Online Adaptive Reinforcement Learning with Echo State Networks for Non-Stationary Dynamics

Aoi Yoshimura[1*] and Gouhei Tanaka[1,2]
[1]Department of Computer Science, Nagoya Institute of Technology, Nagoya 466-8555, Japan
Emails: a.yoshimura.536@nitech.jp[*], gtanaka@nitech.ac.jp
[2]International Research Center for Neurointelligence, The University of Tokyo, Tokyo 113-0033, Japan

***Abstract*—Reinforcement learning (RL) policies trained in simulation often suffer from severe performance degradation when deployed in real-world environments due to non-stationary dynamics. While Domain Randomization (DR) and meta-RL have been proposed to address this issue, they typically rely on extensive pretraining, privileged information, or high computational cost, limiting their applicability to real-time and edge systems. In this paper, we propose a lightweight online adaptation framework for RL based on Reservoir Computing. Specifically, we integrate an Echo State Networks (ESNs) as an adaptation module that encodes recent observation histories into a latent context representation, and update its readout weights online using Recursive Least Squares (RLS). This design enables rapid adaptation without backpropagation, pretraining, or access to privileged information. We evaluate the proposed method on CartPole and HalfCheetah tasks with severe and abrupt environment changes, including periodic external disturbances and extreme friction variations. Experimental results demonstrate that the proposed approach significantly outperforms DR and representative adaptive baselines under out-of-distribution dynamics, achieving stable adaptation within a few control steps. Notably, the method successfully handles intra-episode environment changes without resetting the policy. Due to its computational efficiency and stability, the proposed framework provides a practical solution for online adaptation in non-stationary environments and is well suited for real-world robotic control and edge deployment.**

***Index Terms*—Reservoir Computing, Sim2Real, Recursive Least Squares, Edge Computing**

I. Introduction

Deep Reinforcement Learning (DRL) has demonstrated remarkable success in high-dimensional control tasks, ranging from game playing to complex robotic manipulation. Algorithms such as Proximal Policy Optimization (PPO) [1] and Soft Actor-Critic (SAC) [2] enable agents to learn optimal control policies in an end-to-end manner. However, deploying these policies to real-world robotic systems remains a significant challenge due to the well-known *Sim2Real gap*. Simulators rely on idealized physical models, and inevitable discrepancies in parameters such as friction coefficients, contact dynamics, and sensor noise often lead to severe performance degradation when policies trained in simulation are transferred to the real world [3], [4].

Moreover, real-world environments are inherently non-stationary. Physical properties can change abruptly during manipulation, control, and decision-making tasks due to external disturbances and friction variations. Such changes may occur even within a single episode during RL, rendering static policies with fixed parameters insufficient. Addressing both partial observability and intra-episode environmental variation is therefore essential for robust real-world deployment.

To mitigate the Sim2Real gap, Domain Randomization (DR) has become a widely adopted technique, where physical parameters are randomized during training to improve robustness [3], [4]. While effective for disturbances within the training distribution, DR fundamentally relies on interpolation and struggles under out-of-distribution (OOD) dynamics, often resulting in overly conservative policies that sacrifice optimality for stability [5]. These limitations motivate the development of explicit online adaptation mechanisms.

Recent work has explored history-based adaptation methods that infer latent environmental parameters from observation sequences. A representative example is Rapid Motor Adaptation (RMA) [6], which estimates dynamics-related context variables using a learned encoder. Despite its effectiveness, RMA requires extensive offline pretraining with privileged information, such as ground-truth physical parameters, and large-scale simulation data. Such requirements significantly increase development cost and limit scalability. Alternatively, test-time training approaches, including Policy Adaptation during Deployment (PAD), perform online adaptation using self-supervised learning objectives. However, these methods rely on backpropagation during deployment, leading to high computational overhead and potential instability, which are undesirable for resource-constrained edge devices requiring real-time control.

In this paper, we propose a lightweight real-time adaptation framework, termed *ESN-based Online Adaptation (ESN-OA), in the context of reinforcement learning*. Unlike existing meta-learning or test-time training approaches, our method requires no pretraining phase for adaptation and no access to privileged information. We integrate an Echo State Network (ESN) [7], [8] into a DRL agent to extract temporal context from recent observation histories. Immediate adaptation is achieved by training only the ESN readout layer online using Recursive Least Squares (RLS). This approach avoids the computationally expensive backpropagation required for updating deep neural network weights, enabling rapid and stable adaptation with minimal overhead.

The main contributions of this paper are summarized as follows:

1) **Zero-shot Adaptation to Dynamics Shift:** We propose an online adaptation framework that enables agents to adapt to non-stationary environments from scratch during deployment. We refer to this framework as zero-shot adaptation to dynamics shift, since the adaptation module requires no prior exposure to target dynamics variations (meta-training) or privileged information.
2) **Computational Efficiency for Edge Deployment:** By leveraging RLS for online readout updates, the proposed method achieves fast convergence and low computational overhead, making it suitable for real-time robotic control on resource-constrained platforms.
3) **Robustness to Intra-Episode Dynamics Changes:** We demonstrate that the proposed approach can handle sudden and extreme changes in physical parameters, such as multi-fold variations in friction, within a few control steps, significantly outperforming DR and representative adaptive baselines.

II. Related Work

A. Domain Randomization and its Limits

While DR is the standard approach for bridging the Sim2Real gap [9], [10], standard DR relies on fixed parameter distributions designed manually [11]. Recent advancements include Active DR [12] and entropy-maximization techniques [13] to improve sample efficiency and coverage. Alternatively, recent work proposes bridging this gap by constructing high-fidelity simulators that incorporate realistic physics such as friction and air resistance [14]. However, these methods result in static policies that struggle to generalize to OOD scenarios [15], [16]. As noted in recent surveys [17], [18], static policies remain insufficient for non-stationary real-world environments, necessitating mechanisms that can adapt to environmental changes on the fly.

B. POMDPs and Meta-Reinforcement Learning

To handle partial observability and dynamics shifts, control tasks are often formulated as Partially Observable Markov Decision Processes (POMDPs) requiring history-based inference [19], [20]. Transformer-based architectures have been employed to capture long-term context [21], [22], and Meta-Reinforcement Learning (Meta-RL) approaches like RMA [6] demonstrate robust adaptation by estimating environmental extrinsics. Similarly, factored adaptation frameworks have been proposed to explicitly model time-varying latent factors in non-stationary environments [23]. However, these methods typically require extensive pre-training with privileged information (e.g., ground-truth physics) which is unavailable in many real-world settings. Alternatively, test-time training methods such as PAD [24] adapt without privileged information but rely on backpropagation during deployment. This incurs high computational cost and latency [25], making them ill-suited for resource-constrained edge devices compared to the gradient-free adaptation proposed in this work.

III. Proposed Method: Online Adaptive RL using ESN

In real-world robotic control, the assumption of stationarity often fails due to external disturbances and friction variations. While DR and Meta-RL attempt to address this issue as described in Section II, they typically require extensive pre-training or privileged information. To overcome these limitations, we propose an *Online Adaptive Reinforcement Learning* framework based on Reservoir Computing, specifically utilizing an ESN. Our approach utilizes the RLS-based online learning for the readout layer, enabling rapid adaptation to non-stationary dynamics without any pre-training. The overall system architecture is illustrated in Fig. 1, and the pseudo-code of the learning procedure is presented in Algorithm 1.

A. Echo State Networks

Recurrent Neural Networks (RNNs) and their variants have widely been used for processing time-series data and estimating hidden states. However, despite advances of RNN variants such as LSTM and GRU, training RNN-based models via Backpropagation Through Time (BPTT) remains computationally expensive and can still suffer from vanishing gradient issues in long-sequence modeling. To address this, we employ Reservoir Computing (RC), specifically ESN architecture.

The ESN consists of a fixed, randomly initialized recurrent layer (reservoir) and a trainable output layer (readout). Let $u_t \in \mathbb{R}^{N_u}$ be the input vector, $x_t \in \mathbb{R}^{N_x}$ be the internal reservoir state, and $y_t \in \mathbb{R}^{N_y}$ be the output vector at time step t. The state update rule is defined as follows [7], [26]:

$$x_t = (1-\alpha)x_{t-1} + \alpha \cdot \tanh(W^{\text{in}}u_t + W^{\text{res}}x_{t-1}), \quad (1)$$

where $W^{\text{in}} \in \mathbb{R}^{N_x \times N_u}$ is the input weight matrix, $W^{\text{res}} \in \mathbb{R}^{N_x \times N_x}$ is the recurrent weight matrix, and $\alpha \in (0, 1]$ is the leak rate. To ensure the Echo State Property, the spectral radius of W^{res} is set to $\rho(W^{\text{res}}) < 1$ [8].

The network output $\hat{y}_t$ is computed as a linear combination of the reservoir states:

$$\hat{y}_t = W^{\text{out}}x_t, \quad (2)$$

where $W^{\text{out}} \in \mathbb{R}^{N_y \times N_x}$ is the only trainable weight matrix in our adaptation module.

B. Online Adaptation via Recursive Least Squares

To enable rapid online adaptation, we employ the RLS algorithm for updating W^{out}. RLS provides second-order convergence properties, which allows for immediate adaptation to non-stationary environments within a few time steps [27].

The objective is to minimize the weighted sum of squared errors between the predicted next state $\hat{s}_{t+1}$ and the actual next state s_{t+1}, given by:

$$\mathcal{L}_t = \sum_{i=1}^{t} \lambda^{t-i} \|e_i\|^2, \quad (3)$$

$$e_i = s_{i+1} - W^{\text{out}}x_i. \quad (4)$$

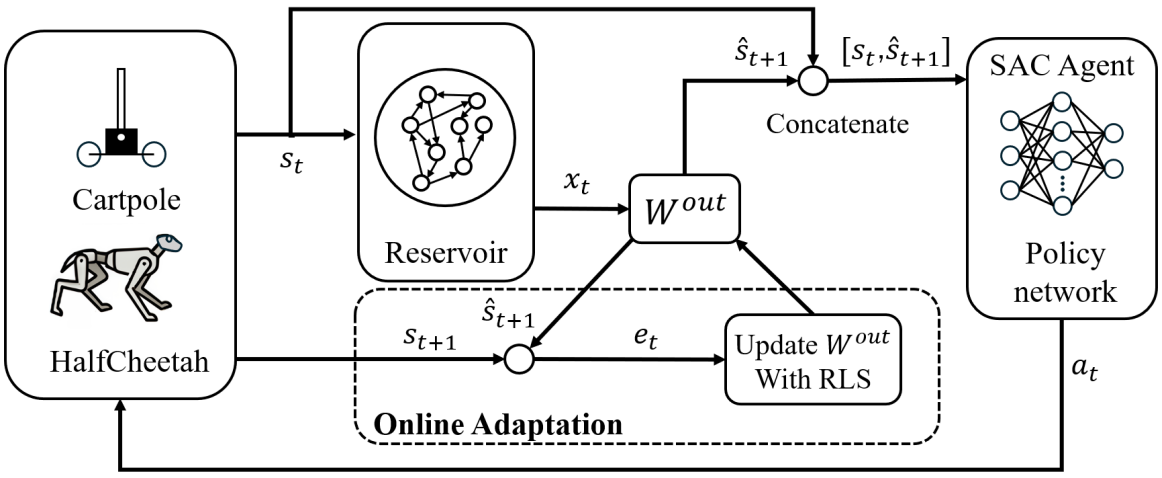

Fig. 1. Overview of the proposed ESN-based online adaptation system. The reservoir captures temporal context, and RLS updates the readout weights W^{out} in real-time based on prediction error e_t. The predicted next state $\hat{s}_{t+1}$ is concatenated with the current state s_t and used as the input to the policy network of the SAC agent.

Here, $\lambda \in (0, 1]$ is the forgetting factor. A smaller λ allows the system to rapidly forget past dynamics and adapt to sudden environmental changes.

The online update procedure for W^{out} at each step t is as follows:

1) Compute Prediction Error:
$$e_t = s_{t+1} - W^{\text{out}}_{t-1} x_t. \tag{5}$$
2) Compute Gain Vector:
$$k_t = \frac{P_{t-1} x_t}{\lambda + x_t^T P_{t-1} x_t}, \tag{6}$$
where the precision matrix P_t approximates the inverse covariance matrix $(\sum_t x_t x_t^T)^{-1}$.
3) Update Output Weights:
$$W^{\text{out}}_t = W^{\text{out}}_{t-1} + e_t k_t^T. \tag{7}$$
4) Update Precision Matrix:
$$P_t = \frac{1}{\lambda}(P_{t-1} - k_t x_t^T P_{t-1}). \tag{8}$$

Crucially, for the "No pre-training" setting, we initialize $W^{\text{out}}(0) = 0$ and $P(0) = \delta I$ (where δ is a large constant, e.g., 100). We note that interpreted as introducing a decaying regularization term to the error function in (3). This setting of $P(0)$ reflects a state of high sensitivity to prediction errors, facilitating rapid initial adaptation of W^{out}.

Algorithm 1 ESN-based Online Adaptation (ESN-OA)

Input: Reservoir (N_x, ρ, α), RLS (λ, δ), Policy π_ϕ
Init: $W^{\text{in}}, W^{\text{res}}$ (random); $W^{\text{out}} \leftarrow \mathbf{0}$, $P \leftarrow \delta I$, $x(0) \leftarrow \mathbf{0}$
for each episode **do**
  Reset environment and observe s_0
  (Nominal for training, Non-stationary for testing)
  for $t = 0$ to T **do**
    Update reservoir state: $x_t \leftarrow (1-\alpha)x_{t-1} + \alpha \cdot \tanh(W^{\text{in}} s_t + W^{\text{res}} x_{t-1})$ ▷ Eq. (1)
    Predict next state: $\hat{s}_{t+1} \leftarrow W^{\text{out}} x_t$ ▷ Eq. (2)
    Augment state: $\tilde{s}_t \leftarrow [s_t, \hat{s}_{t+1}]$ ▷ Eq. (9)
    Select action $a_t \sim \pi_\phi(\cdot|\tilde{s}_t)$
    Execute a_t, observe r_t, s_{t+1}
    // Online Adaptation (RLS)
    $e \leftarrow s_{t+1} - \hat{s}_{t+1}$ ▷ Eq. (5)
    $k \leftarrow \frac{P x_t}{\lambda + x_t^T P x_t}$ ▷ Eq. (6)
    $W^{\text{out}} \leftarrow W^{\text{out}} + e k^T$ ▷ Eq. (7)
    $P \leftarrow \frac{1}{\lambda}(P - k x_t^T P)$ ▷ Eq. (8)
    if training **then**
      Update SAC parameters ϕ using standard loss
    end if
  end for
end for

C. Integration with Soft Actor-Critic (SAC)

We integrate the ESN-RLS adaptation module with a SAC agent.

The standard policy $\pi(a_t|s_t)$ in MDPs relies only on the current observation s_t. However, under partial observability caused by hidden physical parameters (e.g., friction coefficients, wind force), s_t is insufficient for adaptation to non-stationary environments. To address this, our system operates as follows:

1) Context extraction: The reservoir receives the current observation s_t and updates its internal state x_t, encoding the temporal history of the trajectory.
2) Dynamics prediction: The readout layer predicts the next state $\hat{s}_{t+1} = W^{\text{out}} x_t$. Since W^{out} is updated online via RLS to minimize prediction error, $\hat{s}_{t+1}$ implicitly contains information about the current environmental dynamics.
3) State augmentation: The policy network receives an aug-

mented state vector $\tilde{s}_t$ formed by concatenating the raw observation and the ESN prediction:

$$\tilde{s}_t = [s_t, \hat{s}_{t+1}]. \tag{9}$$

4) Action generation: The SAC agent outputs action $a_t = \pi(\tilde{s}_t)$ based on this context-aware state.

By using the predicted next state $\hat{s}_{t+1}$ instead of the high-dimensional reservoir state x_t for concatenation, we avoid the curse of dimensionality while providing the agent with explicit information about the estimated dynamics.

IV. Experiments

To validate the efficacy of the proposed ESN-based online adaptable RL framework, we conducted comparative experiments in two simulated environments: CartPole [28] and HalfCheetah [29]. The primary objective is to evaluate the system's capability for zero-shot adaptation, such as maintaining control performance in non-stationary environments without any pre-training or prior data collection.

A. Experimental Environments and Non-stationarity

We introduced specific non-stationary conditions into the physical parameters of the environments to simulate external disturbances and sudden hardware failures.

1) CartPole with periodic disturbance
The CartPole task requires an agent to balance a pole vertically by applying horizontal forces to a cart. The state space consists of the cart position, cart velocity, pole angle, and pole angular velocity. The reward is defined as $+1$ for every step the pole remains upright within the allowable range. The standard CartPole environment ($g = 9.8\mathrm{m/s}^2$, cart mass $1.0\mathrm{kg}$, pole mass $0.1\mathrm{kg}$) was utilized [30]. In addition, a time-varying external wind force $F_{\mathrm{wind}} = A\cos(\omega t)$ was applied to the cart.

- **Protocol:** During training, the environment was stationary ($A = 0$). During testing, the amplitude A was varied within the range $[1.0, 10.0]$ to assess robustness against non-stationary periodic disturbances.

2) HalfCheetah with a sudden parameter change
HalfCheetah is a high-dimensional (17-dimensional) locomotion task where a planar cheetah-like robot learns to run forward as fast as possible. The agent must coordinate the torques of six joints based on the positions and velocities of its body parts. The reward encourages forward motion while penalizing excessive control effort, defined as $r_t = v_x - 0.1\|a_t\|^2$ [30]. For the HalfCheetah task, we simulated a drastic change in contact dynamics to evaluate the temporal adaptation speed.

- **Protocol:** The friction coefficient of the ground was maintained at its default value ($1.0\times$) for the first 500 time steps. At step 500, the coefficient was abruptly scaled by a factor ranging from $1.0\times$ to $10.0\times$. This setup tests the agent's ability to recover from sudden "failures" in real-time.

B. Baselines and Comparative Analysis

We compared the proposed method against six baseline strategies, including state-of-the-art domain adaptation methods (PAD and RMA). A critical aspect of this comparison is the data collection strategy required for pre-training based methods.

1) Proposed Method
- **ESN-OA (ESN-based Online Adaptation without pre-training):** The readout weights W^{out} are initialized to zero, and adaptation via RLS begins immediately at the start of the test episode. This method requires absolutely no pre-training phase.

2) Pre-trained Baselines
- **ESN-OA-PT (ESN-based Online Adaptation with pre-training):** The weights W^{out} are pre-trained on a stationary environment. To ensure high-quality reservoir feature extraction, the pre-training data must contain long, continuous trajectories. Since purely random actions lead to early termination in CartPole, we employed a "noisy rule-based policy" (30% random actions) to secure trajectory length while maintaining exploration. Similarly, for HalfCheetah, we utilized Ornstein-Uhlenbeck (OU) noise ($\omega = 0.3$) to generate temporally correlated motion data suitable for reservoir dynamics learning.
- **PAD (Policy Adaptation during Deployment):** A self-supervised adaptation method that trains an encoder using an auxiliary loss $\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{IDM}} + \mathcal{L}_{\mathrm{rec}}$ (Inverse Dynamics Model + Reconstruction) [24]. In contrast to the ESN approach, PAD relies on learning a diverse latent representation of the environment rather than capturing long temporal dependencies. Consequently, to prevent overfitting and ensure broad state-space coverage during pre-training, we prioritized diversity over stability by utilizing a high-entropy policy (50% random mixture) for CartPole and DR for HalfCheetah.
- **RMA (Rapid Motor Adaptation):** A two-phase method using an oracle policy and a history-based adaptor network (50-step history) [31].

3) Standard Baselines
- **Standard SAC:** A standard SAC agent trained in a stationary environment, used as a lower-bound reference to highlight the performance degradation under distribution shift.
- **DR:** A robust policy trained with randomized physical parameters, serving as a baseline for comparison against static robustness versus explicit online adaptation.

C. Implementation Details

All algorithms were implemented using Python libraries, PyTorch and Stable Baselines3 [32]. Hyperparameters were selected via grid search. Experiments were conducted on a machine with an AMD Ryzen 9 9950X CPU and 128 GB RAM. To ensure statistical reliability, all experiments for each method were conducted over $N = 10$ independent trials with different random seeds.

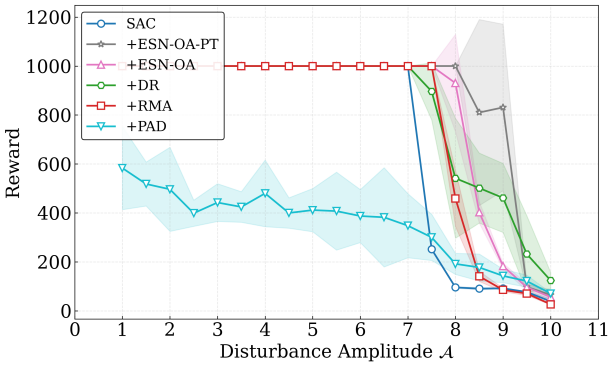

Fig. 2. Average reward in the CartPole environment under increasing periodic wind disturbances.

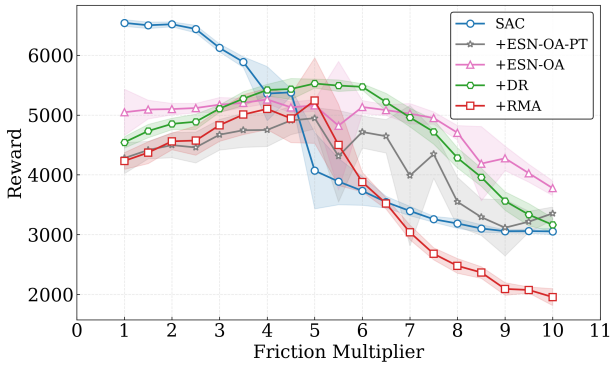

Fig. 4. Average reward in the HalfCheetah environment under varying friction coefficients.

TABLE I
ROBUSTNESS EVALUATION ON CARTPOLE (MEAN ± STD).

Method	A=8.0	A=8.5	A=9.0	A=9.5
SAC	95.5 ± 3.1	90.5 ± 4.4	91.8 ± 2.3	76.8 ± 2.4
+ESN-OA-PT	1000.0 ± 0.0	810.1 ± 379.9	830.9 ± 340.1	97.6 ± 12.5
+ESN-OA	930.2 ± 199.0	402.3 ± 43.9	182.7 ± 11.7	96.2 ± 14.7
+DR	541.6 ± 244.8	501.5 ± 142.9	461.2 ± 140.8	231.8 ± 156.3
+RMA	459.0 ± 114.9	140.9 ± 20.8	85.4 ± 7.3	70.4 ± 7.5
+PAD	191.7 ± 43.0	177.2 ± 55.0	142.7 ± 27.5	121.3 ± 24.4

TABLE II
ROBUSTNESS EVALUATION ON HALFCHEETAH (MEAN ± STD).

Method	F=1	F=4	F=7	F=10
SAC	6540.5 ± 55.8	5358.4 ± 453.7	3396.4 ± 78.1	3054.1 ± 53.6
+ESN-OA-PT	4279.8 ± 259.1	4750.0 ± 276.0	3986.5 ± 1173.6	3354.8 ± 108.4
+ESN-OA	5046.3 ± 386.2	5260.6 ± 68.2	5022.4 ± 170.8	3777.2 ± 123.4
+DR	4543.5 ± 94.9	5418.7 ± 95.4	4960.4 ± 156.2	3164.3 ± 116.6
+RMA	4232.8 ± 140.0	5102.2 ± 344.9	3039.3 ± 123.2	1956.5 ± 141.0

1) Reinforcement Learning (SAC): The SAC agent utilized an Multilayer Perceptron (MLP) architecture consisting of two layers with hidden sizes of $[64, 64]$ for CartPole and $[256, 256]$ for HalfCheetah. The optimizer was Adam [33] with a learning rate of 3×10^{-4}.

2) Reservoir and RLS Settings: The ESN configuration was tailored to the task complexity:

- **Reservoir Size (N_x):** 300 for CartPole, 500 for HalfCheetah.
- **Dynamics:** Spectral radius $\rho = 0.9$, Leak rate $\alpha = 0.3$.
- **Online Adaptation (RLS):** The forgetting factor λ was set to 0.99 for CartPole and 0.95 for HalfCheetah to handle the rapid dynamics changes. The inverse covariance matrix was initialized as $P(0) = 100I$ to facilitate rapid initial learning.

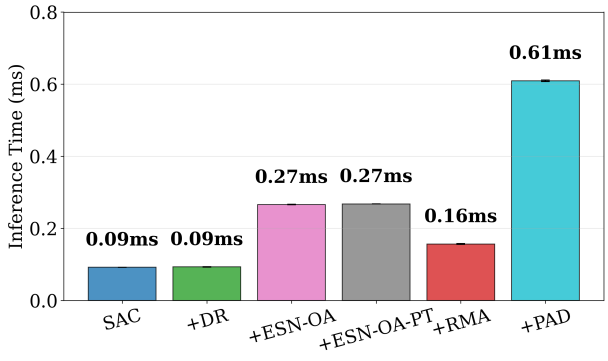

Fig. 3. Inference time per control step on CartPole.

V. RESULTS

A. Performance on CartPole with Non-Stationary Disturbances

We evaluated robustness of the RL algorithms against non-stationary periodic wind disturbances in the CartPole environment. An external force $F_{\text{wind}} = A\cos(\omega t)$ was applied as described in Sec. IV-A, and the disturbance amplitude A was gradually increased during testing. Fig. 2 shows the average reward as a function of A, and Table I summarizes the numerical results.

The standard SAC and representative baselines (DR, RMA) collapse when the disturbance amplitude A exceeds 8.0. In contrast, ESN-OA and ESN-OA-PT maintain near-optimal performance across the tested range ($N = 10$). ESN-OA-PT shows even higher stability at extreme amplitudes ($A = 9.5$), indicating that pre-training further strengthens the reservoir's representation. This robustness indicates that the latent context encoded by the ESN effectively compensates for OOD forces in real-time. Regarding computational overhead, ESN-OA achieves a practical balance between adaptation and latency. Fig. 3 confirms that ESN-OA requires only 0.27 ms per step (3.7 kHz), significantly faster than the gradient-based PAD (0.61 ms). This efficiency makes it well-suited for high-frequency control.

B. Performance on HalfCheetah with Sudden Parameter Shifts

We next evaluated adaptation performance in the HalfCheetah environment under abrupt changes in ground friction. The friction coefficient was scaled by a multiplier ranging from $1.0\times$ to $10.0\times$. PAD was excluded due to unstable behavior in this task.

Fig. 4 and Table II summarize the results. The standard SAC agent achieved the highest performance at the nominal friction

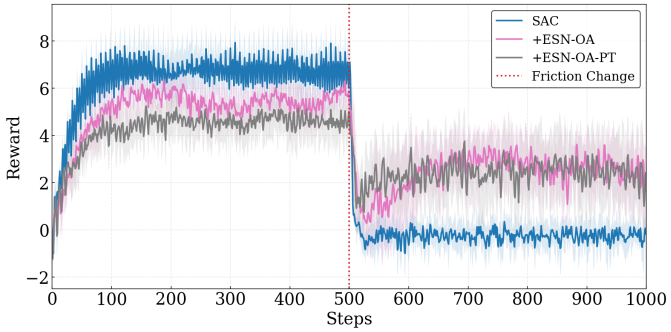

Fig. 5. Reward transition after an abrupt friction change from $1.0\times$ to $10.0\times$ during an episode.

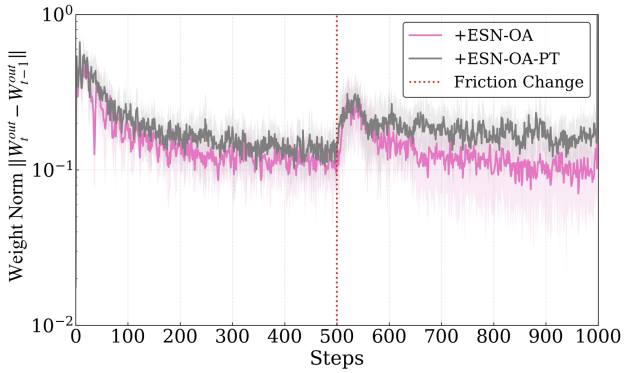

Fig. 6. Temporal evolution of the ESN output weight update norm $\|W_t^{\text{out}} - W_{t-1}^{\text{out}}\|$ during friction change in HalfCheetah.

but degraded significantly as friction increased, indicating strong overfitting to the training dynamics. RMA and DR improved robustness in moderate regimes but collapsed in the extrapolation region. In contrast to the baselines, ESN-OA and ESN-OA-PT maintained stable performance across all friction settings. While ESN-OA achieved the highest reward under extreme conditions, ESN-OA-PT also demonstrated consistent robustness without the collapse seen in DR or RMA. These results demonstrate the effectiveness of the proposed online adaptation mechanism in handling severe and non-stationary dynamics shifts.

1) Intra-Episode Adaptation Speed: We analyzed the temporal adaptation process when the friction coefficient was abruptly switched from $1.0\times$ to $10.0\times$ during an episode. As shown in Fig. 5, the SAC agent failed immediately after the change, whereas the proposed ESN-OA rapidly recovered performance within a few dozen steps and maintained stable locomotion thereafter.

2) ESN Output Weight Dynamics during Adaptation: To further analyze the adaptation behavior, we measured the temporal change in the ESN output weights. Fig. 6 shows the norm of the step-wise difference $\|W_t^{\text{out}} - W_{t-1}^{\text{out}}\|$ when the friction coefficient was abruptly changed during the episode. As shown in the figure, ESN-OA exhibits a pronounced spike in the output weight update norm immediately after the dynamics change, indicating rapid update of the readout parameters in response to large prediction errors. In contrast, ESN-OA-PT shows more moderate updates, suggesting reduced sensitivity to sudden environment shifts. These results highlight the fast and responsive nature of the proposed online adaptation mechanism.

VI. Discussion

This work demonstrated that integrating the ESN-based online adaptation mechanism using the RLS online learning algorithm enables RL agents to rapidly adjust to non-stationary environments. Across both CartPole and HalfCheetah tasks, the proposed method consistently recovered performance within a few steps after abrupt changes in external disturbances or physical parameters, highlighting its effectiveness under severe OOD dynamics.

A. Effectiveness of Online Adaptation

The success of the proposed approach stems from two complementary properties. First, the reservoir dynamics provide a compact yet expressive temporal representation of recent observation histories, allowing the agent to implicitly infer latent environmental context that is not directly observable. This effectively addresses partial observability without relying on heavy recurrent architectures or long history windows. Second, the RLS-based readout adaptation achieves fast and stable convergence without backpropagation, making it suitable for real-time deployment and intra-episode adaptation.

B. Comparison with Existing Methods

Compared to DR, the proposed method does not depend on predefined parameter distributions and therefore remains effective even when test-time dynamics lie far outside the training distribution. In contrast to meta-RL approaches such as RMA, our framework requires neither privileged information nor extensive pretraining, significantly reducing deployment cost. Methods based on test-time training, such as PAD, can adapt online but incur higher computational overhead and may suffer from instability due to gradient-based updates. Restricting learning to a linear readout layer avoids catastrophic forgetting and ensures stable behavior during deployment.

C. Intra-Episode Adaptation

A key contribution of this study is the explicit evaluation of intra-episode adaptation, where environment dynamics change abruptly during task execution. The proposed method can continuously track such changes without resetting the policy or environment, which is critical for real-world robotic systems operating in non-stationary conditions.

D. Limitations and Future Work

Despite these advantages, the method has limitations. RLS can be sensitive to observation noise and outliers, and stability guarantees are not explicitly enforced. While suitable hyperparameter choices mitigated these issues in our experiments, incorporating formal safety constraints and adaptive regularization remains an important direction for future work. Overall, the proposed ESN-based online adaptation framework offers

a lightweight and practical alternative to existing robust and adaptive reinforcement learning methods. Its computational efficiency and ability to adapt without pretraining make it particularly promising for real-world and edge deployment in robotics and autonomous systems.

Code Availability

The code for this work is available at https://github.com/aoi-yoshi/esn-oa.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Numbers JP23K28154, JP25H00451, JST Moonshot R&D Grant Number JPMJMS2021, and JST CREST Grant Number JPMJCR24R2.

References

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

[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018. [Online]. Available: https://arxiv.org/abs/1801.01290

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

[4] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[5] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[6] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021. [Online]. Available: https://arxiv.org/abs/2107.04034

[7] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology gmd technical report*, vol. 148, no. 34, p. 13, 2001.

[8] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013709000173

[9] L. Da, J. Turnau, T. P. Kutralingam, A. Velasquez, P. Shakarian, and H. Wei, "A survey of sim-to-real methods in rl: Progress, prospects and challenges with foundation models," 2025. [Online]. Available: https://arxiv.org/abs/2502.13187

[10] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters, "Robot learning from randomized simulations: A review," *Frontiers in Robotics and AI*, vol. 9, p. 799893, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2022.799893/full

[11] M. Mozian, J. C. G. Higuera, D. Meger, and G. Dudek, "Learning domain randomization distributions for training robust locomotion policies," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6112–6117.

[12] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Proceedings of the Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 1162–1176. [Online]. Available: https://proceedings.mlr.press/v100/mehta20a.html

[13] G. Tiboni, P. Klink, J. Peters, T. Tommasi, C. D'Eramo, and G. Chalvatzaki, "Domain randomization via entropy maximization," 2024. [Online]. Available: https://arxiv.org/abs/2311.01885

[14] L. Bantel, P. Domanski, and D. Pflüger, "High-fidelity simulation of a cartpole for sim-to-real deep reinforcement learning," in *2024 4th Interdisciplinary Conference on Electrics and Computer (INTCEC)*, 2024, pp. 1–6.

[15] S. Zhang, Y. Luo, Q. Wang, H. Chi, X. Chen, B. Han, and J. Li, "Mixture data for training cannot ensure out-of-distribution generalization," 2024. [Online]. Available: https://arxiv.org/abs/2312.16243

[16] A. Curtis, E. Li, M. Noseworthy, N. Gothoskar, S. Chitta, H. Li, L. P. Kaelbling, and N. Carey, "Flow-based domain randomization for learning and sequencing robotic skills," 2025. [Online]. Available: https://arxiv.org/abs/2502.01800

[17] M. Malmir, J. Josifovski, N. Klarmann, and A. Knoll, "Diarel: Reinforcement learning with disturbance awareness for robust sim2real policy transfer in robot control," 2025. [Online]. Available: https://arxiv.org/abs/2306.09010

[18] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in non-stationary environments: A survey," *IEEE Computational intelligence magazine*, vol. 10, no. 4, pp. 12–25, 2015.

[19] C. T. Ponnambalam, D. Kamran, T. D. Simão, F. A. Oliehoek, and M. T. Spaan, "Back to the future: Solving hidden parameter mdps with hindsight," in *Adaptive Learning Agents Workshop at the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2022.

[20] E. Kim, Y. Karunanayake, and H. Kurniawati, "Reference-based pomdps," *Advances in Neural Information Processing Systems*, vol. 36, pp. 40 659–40 675, 2023.

[21] C. Lu, R. Shi, Y. Liu, K. Hu, S. S. Du, and H. Xu, "Rethinking transformers in solving pomdps," 2024. [Online]. Available: https://arxiv.org/abs/2405.17358

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[23] F. Feng, B. Huang, K. Zhang, and S. Magliacane, "Factored adaptation for non-stationary reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31 957–31 971, 2022.

[24] N. Hansen, R. Jangir, Y. Sun, G. Alenyà, P. Abbeel, A. A. Efros, L. Pinto, and X. Wang, "Self-supervised policy adaptation during deployment," 2021. [Online]. Available: https://arxiv.org/abs/2007.04309

[25] B. Yin, F. Corradi, and S. M. Bohté, "Accurate online training of dynamical spiking neural networks through forward propagation through time," *Nature Machine Intelligence*, vol. 5, no. 5, pp. 518–527, 2023.

[26] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky- integrator neurons," *Neural Networks*, vol. 20, no. 3, pp. 335–352, 2007, echo State Networks and Liquid State Machines. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S089360800700041X

[27] S. Haykin, *Adaptive Filter Theory*. Pearson, 2014. [Online]. Available: https://books.google.co.jp/books?id=J4GRKQEACAAJ

[28] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-13, pp. 834–846, 1983. [Online]. Available: https://api.semanticscholar.org/CorpusID:1522994

[29] P. Wawrzyński, "A cat-like robot real-time learning to run," in *International Conference on Adaptive and Natural Computing Algorithms*. Springer, 2009, pp. 380–390.

[30] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. D. Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, H. Tan, and O. G. Younis, "Gymnasium: A standard interface for reinforcement learning environments," 2025. [Online]. Available: https://arxiv.org/abs/2407.17032

[31] A. Kumar *et al.*, "Rapid motor adaptation for legged robots," *Robotics: Science and Systems (RSS)*, 2021.

[32] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[33] D. P. Kingma, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.