Efficient Test-Time Adaptation For Robot Policies

Motasem Alfarra, Pietro Mazzaglia, Markus Peschl, Daniel Dijkman, Christos Louizos
Qualcomm AI Research, Amsterdam, the Netherlands
malfarra@qti.qualcomm.com

Abstract

Reinforcement-learning policies for robot locomotion can fail at deployment when operating conditions shift (e.g., rough or slippery terrain, payload changes, sensing degradation), and retraining or manual recalibration is often infeasible under on-device latency and memory constraints. We study *test-time adaptation* (TTA) for robotic control: adapting a pretrained policy online and self-supervised during execution, without access to rewards. Our starting point is a lightweight baseline that updates only the actor using a frozen pretrained critic as a deployment-time performance surrogate. Building on this, we introduce TEMPO, a test-time entropy-regularized, memory-guided policy optimization method that stabilizes critic-driven updates under severe shift. TEMPO combines (i) an entropy-promoting regularizer on the critic's predictive distribution to mitigate overconfident extrapolation, and (ii) a score-based memory, inspired by prioritized replay, that prioritizes low-value ("hard") observations and applies importance sampling to focus limited computation on the most informative deployment conditions. Across Go1, Go2, and G1 in MuJoCo Playground under four representative shifts and multiple architectures, TEMPO yields consistent gains, reaching up to 80% improvement in a single-device setting. We further validate on a physical Unitree Go2 with an added calf payload, where fully on-device adaptation improves performance over the non-adapted baseline across real-world episodes.

1 Introduction

Robots deployed in the real world must contend with operating conditions that can diverge sharply from those encountered during training [Kumar et al., 2021]. These discrepancies arise from multiple sources: terrain variability (e.g., rocks, low-friction surfaces) [Sun et al., 2025], hardware changes (e.g., localized payloads or partial actuator degradation) [Pouroullis et al., 2025], emergent situations (e.g., a weight stuck on one arm/calf) [Zhang et al., 2025, Chen et al., 2023], and the broader sim-to-real gap between simulated and physical environments [Peng et al., 2018]. In practice, such factors rarely occur in isolation: a deployed robot may simultaneously experience altered contact dynamics, sensor noise, actuator wear, and previously unseen environmental configurations. These distribution shifts can substantially degrade policies that perform well under nominal conditions (refer to Figure 1 left), making online adaptation central to robust autonomy.

A common response to combat this problem is to apply large-scale domain randomization during reinforcement learning to improve zero-shot generalization [Tobin et al., 2017, Peng et al., 2018]. While effective in some cases, this approach is fundamentally constrained by what can be specified in advance: it presumes one can enumerate relevant sources of variability, model them with sufficient fidelity in simulation, and collect enough data to cover their interactions. When robots encounter truly novel conditions, adaptation is performed offline. This interrupts operation to gather new data and fine-tune from scratch, which is misaligned with real-time, on-device compute and reliability

Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

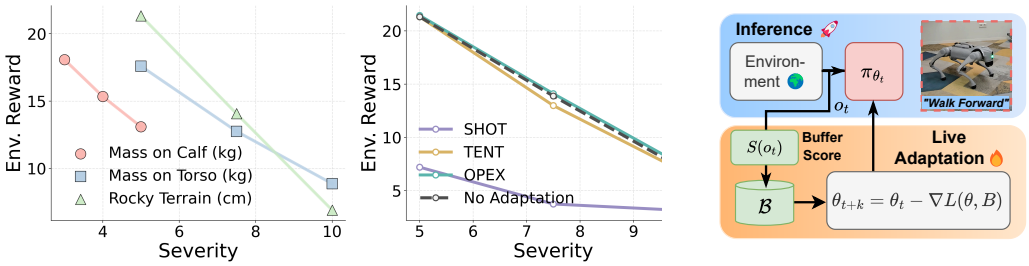

Figure 1: **Motivation and pipeline.** Left: Pretrained locomotion policies degrade as shift severity increases. Middle: Off-the-shelf TTA and prior RL adaptation baselines fail under a rocky-terrain shift. Right: TEMPO loop: execute π_{θ_t}, update memory $\mathcal{B}$, and periodically adapt via Eq. (2).

constraints. Architectural add-ons for adaptation can help but often increase system complexity and reduce portability across tasks, embodiments, and deployment settings [Kumar et al., 2021].

We study **Test-Time Adaptation** (TTA) [Wang et al., 2020, Sun et al., 2020] as a practical alternative for robotic control under distribution shift. TTA is well-suited for tackling this problem as it adapts the pretrained model at test time in an online and self-supervised manner, alleviating the need to disrupt the inference stream for retraining and requiring no labeled data. Thus, our starting point is a simple, general baseline: we update only the actor (policy) parameters at deployment while freezing the critic, using a compact on-device memory of recent observations. At periodic intervals during execution, the actor is adapted by maximizing the critic's estimated action value on stored observations—treating the critic as a learned performance surrogate and requiring no online rewards. Unlike standard deployments where the critic is used primarily for training and often discarded at inference, here we *actively leverage* the pretrained critic during runtime. While distribution shift also affects the critic, we find it still provides a sufficiently informative signal to guide test-time actor updates [Chen et al., 2023]; nonetheless, naive value maximization can overfit to spurious, overconfident estimates [Kumar et al., 2020] and under-explore the loss landscape.

In this work, and to address these limitations, we introduce an enhanced framework with two complementary elements. First, we add an **entropy-promoting regularizer** on the critic's predictive distribution during adaptation, which tempers overconfidence under shift and biases updates toward regions that are simultaneously high-value and high-uncertainty. This yields more informative gradients due to the stability and exploration benefits of maximum-entropy objectives in RL [Eysenbach and Levine, 2021], which we adapt to the *critic*'s uncertainty at test time. Second, and in the spirit of prioritized experience replay [Schaul et al., 2016], we employ a **score-based memory** that prioritizes observations inversely proportional to the critic's predicted value, coupled with importance sampling to focus updates on the most informative deployment conditions under tight memory and latency budgets. We evaluate on Go1, Go2, and G1 in Mujoco Playground simulation [Zakka et al., 2025], spanning quadruped and humanoid morphologies under four representative shifts (mass on torso, mass on calf, rocky terrain, slippery terrain) and three different architectures, and observe up to 50% performance gains very efficiently in a single-device setting. We further validate our approach on a physical Unitree Go2 robot by attaching a $\{3, 2\}$ kg calf payload and performing fully on-device TTA , observing trends consistent with simulation. We can summarize our contributions in four folds:

- **A lightweight, on-device TTA framework for RL control.** We adapt only the actor parameters while freezing the critic, maintain a compact memory of recent observations, and perform periodic, self-supervised updates that fit strict control-loop latency and compute budgets.
- **Entropy-regularized critic-driven adaptation.** We introduce an objective that maximizes the critic's estimated action value while promoting higher-entropy critic outputs during test-time updates—mitigating overconfident value extrapolation and preserving local action coverage; the procedure is architecture-agnostic and supervision-free.
- **Score-based memory with importance sampling.** We propose a memory mechanism that assigns each observation a score inversely proportional to the critic's predicted value, stores harder samples, and employs importance sampling to focus on more informative samples under memory constraints.
- **Comprehensive validation in simulation and on-device.** We show consistent gains on Go1/Go2/G1 embodiments and on four deployment shifts, and confirm the same trends on a physical Unitree Go2 via fully on-device adaptation with a $\{3, 2\}$ kg calf payload.

2 Robot Adaptation as Test Time Adaptation

2.1 Problem Formulation and Evaluation Protocol

RL setting and distribution shift. We consider locomotion as a Markov decision process (MDP) $\mathcal{M} = (\mathcal{O}, \mathcal{A}, P, \gamma)$, where $\mathcal{O} \subseteq \mathbb{R}^d$ denotes the observation space, $\mathcal{A} \subseteq \mathbb{R}^p$ the action space, P the (unknown) transition dynamics over observations,[1] and $\gamma \in (0, 1)$ the discount factor. A policy $\pi_\theta(a \mid o)$ maps an observation $o_t \in \mathcal{O}$ (e.g., proprioception, IMU, joint states, contact signals) to an action $a_t \in \mathcal{A}$ (e.g., target joint positions/torques). During training, π_θ is learned in a *source* training setup, inducing a training distribution over the source observations. At deployment, the robot operates under a *shifted* test-time distribution (e.g., altered friction, payload changes, rocky terrain), which can be viewed as changes in the observations (i.e from o to $\hat{o}$) encountered by the agent and in the induced observation-transition process (i.e. from P to $\hat{P}$). We assume the deployed policy is pretrained using an actor–critic framework [Haarnoja et al., 2018, Fujimoto et al., 2018]. In particular, alongside the pretrained policy π_{θ_0}, training produces a critic $Q_{\psi_0}(o, a)$ that estimates the action-value of executing action a when observing o. In contrast with the common asymmetric setup [Nahrendra et al., 2023] where the critic is fed privileged information from the underlying simulator state, we consider a critic that consumes the same set of observations as the actor.

Test-time adaptation as an online stream problem. At test time, the robot receives a stream of unlabeled observations $\{o_t\}_{t\geq 0}$ governed by the actions taken by the policy π_{θ_t} and the transition dynamics $\hat{P}$. A Test-Time Adaptation (TTA) method is an update rule g that uses only quantities available online at inference time to update either the model parameters, the received observation, or both. This formulation is intentionally general: g may adapt the policy parameters θ, the critic parameters ψ, both, or a subset of either.

Online evaluation protocol. Following the online/streaming evaluation paradigm [Cai et al., 2021, Ghunaim et al., 2023, Alfarra et al., 2024], we simulate the interaction between the deployed agent and the test stream at each time step t as:

1. The environment reveals an observation $o_t \sim \hat{P}(o_{t-1}, a_{t-1})$.
2. The agent selects and executes an action $a_t \sim \pi_{\theta_t}(o_t)$.
3. The adaptation rule updates parameters: $(\theta_{t+1}, \psi_{t+1}) \leftarrow g(\theta_t, \psi_t)$.

Notably, this protocol does not assume access to rewards or labels at test time. Although rewards are not used by the adaptation mechanism, performance is evaluated using task-level metrics computed offline or by the benchmark (e.g., episodic return in simulation, distance traveled, tracking error, stability/fall rate, task success). These metrics are used solely for reporting and comparison, not as supervision for adaptation.

2.2 Constructing a Baseline

Test-time adaptation for robot control is challenging for two reasons. **(i)** As opposed to massively parallel simulation-based training [Li et al., 2023], deployment typically occurs in a single environment with limited diversity over short time horizons. Consecutive observations are strongly correlated and provide a narrow view of the shifted dynamics. **(ii)** The distribution shift can be abrupt (e.g., payload changes or slippery terrain), yet the adaptation budget is small and must fit within on-device latency constraints. These constraints make standard TTA baselines (e.g., Tent [Wang et al., 2020]) as well as RL-based alternatives (e.g., OPEX [Park et al., 2024]) ineffective in our setting (see Fig. 1, middle) For simplicity, we assume throughout that the policy is deterministic (*i.e.* $a_i = \pi_{\theta_t}(o_i)$).

Critic-guided policy update QMAX One naive update strategy is to utilize the critic's action value as a surrogate self-supervised performance measure. With that, one could potentially update the policy parameters (*i.e.* θ) to maximize the performance measure on received observations at test-time (*i.e.* minimizing $\mathcal{L} = -Q_{\psi_0}(o_i, a_i)$). While intuitive, the aforementioned factors **(i, ii)** make this naive per-step updates brittle and motivate aggregating evidence over time via a small memory buffer.

FIFO buffer. We maintain a bounded on-device FIFO buffer $\mathcal{B}$ of capacity N that stores the most recent observations. At each time step, we append o_t and evict the oldest element if $|\mathcal{B}| > N$.

[1] We use an observation-centric formulation since neither the robot nor our method has access to privileged states.

Every K steps, we sample a minibatch $\{o_i\}_{i=1}^{B}$ uniformly from $\mathcal{B}$ to form a more stable and diverse adaptation batch than a single observation. We note that employing an episodic memory for TTA was also proposed by Yuan et al. [2023].

Critic-guided policy update with buffer (QMAX). Given a pretrained actor–critic pair $(\pi_{\theta_0}, Q_{\psi_0})$, we define a simple self-supervised adaptation objective is to maximize the critic's predicted action value on buffered observations:

$$\mathcal{L}_{\text{QMAX}}(\theta; \mathcal{B}) = -\frac{1}{B}\sum_{i=1}^{B} Q_{\psi_0}(o_i, a_i), \qquad a_i = \pi_\theta(o_i). \tag{1}$$

We then perform a gradient step $\theta_{t+1} \leftarrow \theta_t - \eta \nabla_\theta \mathcal{L}_{\text{QMAX}}|_{\theta=\theta_t}$ with step size $\eta > 0$ while freezing the critic's parameters. While distribution shift can affect the critic's calibration, we empirically find that Q_{ψ_0} still provides a useful surrogate signal for guiding test-time updates in practice. This baseline is lightweight and fully self-supervised, requiring only test-time observations and critic predictions (no rewards or labels). Nevertheless, we found that QMAX is pretty effective in recovering significant portion of the performance drop at test time.

3 Test-time Entropy regularized, Memory guided Policy Optimization

In the previous section, we formulated online test-time adaptation for a pretrained policy under distribution shift and introduced QMAX, which adapts the policy by maximizing a pretrained critic on a small buffer of recent observations. As we demonstrate empirically in Section 5.1, QMAX can be effective, but it may become unstable under severe shifts: when the critic is queried out of distribution, directly ascending $Q_{\psi_0}(o, a_t)$ may yield overly sharp updates, reduced action coverage, and brittle closed-loop behavior. We propose **TEMPO** (*Test-time Entropy-regularized, Memory-guided Policy Optimization*), which improves stability and sample-efficiency by combining two lightweight modifications: (i) An entropy-regularized critic objective, and (ii) a score-based memory that prioritizes informative test-time observations. As described in Section 2.1, TEMPO updates only the policy parameters θ at test time.

3.1 Entropy Maximization

While action perturbation broadens the local neighborhood explored by each update, it does not explicitly discourage overconfident critic-driven collapse. TEMPO therefore augments critic maximization with an entropy term defined on the critic's predictive distribution, encouraging updates that are high-value yet avoid overly peaky solutions.

Distributional critic and entropy. Following Seo et al. [2025], we assume the actor–critic pretraining provides a distributional critic $Z_{\psi_0}(o, a) \in \mathcal{P}(\mathcal{Z})$ where $\mathcal{P}(\mathcal{Z})$ is the probability simplex over a discretized value support $\mathcal{Z}$, with $Q_{\psi_0}(o, a) = \mathbb{E}_{z \sim Z_{\psi_0}(o,a)}[z]$. To avoid the high confidence regions over the critic's prediction we propose an **Entropy-regularized objective.** TEMPO optimizes the following entropy-regularized loss on buffered observations with:

$$\mathcal{L}_{\text{TEMPO}}(\theta; \mathcal{B}) = -\frac{1}{B}\sum_{i=1}^{B} \left[Q_{\psi_0}(o_i, a_i) \;+\; \lambda\, \mathcal{H}(Z_{\psi_0}(o_i, a_i)) \right], \tag{2}$$

where $\lambda > 0$ controls the strength of entropy regularization. The entropy term tempers overconfident maximization by favoring updates that remain robust in regions where the critic's predictions are less concentrated, improving stability under large shifts.

3.2 Score-Based Memory

Next, we ask the following question: given the memory and compute constraints for on-device adaptation, which samples shall we store? And which one shall we adapt on? FIFO buffers treat all recent observations as equally useful, even though under distribution shift only a subset of encountered states may be responsible for performance degradation (e.g., slip events or destabilizing contacts). TEMPO therefore replaces FIFO with a score-based memory that preferentially stores and samples the most informative observations.

Scoring and buffer maintenance. Given the current policy and pretrained critic, we assign each observation a score

$$S(o) = \frac{1}{Q_{\psi_0}\big(o, \pi_\theta(o)\big) + \varepsilon}, \tag{3}$$

with a small constant $\varepsilon > 0$ for numerical stability. High scores correspond to low predicted value and thus "hard" states. TEMPO maintains a buffer $\mathcal{B}$ of capacity N storing pairs $(o, S(o))$. When a new observation o_t arrives, we compute $S(o_t)$ and insert it if it exceeds the minimum score currently in the buffer; in that case, the lowest-scoring element is evicted. This yields a compact memory that tracks the most challenging deployment observations under the current policy.

Importance sampling for adaptation. During each adaptation step, TEMPO samples a minibatch from $\mathcal{B}$ using probabilities proportional to the stored scores:

$$p_j \;=\; \frac{\hat{S}_j}{\sum_k \hat{S}_k}, \qquad \text{where } \hat{S}_j = S_j - \min_i S_i + \varepsilon \tag{4}$$

where p_j is the probability of sampling the jth element from the memory, and S_j is computed according to Eq. (3). TEMPO then draws B observations from $\mathcal{B}$ without replacement according to $\{p_j\}$. This focuses limited on-device adaptation compute on states that are most associated with degraded performance, improving sample-efficiency compared to uniform FIFO sampling.

Putting All Together. TEMPO combines three lightweight ingredients into a single on-device adaptation loop. At each control step, the robot executes the current deterministic policy $a_t = \pi_{\theta_t}(o_t)$ and records the observation in a bounded memory. Unlike FIFO, TEMPO maintains a score-based buffer that preferentially keeps "hard" observations (Eq. (3)) and samples adaptation minibatches via importance sampling (Eq. (4)). Every K steps, TEMPO performs a single gradient update on θ (with Q_{ψ_0} frozen) by minimizing the entropy-regularized objective in Eq. (2). Figure 1 (right) summarizes the pipeline. A detailed algorithm for adapting with TEMPO is in Appendix A.

4 Related Work

Domain Shifts on Robots Robots deployed outside controlled training conditions face *domain shifts* such as changes in contact dynamics, inertial properties, and sensing, which can substantially degrade locomotion policies. A common response is to harden policies in simulation via accurate actuator modeling [Tan et al., 2018, Hwangbo et al., 2019, Zhu et al., 2026, Shin et al., 2025], domain randomization [Peng et al., 2018, Lee et al., 2020] and massively parallel training [Rudin et al., 2022, Zakka et al., 2025]. This improves zero-shot robustness within the training envelope. To better handle unstructured environments, many approaches additionally incorporate exteroception (e.g., elevation maps or images) to anticipate terrain-induced shifts [Miki et al., 2022, Lai et al., 2023, He et al., 2025, Yu et al., 2024, Dong et al., 2025], or auxiliary mechanisms (e.g., asymmetric training, distillation, learned estimators) that infer latent factors [Chen et al., 2020, Kumar et al., 2021, Ji et al., 2022, Choi et al., 2023]. However, when deployment conditions drift beyond the training distribution, these approaches often still require retraining or additional structure, motivating lightweight *deployment-time* adaptation under strict hardware constraints. Adapting at deployment time has been explored in adjacent settings: Park et al. [2024] analyze offline RL bottlenecks and propose simple test-time policy improvement methods, while Chen et al. [2023] adapt on-the-fly in single-life deployment by selecting and modulating a *repertoire* of pretrained behaviors using value estimates. In contrast, we target lightweight *on-device* test-time adaptation for locomotion under strict latency and memory budgets, by adapting a *single* deployed actor via critic-guided updates.

Test-Time Adaptation Test-Time Adaptation (TTA) mitigates the performance drop of pre-trained models under test-time distribution shift by adapting at inference using only unlabeled test data [Mancini et al., 2018, Kojima et al., 2022]. Methods either update model parameters online [Liang et al., 2020, Wang et al., 2020] or adapt the test input to better match the training distribution [Gao et al., 2022]. Early approaches focused on updating Batch Normalization statistics [Li et al., 2016], while later work introduced stronger self-supervised objectives, including entropy minimization and stabilized variants [Wang et al., 2020, Niu et al., 2022, 2023], information maximization [Liang et al., 2020], and consistency/teacher–student updates [Yuan et al., 2023], as well as auxiliary-task-driven adaptation [Alfarra et al., 2025]. Most progress has been demonstrated on perception benchmarks—primarily classification under synthetic corruptions and covariate

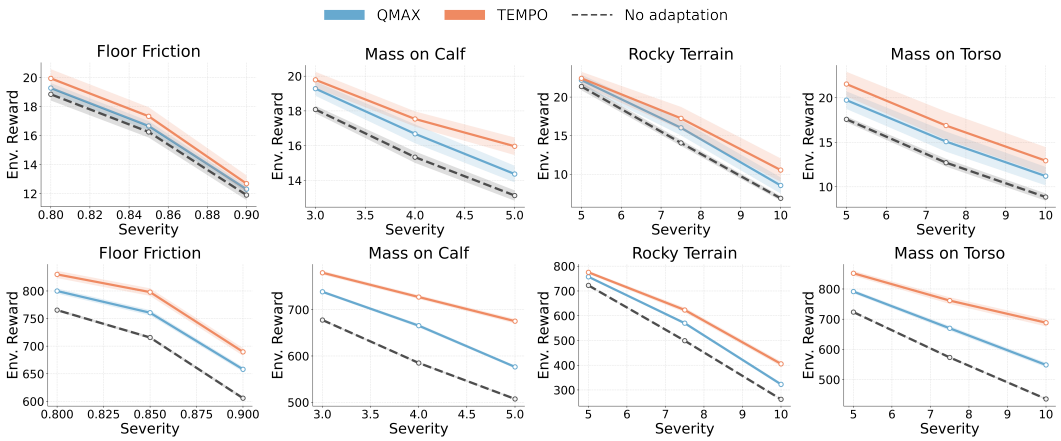

Figure 2: **Main results on Go2 under deployment-time distribution shift.** Top: episodic return. Bottom: average forward linear velocity. Shifts (left→right): 1−friction, calf mass, rough terrain, torso mass (severity increases along x-axis). As the severity of the domain shift increases, performance degrades more rapidly. Naive adaptation via QMAX recovers part of the performance drop. Our proposed TEMPO consistently provides the largest performance gain, especially under severe shifts.

shift [Hendrycks and Dietterich, 2019]—with fewer works on structured prediction [Mirza et al., 2023] or other modalities such as missing modality [Ramazanova et al., 2025]. Translating TTA to robotics is challenging because adaptation must preserve stable closed-loop control under strict latency/memory budgets and highly correlated on-policy streams; our work addresses this by tailoring lightweight, critic-guided TTA to on-device locomotion.

5 Experiments

Environments and pretrained policies. We evaluate TTA on three simulated locomotion tasks from MuJoCo Playground [Zakka et al., 2025]: GO1-JOYSTICK (quadruped), GO2-JOYSTICK (quadruped), and G1-JOYSTICK (humanoid). Since Section 2 places no assumptions on the policy architecture, we consider three representative pretrained policy families: FAST-TD3 [Seo et al., 2025], SIMBAV2 [Lee et al., 2025], and Rapid Motor Adaptation (RMA) [Kumar et al., 2021].

Training protocol and distribution shifts. To disentangle test-time robustness from training-time regularization, we train π_θ on a *flat* terrain *without* domain randomization, and evaluate (and adapt) under a single fixed test environment that emulates a realistic deployment condition. We consider four shift types parameterized by a severity scalar δ: **(i)** *Rough terrain*, with height variation δ; **(ii)** *Calf mass perturbation*, attaching a mass of δ kg to one calf; **(iii)** *Torso mass perturbation*, attaching a mass of δ kg to the torso; and **(iv)** *Slippery terrain*, setting the ground friction coefficient to $1 - \delta$. Unless otherwise stated, each test episode contains a *single* shift type (i.e., no mixed corruptions).

Adaptation protocol. Across all methods, we use Adam with learning rate 5×10^{-6}. We adapt all learnable parameters of π_θ while freezing the critic(s). We maintain a replay buffer of size $|\mathcal{B}| = 16$k and use batches of size $B = 2$k. Note that with 49-dimensional observations and $|\mathcal{B}| = 16$k, the FIFO buffer footprint is merely ≈ 3 MiB; score-based variants require ≈ 3.05 MiB. Adaptation is performed online with period $K = 128$: every K environment steps, we sample a batch from $\mathcal{B}$, compute gradients, and take a single optimizer step. For TEMPO we set the entropy weight to $\lambda = 0.5$. In the following sections, we will provide a comprehensive evaluation demonstrating the benefits of TTA under distribution shift and the consistent performance gains of TEMPO. Unless stated otherwise, we fix the policy architecture to Fast-TD3 and the environment to Go2 on Flat terrain without domain randomization. Further experimental details can be found in Appendix B.1.

5.1 Effectiveness of TEMPO

We first evaluate whether lightweight *test-time adaptation* (TTA) can recover locomotion performance under realistic deployment shifts, and whether TEMPO provides robust gains beyond critic maximization and general-purpose TTA baselines. To do so, we test on four shift families with

Table 1: **Real-world deployment on Unitree Go2 under a calf-mass shift.** TEMPO significantly improves average reward over the non-adapted baseline for various input commands.

v_x	w_z	No adaptation	TEMPO
0.5	0.0	93.2 ± 1.3	$\mathbf{95.2 \pm 0.6}$
-0.5	0.0	98.7 ± 1.6	$\mathbf{113.5 \pm 0.7}$
-0.7	0.0	88.1 ± 1.1	$\mathbf{96.5 \pm 1.0}$
0.7	0.6	88.4 ± 3.3	$\mathbf{101.6 \pm 1.4}$
0.7	-0.6	80.5 ± 1.0	$\mathbf{86.9 \pm 1.2}$
Average		89.8	**98.7**

Figure 3: **Real-world distribution shift on Unitree Go2.** Weight plates mounted on a calf to induce an inertial perturbation during on-device adaptation.



increasing severity: slippery terrain, calf mass perturbation, rough terrain, and torso mass perturbation. Adaptation is performed online during execution using only test-time observations; rewards are never used for updating and are only used for offline evaluation. All results are averaged across six seeds.

Figure 2 reports **episodic return** (top row) and **average forward linear velocity** (bottom row), capturing both task-level performance and the quality of locomotion. Across all shifts, the pretrained policy without adaptation degrades rapidly as severity increases, indicating that even strong policies trained under nominal conditions are brittle when contact dynamics or inertial properties change at deployment. Introducing TTA reliably mitigates this drop: the simple critic-guided baseline QMAX recovers performance across a wide range of severities, confirming that a frozen critic can provide an informative self-supervised signal at test time. However, QMAX often saturates under harder shifts. In contrast, TEMPO is consistently best across all shift families and throughout the severity range, with the clearest advantages emerging in the most challenging regimes. Importantly, these gains are mirrored in forward velocity: TEMPO maintains faster, more continuous locomotion under shift, suggesting that improvements reflect better stabilized behavior rather than reward-only artifacts.

5.2 Real World Deployment

We validate TEMPO on a physical Unitree GO2 quadruped to assess whether the improvements observed in simulation translate to real hardware. To induce a controlled deployment shift, we attach weight plates to both the front left and rear left calves using a custom 3d-printed rigid mount (Fig. 3), altering the leg's inertial properties and contact dynamics while keeping the policy and controller stack unchanged. Adaptation is performed once on-device for 4 minutes and using the same streaming protocol as in simulation: we continuously update the actor parameters online while freezing the critic, using only real observations and critic predictions.

Table 1 reports the environment reward across for various velocity commands $cmd = (v_x, w_z)$, where v_x is the desired forward linear velocity and w_z is the desired angular velocity. The evaluation for each command consists of 5 trials spanning a 3 second window (150 steps each). To define a useful metric for real world performance, we evaluate the undiscounted sum of the per-timestep linear-velocity tracking rewards $r_t = \exp\big(-\|\mathbf{v}_{xy}^{\text{cmd}} - \mathbf{v}_{xy}^{\text{loc}}\|^2\big)$. Thus, each timestep contributes a value in $(0, 1]$, resulting in a maximum possible value of 150. TEMPO outperforms the non-adapted baseline for each of the tested commands, improving the mean score from 89.8 to 98.7. Gains are consistent across command types, with the largest improvements under walking backwards at medium speed ($v_x = -0.5$), where TEMPO increases the score by 15% from 98.7 to 113.5. Overall, these results provide evidence that TEMPO remains effective under real-world distribution shift and can improve locomotion performance in a fully on-device self-supervised setting.

5.3 Mixed Corruption Setting

Real deployments rarely involve a single isolated shift; instead, multiple factors can co-occur (e.g., payload changes together with altered contact dynamics). We therefore evaluate *mixed corruptions* by combining a **payload perturbation** with a **terrain shift** within the same episode. Specifically, we attach a fixed mass either to the **calf** (2 kg) or the **torso** (5 kg), and simultaneously test on either **slippery terrain** ($1-$friction) or **rough terrain** (rocky heightfield), each at two severities.

Table 2: **Performance Comparison Under Mixed Corruptions.** We report the environment reward (six seeds) for the non-adapted policy, QMAX, and our proposed TEMPO when the policy is subjected to a composition of corruptions (e.g. mass on torso and rocky terrain simultaneously). Our proposed TEMPO consistently provides the best performance by a significant margin.

	Mass on Calf = 2 kg				Mass on Torso = 5 kg			
	Floor friction		Rocky terrain		Floor friction		Rocky terrain	
Method	0.90	0.85	0.03	0.06	0.90	0.85	0.03	0.06
No Adapt	$9.2_{\pm 0.1}$	$12.9_{\pm 0.2}$	$19.9_{\pm 0.1}$	$14.5_{\pm 0.2}$	$4.2_{\pm 0.2}$	$6.2_{\pm 0.1}$	$14.7_{\pm 0.3}$	$7.8_{\pm 0.2}$
QMAX	$9.7_{\pm 0.2}$	$13.4_{\pm 0.4}$	$20.7_{\pm 0.1}$	$15.9_{\pm 0.2}$	$4.8_{\pm 0.1}$	$7.0_{\pm 0.3}$	$17.3_{\pm 0.5}$	$9.9_{\pm 0.2}$
TEMPO	$\mathbf{10.4}_{\pm 0.2}$	$\mathbf{14.4}_{\pm 0.1}$	$\mathbf{21.2}_{\pm 0.3}$	$\mathbf{17.5}_{\pm 0.2}$	$\mathbf{5.6}_{\pm 0.2}$	$\mathbf{9.0}_{\pm 0.4}$	$\mathbf{20.1}_{\pm 0.2}$	$\mathbf{14.6}_{\pm 0.2}$

Table 2 summarizes the results. Across *all 8* mixed conditions, TEMPO is best: averaged over the table, it improves the environment reward from 11.18 (NO ADAPT) and 12.48 (QMAX) to 14.09, i.e., a gain of +2.91 (+26.1%) over NO ADAPT and +1.61 (+12.9%) over QMAX. The advantage is especially pronounced for the harder torso-payload compositions, where TEMPO raises the average reward from 8.23 to 12.33 (+4.10, +49.8%), and from 10.00 to 12.33 versus QMAX (+2.33, +23.3%). The largest single-case improvement occurs under torso mass with rocky terrain at the highest severity, where TEMPO reaches 14.6 compared to 10.5 for QMAX (+4.1, +39.0%) and 7.8 without adaptation (+6.8, +87.2%), demonstrating clear benefits under compounding shifts.

5.4 Continual TTA

So far, we evaluated adaptation under a fixed deployment condition. In practice, however, distribution shift is often *non-stationary*: terrain properties can change gradually, payloads can vary over time, and contact dynamics can drift during prolonged operation. We therefore consider a *continual TTA* [Wang et al., 2022] setting in which the deployed policy experiences a sequence of increasingly difficult conditions, implemented by continually increasing the severity of the domain shift during evaluation.

Figure 4 reports average return as adaptation progresses in this non-stationary stream. The results highlight a clear separation between methods: TEMPO exhibits faster recovery after the shift intensifies and reaches a higher performance plateau than QMAX. This gap is most evident as the stream becomes more challenging, where naive critic maximization improves initially but saturates earlier, while TEMPO continues to make progress and maintains stronger steady-state performance. Overall, these results indicate that TEMPO is not only effective under a fixed test condition, but also well-suited to *continual* deployment where the test distribution evolves over time and adaptation must remain stable under compounding shift severity.

Performance on Source Domain. A key concern in continual test-time adaptation is *forgetting*: gains under shifted conditions should not degrade behavior when the environment returns to nominal. To evaluate this, we append the source domain (flat terrain without domain shift) at the end of the continual sequence and directly assess the adapted policies (Appendix B.4, Fig. 9). We observe that both TEMPO and QMAX recover source-domain performance *instantly*, matching the original policy without any transient degradation or additional adaptation, indicating no observable forgetting.

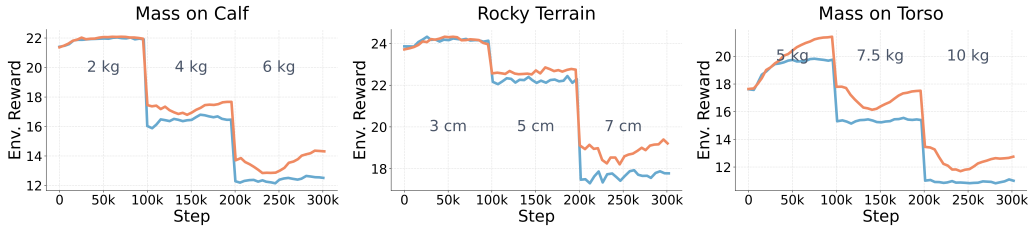

Figure 4: **Continual TTA under progressively increasing shift severity.** We report environment rewards versus adaptation steps as the deployment shift intensifies over time. TEMPO recovers faster and converges to a higher performance plateau than QMAX across all corruptions.

Table 3: **Ablating TEMPO.** Average episodic return under increasing shift severity for three corruption families. Adding entropy regularization and score-based memory each improves over QMAX, and combining both (TEMPO) yields the best robustness across all severities.

Corruption	Mass on Calf (kg)		Mass on Torso (kg)		Rocky Terrain (cm)	
Method \ Severity	4	5	5	7.5	7.5	10
No Adapt	15.3	13.0	17.5	12.7	14.0	7.0
QMAX	16.5	14.4	19.7	15.4	16.6	9.0
+ Entropy Max.	17.0	15.3	21.0	17.0	17.3	10.6
+ Score-Based Mem.	17.5	15.9	21.0	17.1	17.8	11.3
+ ALL (TEMPO)	**17.9**	**16.0**	**21.5**	**17.6**	**18.0**	**11.5**

5.5 Ablating Components of TEMPO

Table 3 ablates the two components of TEMPO—entropy regularization and the score-based memory—across increasing severities of payload and terrain shifts. Starting from QMAX, adding *entropy regularization* yields consistent gains of $+0.5$ to $+1.6$ return across the six settings. Replacing FIFO with the *score-based memory* provides larger improvements of $+1.0$ to $+2.3$. Combining both (TEMPO) gives the best results in every column, improving over QMAX by $+1.4$ to $+2.5$. This indicates that the design of the memory can play a crucial role in boosting the adaptation performance. Relative to NO ADAPT, TEMPO improves by $+2.6$ to $+4.9$ per setting, indicating that both components contribute and their combination is complementary.

5.6 RMA as an Adaptation-Aware Backbone

Rapid Motor Adaptation (RMA) [Kumar et al., 2021] explicitly targets fast adaptation to dynamics changes by conditioning the policy on a learned latent embedding that is inferred online during deployment. Unlike the policies considered in previous sections, RMA is therefore not a static pretrained controller, but already incorporates an internal adaptation mechanism, making it a strong test for whether our proposed TEMPO can provide additional gains beyond architecture-specific adaptation.

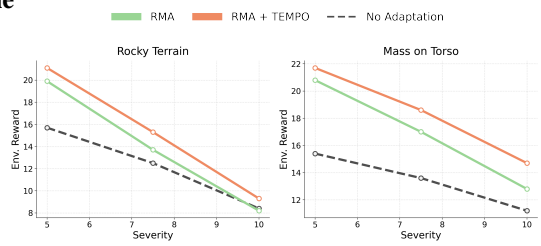

Figure 5: **RMA ablation.** Frozen RMA vs native RMA adaptation vs RMA+TEMPO. TEMPO provides consistent and significant additional gains.

We evaluate three variants under the same shifted test streams: *(i) No Adaptation* (freeze RMA at deployment), *(ii) RMA* (enable RMA's native online adaptation), and *(iii) RMA + TEMPO* (apply TEMPO's critic-guided test-time updates on top of the deployed RMA policy). Figure 5 shows that native RMA adaptation improves over the frozen baseline, while augmenting RMA with TEMPO yields the best performance across the evaluated shift severities, demonstrating that TEMPO complements (rather than replaces) RMA's built-in adaptation. Further experiments on different architectures, and robustness analyses of TEMPO can be found in Appendix B.

6 Conclusions

In this work, we studied reward-free, on-device test-time adaptation for robot locomotion under deployment-time distribution shift. Starting from a lightweight baseline that updates only the actor using a frozen pretrained critic, we introduced TEMPO, which stabilizes critic-guided adaptation through entropy regularization and a score-based memory that prioritizes informative test-time observations. Across simulated locomotion benchmarks and multiple shift types, architectures, and environments, TEMPO consistently improves over no adaptation and naive critic maximization, with the clearest gains under severe shifts. We further confirm the same trend on a physical Unitree Go2 via fully on-device adaptation under a calf-payload perturbation.

References

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid Motor Adaptation for Legged Robots. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.011.

Peng Sun, Qi Li, Hao Hu, Junjie Qiang, Weiwei Wu, and Xin Luo. Proprioceptive slip detection and state estimation of multi-legged robots in slippery scenarios. *Frontiers of Mechanical Engineering*, 20(5):36, 2025.

Alexandros Pouroullis, David Blore, Michael Scott, Julius Smith, Sindiso Mkhatshwa, and Geoff Nitschke. Automating damage recovery in a legged robot. In *2025 IEEE Symposium on Computational Intelligence in Artificial Life and Cooperative Intelligent Systems (ALIFE-CIS)*, pages 1–7. IEEE, 2025.

Yang Zhang, Buqing Nie, Zhanxiang Cao, Yangqing Fu, and Yue Gao. Disturbance-aware adaptive compensation in hybrid force-position locomotion policy for legged robots. *arXiv preprint arXiv:2506.00472*, 2025.

Annie S Chen, Govind Chada, Laura Smith, Archit Sharma, Zipeng Fu, Sergey Levine, and Chelsea Finn. Adapt on-the-go: Behavior modulation for single-life robot deployment. *arXiv preprint arXiv:2311.01059*, 2023.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018. doi: 10.1109/ICRA.2018.8460528.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR, 2020.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33:1179–1191, 2020.

Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. *arXiv preprint arXiv:2103.06257*, 2021.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016. URL https://arxiv.org/abs/1511.05952.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A Kahrs, et al. Mujoco playground. *arXiv preprint arXiv:2502.08844*, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL https://arxiv.org/abs/1801.01290.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018. URL https://arxiv.org/abs/1802.09477.

I Made Aswin Nahrendra, Byeongho Yu, and Hyun Myung. Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5078–5084. IEEE, 2023.

Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8281–8290, 2021.

Yasir Ghunaim, Adel Bibi, Kumail Alhamoud, Motasem Alfarra, Hasan Abed Al Kader Hammoud, Ameya Prabhu, Philip HS Torr, and Bernard Ghanem. Real-time evaluation in online continual learning: A new paradigm. *arXiv preprint arXiv:2302.01047*, 2023.

Motasem Alfarra, Hani Itani, Alejandro Pardo, Shyma Alhuwaider, Merey Ramazanova, Juan C Pérez, Zhipeng Cai, Matthias Müller, and Bernard Ghanem. Evaluation of test-time adaptation under computational time constraints. *International Conference on Machine Learning*, 2024.

Zechu Li, Tao Chen, Zhang-Wei Hong, Anurag Ajay, and Pulkit Agrawal. Parallel q-learning: Scaling off-policy reinforcement learning under massively parallel simulation, 2023. URL https://arxiv.org/abs/2307.12983.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is value learning really the main bottleneck in offline rl? *Advances in Neural Information Processing Systems*, 37:79029–79056, 2024.

Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15922–15932, 2023.

Younggyo Seo, Carmelo Sferrazza, Haoran Geng, Michal Nauman, Zhao-Heng Yin, and Pieter Abbeel. Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control. *arXiv preprint arXiv:2505.22642*, 2025.

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.010.

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

Chengrui Zhu, Zhen Zhang, Weiwei Liu, Siqi Li, and Yong Liu. Learning accurate and robust velocity tracking for quadrupedal robots. *Journal of Field Robotics*, 43(1):76–88, 2026. doi: https://doi.org/10.1002/rob.70028. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.70028.

Young-Ha Shin, Tae-Gyu Song, Gwanghyeon Ji, and Hae-Won Park. Reinforcement learning for high-speed quadrupedal locomotion with motor operating region constraints: Mitigating motor model discrepancies through torque clipping in realistic motor operating region. *IEEE Robotics & Automation Magazine*, 32(2): 49–59, 2025. doi: 10.1109/MRA.2024.3487322.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi: 10.1126/scirobotics.abc5986. URL https://www.science.org/doi/abs/10.1126/scirobotics.abc5986.

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

Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7(62):eabk2822, 2022. doi: 10.1126/scirobotics.abk2822. URL https://www.science.org/doi/abs/10.1126/scirobotics.abk2822.

Hang Lai, Weinan Zhang, Xialin He, Chen Yu, Zheng Tian, Yong Yu, and Jun Wang. Sim-to-real transfer for quadrupedal locomotion via terrain transformer. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5141–5147, 2023. doi: 10.1109/ICRA48891.2023.10160497.

Junzhe He, Chong Zhang, Fabian Jenelten, Ruben Grandia, Moritz Bächer, and Marco Hutter. Attention-based map encoding for learning generalized legged locomotion. *Science Robotics*, 10(105):eadv3604, 2025. doi: 10.1126/scirobotics.adv3604. URL https://www.science.org/doi/abs/10.1126/scirobotics.adv3604.

Alan Yu, Ge Yang, Ran Choi, Yajvan Ravan, John Leonard, and Phillip Isola. Learning visual parkour from generated images. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=cGswIOxHcN.

Yinzhao Dong, Ji Ma, Liu Zhao, Wanyue Li, and Peng Lu. Marg: Mastering risky gap terrains for legged robots with elevation mapping. *IEEE Transactions on Robotics*, 41:6123–6139, 2025. doi: 10.1109/TRO.2025.3619041.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on robot learning*, pages 66–75. PMLR, 2020.

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

Suyoung Choi, Gwanghyeon Ji, Jeongsoo Park, Hyeongjun Kim, Juhyeok Mun, Jeong Hyun Lee, and Jemin Hwangbo. Learning quadrupedal locomotion on deformable terrain. *Science Robotics*, 8(74):eade2256, 2023. doi: 10.1126/scirobotics.ade2256. URL https://www.science.org/doi/abs/10.1126/scirobotics.ade2256.

Massimiliano Mancini, Hakan Karaoguz, Elisa Ricci, Patric Jensfelt, and Barbara Caputo. Kitting in the wild through online domain adaptation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1103–1109. IEEE, 2018.

Takeshi Kojima, Yutaka Matsuo, and Yusuke Iwasawa. Robustifying vision transformer without retraining from scratch by test-time class-conditional feature alignment. *arXiv preprint arXiv:2206.13951*, 2022.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR, 2020.

Jin Gao, Jialing Zhang, Xihui Liu, Trevor Darrell, Evan Shelhamer, and Dequan Wang. Back to the source: Diffusion-driven test-time adaptation. *arXiv preprint arXiv:2207.03442*, 2022.

Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pages 16888–16905. PMLR, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. *International Conference on Learning Representations*, 2023.

Motasem Alfarra, Alvaro Correia, Bernard Ghanem, and Christos Louizos. Test-time adaptation with source based auxiliary tasks. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=XWAXcxNg4n.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

Muhammad Jehanzeb Mirza, Pol Jané Soneira, Wei Lin, Mateusz Kozinski, Horst Possegger, and Horst Bischof. Actmad: Activation matching to align distributions for test-time-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24152–24161, 2023.

Merey Ramazanova, Alejandro Pardo, Bernard Ghanem, and Motasem Alfarra. Test-time adaptation for combating missing modalities in egocentric videos. In *International Conference on Learning Representations*, volume 2025, pages 82755–82774, 2025.

Hojoon Lee, Youngdo Lee, Takuma Seno, Donghu Kim, Peter Stone, and Jaegul Choo. Hyperspherical normalization for scalable deep reinforcement learning. *arXiv preprint arXiv:2502.15280*, 2025.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022.

Ross Hartley, Maani Ghaffari Jadidi, Jessy W Grizzle, and Ryan M Eustice. Contact-aided invariant extended kalman filtering for legged robot state estimation. *arXiv preprint arXiv:1805.10410*, 2018.

A Test-time Entropy regularized, Memory guided Policy Optimization

In Section 3, we proposed TEMPO, an efficient, on-device friendly TTA method for adapting robotic policies. In the construction of TEMPO, we followed Seo et al. [2025] in accessing the distributional critic. We note that the output of the distributional critic is nothing but the output of the penultimate layer of the critic network. Refer to the works of Seo et al. [2025], Lee et al. [2025] for further details. Due to the space constraints, we did not provide a full explicit algorithm to describe how one can implement TEMPO in practice. For completeness, we provide these algorithms in this Section.

Algorithm 1 summarizes the high level interaction between the test stream and the pretrained policy equipped with our proposed TEMPO. Given an observation vector o_t, the policy decides on the action a_t. TEMPO leverages this observation vector and its corresponding action in updating its memory bank $\mathcal{B}$ according to Algorithm 2. When there are enough samples in the memory to conduct an adaptation step, TEMPO samples a batch with importance sampling according to Eq. 4 and runs a single update step with the loss in Eq 2. We note that for efficiency purposes, TEMPO conducts a single update step every K received observations to keep the additional compute requirement light.

Algorithm 2 describes the score based memory update that we proposed in section 3.2. In a nutshell, as long as the memory bank not full, we store all received observation accompanied by their corresponding score. When hitting the capacity limit, we check if the newly received observation has a better score than the sample with the lowest score in memory, then we evict that sample and store the newly received one. We note that for additional implementation efficiency purposes, one can leverage directly the action from step 4 in Algorithm 1 instead of recomputing it in step 1 in Algorithm 2 (one less forward pass).

B Experiments

In Section 5, we provided comprehensive evaluation of how TEMPO is effective under different scenarios in both simulation and real world setting. In this section, we provide additional details about the experimental setup for full reproducibility, and additional analysis on TEMPO.

B.1 Setup.

The policy observation consists of proprioceptive and command-related signals commonly used in legged locomotion. Specifically, it includes IMU measurements in the form of angular velocity (gyro) and a gravity-aligned vector, base linear velocity, joint angles expressed relative to a default pose, joint velocities, the previous action, and a commanded motion vector (v_x, v_y, w_z) specifying desired forward/lateral linear velocities and yaw rate. The critic uses a symmetric, non-privileged observation space identical to the policy input, which enables on-device test-time adaptation directly from robot sensors. Since our real-world Go2 robot does not enable linear velocity estimation out of the box, we utilize an Extended Kalman Filter to calculate velocities on the fly using the built-in foot force sensors [Hartley et al., 2018].

Algorithm 1: TEMPO (high-level online loop)

Input: pretrained π_{θ_0}; frozen critic Q_{ψ_0}; step size η.
Output: actions $\{a_t\}_{t\geq 0}$ and adapted parameters $\{\theta_t\}_{t\geq 0}$.

1: Initialize θ_0, $\mathcal{B} \leftarrow \emptyset$
2: **for** $t = 0, 1, 2, \ldots$ **do**
3: Observe $o_t \sim \hat{P}(o_{t-1}, a_{t-1})$
4: Act $a_t \leftarrow \pi_{\theta_t}(o_t)$
5: $\mathcal{B} \leftarrow \textsc{UpdateMemory}(\mathcal{B}, o_t)$
6: **if** $\textsc{ShouldAdapt}(t, \mathcal{B})$ **then**
7: $\theta_{t+1} \leftarrow \text{TempoUpdate}(\theta_t, \mathcal{B}, Q_{\psi_0})$
8: **else**
9: $\theta_{t+1} \leftarrow \theta_t$
10: **end if**
11: **end for**

SHOULDADAPT: true every K steps when $|\mathcal{B}| \geq B$.

Algorithm 2: UpdateMemory (Score Based)

Input: observation o; memory bank $\mathcal{B}$;
Output: updated memory bank $\mathcal{B}$

1: $a \leftarrow \pi_\theta(o)$
2: $S \leftarrow \frac{1}{Q_{\psi_0}(o,a)+\varepsilon}$ {Eq. 3}
3: **if** $|\mathcal{B}| < N$ **then**
4: Insert (o, S) into $\mathcal{B}$
5: **else**
6: $(o_{\min}, S_{\min}) \leftarrow \arg\min_{(o',S')\in\mathcal{B}} S'$
7: **if** $S > S_{\min}$ **then**
8: Remove $(o_{\min}, S_{\min})$ from $\mathcal{B}$
9: Insert (o, S) into $\mathcal{B}$
10: **end if**
11: **end if**

TEMPOUPDATE: Sample with Eq.4 and optimize Eq 2.

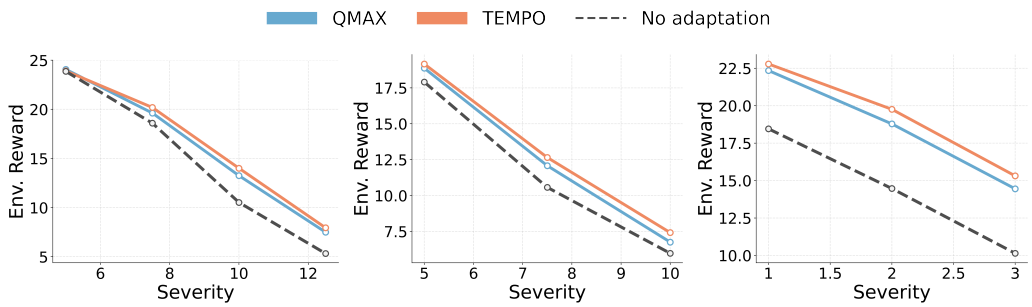

Figure 6: **Generalization across architectures and embodiments.** Rocky-terrain results (left→right): Fast-TD3 trained with domain randomization, SimBA-V2, and the G1 humanoid. TEMPO transfers without architecture-specific modifications and remains effective across settings.

B.2 Generalization to Different Architectures and Environments

A central goal of our approach is *portability*: test-time adaptation should improve a deployed controller without relying on architecture-specific components, privileged signals, or task-tailored objectives. We therefore evaluate whether TEMPO transfers beyond our default setting (Fast-TD3 on Go2) to substantially different pretrained policy families and to a different embodiment, without modifying the algorithm or introducing additional modules.

Figure 6 summarizes this evaluation on the *rocky terrain* test environment. On the left, we consider a Fast-TD3 policy trained *with domain randomization*, representing a stronger robustness-oriented pretraining baseline. Table 4 reports results on Go1 and G1 under mass perturbations. Even in this regime, TEMPO continues to provide consistent improvements, indicating that its gains are not limited to policies trained without robustness regularization and that it can complement domain randomization at deployment. In the middle, we adapt a policy trained with the SIMBA-V2 architecture, which differs in inductive biases and learning dynamics from Fast-TD3; TEMPO remains effective without any architecture-specific tuning, supporting the claim that our objective and update rule are broadly applicable to actor–critic policies. Finally, on the right, we evaluate on G1, a humanoid with markedly different morphology and contact dynamics. Despite this shift in embodiment, TEMPO still improves performance on rocky terrain, suggesting that the method generalizes beyond quadrupeds and can operate in more challenging control regimes.

B.3 Sensitivity to Hyperparameters, Memory Size, Batch Size, and Adaptation Interval

On-device test-time adaptation must operate under strict memory and compute budgets; accordingly, we study how TEMPO behaves as we vary the three deployment-critical hyperparameters that directly control resource usage: replay memory size $|\mathcal{B}|$ (storage footprint), minibatch size B (compute per update), and adaptation interval K (update frequency, i.e., one gradient step every K environment steps). Figure 7 reports performance under distribution shift across these ablations.

Overall, TEMPO is robust across a broad range of settings and remains the best-performing method throughout the three sweeps. Importantly, this robustness translates into practical efficiency on device. First, TEMPO maintains strong performance even with relatively small $|\mathcal{B}|$, implying that effective adaptation does not require storing large histories; this keeps the on-device memory footprint small and compatible with embedded constraints. Second, TEMPO performs well with small minibatches,

Table 4: Additional results on Go1 and G1 under payload perturbations. MC denotes mass on calf and MT denotes mass on torso.

	Go1				G1		
Method	MC=4	MC=5	MT=7.5	MT=10	MT=2	MT=3	MT=5
No Adapt	18.9	16.3	13.6	6.8	21.3	17.5	10.0
QMAX	19.6	17.6	13.6	6.9	24.1	22.1	17.4
TEMPO	**20.5**	**18.6**	**16.9**	**11.0**	**24.5**	**23.4**	**18.5**

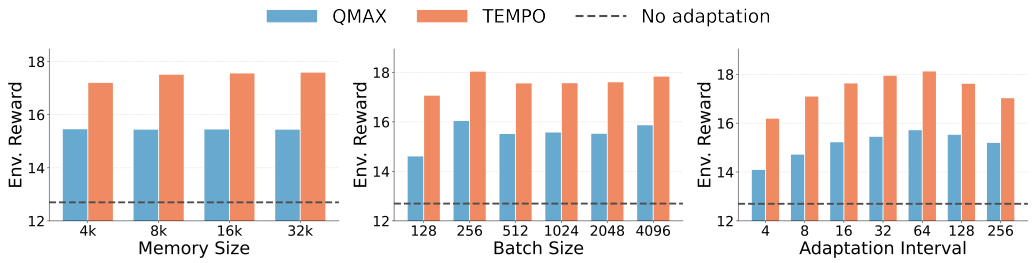

Figure 7: **Robustness to on-device adaptation hyperparameters.** TEMPO is robust and performs best across different memory sizes (left), batch sizes (middle), and adaptation intervals (right). All experiments are run with Fast-TD3 on Go2 with Mass on Torso Corruption with severity of 7.5 kg.

showing that adaptation can be carried out with limited per-update compute (and correspondingly lower latency), rather than relying on large, expensive updates. Third, TEMPO remains effective for larger adaptation intervals K, meaning updates can be applied infrequently—reducing the total number of backward passes per unit time and making the method applicable even when compute budgets are very tight.

As expected, extreme settings reveal intuitive trade-offs. Very small memories reduce state diversity and can limit coverage of informative "hard" states, while very large memories yield diminishing returns given a fixed update schedule. Similarly, very small batches increase gradient noise, whereas very large batches offer limited additional benefit relative to their higher compute cost. Finally, K exposes a responsiveness–efficiency trade-off: smaller K increases adaptation frequency but raises compute load, while larger K lowers overhead at the cost of slower reaction to abrupt changes. Taken together, the results indicate that TEMPO provides stable improvements without delicate tuning and can be configured to meet stringent on-device constraints via small $|\mathcal{B}|$, small B, and large K.

At last, we assess the performance of TEMPO when varying its only hyperparameter; λ. We follow the experimental setup in Figure 7 and vary $\lambda \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$ and report the results in Figure 8. We observe a stable behavior under different values of λ.

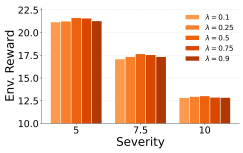

Figure 8: **Ablating λ.**

B.4 Performance on Source Domain

A key concern in continual test-time adaptation is catastrophic forgetting: parameters adapted to severe deployment shifts may degrade performance once the environment returns to nominal conditions. This risk is particularly acute in on-device, online settings, where updates are applied continuously without explicit supervision, replay from the source domain, or reset mechanisms. To assess whether critic-guided test-time adaptation induces such forgetting, we evaluate both QMAX and TEMPO after prolonged exposure to increasingly severe distribution shifts, followed by a return to the source domain (flat terrain without perturbations). Specifically, we adopt the continual TTA protocol from Section 5.4 and append a final evaluation phase in which the environment dynamics match the original training conditions, while keeping the adapted policy parameters fixed—without additional fine-tuning or recovery steps.

Figure 9 reports performance for two representative shift families (mass on calf and mass on torso) using the Fast-TD3 architecture. In both cases, QMAX and TEMPO immediately recover source-domain performance, matching the non-adapted policy without any transient degradation. This indicates that critic-guided test-time updates do not overwrite nominal behaviors, and that adaptation remains localized to state regions associated with deployment-time shifts. Overall, these results suggest that both methods—particularly TEMPO—support stable continual adaptation without observable forgetting, making them well suited for long-horizon deployment scenarios in which operating conditions drift temporarily but may later revert to nominal regimes.

Further, and to complement the previous experiments, We evaluate the trained policy under the source domain (flat terrain, no shift) with no adaptation, QMAX, and TEMPO, across all three reward signals used in the paper (averaged across six seeds). We observe that all three methods are numerically

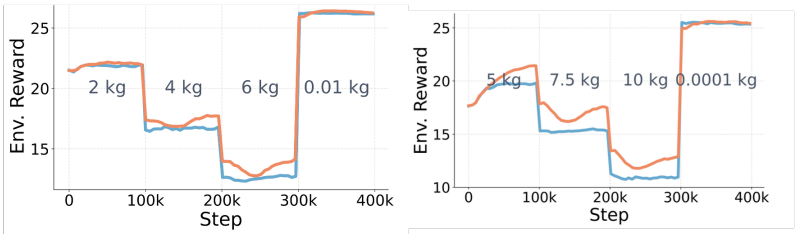

Figure 9: **Continual TTA followed by return to the source domain.** Performance under a non-stationary deployment stream in which the severity of the domain shift increases over time and is then reset to nominal conditions. Left: mass-on-calf shift. Right: mass-on-torso shift. Architecture: Fast-TD3. Both QMAX and TEMPO recover source-domain performance immediately after the shift is removed, indicating no catastrophic forgetting despite prolonged adaptation under shift.

indistinguishable, and both velocity-tracking signals sit within run-to-run noise (environment reward of 25.6, Linear Velocity tracking reward of 864, and Angular Velocity Tracking reward of 457).

B.5 Improving Angular Velocity Tracking

While the main paper focuses on task-level performance and forward locomotion metrics, stable angular velocity tracking is critical for practical deployment, especially under distribution shifts that alter contact dynamics or inertial properties. Poor tracking of angular velocity often manifests as oscillatory turning, delayed heading response, or loss of stability, which may not be fully reflected in episodic return alone. To examine this aspect more closely, we evaluate the angular velocity tracking performance of QMAX and TEMPO on Go2 under the same deployment-time distribution shifts considered in Section 5.1 For each method, we measure the tracking angular velocity reward, and report results averaged across 6 random seeds.

Figure 10 summarizes the results across four shift families with increasing severity. Without adaptation, the pretrained policy exhibits rapidly degrading angular velocity tracking as the shift intensifies. Introducing critic-guided test-time adaptation substantially mitigates this degradation: QMAX consistently improves tracking accuracy relative to no adaptation, while TEMPO provides the most stable and accurate tracking across all severities. The gains are most pronounced under harder shifts, such as rocky terrain and torso mass perturbations, where overconfident critic-driven updates can otherwise lead to unstable turning behavior. Overall, these results indicate that the benefits of TEMPO extend beyond forward velocity and reward metrics to finer-grained control objectives, demonstrating improved stability and control fidelity under deployment-time distribution shift.

B.6 Does TEMPO Improve Local Action Coverage?

To analyze the impact of TEMPO on the local action coverage, we measure the critic's distributional entropy over the course of adaptation. On the severe mixed-corruption setting (Mass on Torso=5, Rough Terrain=0.06; rightmost column of Table 2), starting from an initial entropy of 1.15, QMAX drives the entropy of $Z_{\psi_0}(o, \pi_\theta(o))$ down to 1.05 while TEMPO raises it to 1.41 — a 34% relative gap

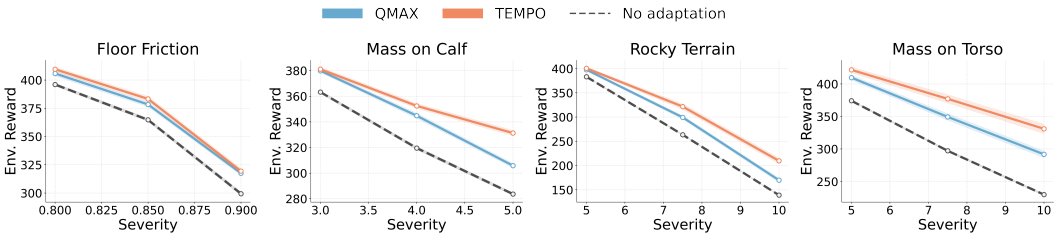

Figure 10: **Angular velocity tracking on Go2 under deployment-time distribution shift.** We report angular velocity tracking error as a function of shift severity for four corruption families. Results are averaged over five seeds. Critic-guided test-time adaptation improves tracking accuracy relative to no adaptation, with TEMPO achieving the most stable and accurate angular velocity tracking, particularly under severe distribution shifts.

that is temporally aligned with the return gap (9.9 vs 14.0, +41% relative). Elevated critic entropy preserves broader gradient support and thus wider local action coverage, exactly the mechanism the regularizer is designed to induce. This is also consistent with our results in Table 3, where removing the entropy term reduces performance.

B.7 RMA Setup

Due to the space constraints, we breifly outlined the experimental setup of RMA in Section 5.6. Here we provide the remaining training details for full reproducibility. RMA policy is trained on Go2JoystickFlatTerrain with MuJoCo Playground domain randomization enabled. The training randomization covers floor friction U(0.4,1.0), joint friction scale U(0.9,1.1), armature scale U(1.0,1.05), center-of-mass shift U(-0.05,0.05), link-mass scale U(0.9,1.1), torso mass offset U(-1,1) kg, and initial joint jitter U(-0.05,0.05). The RMA target conditions are deliberately outside these ranges: e.g., floor friction 0.1–0.2, torso payload 5–10 kg, calf payload 3–5 kg, and rough-terrain height 0.05–0.1, where terrain height is not part of the flat-terrain training randomization.

B.8 Refreshing Scores of Memory

Under long hoizon streams, scores in the memory bank for previously observed observations could become stale hindering the positive impact of adaptation. A natural fix to this problem is to integrate refreshing the scores for the observations in the memory bank. For that, we equipped TEMPO with either of the following two score refreshing variants: (i) periodic score refresh, and (ii) age-decayed scoring, where Eq. (3) is replaced by $\hat{S}(o) = S(o)/(1 + \gamma \cdot \mathrm{age}(o))$ with $\mathrm{age}(o)$ the number of steps o has resided in the memory. We swept $\gamma \in 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100$. We observe that both variants added per-step compute without improving returns over the current minimum-score replacement rule. We attribute this to the limited state diversity of our simulation; in real-world, long-horizon deployment with richer diversity, we expect stale-score handling to matter more.

B.9 Comparison to Test-Time Training

Recently, Park et al. [2024] proposed OPEX and Test Time Training as adaptation methods for policies subjected to shifts. In Figure1, we analyzed the efficacy of OPEX. For completeness, we discuss here the performance of TTT. In particular, one can express the adaptation loss for TTT as: $\mathcal{L}_{\mathrm{TTT}} = \mathcal{L}_{\mathrm{QMAX}} + \beta\, \mathrm{KL}(\pi_\theta \,\|\, \pi_{\theta_0})$. Thus, we compare our proposed baseline; QMAX versus TTT. We evaluate on Go2 under the mass-on-calf shift and sweep the regularization coefficient β.

We observe a monotonic decrease in performance as β increases. For MC=4, the return drops from 16.8 at $\beta = 0$ (QMAX) to 16.3, 16.1, 15.7, and 15.6 for $\beta = \{0.02, 0.04, 0.06, 0.08\}$, respectively. Similarly, for MC=5, performance decreases from 14.7 at $\beta = 0$ to 14.4, 13.7, 13.6, and 13.3. Thus, the strongest TTT configuration corresponds to $\beta = 0$, which exactly recovers QMAX.

C Compute Requirements, Limitations, and Broader Impact

Compute Requirements. All our experiments that were conducted on simulation were run on NVIDIA V100 GPUs including training, evaluation, and TTA (each experiment was run on a single GPU). Regarding the real world experiments, we ran our TTA pipeline on the Unitree Go2 robot leveraging four CPU cores (2 for inference stream and 1 for adaptation and 1 for odometry).

Limitations. Our study focuses on reward-free, critic-guided test-time adaptation for locomotion policies under deployment-time distribution shifts, and some limitations remain. First, while TEMPO is designed to be lightweight and on-device friendly, it still relies on access to a pretrained critic that is reasonably informative under shift; if the critic is severely miscalibrated or trained on a highly mismatched domain, adaptation quality may degrade. Second, TEMPO induces an additional, albeit small, memory and compute cost during inference.

Broader Impact. This work aims to improve the robustness of robotic systems under deployment-time distribution shifts, reducing the need for manual recalibration and improving autonomy. However, adaptive systems may introduce risks if deployed without adequate testing and oversight, especially in safety-critical environments. We emphasize that our method is intended to complement, not replace, careful system design, and we encourage responsible use in real-world settings.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims reflect the paper's contributions, as we demonstrated in Sections 2 and 3 (constructing TEMPO), and conducting comprehensive experimental analysis (9 Figures and 3 Tables of results).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix C has an explicit paragraph outlining the main limitations of this work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.

- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer:[N/A]

   Justification: In this paper, we did not propose any new theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We included three paragraphs in Section 5 supported by a full section in the appendix (B.1) to describe all experimental details for full reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.

- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code and data are proprietary.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We included three paragraphs in Section 5 supported by a full section in the appendix (B.1) to describe all experimental details for full reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For all main results (Sections 5.1 and 5.2 – Figures 1, 9, Table 1), we ran our experiments across several random seeds and reported the average along with standard error.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In appendix C, we dedicated a full paragraph discussing the compute resources used in all of our experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research does not involve human subjects, personal data, or crowdsourcing, and all experiments are conducted in simulation or on robotic hardware under controlled conditions. The work focuses on improving robustness and reliability of robotic control, and potential deployment considerations are discussed in the Limitations and Broader Impact sections

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In appendix C, we dedicated a full paragraph discussing the compute resources used in all of our experiments.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

Justification: The paper does not release data or models with a high risk of misuse, such as large-scale generative models, scraped datasets, or user-facing systems. As a result, no additional safeguards for responsible release are required.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The authors of this work are the sole contributors to the assets of this work (code, data models). All relevant references are appropriately cited and credited.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not introduce new datasets or benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not include crowdsourcing experiments or research involving human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The research does not involve human subjects, study participants, or user-facing evaluations. All experiments are conducted in simulation or on robotic hardware without interaction with human participants, and therefore no participant risks, disclosures, or IRB approvals are applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The LLM usage in this paper does *not* impact the core methodology, scientific rigor, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.