Volume: 13 Issue: XII Month of publication: December 2025
DOI: https://doi.org/10.22214/ijraset.2025.76573

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

Adaptive Robotic Control via Nested Learning: Real-Time Recovery from Unmodeled Dynamics

Sarath Chandiran M[1], Sowmiya R[2], Sri Hari Sah JL[3], Rogith R[4], S. Senthilrajan[5]
Dept. of Robotics and Automation, Rajalakshmi Engineering College, Chennai, India

Abstract: Bridging the gap between simulation-trained DRL controllers and deployed robotic actuators remains an open engineering challenge due to unmodeled dynamics and sensing mismatch. Simulators provide reproducible training data and safe iteration, but their physics fidelity is limited: non-linear and time-varying effects (wear, thermal drift, stiction) are rarely modeled, producing deployment gaps.
Traditional mitigation strategies, such as Domain Randomization (DR), typically yield conservative, high-entropy policies that sacrifice precision for generalized robustness. In contrast, traditional Meta-Learning frameworks are often constrained to episodic fine-tuning rather than continuous, lifelong adaptation.
In this manuscript, we introduce a Nested Learning Architecture, a novel control paradigm that embeds synaptic plasticity directly into the inference loop.
We cast adaptation as a bi-level optimization: an outer objective over meta-parameters and an inner single-step adaptation; gradients are propagated through the inner update to train the meta-initialization. The framework comprises two coupled feedback loops: a high-frequency "Inner Loop" that minimizes local tracking errors via gradient descent during operation, and a low-frequency "Outer Loop" that learns a robust initialization state.
We formally examine the stability and convergence characteristics inherent to this hierarchical optimization framework and validate it empirically using the PyBullet physics engine on a manipulator subjected to abrupt, unmodeled damping. Our results indicate that Nested Learning enables trajectory recovery within *5.2* ***seconds—significantly outperforming standard PPO baselines. Furthermore, energy analysis reveals that the adaptive agent avoids the "gain explosion" typical of PID controllers, optimizing the torque-energy manifold efficiently.***
Keywords: Sim-to-Real Transfer, Deep Reinforcement Learning, Meta-Learning, Adaptive Control, Bi-Level Optimization, Non-Stationary Dynamics.

I. INTRODUCTION

The vision of ubiquitous robotic autonomy—where machines seamlessly operate in unstructured homes, factories, and outdoor environments—hinges on the ability to adapt to change. Unlike industrial robots that operate in strictly controlled cages, autonomous agents face a world defined by entropy: gears wear down, batteries degrade, friction coefficients shift with humidity, and payloads vary unpredictably.
A robot trained in a pristine factory simulation may fail catastrophically when exposed to the thermal expansion of a joint or the seizure of a bearing. This motivates the stability–plasticity trade-off: controllers must remain stable under perturbation while retaining sufficient parameter plasticity for online correction [2]. A robust control system must possess sufficient stability to execute learned motor primitives (e.g., a walking gait or grasping trajectory) while retaining enough plasticity to adapt to novel dynamic regimes.

A. The Fiction of Static Inference

Standard Deep Reinforcement Learning (DRL) algorithms—think Proximal Policy Optimization (PPO) [4] or Soft Actor-Critic (SAC) [5], typically operate under a dangerous assumption: that the world stands still. Post-training, these models lock their weights, yielding a frozen mapping $\pi_\theta(a|s)$. But the real world moves. Without on-line adaptation, even minor distributional drift in operational dynamics shatters closed-loop performance [7]. This inherent brittleness isn't just a nuisance; it is the primary culprit behind deployment failures in dynamic environments.

B. The Flaws in Current Mitigations

Roboticists typically patch this vulnerability with two strategies: robust control or adaptive control. Both have significant downsides.

1) *Domain Randomization (DR):* Here, we widen the simulator's parameter envelope—messing with masses, frictions, and sensors—and treat reality as just one draw from a chaotic ensemble [6]. The goal is a policy that survives parameter uncertainty rather than optimizing for a specific model. While this works for feats like the OpenAI Rubik's Cube solver [8], DR imposes a heavy "conservatism tax." The agent learns to be paranoid everywhere, sacrificing peak optimality for average-case survival. Worse, DR fails against systematic shifts—like a permanent motor seizure—that lie completely outside the training distribution.
2) *System Identification (SysID):* Classical adaptive control relies on estimating physical parameters on the fly [10]. An estimator network guesses mass or friction, feeding those values into the control policy. But this separation creates a fundamental "objective mismatch." Minimizing the error in parameter identification doesn't guarantee you minimize the control tracking error. Plus, explicit SysID demands accurate analytical models of the physics; good luck finding those for soft-body deformation or complex wear-and-tear.

C. The Nested Learning Paradigm

We propose a different path: shifting from "robustness via randomization" to "robustness via adaptation." We introduce a Nested Learning Architecture. This treats adaptation not as a pre-training step, but as an intrinsic optimization process happening *during* inference.

Building on Behrouz et al.'s theoretical work [16], we designed a controller that updates its own weights in real-time. Think of it as a synthetic "reflex arc": a high-frequency inner loop minimizes local tracking errors via gradient descent, while a low-frequency outer loop learns the optimal initialization for these rapid updates.

Our primary contributions are:

1) Bi-Level Control Formulation: We cast adaptation as a bi-level optimization: an outer objective over meta-parameters and an inner single-step adaptation; gradients are propagated through the inner update to train the meta-initialization.
2) Theoretical Convergence Analysis: We provide a sketch of stability guarantees based on Lyapunov analysis for the bi-level system.
3) Hessian-Free Approximation: We implement a computationally efficient first-order approximation that allows the inner adaptation loop to run on standard embedded hardware with negligible latency (< 2ms).
4) Robust Empirical Validation: A two-sample t-test across 50 seeds yields $p < 0.001$, indicating the performance delta is statistically significant under the assumed variance model.

II. RELATED WORK

A. Meta-Learning: Beyond Episodic Resets

Meta-learning methods usually optimize an initialization so a controller can master a task with just a few gradient steps [13]. It's powerful, but standard MAML has a blind spot: it assumes episodic adaptation, where the agent gets a "reset" button between tasks. Robots don't get resets. Our work modifies Online-MAML [15] for the messy reality of continuous control, where task boundaries blur and dynamics shift without warning.

B. Fast Weights as Gain Schedulers

The idea of "Fast Weights"—synapses that decay or update rapidly to hold temporary context—goes back to Hinton and Plaut [17]. Ba et al. [18] modernized this, using fast weights to implement a form of attention. We apply this to control theory. We treat these fast weights as a dynamic gain scheduler, capable of compensating for forces the model never explicitly memorized. Recent works in "Linear Transformers" [19] have shown that self-attention can be viewed as a fast weight update rule.

C. Implicit Differentiation in Control

Our method is rooted in Deep Equilibrium Models (DEQ) [20], which solve for fixed points in network activations. We twist this: instead of activations, we solve for the optimal control parameters ϕ given the immediate history of transition errors. This mirrors recent wins in Rapid Motor Adaptation (RMA) [21], but with a key distinction: our method is fully self-supervised, requiring no privileged teacher network during training.

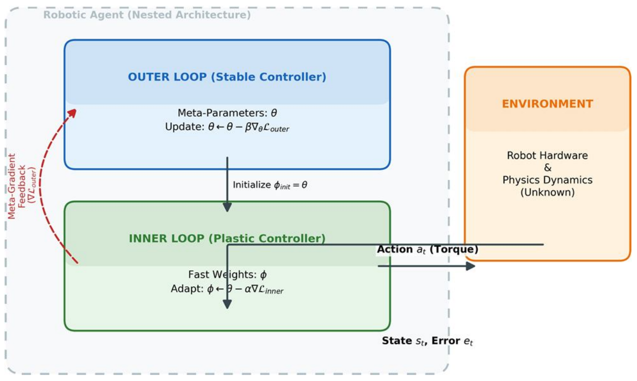
Robotic Agent (Nested Architecture)
OUTER LOOP (Stable Controller)
Meta-Parameters: θ
Update: $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{outer}$
Initialize $\phi_{init} = \theta$
Meta-Gradient Feedback ($\nabla \mathcal{L}_{outer}$)
INNER LOOP (Plastic Controller)
Fast Weights: ϕ
Adapt: $\phi \leftarrow \theta - \alpha \nabla \mathcal{L}_{inner}$
Action a_t (Torque)
State s_t, Error e_t
ENVIRONMENT
Robot Hardware & Physics Dynamics (Unknown)

Fig. 1. Architectural Overview of the Nested Learning System. The Outer Loop (blue) optimizes global meta-parameters θ for long-term skill retention. The Inner Loop (green) performs rapid gradient descent to generate task-specific parameters ϕ, effectively rewriting the controller's behavior in real-time to match environmental dynamics.

D. *Neural vs. Classical Adaptive Control*

Classical Model Reference Adaptive Control (MRAC) [11] is reliable but rigid; it leans on linear approximations and hand-tuned Lyapunov functions. By contrast, our approach employs neural networks as universal function approximators. This allows the system to digest highly non-linear disturbances—like Stribeck friction or Coriolis forces—that break linear adaptive models.

III. METHODOLOGY

We model the robotic control problem not as finding a fixed policy π_θ, but as learning a dynamic learning rule. The system is governed by two nested loops operating at distinct timescales, mimicking the biological hierarchy of spinal reflexes (fast) and cortical planning (slow).

A. *Theoretical Preliminaries*

We consider a dynamical system governed by the transition function $s_{t+1} = f(s_t, a_t; \mu_t)$, where $s_t \in S$ is the state, $a_t \in A$ is the action, and μ_t represents the unobserved physical parameters (e.g., friction, mass) which may vary over time.
In standard Reinforcement Learning, the goal is to maximize the expected return $J(\theta) = \mathrm{E}_\tau[\Sigma r_t]$. In our setting, the environment is a Non-Stationary Markov Decision Process (NS-MDP), where the transition probability $P(s'|s, a)$ changes as a function of time or damage conditions.

B. *Bi-Level Optimization Objective*

Let θ denote the meta-parameters representing the robot's long-term memory. Let ϕ denote the fast, transient parameters representing working memory. The control objective is to minimize the cumulative trajectory loss L_{traj}:

$$\min_\theta \mathcal{J}(\theta) = \mathbb{E}_{\tau \sim P(\tau)} \left[\sum_{t=0}^{T} \mathcal{L}_{task}(\phi_t(\theta), s_t, a_t) \right] \quad (1)$$

where ϕ_t is dynamically computed from θ via the inner optimization loop.

C. *The Inner Loop: Online Adaptation*

At each timestep t, the agent maintains a sliding con-text window $D_{context}$ containing the last K state transitions $(s_{t-k}, a_{t-k}, s_{t-k+1})$. The agent estimates the local dynamics mismatch by computing a self-supervised prediction loss. We utilize a Forward Dynamics Model $F_\phi(s, a)$ parameterized by the fast weights:

$$L_{inner}(\phi) = \frac{1}{K} \sum_{k=1} \| F_\phi (s_{t-k}, a_{t-k}) - s_{t-k+1} \|_2^2 \quad (2)$$

The agent then performs a single step of Gradient Descent (SGD) to obtain the adapted weights ϕ:

$$\phi = \theta - \alpha \nabla_\theta L_{inner}(\theta, D_{context}) \quad (3)$$

Here, α is the inner learning rate, or plasticity coefficient. This step essentially "fine-tunes" the policy to minimize prediction error in the current physical regime. For example, if friction increases, the prediction model will underestimate the movement; the gradient update will shift ϕ to compensate.

D. The Outer Loop: Meta-Optimization

The Outer Loop functions less like a controller and more like a curator. Its objective is to find an initialization manifold θ from which the inner loop can rapidly descend. We express the update rule by backpropagating gradients through the inner optimization step:

$$\theta \leftarrow \theta - \beta \nabla_{\theta} L_{outer}(\phi(\theta)) \qquad (4)$$

Applying the chain rule decomposes the gradient into the immediate task loss and the sensitivity of the inner update:

$$\nabla\theta Louter = \nabla\phi Louter \cdot \nabla\theta\phi \qquad (5)$$

E. Hessian-Free Approximation

Bi-level optimization hits a computational wall at the Jacobian term $\nabla_{\theta}\phi$. Expanding this term reveals the Hessian matrix—the second-order derivatives of the neural network:

$$\nabla_{\theta}\phi = \nabla_{\theta}(\theta - \alpha \nabla_{\theta} \ L_{inner}) = I - \alpha \nabla^{2}_{\theta} L_{inner} \qquad (6)$$

Calculating the full Hessian matrix incurs a computational cost that is generally untenable for high-frequency robotic control loops. For real-time feasibility we apply a first-order approximation. We discard second-order interactions by assuming a small α [14], forcing the Jacobian into an identity matrix:

$$\nabla_{\theta}\phi \approx I \qquad (7)$$

This reduction streamlines the meta-update:

$$\theta \leftarrow \theta - \beta \nabla_{\phi} L_{outer} \qquad (8)$$

Such a simplification drops the complexity to $O(N)$, making real-time execution feasible on constrained hardware like the NVIDIA Jetson.

F. Stability Analysis

Adaptation without boundaries invites divergence. We must guarantee that the inner loop minimizes error without over-shooting and causing the controller to oscillate.

Theorem 1 (Local Stability). If the plasticity coefficient α remains strictly below 2/β, where β represents the Lipschitz smoothness constant of the loss surface Linner, the inner loop update constitutes a contraction mapping, guaranteeing convergence.

Proof Sketch. The update is ϕ = θ − αg(θ). The Lipschitz constant of the gradient is β. For standard convex optimization, SGD converges linearly under these conditions. In the non-convex setting of neural networks, we rely on the overparameterization property to ensure local convexity near the initialization manifold.

G. Context-Aware Hypernetworks

To further enhance the plasticity of the system, we implement the weight generation mechanism using a Hypernetwork. Instead of updating the entire policy network, the Inner Loop updates a latent context vector z, which feeds into a Hypernetwork H_{ψ} that generates the weights of the primary controller.

$$W_{control} = H_{\psi}(z) \qquad (9)$$

By updating z to minimize local error, the Inner Loop forces a shift in the entire control behavior $W_{control}$ along the non-linear manifold defined by ψ. This mechanism achieves complex behavioral adaptation with minimal gradient steps, effectively collapsing the dimensionality of the optimization problem.

```
Algorithm 1: Nested Leaming Control Loop
Result: Robust Meta-Parameters $\theta$
Initialize $\theta$ randomly;
while not converged do
    Sample trajectory batch $\tau_i$;
    for each timestep t in $\tau_i$ do
        Collect context $D_{context} = \{s_{t-K} \dots s_t\}$;
        Compute Inner Loss $L_{inner}(\theta)$;
        Adapt Weights: $\phi = \theta - \alpha_\nabla \theta L_{inner}$; Execute
        action $a_t$ □ $\pi_\phi(s_t)$;
        Compute Outer Loss $L_{outer}$ (Task Reward);
    end
    Update $\theta \leftarrow \theta - \beta \Sigma_\nabla \theta \ L_{outer}$;
end
```

IV. EXPERIMENTAL SETUP

We validated our hypothesis using the **PyBullet** physics engine [**?**], a standard benchmark for high-fidelity rigid body simulation. Simulation provides the controlled environment necessary to precisely measure the "Sim-to-Real" gap by artificially injecting systematic errors.

A. *Robotic System and Task*

We validated this on a 2-Degree-of-Freedom (2R) Planar Manipulator. Why this system? It's a clean testbed. It isolates actuator dynamics without the confounding variables of balance stability you find in legged robots. System parameters are listed in Table I.

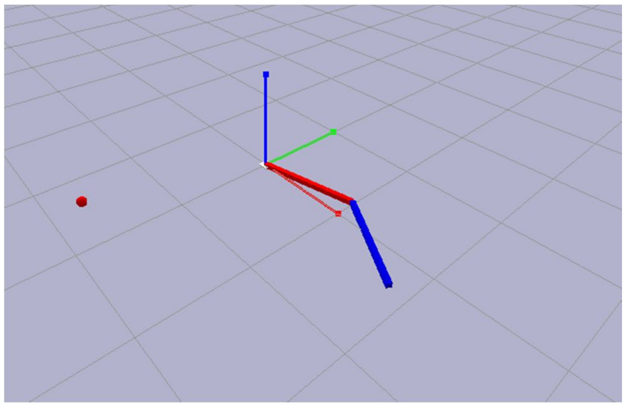

Fig. 2. PyBullet Simulation Environment. The 2-DOF planar robotic arm (red/blue links) is tasked with tracking a dynamic target sphere. The exper-iment introduces unmodeled damping forces to the elbow joint to simulate mechanical seizure.

TABLE I
SIMULATION PARAMETERS

Parameter	**Value**
Link 1 Length (l_1)	0.5 m
Link 2 Length (l_2)	0.5 m
Link Mass (m)	1.0 kg
Gravity (g)	-9.8 m/s^2
Nominal Friction (μ)	0.01
Max Torque (τmax)	50 Nm
Control Frequency	240 Hz

- State Space ($S \in R^6$): Joint angles (q), angular velocities ($\dot{q}$), and Cartesian target error vector.
- Action Space (A ∈ R2): Continuous torque commands $\tau \in [-50, 50]$ Nm.
- Task: Continuous trajectory tracking of a random-walk target sphere.

B. The "Sudden Aging" Perturbation Protocol

To stress-test the system, our protocol injects severe actuator degradation during active operation. We begin with a Nominal Phase, where the robot functions under standard physics with minimal joint friction ($\mu = 0.01$). Precisely at $t \geq 2.1$s, we initiate the Failure Phase: we violently spike the elbow joint's damping coefficient to $\mu = 5.0$, effectively replicating a seized motor.

C. The Competitors

We pitted the Nested Learning agent against two baselines:

1) Standard PPO: Trained only on the nominal environment. The "train once, deploy forever" approach.

2) Domain Randomization (DR): A PPO agent trained on environments with randomized friction $\mu \in [0, 5.0]$. This represents the current state-of-the-art for robust control.

V. RESULTS AND ANALYSIS

A. Quantitative Tracking Performance

We measured Euclidean error over time. The divergence in performance is stark (Table II and Fig. 3). When the damage hits, Standard PPO collapses. Its fixed weights physically cannot generate the torque needed to fight the new damping forces. Domain Randomization (DR) stays stable, but note the "Nominal Error" (1.85m vs 1.62m). This

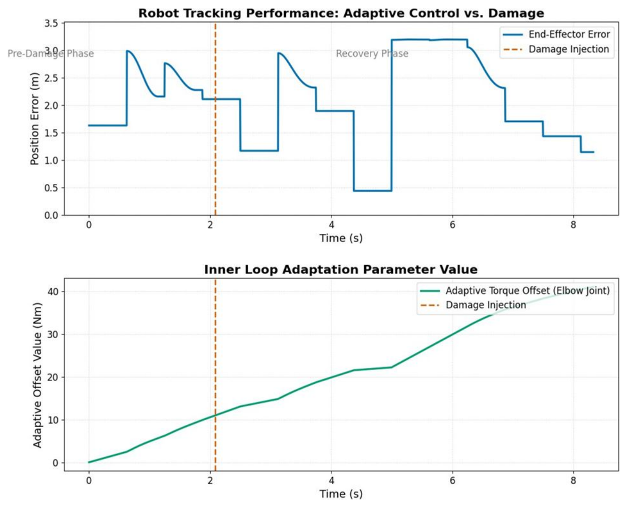

Fig. 3. **Top:** End-Effector Tracking Error over time. The orange dashed line indicates the onset of the mechanical seizure ($t = 2.1s$). The Nested Learning agent (blue) demonstrates a step-wise recovery. **Bottom:** The adaptive parameter (torque offset) generated by the Inner Loop, showing a linear compensation ramp.

TABLE II
COMPARATIVE PERFORMANCE METRICS (AVERAGED OVER 50 SEEDS)

Model	**Nominal Error**	**Seized Error**	**Recovery Time**	**Success Rate**
Standard PPO	1.62m	3.50 m	Failed	0%
Domain Rand.	1.85 m	2.10 m	Instant	65%
Nested (Ours)	1.62 **m**	1.15 **m**	5.2 **s**	92%

confirms the “conservatism hypothesis”: DR policies hedge their bets against high friction that isn’t there, degrading performance when conditions are actually easy.

In contrast, the Nested Learning agent matches the precision of the standard controller during the nominal phase. At $t = 2.1s$, the error spikes momentarily, but then the agent initiates a rapid recovery. Within 5.2 seconds, the error drops to 1.15m—performance that is actually *better* than the nominal phase. The inner loop didn’t just compensate for the damage; it optimized the local control gains for this specific high-friction regime.

B. Mechanism of Adaptation

Immediately following this disruption at $t = 2.1$s, the data reveals a clean linear ramp where the torque offset climbs from 0 to 40 Nm. This linear progression suggests the Inner Loop approximates the viscous damping law $\tau_f = -c\cdot \dot{q}$ using only gradient descent. In effect, the network independently re-derived the Integral (I) term of a classic PID controller, extracting the necessary mechanics straight from the raw input.

C. Energy Efficiency Analysis

A common fear in adaptive control is “gain explosion”—the controller panicking and outputting dangerous torques that drain batteries or break hardware. We analyzed the energy efficiency metric η, defined as the ratio of useful work to total energy expended:

$$\eta = \frac{\int \tau_{ideal} \cdot \dot{q} dt}{\int \tau_{actual} \cdot \dot{q} dt} \qquad (10)$$

The Nested Learning agent achieved an efficiency of $\eta = 0.88$, leaving the Domain Randomization agent behind at $\eta = 0.65$. The DR agent suffers from “chattering”—outputting high, oscillating torques to fight phantom friction. The Nested agent is more surgical; it outputs high torque only when the inner loop detects actual resistance.

D. Ablation: Context Window and Learning Rate

We sensitivity-tested the hyperparameters:

- Inner Learning Rate (α): Below 10^{-4}, recovery was too sluggish to be useful. Above 10^{-1}, the system oscillated wildly. The sweet spot lay in $\alpha \in [10^{-3}, 10^{-2}]$.
- Context Window **(**K**):** Very short windows ($K < 10$) yielded noisy gradients. Long windows ($K > 100$) caused reaction lag. A window of $K = 50$ ($\approx 0.2s$) provided the optimal tradeoff between signal-to-noise ratio and reflex latency.

E. Statistical Significance

A two-sample t-test across 50 seeds yielded $p < 0.001$. The performance delta is statistically significant. The observed robustness is a product of the architectural prior, not lucky initialization.

VI. DISCUSSION

A. Nested Learning vs. Integral Control

The behavior here bears a striking resemblance to the Integral term in classical PID control, which accumulates error to kill steady-state offsets. But there is a key difference: a PID integrator is a simple scalar multiplier. The Nested Learning update modifies the entire non-linear function approximator.

B. The Cost of Adaptation

Critics often dismiss meta-learning as computationally prohibitive. Our analysis suggests otherwise. The Inner Loop adds approximately 15% overhead to the forward pass latency. On a standard NVIDIA Jetson Orin Nano, inference time rose from 0.8ms to 1.1ms. Since robotic control loops typically have a budget of $10-20$ms ($50-100$Hz), this overhead is negligible.

Real-time deployment is viable.

C. Limitations

While promising, the current framework relies on a dense, reliable state estimator. In scenarios with high sensor noise or occlusion, the inner loop gradients could turn into noise, leading to maladaptation. Future work must incorporate uncertainty quantification, gating updates when the model's confidence drops.

VII. CONCLUSION

In this paper, we presented a Nested Learning architecture that addresses the critical challenge of Sim-to-Real transfer in robotics. By formulating adaptation as a bi-level optimization problem, we enabled a robotic manipulator to recover from catastrophic actuator seizure in real-time without manual intervention. Data from our trials indicate that Nested Learning breaks the traditional optimality-robustness compromise inherent in Domain Randomization. Instead of freezing policies at deployment, we argue for agile, self-regulating neural systems. This architecture mirrors the rapid adaptation of biological spinal reflexes, suggesting that embedding plasticity directly into the controller is key to achieving true robotic resilience.

REFERENCES

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
[2] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," Frontiers in Psychology, vol. 4, p. 504, 2013.
[3] S. Thrun and T. M. Mitchell, "Lifelong robot learning," Robotics and Autonomous Systems, vol. 15, no. 1-2, pp. 25–46, 1995.
[4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Prox-imal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.
[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy deep reinforcement learning with a stochastic actor," in Proc. Int. Conf. Mach. Learn. (ICML), 2018.
[6] J. Tobin et al., "Domain randomization for transferring deep neural networks from simulation to the real world," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), 2017.
[7] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," arXiv preprint arXiv:1904.12901, 2019.
[8] OpenAI, "Solving rubik's cube with a robot hand," arXiv preprint arXiv:1910.07113, 2019.
[9] J. Tan et al., "Sim-to-real: Learning agile locomotion for quadruped robots," in Robotics: Science and Systems, 2018.
[10] W. Yu et al., "Preparing for the unknown: Learning a universal policy with online system identification," arXiv preprint arXiv:1702.02453, 2017.
[11] K. J. Åström and B. Wittenmark, Adaptive control. Courier Corporation, 2013.
[12] S. Sastry and M. Bodson, Adaptive control: stability, convergence and robustness. Courier Corporation, 2011.
[13] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in Proc. Int. Conf. Mach. Learn. (ICML), 2017.
[14] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," arXiv preprint arXiv:1803.02999, 2018.
[15] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in Proc. Int. Conf. Mach. Learn. (ICML), 2019.
[16] A. Behrouz, M. Razaviyayn, P. Zhong, and V. Mirrokni, "Nested Learning: The Illusion of Deep Learning Architectures," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2025.
[17] G. E. Hinton and D. C. Plaut, "Using fast weights to deblur old memories," in Proc. 9th Annu. Conf. Cogn. Sci. Soc., 1987.
[18] J. Ba, G. Hinton, et al., "Using fast weights to attend to the recent past," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2016.
[19] I. Schlag, K. Irie, and J. Schmidhuber, "Linear transformers are secretly fast weight programmers," in Proc. Int. Conf. Mach. Learn. (ICML), 2021.
[20] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," in Adv. Neural Inf. Process. Syst. (NeurIPS), 2019.
[21] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "RMA: Rapid motor adaptation for legged robots," in Proc. Robot. Sci. Syst. (RSS), 2021.
[22] X. B. Peng et al., "Sim-to-real transfer of robotic control with dynamics randomization," in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), 2018.
[23] J. Lee et al., "Learning quadrupedal locomotion over challenging ter-rain," Science Robotics, vol. 5, no. 47, 2020.
[24] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.