.00	0.62	0.09	0.19	0.35	0.01	0.13
$\mathbf{T}_2$	0.05	0.98	0.01	0.01	0.09	0.00	0.37	0.34	0.12	0.17
$\mathbf{T}_3$	0.00	0.00	0.91	0.01	0.28	0.81	0.18	0.22	0.10	0.29
$\mathbf{T}_4$	0.00	0.01	0.02	0.98	0.01	0.10	0.26	0.09	0.77	0.41

- **RMA** [6]: We adapt RMA by learning the policy conditioned on latent contextual dynamics representations and an adaptation module to infer these from trajectory histories.

2) Results and Analysis: Table II presents the contextual similarity weights learned for four source tasks in different target tasks. We observe that the weight is negatively correlated with their geometric distance in the 2-D parameter space.

Fig. 5 shows the reward curves. In comparison with baselines, we see that PD exhibits a consistently lower reward convergence than CARoL across most target tasks. Comparing LfS to CARoL, we observe that LfS demonstrates a significantly slower convergence speed, particularly for (f) and (h), and generally achieves lower final rewards. This is because Lfs has a same network size as CARoL but cannot leverage prior knowledge. It is worth noting that in target tasks (i) and some trials of (j), LfS achieves higher rewards than CARoL. This is because the source task may not provide sufficient useful knowledge to the target tasks, as these target tasks use parameters outside the combination of the source tasks. In these cases, LfS benefits from learning new knowledge from the new task and thus outperforms CARoL.

However, this issue can be addressed by CARoL+ (cf. Section IV-A4). CARoL+ integrates interaction-based learning during adaptation, allowing it to achieve the highest reward across all cases. This compensates for CARoL's potential limitations in final convergence values. Additionally, CARoL+ retains CARoL's advantage of fast convergence while also being the most stable one during the convergence process.

Authorized licensed use limited to: George Mason University. Downloaded on October 26,2025 at 05:42:33 UTC from IEEE Xplore. Restrictions apply.

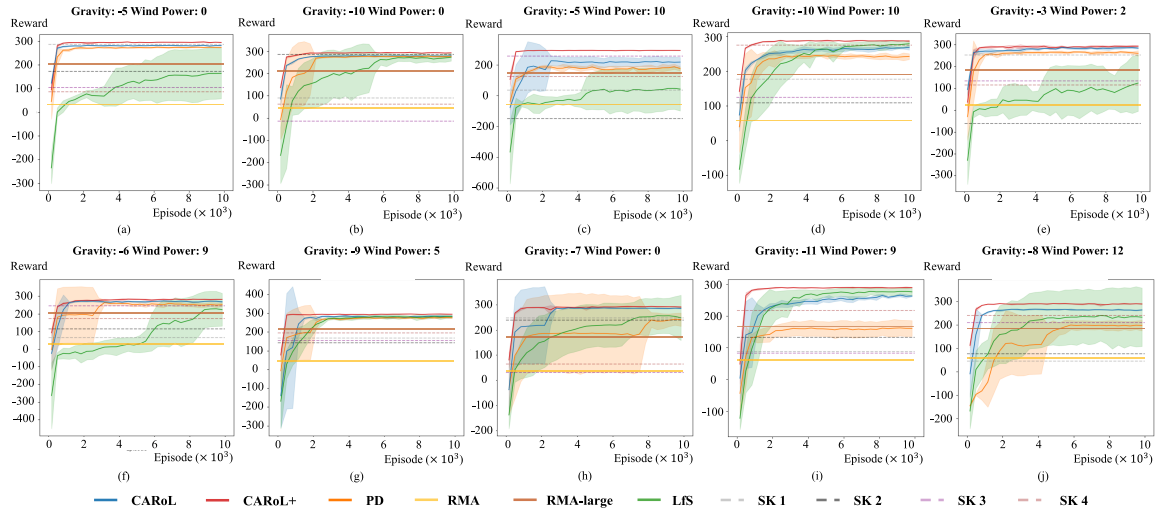

Fig. 5. Comparison of cumulative reward for CARoL, CARoL+ and baselines under different LunarLander tasks. The x-axis represents the running episodes and the y-axis represents the reward.

We also evaluate RMA, which is trained under randomized contextual dynamics, simultaneously learning a latent representation of the environment. For a new task, it can be directly deployed without training. It first collects short trajectories (e.g., 10 trials of 50 steps) from the target task and uses them to infer the latent encoding, which is then fed into the policy for execution. The results are presented as the straight lines in Fig. 5. Initially, when using an adaptation module with a similar network size as CARoL, RMA (yellow line) fails to achieve high cumulative rewards in most target tasks. We attribute this limitation to the lack of structured generalization in the learned latent space. Specifically, the adaptation module relies on fitting the latent representation from short historical trajectories, which may lead to estimation drift or errors, especially in unseen tasks. However, when we increased the network complexity of the adaptation module, RMA-large (brown line) obtains comparable performance, suggesting that the representational capacity of the latent encoding network is a critical factor for RMA's effectiveness.

C. Real World: Ground Vehicle Off-Road Navigation

We apply CARoL to an off-road mobility problem with a physical robot. As shown in Fig. 3(c), the goal is to enable the robot to autonomously and successfully navigate from the start to the goal. RL for real-world off-road mobility is particularly inefficient due to the prohibitive cost of collecting physical vehicle-terrain interaction data. Therefore, in this experiment, we aim to apply CARoL to adapt policies learned in simulation to the real world.

1) Experiment Setup: We leverage a high-fidelity multi-physics simulator, Chrono [29], to train two different off-road navigation policies: SK1 for concrete terrain and SK2 for grass terrain. We generate off-road terrain with undulating topographies and vary the friction coefficient between the vehicle tires and the underlying terrain, i.e., 0.9 and 0.4. Here, different friction levels influence traversability. For example, for the same slope, friction determines whether the robot can climb it or must bypass it. In the real world, we construct an off-road mobility testbed with hundreds of rocks and boulders. We posit the friction coefficient of the real-world rocks and boulders to be between 0.57 and 0.73, necessitating adaptation from the two SKs into a new policy.

2) Implementation of CARoL: In the simulators, we use PPO to train two source knowledge policies from scratch and train state transition functions for each environment as context representation. To implement CARoL in the real world, we first compute the contextual similarity. We manually control the vehicle to randomly navigate 20 trials on the real-world off-road terrain to collect trajectory data, instead of relying on random exploration, for real-world sample efficiency. The contextual similarity is quantified by comparing the prediction errors of the source task transition functions over trajectories collected in the real-world environment.

Based on the obtained contextual similarity and source task policies, we employ CARoL-Algorithm 1 to adapt to the real world. We perform a total of 10 iterations with 8 navigation trials each (approximately 130 minutes in total), during which the vehicle gradually improve from initially failing (veering off the experimental field, getting stuck halfway, or flipping over) to successfully reaching the goal. In the final two iterations, the vehicle successfully reaches the goal in every episode, indicating a good adaptation performance. We then compare this converged model with the source knowledge policies and the fusion of knowledge with specific weights (not the correct weights obtained from contextual similarity). We do not implement LfS and MAML methods due to the cost of collecting extensive physical vehicle-terrain interaction data. For the same reason, we do not employ CARoL+. We also do not employ RMA since training the adaptation module of RMA requires substantial environment-policy interaction across diverse terrain settings, which is expensive in the Chrono. The real-world experiment delivers two key contributions: 1) Demonstrating the feasibility of CARoL in real robot adaptation scenarios; 2) Providing evidence of the rationality of the learned contextual similarity weights.

3) Experiment Results and Analysis: In the comparison experiments, each model undergoes five trials from the start to the goal. The source knowledge refers to the model trained in the Chrono simulator, which is directly deployed on the robot.

Authorized licensed use limited to: George Mason University. Downloaded on October 26,2025 at 05:42:33 UTC from IEEE Xplore. Restrictions apply.

TABLE III
PHYSICAL EXPERIMENT RESULTS

Method	Success ↑	Roll ↓	Pitch ↓	Time (no penalty)
CARoL	**5/5**	**4.08°±4.65°**	**8.55°±5.13°**	16.88s±1.97s
SK1	3/5	5.44°±7.13°	10.35°±5.29°	11.78s±1.44s
SK2	2/5	5.75°±5.00°	8.61°±5.12°	**8.55s±0.67s**
PD (0.5 SK1 + 0.5 SK2)	4/5	5.62°±6.90°	10.12°±5.38°	12.21s±1.35s
0.8 SK1 + 0.2 SK2	2/5	5.30°±7.45°	10.48°±5.12°	11.95s±1.51s
0.2 SK1 + 0.8 SK2	2/5	5.67°±6.98°	10.08°±5.40°	12.03s±1.46s

The evaluation metrics for this experiment, aligned with the RL rewards, include success rate, roll, and pitch. While traveling time is not evaluated, it is provided as a reference.

As shown in the results Table III, CARoL achieves the highest success rate, and the robot demonstrates the most stable navigation under CARoL's adapted model. Although CARoL takes the longest time to reach the goal, we do not impose any reward or penalty on travel time in this task. The longer duration may suggest that the robot is carefully selecting paths based on traversability. The primary objective is to ensure that the robot autonomously reaches the goal successfully, and CARoL has high stability in terms of roll and pitch. To further validate the effectiveness of CARoL's contextual weights, we include three additional baselines for comparison. We observe that if the weights cannot be properly learned and assigned, the performance degrades substantially. Additionally, we observe that while SK2 achieves the shortest average traveling time, its low success rate and higher instability highlight the risk of directly deploying single-source policies to the real world, even when the terrain seems similar (e.g., friction level). Overall, these results demonstrate that CARoL not only successfully transfers learned behaviors to the real world, but also achieves superior navigation stability and robustness through adaptive policy fusion based on contextual similarity.

VI. CONCLUSION AND LIMITATIONS

We presented CARoL, a framework that first evaluates contextual similarities between source and target tasks based on their state transitions. Given the prior knowledge from source tasks, CARoL then performs contextual knowledge adaptation for target tasks. A key advantage of CARoL is its broad applicability to policy-based, value-based, and actor-critic RL methods. Experimental results demonstrated that our algorithm effectively adapts to new tasks. Limitations of CARoL include: (i) in task space, source task configurations need sufficient coverage over target tasks; (ii) sufficient state-action space exploration is needed for quantification of contextual similarity; (iii) we limit the scope to dynamics-shifted tasks with shared reward functions. For future work, we plan to explore learning continuous task embeddings to improve coverage and flexibility of adaptation across smoothly varying tasks, and on tasks with dynamics and reward shifts to validate the generalization of CARoL+.

REFERENCES

[1] Z. Hu, D. Shishika, X. Xiao, and X. Wang, "Bi-CL: A reinforcement learning framework for robots coordination through bi-level optimization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 581–586.

[2] M. Limbu, Z. Hu, X. Wang, D. Shishika, and X. Xiao, "Scaling team coordination on graphs with reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 16538–16544.

[3] Z. Xu, B. Liu, X. Xiao, A. Nair, and P. Stone, "Benchmarking reinforcement learning techniques for autonomous navigation," in *Proc. 2023 IEEE Int. Conf. Robot. Automat.*, 2023, pp. 9224–9230.

[4] S. Ruder, P. Ghaffari, and J. G. Breslin, "Knowledge adaptation: Teaching to adapt," 2017, *arXiv:1702.02052*.

[5] F. Wang, J. Yan, F. Meng, and J. Zhou, "Selective knowledge distillation for neural machine translation," in *Proc. ACL-IJCNLP*, 2021.

[6] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in *Proc. Robot.: Sci. Syst.*, 2021.

[7] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Sci. Robot.*, vol. 5, no. 49, 2020, Art. no. eabb2174.

[8] D. Kalashnikov et al., "MT-Opt: Continuous multi-task robotic reinforcement learning at scale," 2021, *arXiv:2104.08212*.

[9] J. Xing, I. Geles, Y. Song, E. Aljalbout, and D. Scaramuzza, "Multi-task reinforcement learning for quadrotors," *IEEE Robot. Autom. Lett.*, vol. 10, no. 3, pp. 2112–2119, Mar. 2025.

[10] Y. Teh et al., "Distral: Robust multitask reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4499–4509.

[11] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," in *Proc. ICLR*, 2016.

[12] Y. Zhou, W. Jin, and X. Wang, "D3G: Learning multi-robot coordination from demonstrations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 8898–8904.

[13] B. Liu, X. Xiao, and P. Stone, "A lifelong learning approach to mobile robot navigation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1090–1096, Apr. 2021.

[14] A. Logacjov, M. Kerzel, and S. Wermter, "Learning then, learning now, and every second in between: Lifelong learning with a simulated humanoid robot," *Front. Neurorobot.*, vol. 15, 2021, Art. no. 669534.

[15] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "RL^2: Fast reinforcement learning via slow reinforcement learning," in *Proc. ICLR*, 2017.

[16] 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.

[17] L. Zintgraf et al., "VariBAD: Variational bayes-adaptive deep RL via meta-learning," *J. Mach. Learn. Res.*, vol. 22, no. 289, pp. 1–39, 2021.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[19] F. Alet, T. Lozano-Pérez, and L. P. Kaelbling, "Modular meta-learning," in *Proc. Conf. Robot Learn.*, 2018, pp. 856–868.

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

[21] S. Li, F. Gu, G. Zhu, and C. Zhang, "Context-aware policy reuse," in *Proc. AAMAS*, 2019.

[22] 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.

[23] A. Barreto et al., "Successor features for transfer in reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4058–4068.

[24] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik, "Adapting rapid motor adaptation for bipedal robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 1161–1168.

[25] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik, "In-hand object rotation via rapid motor adaptation," in *Proc. Conf. Robot Learn.*, 2023, pp. 1722–1732.

[26] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

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

[28] A. A. Rusu et al., "Policy distillation," in *Proc. ICLR*, 2016.

[29] A. Tasora et al., "Chrono: An open source multi-physics dynamics engine," in *Proc. 2nd Int. Conf. High Perform. Comput. Sci. Eng.*, 2016, pp. 19–49.

Authorized licensed use limited to: George Mason University. Downloaded on October 26,2025 at 05:42:33 UTC from IEEE Xplore. Restrictions apply.