DECOUPLED POLICY ADAPTATION FOR NON-STATIONARY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Catastrophic forgetting is a key obstacle for reinforcement learning (RL) in non-stationary environments, where dynamics may repeatedly alternate across related regimes, causing adaptation to the current context to overwrite behaviors needed under previously encountered ones. Many prior methods rely on task identifiers or restrictive update constraints, making them less suitable for continuously varying contexts or online sample-efficient adaptation. We propose Decoupled Adaptation of Policies via HyperNetwork Interactions (DAPHNI), a sample-efficient architecture for continuous context-driven non-stationary RL that mitigates forgetting through decoupled, similarity-driven adaptation. DAPHNI decomposes a Gaussian policy into a state-only base policy to preserve shared behavior and a context-dependent residual modulator to localize context-specific adaptation. To handle continuously varying contexts, the residual modulator is implemented by a state-conditioned hypernetwork that induces multiplicative state–context interactions. A Kullback-Leibler (KL) divergence constraint further anchors the final policy to the base policy. We theoretically show that DAPHNI yields similarity-driven updates, transferring adaptation across aligned contexts while suppressing interference on dissimilar ones to mitigate forgetting. Comparison experiments and ablation studies show consistent gains over representative baselines across multiple benchmarks and diverse context dynamics, while mechanism analysis supports our design.

1 INTRODUCTION

Reinforcement learning (RL) provides a framework for sequential decision making, but standard RL algorithms typically assume stationary transition and reward dynamics (Sutton & Barto, 2018; Padakandla, 2021). This assumption is often violated in deployed systems (Khetarpal et al., 2022; Kumar et al., 2021; Huleihel et al., 2021; Chakraborty & Murphy, 2014). A common formalization of such non-stationarity is the Contextual Markov Decision Process (CMDP), where the environment is governed by a time-varying context variable $\mathbf{z}_t$ (Hallak et al., 2015). The context may switch among a finite set of regimes, as in classical continual-learning settings, or vary continuously at every time step (Khetarpal et al., 2022; Feng et al., 2022; Hamadanian et al., 2025). When $\mathbf{z}_t$ is unobservable, a common approach is to infer a latent context $\hat{\mathbf{z}}_t$ from recent trajectories and train a context-conditioned policy, often with backbones such as Soft Actor-Critic (SAC) (Xie et al., 2021; Feng et al., 2022; Haarnoja et al., 2018).

Observed or inferred contexts help adaptation, but they do not eliminate *catastrophic forgetting*: updates that improve performance under the current dynamics can overwrite behaviors needed when previously encountered dynamics reappear (Kirkpatrick et al., 2017; Khetarpal et al., 2022; Ribeiro et al., 2019). This risk is especially pronounced when environments repeatedly alternate across related regimes, since the agent must adapt online without knowing when a learned behavior will become useful again. In continuous context spaces, forgetting should therefore be viewed as a similarity-sensitive interference problem: adaptation should transfer across similar contexts while causing minimal interference on dissimilar ones (Hamadanian et al., 2025).

Existing forgetting-mitigation methods mainly fall into three categories. Change-point-based methods detect discrete environmental shifts and isolate learning through resets, policy reuse, or specialized controllers (Padakandla et al., 2020; Alegre et al., 2021; Canonaco et al., 2020). Regularization-based methods preserve prior knowledge by penalizing important parameter changes or projecting gradients

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

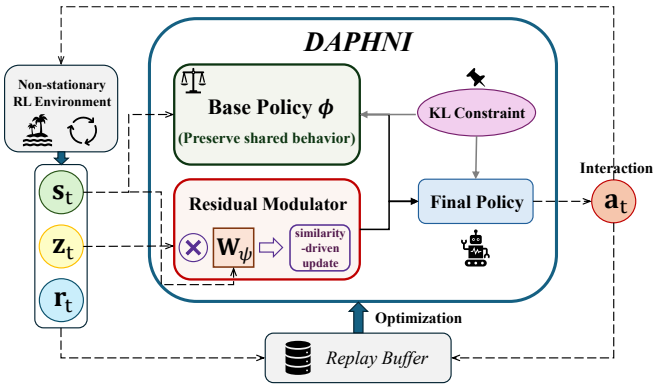


Figure 1: Overview of DAPHNI. The state-only base policy preserves shared behavior, while the hypernetwork-based residual modulator uses the current context to produce localized adaptation. The final policy interacts with the non-stationary environment and is updated from replayed experience under a KL constraint and similarity-driven residual updates.

away from interference-prone directions (Farajtabar et al., 2020; Woo et al., 2022; Kirkpatrick et al., 2017). Architecture-based methods reduce interference by allocating, composing, or modulating network components across tasks (Fernando et al., 2017; Anand & Precup, 2023; Hu et al., 2025). Unlike these three lines of work, Locally Constrained Policy Optimization (LCPO) makes an early attempt to explicitly study catastrophic forgetting under continuous context shifts, but relies on on-policy constrained updates (Hamadanian et al., 2025).

Despite these advances, two limitations remain. **First**, many methods implicitly assume segmented non-stationarity: change-point mechanisms require identifiable regime boundaries, while many architectural methods are designed for discrete task identities. They are therefore ill-suited to environments where $\mathbf{z}_t$ drifts smoothly or changes at every time step. **Second**, methods based on restrictive regularization or on-policy constrained optimization often adapt slowly and require many interactions, limiting sample efficiency in online RL.

To address these limitations, we propose ***D**ecoupled **A**daptation of **P**olicies via **H**yper**N**etwork **I**nteractions* (**DAPHNI**), a policy-side architecture for sample-efficient adaptation under continuous context shifts. DAPHNI is motivated by the observation that, across related dynamics, some of the policy should remain shared while only context-sensitive decisions need to adapt. It decomposes a Gaussian policy into a state-only base policy, which preserves context-invariant behavior, and a context-dependent residual modulator, which localizes context-specific adaptation. To avoid discrete task segmentation, DAPHNI uses a state-conditioned hypernetwork to generate weights applied to $\mathbf{z}_t$, inducing multiplicative state–context interactions that vary smoothly over the continuous context space. A Kullback-Leibler (KL) divergence constraint anchors the final policy to the base policy, preventing the residual branch from dominating shared behavior. Our design yields similarity-driven adaptation: under gradient-based updates, training on context $\mathbf{z}_B$ changes the residual policy associated with context $\mathbf{z}_A$ approximately in proportion to $\mathbf{z}_A^\top \mathbf{z}_B$. Thus, adaptation transfers across aligned contexts while interference on unrelated contexts is suppressed. This decoupled, similarity-driven mechanism mitigates forgetting without relying on restrictive update mechanisms, thereby supporting sample-efficient online adaptation under continuous context shifts. Figure 1 illustrates the overall interaction loop of DAPHNI.

Our main contributions are summarized as follows: **1)** We introduce DAPHNI, a decoupled policy architecture that mitigates catastrophic forgetting in continuous context-driven non-stationary RL by separating context-invariant behavior from context-specific adaptation, enabling sample-efficient adaptation without restrictive update mechanisms. **2)** We theoretically show that the proposed hypernetwork interaction induces similarity-driven updates, explaining how DAPHNI transfers adaptation across aligned contexts and suppresses interference on dissimilar ones. **3)** We conduct comparison experiments and ablation studies across multiple benchmarks and dynamics, showing consistent improvements over representative baselines. Mechanism analysis supports the design.

2 Related Work

Non-Stationary Reinforcement Learning. Non-stationary RL studies sequential decision making when transition or reward dynamics change over time (Padakandla, 2021; Hamadanian et al., 2025). A common formalism is the CMDP, where transition and reward functions are governed by a time-varying context that may be observed or latent (Hallak et al., 2015). When the context is latent, an approach is to infer it from recent trajectories and condition the policy on the inferred representation.

Latent Context Variable Inference. A large body of work focuses on latent context inference for adaptation in non-stationary or multi-task environments. Probabilistic meta-RL methods use latent variables for task adaptation (Rakelly et al., 2019; Zintgraf et al., 2020), and Lifelong Latent Actor-Critic (LILAC) (Xie et al., 2021) extends this idea to continual structured non-stationarity through online probabilistic inference. Subsequent methods track evolving tasks (Poiani et al., 2021; Lee et al., 2020; Woo et al., 2022; Liang et al., 2024), disentangle causal change factors (Feng et al., 2022), use contrastive objectives (Choshen & Tamar, 2023) or Transformer-based encoders (Bing et al., 2023; Qi et al., 2025), and extend latent context inference to visual and safe RL settings (Zhao et al., 2021; Chen et al., 2021). These methods improve environment identification, but context inference alone does not provide a mechanism for preserving previously acquired behaviors. As a result, policies trained across continuously changing contexts can suffer from catastrophic forgetting.

Mitigating Catastrophic Forgetting. *1) Change-point-based methods* mitigate forgetting by detecting discrete environmental shifts and isolating subsequent policy updates (Kessler et al., 2022; Li et al., 2019). For example, Padakandla et al. (2020) use statistical change-point tests to trigger resets, while Model-Based RL Context Detection (MBCD) (Alegre et al., 2021) uses online high-confidence detection to instantiate or reuse specialized policies. Related methods use CUSUM tests (Canonaco et al., 2020), Bayesian detection (Harrison et al., 2020), or reactive exploration (Steinparz et al., 2022) to track environmental changes. Their reliance on boundary detection, however, assumes piecewise stationarity and is less suitable for continuously changing environments, where gradual shifts may evade threshold-based detectors. Frequent resets or policy switches can also reduce sample efficiency.

2) Regularization-based methods preserve prior knowledge by constraining policy updates. They include parameter-importance penalties (Kirkpatrick et al., 2017; Zenke et al., 2017; Nguyen et al., 2018), synaptic modeling (Kaplanis et al., 2018; 2019), and progressive distillation (Schwarz et al., 2018). Gradient-projection approaches further constrain updates. For instance, Woo et al. (2022) adapt Orthogonal Gradient Descent (OGD) (Farajtabar et al., 2020) with a sliding window for non-stationary MDPs. However, strong parameter penalties or orthogonal gradient constraints can restrict useful update directions, slowing adaptation and increasing interaction requirements.

3) Architecture-based methods reduce interference by structurally isolating or composing knowledge. They allocate sub-networks, evolutionary pathways, or compositional modules to sequential tasks (Rusu et al., 2016; Fernando et al., 2017; Mendez et al., 2022), or use task-specific masks and dynamic knowledge pools (Ben-Iwhiwhu et al., 2023; Hu et al., 2025; Geishauser et al., 2022). Anand & Precup (2023) decompose the value function into a permanent component for persistent knowledge and a transient component for rapid adaptation. However, many architectural methods are designed around segmented task sequences, limiting their applicability to continuously shifting dynamics.

Unlike the above three types of methods, LCPO (Hamadanian et al., 2025) is an early attempt to explicitly study catastrophic forgetting under continuous context shifts, but its constrained optimization can limit sample efficiency. DAPHNI addresses these limitations with a decoupled architecture: a context-invariant base policy preserves shared behavior, while a hypernetwork-based residual modulator enables similarity-aware adaptation over continuous contexts without restrictive updates.

3 Problem Formulation

Contextual Markov Decision Process. We model non-stationary environments with a CMDP (Hallak et al., 2015), defined as $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. Here, $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{Z}$ denote the state, action, and context spaces, and $\gamma \in [0, 1]$ is the discount factor. At time step t, the transition function $\mathcal{P}$ and reward function $\mathcal{R}$ are governed by a context $\mathbf{z}_t \in \mathcal{Z}$: $\mathbf{s}_{t+1} \sim \mathcal{P}(\cdot \mid \mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_t)$ and $r_t = \mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_t)$. The agent observes the state $\mathbf{s}_t$ and either receives the current context $\mathbf{z}_t$ or, when it is unobservable,

infers an estimate $\hat{\mathbf{z}}_t$ from past trajectories using a context-inference method such as LILAC (Xie et al., 2021). In both cases, context information is restricted to observations available up to time t.

Continuous and Arbitrary Context Shifts. Unlike piecewise-stationary settings with fixed task boundaries, we consider context-driven non-stationarity where $\mathbf{z}_t$ may change at every time step. The context dynamics are not assumed to follow a known form: they may be deterministic or stochastic, smooth or abrupt, and linear or nonlinear. This formulation covers a broad class of deployed settings, such as robotic locomotion with changing dynamics (Kumar et al., 2021) and healthcare decision making with evolving patient conditions (Komorowski et al., 2018). The agent is therefore required to adapt without knowing the underlying context-transition process.

Objective. We study online RL, where interaction, policy adaptation, and evaluation occur concurrently rather than in separate training and testing phases as in standard meta-RL (Finn et al., 2017). Following Hamadanian et al. (2025), we use online cumulative return as the primary evaluation metric because it is suitable for continuous context environments, and degradation caused by forgetting is reflected directly in return when previously useful behaviors are needed again under recurring dynamics. We additionally design Context Retention Change (CRC) to measure retention under continuous context shifts (Section 5.3). The goal is to learn a context-conditioned policy $\pi(\cdot \mid \mathbf{s}_t, \mathbf{z}_t)$ that maximizes episodic return while adapting to changing contexts and limiting performance degradation on previously encountered dynamics. For an episode of horizon T, we define

$$J_{\text{eval}}(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1} \gamma^t \mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_t)\right], \tag{1}$$

where $\tau = (\mathbf{s}_0, \mathbf{z}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{z}_1, \ldots)$ is generated by $\mathbf{a}_t \sim \pi(\cdot \mid \mathbf{s}_t, \mathbf{z}_t)$ and $\mathbf{s}_{t+1} \sim \mathcal{P}(\cdot \mid \mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_t)$. When the context is latent, $\mathbf{z}_t$ in the policy input denotes the inferred estimate $\hat{\mathbf{z}}_t$.

4 Decoupled Adaptation of Policies via HyperNetwork Interactions

4.1 A Simple Grid-World Example

We first use a simple grid-world example from (Hamadanian et al., 2025) to illustrate the structure of policy changes under context shifts. The agent starts from a fixed state and aims to reach the terminal goal. At each step before reaching the goal, it receives a reward of -1 and can move in four cardinal directions or stay in place. The center cell contains a trap whose status defines the context: $\mathbf{z} = [1, 0]^\top$ when the trap is disabled and $\mathbf{z} = [0, 1]^\top$ when it is enabled. Entering the trap under the enabled context incurs an additional penalty of -10. Figures 2a and 2b show the optimal policies under the two contexts. The context change affects only a localized subset of state-action decisions near the trap, while most goal-directed behavior remains unchanged. This illustrates a common structure in context-conditioned control: different contexts often share a context-invariant policy component, and adaptation is only needed for context-sensitive decisions.

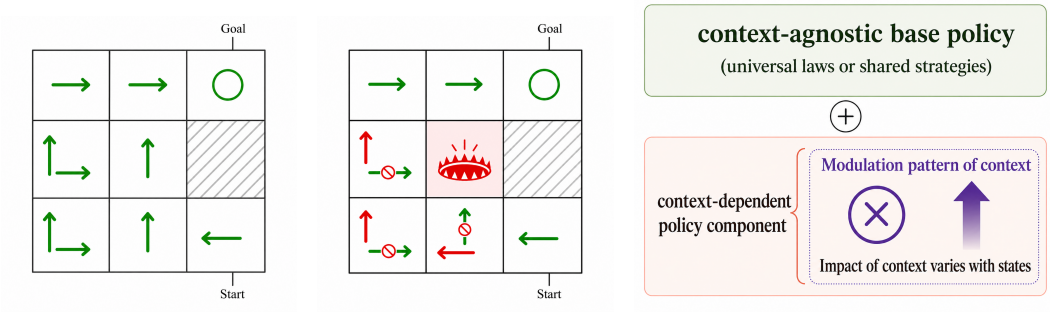
This observation motivates a decoupled policy parameterization. Instead of learning separate policies for different contexts or allowing the entire policy to drift during adaptation, one can represent the policy as a context-invariant base behavior plus a context-dependent residual modulation, as illustrated in Figure 2c. The base policy preserves shared behavior while the residual modulator localizes context-dependent adaptation, directly targeting catastrophic forgetting.

4.2 Decoupled Adaptation of Policies

We now formalize the decomposition suggested by the grid-world example. Following standard maximum-entropy actor-critic methods for continuous control, we parameterize the target optimal policy as a Gaussian with diagonal covariance, $\pi^\star(\cdot \mid \mathbf{s}, \mathbf{z}) = \mathcal{N}\big(\boldsymbol{\mu}^\star(\mathbf{s}, \mathbf{z}), \boldsymbol{\Sigma}^\star(\mathbf{s}, \mathbf{z})\big)$, where $\boldsymbol{\mu}^\star(\mathbf{s}, \mathbf{z}) \in \mathbb{R}^{|\mathcal{A}|}$ and $\boldsymbol{\Sigma}^\star(\mathbf{s}, \mathbf{z})$ is diagonal. We use the vector of log diagonal variances and define the policy-output vector

$$\boldsymbol{\theta}^\star(\mathbf{s}, \mathbf{z}) = [\boldsymbol{\mu}^\star(\mathbf{s}, \mathbf{z}), \log \boldsymbol{\Sigma}^\star(\mathbf{s}, \mathbf{z})]^\top \in \mathbb{R}^{2|\mathcal{A}|}. \tag{2}$$

The log-variance parameterization ensures positive covariance entries after exponentiation.



(a) **(b)** **(c)**

Figure 2: A grid-world illustration of policy decomposition under context shifts. (a) When the trap is disabled, $\mathbf{z} = [1, 0]^\top$; (b) when the trap is enabled, $\mathbf{z} = [0, 1]^\top$. Arrows denote optimal movement directions, and circles denote staying in place. The context shift changes only a localized subset of decisions, motivating DAPHNI to preserve shared behavior with a context-invariant base policy while localizing context-specific adaptation through a residual modulation.

Local decomposition in context space. We assume that, for each state $\mathbf{s}$, $\boldsymbol{\theta}^\star(\mathbf{s}, \mathbf{z})$ is locally differentiable with respect to $\mathbf{z}$ around a reference context $\mathbf{z}_0$. This is a local regularity assumption used only to motivate the architectural form: $\mathbf{z}_0$ need not be known to the algorithm, and the resulting decomposition is not enforced as a global approximation during training.

Applying a first-order multivariate Taylor expansion around $\mathbf{z}_0$ gives

$$\boldsymbol{\theta}^\star(\mathbf{s}, \mathbf{z}) = \boldsymbol{\theta}^\star(\mathbf{s}, \mathbf{z}_0) + \nabla_{\mathbf{z}} \boldsymbol{\theta}^\star(\mathbf{s}, \mathbf{z})|_{\mathbf{z}=\mathbf{z}_0} (\mathbf{z} - \mathbf{z}_0) + \mathcal{O}(\|\mathbf{z} - \mathbf{z}_0\|^2). \tag{3}$$

Let $\mathcal{J}_{\theta^\star}(\mathbf{s}, \mathbf{z}_0) \triangleq \nabla_{\mathbf{z}} \boldsymbol{\theta}^\star(\mathbf{s}, \mathbf{z})|_{\mathbf{z}=\mathbf{z}_0} \in \mathbb{R}^{2|\mathcal{A}|\times|\mathcal{Z}|}$ denote the Jacobian of the optimal output vector with respect to the context at $\mathbf{z}_0$. Ignoring the second-order remainder in Eq. 3, we obtain

$$\boldsymbol{\theta}^\star(\mathbf{s}, \mathbf{z}) \approx \underbrace{[\boldsymbol{\theta}^\star(\mathbf{s}, \mathbf{z}_0) - \mathcal{J}_{\theta^\star}(\mathbf{s}, \mathbf{z}_0)\mathbf{z}_0]}_{\text{context-agnostic}} + \underbrace{\mathcal{J}_{\theta^\star}(\mathbf{s}, \mathbf{z}_0)\mathbf{z}}_{\text{context-dependent}} . \tag{4}$$

Equation 4 motivates a context-agnostic component for shared behavior and a context-dependent residual for local adaptation, thereby mitigating catastrophic forgetting without modifying the entire policy. In practice, we require neither $\boldsymbol{\theta}^\star$, $\mathbf{z}_0$, nor $\mathcal{J}_{\theta^\star}(\mathbf{s}, \mathbf{z}_0)$. All networks are learned end-to-end and may represent nonlinear context-conditioned policies.

Base policy. We instantiate the context-agnostic component in Eq. 4 with a state-only base policy network parameterized by ϕ: $\boldsymbol{\theta}_\phi(\mathbf{s}) = [\boldsymbol{\mu}_\phi(\mathbf{s}), \log \boldsymbol{\Sigma}_\phi(\mathbf{s})]^\top$. The corresponding base policy is

$$\pi_\phi(\cdot \mid \mathbf{s}) = \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{s}), \boldsymbol{\Sigma}_\phi(\mathbf{s})). \tag{5}$$

Because π_ϕ receives only the state, it is structurally biased toward context-invariant behavior and serves as a shared anchor for context-specific residual corrections.

Residual modulator. To model the context-dependent term in Eq. 4, we introduce a residual modulator parameterized by ψ. Hypernetworks generate the weights of another network and have been used in meta-RL (Beck et al., 2023). In many context-conditioned policies, the context generates weights operating on state features. DAPHNI instead uses the state to generate weights applied to the context. This avoids discrete task-specific modules and induces a multiplicative state–context interaction for smooth residual adaptation over continuous contexts.

The residual modulator generates state-dependent weights applied to $\mathbf{z}$:

$$\mathbf{W}_\psi(\mathbf{s}) = \text{HyperNet}_\psi(\mathbf{s}) = \begin{bmatrix} \mathbf{W}_\mu(\mathbf{s}) \\ \mathbf{W}_{\Delta\sigma}(\mathbf{s}) \end{bmatrix}, \tag{6}$$

where $\mathbf{W}_\mu(\mathbf{s}), \mathbf{W}_{\Delta\sigma}(\mathbf{s}) \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{Z}|}$ correspond to the mean and log-variance modulation components, respectively. The residual outputs are then

$$\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}) = \mathbf{W}_\mu(\mathbf{s})\mathbf{z}, \qquad \Delta\boldsymbol{\sigma}_\psi(\mathbf{s}, \mathbf{z}) = \mathbf{W}_{\Delta\sigma}(\mathbf{s})\mathbf{z}. \tag{7}$$

Here, $\Delta\boldsymbol{\sigma}_\psi(\mathbf{s}, \mathbf{z})$ is added in log-variance space, preserving positive covariance after exponentiation. The residual modulator is a learned, state-dependent approximation to the local context sensitivity in Eq. 4. Its multiplicative interaction induces similarity-driven updates that share adaptation across related contexts while reducing interference on dissimilar ones, as shown in the next section.

4.3 HyperNetwork Interactions with Context

We next characterize how the hypernetwork-based residual modulator controls interference across contexts. The result below shows that the multiplicative state–context interaction induces an update bias proportional to contextual alignment, providing forgetting mitigation under continuous contexts without discrete task boundaries or restrictive update mechanisms.

Proposition 1 (Similarity-Driven Policy Update)**.** *Consider a hypernetwork architecture parameterized by ψ, which takes the state variable $\mathbf{s}$ as input and employs a multiplicative interaction with the context variable $\mathbf{z}$. Under the standard Neural Tangent Kernel (NTK) approximation and continuous-time gradient descent dynamics while training on a current context $\mathbf{z}_B$, the rate of change of the residual policy mean $\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)$ corresponding to context $\mathbf{z}_A$ satisfies:*

$$\frac{d\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)}{dt} \propto \mathbf{z}_A^\top \mathbf{z}_B. \tag{8}$$

The proof and additional discussion are provided in Appendix A.

Proposition 1 shows that the residual modulator induces similarity-driven updates. The inner product $\mathbf{z}_A^\top \mathbf{z}_B$ is a scale-weighted alignment, a natural similarity signal for one-hot, normalized, or otherwise comparably scaled observed or learned contexts. Positive alignment transfers updates from $\mathbf{z}_B$ to $\mathbf{z}_A$, near-orthogonality suppresses interference, and negative alignment reverses the update, consistent with opposing policy corrections. DAPHNI therefore transfers adaptation across related contexts while reducing destructive interference on dissimilar ones, directly targeting catastrophic forgetting under continuous non-stationarity in a sample-efficient manner.

4.4 KL Divergence Constraint on Policies

Combining the base policy and the residual modulator gives the final context-conditioned Gaussian policy $\pi_{\phi,\psi}(\cdot \mid \mathbf{s}, \mathbf{z}) = \mathcal{N}\big(\boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}, \mathbf{z}), \boldsymbol{\Sigma}_{\phi,\psi}(\mathbf{s}, \mathbf{z})\big)$, where

$$\boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}, \mathbf{z}) = \boldsymbol{\mu}_\phi(\mathbf{s}) + \boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}), \qquad \log \boldsymbol{\Sigma}_{\phi,\psi}(\mathbf{s}, \mathbf{z}) = \log \boldsymbol{\Sigma}_\phi(\mathbf{s}) + \Delta\boldsymbol{\sigma}_\psi(\mathbf{s}, \mathbf{z}). \tag{9}$$

The residual modulator enables context-specific adaptation, but its access to the context also makes it more expressive than the state-only base policy. Without control, the residual branch may dominate the policy output and weaken the context-invariant behavior needed to prevent forgetting. We therefore anchor the final policy to the base policy with $D_{\mathrm{KL}}\big(\pi_{\phi,\psi}(\cdot \mid \mathbf{s}, \mathbf{z}) \,\|\, \pi_\phi(\cdot \mid \mathbf{s})\big)$.

For diagonal Gaussian policies, this KL divergence has a closed-form expression. Let $\sigma^2_{\phi,k}(\mathbf{s})$ denote the k-th diagonal variance of $\boldsymbol{\Sigma}_\phi(\mathbf{s})$, and let $\mu_{\psi,k}(\mathbf{s}, \mathbf{z})$ and $\Delta\sigma_{\psi,k}(\mathbf{s}, \mathbf{z})$ denote the corresponding residual mean and log-variance modulation. Then

$$D_{\mathrm{KL}} = \frac{1}{2}\sum_{k=1}^{|\mathcal{A}|}\left(\underbrace{\exp(\Delta\sigma_{\psi,k}) - \Delta\sigma_{\psi,k} - 1}_{\text{variance modulation penalty}} + \underbrace{\frac{(\mu_{\psi,k})^2}{\sigma^2_{\phi,k}}}_{\substack{\text{uncertainty-weighted} \\ \text{mean penalty}}}\right). \tag{10}$$

This form gives the KL constraint a direct interpretation. The first term penalizes excessive variance modulation. The second term is a variance-weighted ℓ_2 penalty on the residual mean: residual deviations are penalized more strongly when the base policy is confident, and are allowed to be larger when the base policy is uncertain. Thus, the residual modulator is encouraged to adapt mainly along dimensions where context-specific changes are needed, while preserving stable shared behavior.

We train DAPHNI with a maximum-entropy objective regularized by the expected policy KL:

$$J(\pi_{\phi,\psi}) = \mathbb{E}_{\tau\sim\pi_{\phi,\psi}}\left[\sum_{t=0}^{\infty}\gamma^t\big(\mathcal{R}(\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_t) - \alpha \log \pi_{\phi,\psi}(\mathbf{a}_t \mid \mathbf{s}_t, \mathbf{z}_t)\big)\right] - \beta\mathbb{E}_{\mathbf{s},\mathbf{z}}[D_{\mathrm{KL}}] \tag{11}$$

Algorithm 1 The Pipeline of the Proposed DAPHNI

Input: Environment $\mathcal{M}$, replay buffer $\mathcal{D}$, base policy parameters ϕ, residual modulator parameters ψ, initial KL coefficient β, target KL D_{target}

for each environment step t **do**
  Context Acquisition: observe or infer the current context $\mathbf{z}_t$
  Base Policy: $\boldsymbol{\theta}_\phi(\mathbf{s}_t) = [\boldsymbol{\mu}_\phi(\mathbf{s}_t), \log \boldsymbol{\Sigma}_\phi(\mathbf{s}_t)]^\top$
  Residual Modulation: $\mathbf{W}_\psi(\mathbf{s}_t) = \text{HyperNet}_\psi(\mathbf{s}_t)$, $\boldsymbol{\mu}_\psi = \mathbf{W}_\mu(\mathbf{s}_t)\mathbf{z}_t$, $\Delta\boldsymbol{\sigma}_\psi = \mathbf{W}_{\Delta\sigma}(\mathbf{s}_t)\mathbf{z}_t$
  Policy Construction: $\boldsymbol{\mu}_{\phi,\psi} = \boldsymbol{\mu}_\phi + \boldsymbol{\mu}_\psi$, $\log \boldsymbol{\Sigma}_{\phi,\psi} = \log \boldsymbol{\Sigma}_\phi + \Delta\boldsymbol{\sigma}_\psi$
  Interaction: sample $\mathbf{a}_t \sim \pi_{\phi,\psi}(\cdot \mid \mathbf{s}_t, \mathbf{z}_t)$ and store the transition in $\mathcal{D}$
  Optimization: update ϕ, ψ by SAC with KL regularization $D_{\text{KL}}(\pi_{\phi,\psi}(\cdot \mid \mathbf{s}, \mathbf{z}) \parallel \pi_\phi(\cdot \mid \mathbf{s}))$
  KL Regulation: update β with the target-KL dual objective
end for

Output: Learned policy $\pi_{\phi,\psi}$

where α is the entropy temperature and $\beta \geq 0$ controls the KL strength. We adapt β with a target-KL dual update. Appendix B.1 gives the constrained formulation and update rule, and Appendix B.2 reports the sensitivity analysis.

Overall, DAPHNI mitigates catastrophic forgetting by separating context-invariant behavior from context-specific adaptation: the base policy anchors shared behavior, the KL constraint limits residual overwriting, and the hypernetwork transfers updates across related contexts while limiting interference on dissimilar ones. It addresses the two main limitations of prior methods by operating on continuous contexts without discrete change points or task-specific modules, and by controlling interference through decoupled, similarity-driven adaptation rather than restrictive update mechanisms, supporting sample-efficient learning under continuous shifts. Algorithm 1 summarizes the pipeline.

5 Experiments

5.1 Experimental Setup

Following standard practice in non-stationary RL (Bing et al., 2023), we evaluate DAPHNI on four Gymnasium MuJoCo continuous-control benchmarks: Half Cheetah-v4, Hopper-v4, Walker2d-v4, and Ant-v4, spanning diverse locomotion morphologies, state–action dimensionalities, and contact dynamics. Following prior work (Feng et al., 2022; Hamadanian et al., 2025), time-varying external wind induces four context dynamics: sinusoidal, triangular, stochastic-noise-perturbed, and piecewise discrete, covering smooth predictable changes, noisy fluctuations, and abrupt shifts. A multivariate dynamic form jointly varies wind, body mass, and friction. Its construction and the wind profiles are provided in Appendix D.1. We also evaluate Lunar Lander with a context varying gravity (transition) and fuel cost (reward). All experiments use 10 random seeds and observed contexts to isolate forgetting mitigation from context inference. Experiments with inferred-context are in Appendix C.

5.2 Comparison Experiments

We compare DAPHNI with LCPO (Hamadanian et al., 2025), context-augmented SAC (Haarnoja et al., 2018), the regularization-based Sliding OGD (Woo et al., 2022), the change-point-based MBCD (Alegre et al., 2021), and the architecture-based Permanent–Transient Deep Q-Network (PT-DQN) (Anand & Precup, 2023). SAC receives $\mathbf{z}$ as an additional policy input without decoupled residual adaptation. Since LCPO is on-policy, we allocate it ten times DAPHNI's environment-step budget to match wall-clock training time. More fairness details are in Appendix D.2.

Figure 3 shows smoothed learning curves on Half Cheetah under four wind dynamics. DAPHNI achieves higher, more stable returns than LCPO and SAC throughout these settings. LCPO improves with training, but locally constrained on-policy updates slow adaptation. SAC reuses experience through replay, yet state–context concatenation leaves shared behavior and context-specific adaptation entangled, with larger fluctuations under recurring or smoothly varying dynamics. These results suggest that DAPHNI achieves sample-efficient adaptation while preserving useful behaviors.

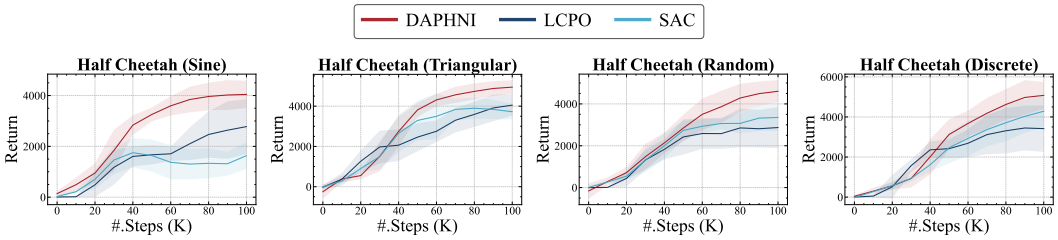


Figure 3: Learning curves on Half Cheetah under four wind dynamics. Curves report the mean evaluation episode return over random seeds, and shaded regions indicate one standard deviation.

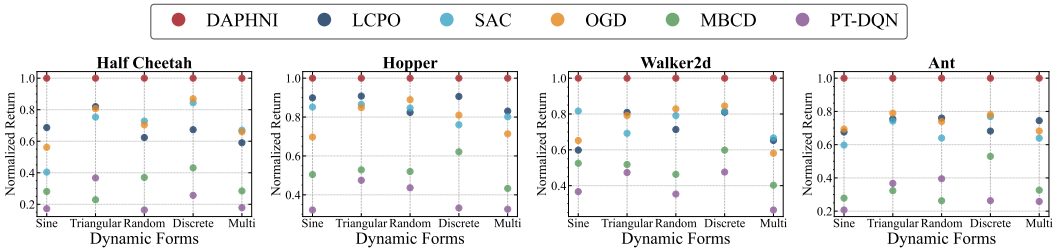


Figure 4: Normalized average return across four benchmarks and various contextual dynamic forms. Each point is scaled within its setting such that the best-performing method has value 1.

Figure 4 shows that DAPHNI achieves the highest mean return across all four MuJoCo benchmarks and five dynamic forms, extending its advantage across morphologies to jointly varying physical factors. Sliding OGD can stabilize updates along restrictive directions, but does not exploit continuous context similarity. MBCD's discrete change-point detection can fragment gradual drift, while PT-DQN's discrete task-style value decomposition does not explicitly exploit such similarity. DAPHNI instead supports smooth adaptation through similarity-driven residual updates, without discrete segmentation or restrictive update mechanisms. Lunar Lander results are provided in Appendix D.4.

The gains are particularly pronounced in higher-dimensional environments such as Ant and under continuously varying dynamics. Direct concatenation can dilute context among rich state features. DAPHNI instead preserves state-only base behavior and assigns context a dedicated residual branch. Similarity-driven updates share adaptation across related contexts while limiting interference on dissimilar ones, mitigating forgetting where interference can accumulate.

5.3 Ablation Studies

We evaluate five variants: (i) w/o modulator, using context-conditioned SAC instead of decoupled residual adaptation (Section 5.2); (ii) w/o KL, removing the KL constraint; (iii) concat modulator, replacing the hypernetwork-based residual with state–context concatenation; (iv) w/o base policy, removing the base policy; and (v) inverted modulator, using context to generate weights for state.

Figure 5 shows that DAPHNI achieves the highest mean return across all five forms, with pronounced gains under continuous changes. Removing the modulator eliminates a dedicated adaptation branch. Removing KL weakens the context-invariant anchor and risks residual adaptation overwriting shared behavior, while removing the base policy eliminates the shared state-to-action pathway. The lower returns of concatenation-based and inverted modulators further corroborate the necessity of similarity-driven interactions in decoupled adaptation to mitigate forgetting.

We introduce Context Retention Change, a forgetting-resistance metric designed for continuously varying contexts. We first evaluate a previously encountered high-context (wind) region, continue training toward the opposite context extreme, and reevaluate the same region along the same drifting context trajectory. If R_{before} and R_{after} are the evaluation returns, $\text{CRC} = 100(R_{\text{after}} - R_{\text{before}})/R_{\text{before}}$. Larger values indicate better retention. Full experimental details are in Appendix D.3.

In Table 1, w/o base policy achieves the highest CRC in three settings, consistent with concentrating adaptation in the similarity-driven modulator while avoiding potential drift in the shared base. Its lower return nevertheless shows that strong relative retention does not ensure effective adaptation.

Table 1: Context Retention Change (CRC) (%) on Half Cheetah under four wind dynamics. CRC measures the percentage change in return when previously encountered high-context regions are revisited after continued training. Larger values indicate better context retention.

	DAPHNI	w/o modulator	w/o KL	concat modulator	w/o base policy	inverted modulator
Sine	16.6%	10.8%	6.5%	12.3%	**18.7%**	7.7%
Triangular	22.7%	13.5%	8.8%	18.6%	**27.1%**	13.1%
Random	19.4%	9.1%	13.4%	16.6%	**22.5%**	13.9%
Discrete	**22.6%**	9.6%	13.8%	7.6%	20.4%	5.5%

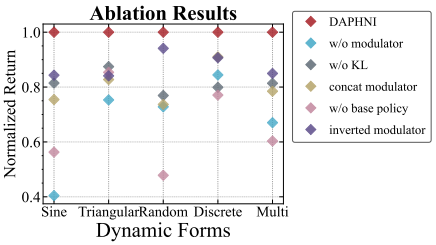


Figure 5: Ablation results (normalized average return) on Half Cheetah under various contextual dynamic forms.

Table 2: Base policy extraction results (average return) on Half Cheetah, compared with the full DAPHNI and a random policy.

	DAPHNI	extracted base policy	random policy
Sine	4044.7	2606.4	75.8
Triangular	4944.1	2491.2	66.5
Random	4604.1	2357.6	59.9
Discrete	5076.9	2830.4	58.8
Multi	3452.8	1311.5	43.7

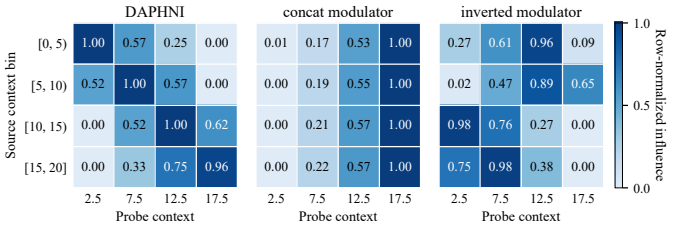


Figure 6: Empirical analysis of the hypernetwork interaction mechanism underlying Proposition 1 on Half Cheetah. Each matrix reports the row-wise min–max normalized influence of an update from a source context bin (rows) on residual outputs evaluated at probe contexts (columns), averaged across runs.

DAPHNI combines high CRC with the highest return, supporting the complementary roles of similarity-driven residual updates in retention and shared base behavior in adaptation.

5.4 Mechanism Analysis

Base policy extraction. To test whether the state-only branch learns shared behavior, we extract the trained base policy and evaluate it without further training alongside the full policy and a random policy under the same protocol. In Table 2, the extracted policy substantially outperforms the random policy, demonstrating that the base learns useful shared behavior. Full DAPHNI further improves return, supporting the residual modulator's contribution to context-specific adaptation.

Hypernetwork interaction. To test whether the residual modulator exhibits similarity-driven updates during training, we partition each batch into four source context (wind) intervals. Using the same batch states before and after that update, we evaluate the residual mean at probe contexts 2.5, 7.5, 12.5, and 17.5, and use the ℓ_2 norm of the mean-output change as the influence. Figure 6 shows the relative influence within each source interval after row-wise min–max normalization. DAPHNI concentrates influence on the same and nearby contexts, whereas concat and inverted modulators lack this consistent local pattern. This supports the similarity-driven update bias of Proposition 1 in the trained policy. Full designs for both analyses are in Appendix D.3. Complete results and hyperparameters for Section 5 are in Appendices D.4 and D.5, respectively.

6 Conclusion

We presented DAPHNI, a policy-side approach for mitigating catastrophic forgetting in continuous context-driven non-stationary RL. DAPHNI decouples a state-only base policy from a hypernetwork-based residual modulator, with a KL constraint anchoring shared behavior. This design induces similarity-driven updates, transferring adaptation across aligned contexts while reducing interference on dissimilar ones without relying on discrete task segmentation or restrictive update mechanisms. Theory characterizes this mechanism. Comparison experiments and ablation studies show consistent gains over representative baselines, while mechanism analysis supports the proposed design.

AI USE STATEMENT

Generative AI tools were used only to aid or polish writing and for retrieval and discovery. They were not used to define the research problem, develop DAPHNI, derive or verify theoretical results, design or run experiments, generate or interpret results, or make research decisions. The authors led and verified all core research, implementation, analysis, and final content. All AI-assisted material was reviewed by the authors, who take full responsibility for the final text, claims, and artifacts.

ETHICS STATEMENT

This work uses publicly available simulation benchmarks and involves no human or animal subjects, private data, or personal information. We identify no specific ethical or privacy concerns arising from the methods or experiments. The study follows applicable ethical standards.

REPRODUCIBILITY STATEMENT

We ensure the reproducibility of our work through the following aspects. The complete DAPHNI procedure is specified in Algorithm 1. All benchmarks used in the study are publicly available. The construction of the contextual dynamic forms is described in Appendix D.1, baseline configurations in Appendix D.2, experimental protocols in Appendix D.3, and hyperparameters in Appendix D.5. Together, these specifications provide the information needed to reproduce the reported results.

REFERENCES

Lucas N. Alegre, Ana L. C. Bazzan, and Bruno C. da Silva. Minimum-delay adaptation in non-stationary reinforcement learning via online high-confidence change-point detection. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 97–105, 2021.

Nishanth Anand and Doina Precup. Prediction and control in continual reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 63779–63817, 2023.

Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning*, volume 205, pp. 1478–1487, 2023.

Eseoghene Ben-Iwhiwhu, Saptarshi Nath, Praveen Kumar Pilly, Soheil Kolouri, and Andrea Soltoggio. Lifelong reinforcement learning with modulating masks. *Transactions on Machine Learning Research*, 2023.

Zhenshan Bing, David Lerch, Kai Huang, and Alois Knoll. Meta-reinforcement learning in non-stationary and dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3476–3491, 2023.

Giuseppe Canonaco, Marcello Restelli, and Manuel Roveri. Model-free non-stationarity detection and adaptation in reinforcement learning. In *Proceedings of the 24th European Conference on Artificial Intelligence*, volume 325, pp. 1047–1054, 2020.

Bibhas Chakraborty and Susan A. Murphy. Dynamic treatment regimes. *Annual Review of Statistics and Its Application*, 1:447–464, 2014.

Baiming Chen, Zuxin Liu, Jiacheng Zhu, Mengdi Xu, Wenhao Ding, Liang Li, and Ding Zhao. Context-aware safe reinforcement learning for non-stationary environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10689–10695, 2021.

Era Choshen and Aviv Tamar. ContraBAR: Contrastive bayes-adaptive deep RL. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 6005–6027, 2023.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, volume 108, pp. 3762–3773, 2020.

Fan Feng, Biwei Huang, Kun Zhang, and Sara Magliacane. Factored adaptation for non-stationary reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 31957–31971, 2022.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A. Rusu, Alexander Pritzel, and Daan Wierstra. PathNet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1126–1135, 2017.

Christian Geishauser, Carel van Niekerk, Hsien-chin Lin, Nurul Lubis, Michael Heck, Shutong Feng, and Milica Gašić. Dynamic dialogue policy for continual reinforcement learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 266–284, 2022.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual Markov decision processes. *arXiv preprint arXiv:1502.02259*, 2015.

Pouya Hamadanian, Arash Nasr-Esfahany, Malte Schwarzkopf, Siddhartha Sen, and Mohammad Alizadeh. Online reinforcement learning in non-stationary context-driven environments. In *International Conference on Learning Representations*, 2025.

James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. Continuous meta-learning without tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17571–17581, 2020.

Jinwu Hu, Zihao Lian, Zhiquan Wen, Chenghao Li, Guohao Chen, Xutao Wen, Bin Xiao, and Mingkui Tan. Continual knowledge adaptation for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Wasim Huleihel, Soumyabrata Pal, and Ofer Shayevitz. Learning user preferences in non-stationary environments. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pp. 1432–1440, 2021.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 2497–2506, 2018.

Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 3242–3251, 2019.

Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J. Roberts. Same state, different task: Continual reinforcement learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7143–7151, 2022.

Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Matthieu Komorowski, Leo A. Celi, Omar Badawi, Anthony C. Gordon, and A. Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine*, 24(11):1716–1720, 2018.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2021.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 5757–5766, 2020.

Siyuan Li, Fangda Gu, Guangxiang Zhu, and Chongjie Zhang. Context-aware policy reuse. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 989–997, 2019.

Anthony Liang, Guy Tennenholtz, Chih-Wei Hsu, Yinlam Chow, Erdem Biyik, and Craig Boutilier. Dynamite-rl: A dynamic model for improved temporal meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 141390–141416, 2024.

Jorge A. Mendez, Harm van Seijen, and Eric Eaton. Modular lifelong reinforcement learning via neural composition. In *International Conference on Learning Representations*, 2022.

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational continual learning. In *International Conference on Learning Representations*, 2018.

Sindhu Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys*, 54(6):1–25, 2021. Article No.: 127.

Sindhu Padakandla, K. J. Prabuchandran, and Shalabh Bhatnagar. Reinforcement learning algorithm for non-stationary environments. *Applied Intelligence*, 50:3590–3606, 2020.

Riccardo Poiani, Andrea Tirinzoni, and Marcello Restelli. Meta-reinforcement learning by tracking task non-stationarity. *arXiv preprint arXiv:2105.08834*, 2021.

Chenyang Qi, Huiping Li, and Panfeng Huang. TIMRL: A novel meta-reinforcement learning framework for non-stationary and multi-task environments. *arXiv preprint arXiv:2501.07146*, 2025.

Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pp. 5331–5340, 2019.

João Ribeiro, Francisco S. Melo, and João Dias. Multi-task learning and catastrophic forgetting in continual reinforcement learning. In *Proceedings of the 5th Global Conference on Artificial Intelligence*, volume 65 of *EPiC Series in Computing*, pp. 163–175. EasyChair, 2019.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 4528–4537, 2018.

Christian Alexander Steinparz, Thomas Schmied, Fabian Paischer, Marius-Constantin Dinu, Vihang Prakash Patil, Angela Bitto-Nemling, Hamid Eghbal-Zadeh, and Sepp Hochreiter. Reactive exploration to cope with non-stationarity in lifelong reinforcement learning. In *Proceedings of The 1st Conference on Lifelong Learning Agents*, volume 199, pp. 441–469, 2022.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition, 2018.

Pingjie Wang, Hongcheng Liu, Yusheng Liao, Ziqing Fan, Yaxin Du, Shuo Tang, Yanfeng Wang, and Yu Wang. Selecting auxiliary data via neural tangent kernels for low-resource domains. *arXiv preprint arXiv:2511.07380*, 2025.

Honguk Woo, Gwangpyo Yoo, and Minjong Yoo. Structure learning-based task decomposition for reinforcement learning in non-stationary environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8657–8665, 2022.

Annie Xie, James Harrison, and Chelsea Finn. Deep reinforcement learning amidst continual structured non-stationarity. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 11393–11403, 2021.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 3987–3995, 2017.

Zihao Zhao, Anusha Nagabandi, Kate Rakelly, Chelsea Finn, and Sergey Levine. MELD: Meta-reinforcement learning from images via latent state models. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155, pp. 1246–1261, 2021.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: A very good method for bayes-adaptive deep RL via meta-learning. In *International Conference on Learning Representations*, 2020.

A PROOFS

A.1 A LEMMA BASED ON NEURAL TANGENT KERNEL (NTK) THEOREM

To establish the proof of Proposition 1, we first formalize a critical supporting lemma regarding the geometric properties of the gradients under the NTK regime. While this is rooted in the tensor limits of Jacot et al. (2018), we explicitly state it here in terms of gradient inner products for clarity and self-containment.

Lemma 1 (Asymptotic Orthogonality of Parameter Gradients Across Output Dimensions)**.** *Let $f_\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ be a fully connected neural network of depth L, parameterized by $\theta \in \mathbb{R}^P$. Assume the standard NTK parameterization, where the weights and biases are initialized as independent and identically distributed (i.i.d.) random variables with zero mean. Then, in the infinite-width limit ($n_1, \ldots, n_{L-1} \to \infty$), the parameter gradients corresponding to any two distinct output dimensions $i, j \in \{1, \ldots, n_L\}$ with $i \neq j$ are asymptotically orthogonal. Explicitly, for any input pair $x, x' \in \mathbb{R}^{n_0}$, we have:*

$$\lim_{n_1,\ldots,n_{L-1}\to\infty} \langle \nabla_\theta f_{\theta,i}(x,t), \nabla_\theta f_{\theta,j}(x',t) \rangle = 0 \tag{12}$$

where the limit holds in probability at initialization ($t = 0$) and uniformly over any finite training time interval $t \in [0, T]$ during continuous-time gradient descent.

Proof. We draw upon the foundational multidimensional NTK theory established by Jacot et al. (2018). Let $J_\theta(x) \in \mathbb{R}^{n_L \times P}$ denote the Jacobian matrix of the network output with respect to the parameters, whose i-th row is the transposed gradient vector $\nabla_\theta f_{\theta,i}(x)^\top$. The empirical NTK is formally defined as the $n_L \times n_L$ matrix-valued kernel given by:

$$\Theta^{(L)}(\theta; x, x') = J_\theta(x) J_\theta(x')^\top = \sum_{p=1}^{P} \partial_{\theta_p} f_\theta(x) \otimes \partial_{\theta_p} f_\theta(x'). \tag{13}$$

By this definition, the (i, j)-th element of the empirical NTK matrix exactly corresponds to the inner product of the parameter gradients for the i-th and j-th output dimensions:

$$\Theta^{(L)}_{i,j}(\theta; x, x') = \langle \nabla_\theta f_{\theta,i}(x), \nabla_\theta f_{\theta,j}(x') \rangle. \tag{14}$$

According to Theorem 1 (for initialization) and Theorem 2 (for training dynamics) in Jacot et al. (2018), as the hidden layer widths $n_1, \ldots, n_{L-1} \to \infty$, the empirical NTK converges in probability to a deterministic, time-invariant limit kernel:

$$\Theta^{(L)}(\theta(t); x, x') \xrightarrow{n\to\infty} \Theta^{(L)}_\infty(x, x') \otimes I_{n_L}, \tag{15}$$

where $\Theta^{(L)}_\infty(x, x') \in \mathbb{R}$ is a scalar kernel entirely determined by the depth L, the activation function σ, and the initialization variance, while I_{n_L} is the $n_L \times n_L$ identity matrix.

The Kronecker product with the identity matrix forces the off-diagonal entries of the limiting tensor to strictly vanish. Consequently, applying this limit to Eq. 14, for any distinct output dimensions $i \neq j$, the asymptotic limit of the (i, j)-th entry is identically zero:

$$\lim_{n_1,\ldots,n_{L-1}\to\infty} \Theta^{(L)}_{i,j}(\theta(t); x, x') = \Theta^{(L)}_\infty(x, x') \cdot \delta_{ij} = 0, \tag{16}$$

where δ_{ij} is the Kronecker delta. This proves the exact asymptotic orthogonality of the parameter gradients across different output channels, concluding the proof. □

Remark 1 (Near-Orthogonality in Finite-Width Models)**.** *While Lemma 1 establishes exact orthogonality in the strict infinite-width regime, in practical, finite-width neural networks, the parameter gradients for distinct output dimensions exhibit a "nearly orthogonal" property. Recent literature leveraging the NTK for parameter-efficient fine-tuning has both empirically observed and theoretically exploited this phenomenon. For instance, Wang et al. (2025) empirically demonstrated that approximating the empirical NTK by ignoring cross-dimensional gradient products (relying on near-orthogonality) yields a highly stable and memory-efficient Jacobian-free selector, enabling effective transfer learning and data selection in low-resource domains.*

A.2 Notation and Preliminaries

To facilitate the reading of the proof, especially the main result in Proposition 1, we summarize the specific symbols in Table 3.

Table 3: Summary of notation in the proof.

Symbol	Description
$\mathbf{s}, \mathbf{z}, \mathbf{a}$	State variable, context variable, and action variable, respectively.
$\mathcal{A}, \mathcal{Z}$	Action space and context space, respectively.
ϕ, ψ	Parameters of base policy network and residual modulator, respectively.
Ψ	Parameter space of the residual modulator.
$\boldsymbol{\mu}_\phi(\cdot), \boldsymbol{\mu}_\psi(\cdot), \boldsymbol{\mu}_{\phi,\psi}(\cdot)$	Mean term output of base policy network, residual modulator, and final policy network, respectively.
$\boldsymbol{\Sigma}_\phi(\cdot), \Delta\boldsymbol{\sigma}_\psi(\cdot), \boldsymbol{\Sigma}_{\phi,\psi}(\cdot)$	Variance term output of base policy network, variance factor output of residual modulator, and variance term output of final policy network, respectively.
η	Learning rate.
d_B	Stationary state distribution under context $\mathbf{z}_B$.
$A^{\pi_{\phi,\psi}}(\cdot)$	Advantage function associated with the current policy $\pi_{\phi,\psi}$.
$\pi_{\phi,\psi}(\mathbf{a} \mid \cdot)$	Density output of the multivariate Gaussian policy $\pi_{\phi,\psi}$.
$\mathbf{W}_\psi(\cdot)$	Weight matrix generated by the hypernetwork.

A.3 Proof of Proposition 1

Proposition 1 (Similarity-Driven Policy Update). Consider a hypernetwork architecture parameterized by ψ, which takes the state variable $\mathbf{s}$ as input and employs a multiplicative interaction with the context variable $\mathbf{z}$. Under the standard Neural Tangent Kernel (NTK) approximation and continuous-time gradient descent dynamics while training on a current context $\mathbf{z}_B$, the rate of change of the residual policy mean $\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)$ corresponding to context $\mathbf{z}_A$ satisfies:

$$\frac{d\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)}{dt} \propto \mathbf{z}_A^\top \mathbf{z}_B.$$

Proof. Under the proposed decoupled policy mechanism, we have $\boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}, \mathbf{z}) = \boldsymbol{\mu}_\phi(\mathbf{s}) + \boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z})$ and $\boldsymbol{\Sigma}_{\phi,\psi}(\mathbf{s}, \mathbf{z}) = \boldsymbol{\Sigma}_\phi(\mathbf{s}) \odot \exp(\Delta\boldsymbol{\sigma}_\psi(\mathbf{s}, \mathbf{z}))$. Because our primary interest lies in the shift of the policy mean, we slightly abuse the notation ψ to denote all parameters in the residual modulator associated with $\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z})$ for the sake of simplicity. Since there is no parameter sharing between the mean and variance heads within the residual modulator, we abbreviate $\boldsymbol{\Sigma}_{\phi,\psi}(\mathbf{s}, \mathbf{z})$ as $\boldsymbol{\Sigma}$ to streamline the notation without compromising mathematical rigor. Assume the agent is currently operating in an environment characterized by the context $\mathbf{z}_B$. Updating the parameter ψ via continuous-time gradient descent based on the policy gradient theorem yields the following dynamics:

$$\frac{d\psi}{dt} = \eta \cdot \mathbb{E}_{\mathbf{s}' \sim d_B, \mathbf{a} \sim \pi_{\phi,\psi}} \left[\nabla_\psi \log \pi_{\phi,\psi}(\mathbf{a} \mid \mathbf{s}', \mathbf{z}_B) A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a}) \right], \tag{17}$$

where the action $\mathbf{a} \in \mathbb{R}^{|\mathcal{A}|}$ is sampled from the multivariate Gaussian distribution $\mathbf{a} \sim \mathcal{N}(\boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}', \mathbf{z}_B), \boldsymbol{\Sigma})$. By the properties of the multivariate Gaussian distribution, the score function can be written as $\nabla_\psi \log \pi_{\phi,\psi}(\mathbf{a} \mid \mathbf{s}', \mathbf{z}_B) = \nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}', \mathbf{z}_B)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{a} - \boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}', \mathbf{z}_B))$, where

$\nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}', \mathbf{z}_B)$ is the Jacobian matrix of shape $|\mathcal{A}| \times |\Psi|$. Therefore, we obtain:

$$\frac{d\psi}{dt} = \eta \cdot \mathbb{E}_{\mathbf{s}' \sim d_B, \mathbf{a} \sim \pi_{\phi,\psi}} \left[\nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}', \mathbf{z}_B)^\top \boldsymbol{\Sigma}^{-1} (\mathbf{a} - \boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}', \mathbf{z}_B)) A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a}) \right]. \tag{18}$$

Now, consider a specific state $\mathbf{s}$. The rate of change of the policy mean $\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)$ corresponding to context $\mathbf{z}_A$, induced by the update of ψ under context $\mathbf{z}_B$, is given by $\frac{d\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)}{dt}$. Applying the chain rule, we have $\frac{d\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)}{dt} = \nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A) \frac{d\psi}{dt}$. To preserve theoretical generality and formally account for the state distribution, we consider the instantaneous parameter update induced by an arbitrary sampled state $\mathbf{s}' \sim d_B$, which evaluates to $\frac{d\psi}{dt} = \eta \cdot \nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}', \mathbf{z}_B)^\top \mathbb{E}_{\mathbf{a} \sim \pi_{\phi,\psi}} \left[\boldsymbol{\Sigma}^{-1} (\mathbf{a} - \boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}', \mathbf{z}_B)) A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a}) \right]$. Consequently, we can write:

$$\begin{aligned} \frac{d\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)}{dt} = \eta \, & \left[\nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A) \nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}', \mathbf{z}_B)^\top \right] \\ & \cdot \mathbb{E}_{\mathbf{a} \sim \pi_{\phi,\psi}} \left[\boldsymbol{\Sigma}^{-1} (\mathbf{a} - \boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}', \mathbf{z}_B)) A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a}) \right]. \end{aligned} \tag{19}$$

Let $G(\mathbf{s}', \mathbf{z}_B)$ be defined as:

$$G(\mathbf{s}', \mathbf{z}_B) = \mathbb{E}_{\mathbf{a} \sim \pi_{\phi,\psi}} \left[\boldsymbol{\Sigma}^{-1} (\mathbf{a} - \boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}', \mathbf{z}_B)) A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a}) \right].$$

Expanding this expectation and leveraging the property that the expected score of a probability distribution is zero, we obtain:

$$\begin{aligned} G(\mathbf{s}', \mathbf{z}_B) &= \boldsymbol{\Sigma}^{-1} \big\{ \mathbb{E}_{\mathbf{a} \sim \pi_{\phi,\psi}} [\mathbf{a} - \boldsymbol{\mu}_{\phi,\psi}(\mathbf{s}', \mathbf{z}_B)] \, \mathbb{E}_{\mathbf{a} \sim \pi_{\phi,\psi}} [A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a})] \\ &\quad + \operatorname{Cov}(\mathbf{a}, A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a})) \big\} \\ &= \boldsymbol{\Sigma}^{-1} \operatorname{Cov}(\mathbf{a}, A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a})). \end{aligned}$$

According to Stein's Lemma, we have $\operatorname{Cov}(\mathbf{a}, A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a})) = \boldsymbol{\Sigma} \, \mathbb{E}_{\mathbf{a} \sim \pi_{\phi,\psi}} [\nabla_{\mathbf{a}} A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a})]$. Hence, the term simplifies to:

$$G(\mathbf{s}', \mathbf{z}_B) = \mathbb{E}_{\mathbf{a} \sim \pi_{\phi,\psi}} \left[\nabla_{\mathbf{a}} A^{\pi_{\phi,\psi}}(\mathbf{s}', \mathbf{z}_B, \mathbf{a}) \right].$$

Under the hypernetwork architecture, which takes the state variable $\mathbf{s}$ as input and employs a multiplicative interaction with the context variable $\mathbf{z}$, we have $\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}) = \mathbf{W}_\psi(\mathbf{s})\mathbf{z}$, where the weight matrix $\mathbf{W}_\psi(\mathbf{s})$ has a shape of $|\mathcal{A}| \times |\mathcal{Z}|$. Therefore, the matrix product becomes $\nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A) \nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}', \mathbf{z}_B)^\top = (\nabla_\psi \mathbf{W}_\psi(\mathbf{s}) \mathbf{z}_A)(\nabla_\psi \mathbf{W}_\psi(\mathbf{s}') \mathbf{z}_B)^\top$, where the Jacobian $\nabla_\psi \mathbf{W}_\psi(\mathbf{s})$ is a third-order tensor of shape $|\mathcal{A}| \times |\mathcal{Z}| \times |\Psi|$. Let $\mathcal{J} = \nabla_\psi \mathbf{W}_\psi(\mathbf{s})$, whose elements are given by $\mathcal{J}_{ijk} = \frac{\partial \mathbf{W}_{ij}(\mathbf{s})}{\partial \psi_k}$, with $\mathbf{W}_{ij}(\mathbf{s})$ and ψ_k denoting the (i, j)-th element of $\mathbf{W}_\psi(\mathbf{s})$ and the k-th parameter in ψ, respectively. The operation $\nabla_\psi \mathbf{W}_\psi(\mathbf{s}) \mathbf{z}_A$ represents the contraction of the tensor along the context dimension. Let the result of this contraction be $\mathbf{M}_A \in \mathbb{R}^{|\mathcal{A}| \times |\Psi|}$, whose elements are defined as $(\mathbf{M}_A)_{ik} = \sum_{j=1}^{|\mathcal{Z}|} \frac{\partial \mathbf{W}_{ij}(\mathbf{s})}{\partial \psi_k} (\mathbf{z}_A)_j$. We can then define $\boldsymbol{\Theta} = \nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A) \nabla_\psi \boldsymbol{\mu}_\psi(\mathbf{s}', \mathbf{z}_B)^\top = \mathbf{M}_A \mathbf{M}_B{}^\top$. Its (i, l)-th element is:

$$\begin{aligned} \boldsymbol{\Theta}_{il} = \sum_{k=1}^{|\Psi|} (\mathbf{M}_A)_{ik} (\mathbf{M}_B)_{lk} &= \sum_{k=1}^{|\Psi|} \left(\sum_{j=1}^{|\mathcal{Z}|} \frac{\partial \mathbf{W}_{ij}(\mathbf{s})}{\partial \psi_k} (\mathbf{z}_A)_j \right) \left(\sum_{m=1}^{|\mathcal{Z}|} \frac{\partial \mathbf{W}_{lm}(\mathbf{s}')}{\partial \psi_k} (\mathbf{z}_B)_m \right) \\ &= \sum_{j=1}^{|\mathcal{Z}|} \sum_{m=1}^{|\mathcal{Z}|} (\mathbf{z}_A)_j (\mathbf{z}_B)_m \left(\sum_{k=1}^{|\Psi|} \frac{\partial \mathbf{W}_{ij}(\mathbf{s})}{\partial \psi_k} \frac{\partial \mathbf{W}_{lm}(\mathbf{s}')}{\partial \psi_k} \right). \end{aligned}$$

Notice that the inner summation term $\sum_{k=1}^{|\Psi|} \frac{\partial \mathbf{W}_{ij}(\mathbf{s})}{\partial \psi_k} \frac{\partial \mathbf{W}_{lm}(\mathbf{s}')}{\partial \psi_k} = \langle \nabla_\psi \mathbf{W}_{ij}(\mathbf{s}), \nabla_\psi \mathbf{W}_{lm}(\mathbf{s}') \rangle$ is exactly the inner product between the gradients of the (i, j)-th and (l, m)-th elements of the hypernetwork's output weight matrix with respect to the parameter vector ψ.

From Lemma 1 and Remark 1, we have the approximation $\langle \nabla_\psi \mathbf{W}_{ij}(\mathbf{s}), \nabla_\psi \mathbf{W}_{lm}(\mathbf{s}') \rangle \approx \delta_{il} \delta_{jm} \cdot \kappa(\mathbf{s}, \mathbf{s}')$, where the Kronecker delta δ_{il} (resp. δ_{jm}) equals 1 if and only if $i = l$ (resp. $j = m$),

and 0 otherwise. Here, $\kappa(\mathbf{s}, \mathbf{s}')$ is a scalar NTK value uniquely determined by the hypernetwork's architecture. Consequently, we can simplify:

$$\begin{aligned}\boldsymbol{\Theta}_{il} &\approx \sum_{j=1}^{|\mathcal{Z}|}\sum_{m=1}^{|\mathcal{Z}|}(\mathbf{z}_A)_j(\mathbf{z}_B)_m\big(\delta_{il}\delta_{jm}\cdot\kappa(\mathbf{s},\mathbf{s}')\big)\\ &= \sum_{j=1}^{|\mathcal{Z}|}(\mathbf{z}_A)_j(\mathbf{z}_B)_j\cdot\delta_{il}\cdot\kappa(\mathbf{s},\mathbf{s}')\\ &= \delta_{il}\cdot\kappa(\mathbf{s},\mathbf{s}')\cdot\big(\mathbf{z}_A^\top\mathbf{z}_B\big).\end{aligned}$$

This implies that $\boldsymbol{\Theta}$ is a diagonal matrix with $\kappa(\mathbf{s}, \mathbf{s}') \cdot (\mathbf{z}_A^\top \mathbf{z}_B)$ on its diagonal; that is, $\boldsymbol{\Theta} = \kappa(\mathbf{s}, \mathbf{s}') \cdot (\mathbf{z}_A^\top \mathbf{z}_B) \cdot \mathbf{I}_{|\mathcal{A}|}$. Substituting this result back into the expression for $\frac{d\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)}{dt}$, we finally obtain:

$$\frac{d\boldsymbol{\mu}_\psi(\mathbf{s}, \mathbf{z}_A)}{dt} \approx \eta \cdot \kappa(\mathbf{s}, \mathbf{s}') \cdot G(\mathbf{s}', \mathbf{z}_B) \cdot (\mathbf{z}_A^\top \mathbf{z}_B) \tag{20}$$

This concludes the proof. □

Remark 2 (Architectural Justification for the NTK Approximation)**.** *We emphasize that the structural requirements of the proposed hypernetwork naturally facilitate the NTK-based approximations invoked in the proof of Proposition 1. Specifically, the hypernetwork is tasked with generating the weights (and potentially bias terms) that operate on the context variable* $\mathbf{z}$*. Consequently, its output space is inherently high-dimensional. For instance, generating merely the primary weight matrices and biases acting on* $\mathbf{z}$ *typically necessitates an output dimension of at least* $2 \times |\mathcal{Z}| \times |\mathcal{A}|$*. To ensure sufficient representational capacity for mapping the state* $\mathbf{s}$ *to such a high-dimensional parameter space, practical implementations of this hypernetwork intrinsically dictate the use of relatively wide hidden layers. This architectural necessity naturally aligns the model with the wide-network regime, thereby substantially bridging the gap between practical finite-width instantiations and the infinite-width asymptotic properties. As a result, the near-orthogonality of parameter gradients across distinct output dimensions, as established in Lemma 1 and discussed in Remark 1, is preserved in our experimental setup, providing justification for the theoretical simplifications made in Proposition 1.*

Remark 3 (Extension to Maximum Entropy Reinforcement Learning)**.** *To ensure the generality and standard formulation of our proof, Proposition 1 is derived utilizing the canonical Policy Gradient Theorem. However, state-of-the-art continuous control architectures commonly operate within the Maximum Entropy (MaxEnt) reinforcement learning framework to foster robust exploration. We demonstrate below that our proof seamlessly and rigorously extends to this framework.*

The MaxEnt objective incorporates an entropy regularization term scaled by a temperature parameter $\alpha > 0$*:*

$$J_{\textit{MaxEnt}}(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\big(r(\mathbf{s}_t,\mathbf{a}_t) - \alpha\log\pi(\mathbf{a}_t \mid \mathbf{s}_t)\big)\right].$$

Under this framework, the soft action-value function $Q^\pi_{\textit{soft}}$ *and the soft state-value function* $V^\pi_{\textit{soft}}$ *are defined respectively as:*

$$\begin{aligned}Q^\pi_{\textit{soft}}(\mathbf{s}_t,\mathbf{a}_t) &= r(\mathbf{s}_t,\mathbf{a}_t)\\ &\quad + \mathbb{E}_{\tau\sim\pi}\left[\sum_{k=1}^{\infty}\gamma^k\Big(r(\mathbf{s}_{t+k},\mathbf{a}_{t+k}) - \alpha\log\pi(\mathbf{a}_{t+k} \mid \mathbf{s}_{t+k})\Big)\;\Big|\;\mathbf{s}_t,\mathbf{a}_t\right],\\ V^\pi_{\textit{soft}}(\mathbf{s}_t) &= \mathbb{E}_{\mathbf{a}\sim\pi(\cdot|\mathbf{s}_t)}\Big[Q^\pi_{\textit{soft}}(\mathbf{s}_t,\mathbf{a}) - \alpha\log\pi(\mathbf{a} \mid \mathbf{s}_t)\Big].\end{aligned}$$

Applying the log-derivative trick to $J_{\textit{MaxEnt}}(\pi)$*, the exact policy gradient using the score function estimator evaluates to:*

$$\nabla_\psi J_{\textit{MaxEnt}} = \mathbb{E}_{\mathbf{s},\mathbf{a}}\left[\nabla_\psi\log\pi_\psi(\mathbf{a} \mid \mathbf{s})\big(Q^\pi_{\textit{soft}}(\mathbf{s},\mathbf{a}) - \alpha\log\pi_\psi(\mathbf{a} \mid \mathbf{s})\big) - \alpha\nabla_\psi\log\pi_\psi(\mathbf{a} \mid \mathbf{s})\right].$$

Because the expected score vanishes (i.e., $\mathbb{E}_{\mathbf{a}}[\nabla_\psi \log \pi_\psi(\mathbf{a} \mid \mathbf{s})] = 0$), the rightmost term disappears. By subsequently introducing $V^\pi_{soft}(\mathbf{s})$ as a standard state-dependent baseline to reduce variance, we formulate the rigorously defined MaxEnt advantage function:

$$A^\pi_{MaxEnt}(\mathbf{s}, \mathbf{a}) \triangleq Q^\pi_{soft}(\mathbf{s}, \mathbf{a}) - \alpha \log \pi_\psi(\mathbf{a} \mid \mathbf{s}) - V^\pi_{soft}(\mathbf{s}).$$

By definition, its expectation over the policy distribution is identically zero: $\mathbb{E}_{\mathbf{a}\sim\pi}[A^\pi_{MaxEnt}(\mathbf{s}, \mathbf{a})] = 0$. This zero-expectation property ensures that the covariance expansion step $G(\mathbf{s}, \mathbf{z}_B) = \Sigma^{-1} Cov(\mathbf{a}, A^\pi_{MaxEnt}(\mathbf{s}, \mathbf{z}_B, \mathbf{a}))$, identical to our derivation in the above proof, holds exactly.

Crucially, when we apply Stein's Lemma to this expanded advantage function, we obtain:

$$\mathbb{E}_{\mathbf{a}\sim\pi}[\nabla_{\mathbf{a}} A^\pi_{MaxEnt}(\mathbf{s}, \mathbf{a})] = \mathbb{E}_{\mathbf{a}\sim\pi}[\nabla_{\mathbf{a}} Q^\pi_{soft}(\mathbf{s}, \mathbf{a})] - \alpha \mathbb{E}_{\mathbf{a}\sim\pi}[\nabla_{\mathbf{a}} \log \pi_\psi(\mathbf{a} \mid \mathbf{s})].$$

Recall that the action is sampled from a parameterized multivariate Gaussian distribution, $\mathbf{a} \sim \mathcal{N}(\mu_{\phi,\psi}(\mathbf{s}, \mathbf{z}_B), \Sigma)$. By the analytical properties of the Gaussian distribution, the gradient of its log-density with respect to the action is given by: $\nabla_{\mathbf{a}} \log \pi_\psi(\mathbf{a} \mid \mathbf{s}) = -\Sigma^{-1}(\mathbf{a} - \mu_{\phi,\psi}(\mathbf{s}, \mathbf{z}_B))$. Taking the expectation of this gradient over the policy distribution yields:

$$\mathbb{E}_{\mathbf{a}\sim\pi}[\nabla_{\mathbf{a}} \log \pi_\psi(\mathbf{a} \mid \mathbf{s})] = -\Sigma^{-1}(\mathbb{E}_{\mathbf{a}\sim\pi}[\mathbf{a}] - \mu_{\phi,\psi}(\mathbf{s}, \mathbf{z}_B)) = -\Sigma^{-1}(\mu_{\phi,\psi}(\mathbf{s}, \mathbf{z}_B) - \mu_{\phi,\psi}(\mathbf{s}, \mathbf{z}_B)) = 0.$$

Consequently, the expected gradient of the log-density precisely vanishes, meaning the entropy regularization term introduces strictly zero expected drift to the policy mean. The core continuous-time dynamic mathematically reduces to exactly the same form derived in the proof of Proposition 1: $G(\mathbf{s}, \mathbf{z}_B) = \mathbb{E}_{\mathbf{a}\sim\pi}[\nabla_{\mathbf{a}} Q^\pi_{soft}(\mathbf{s}, \mathbf{z}_B, \mathbf{a})]$. Therefore, the similarity-driven update rule $\frac{d\mu_\psi}{dt} \propto \mathbf{z}_A^\top \mathbf{z}_B$ is preserved under Maximum Entropy paradigms.

Remark 4 (Extension to Batch-Based Empirical Updates)**.** *While Proposition 1 is theoretically derived based on the interaction between two individual context variables $\mathbf{z}_A$ and $\mathbf{z}_B$, practical deep RL algorithms almost exclusively rely on mini-batch optimization. In this empirical setting, $\mathbf{z}_A$ and $\mathbf{z}_B$ can be naturally interpreted as the expected or average context vectors of two distinct training batches. Similarly, algorithms like Locally Constrained Policy Optimization (LCPO) by Hamadanian et al. (2025) calculate the empirical mean of the context within a recent batch (e.g., $\bar{w}_r = \mathbb{E}_{w\sim B_r}[w]$). The distance between this average recent context and historical samples is then used to identify out-of-distribution (OOD) experiences. By applying this batch-level contextual distance metric, the agent can locally constrain policy optimization on these OOD samples, anchoring prior knowledge while safely optimizing for the current batch. Therefore, the similarity metric $\mathbf{z}_A^\top \mathbf{z}_B$ that drives our policy updates directly translates to the practical batch-to-batch contextual similarities employed in practical algorithmic designs, reinforcing the empirical applicability of our theoretical framework.*

B Automatic Adjustment of the KL Coefficient

B.1 Target-KL Dual Update

The coefficient β in Eq. 11 controls the strength of the KL constraint between the final context-conditioned policy $\pi_{\phi,\psi}(\cdot \mid \mathbf{s}, \mathbf{z})$ and the base policy $\pi_\phi(\cdot \mid \mathbf{s})$. Instead of treating β as a manually tuned hyperparameter, we adapt it automatically using a target KL value D_{target}, following the dual-adjustment principle used for the entropy temperature in SAC (Haarnoja et al., 2018).

We first rewrite the KL-regularized optimization as the following constrained maximum-entropy problem:

$$\max_\pi J_{\text{MaxEnt}}(\pi) \quad \text{s.t.} \quad \mathbb{E}_{\mathbf{s},\mathbf{z}}\left[D_{\text{KL}}\big(\pi_{\phi,\psi}(\cdot \mid \mathbf{s}, \mathbf{z}) \parallel \pi_\phi(\cdot \mid \mathbf{s})\big)\right] \le D_{\text{target}}, \tag{21}$$

where $J_{\text{MaxEnt}}(\pi)$ denotes the standard maximum-entropy RL objective. Introducing the Lagrange multiplier $\beta \ge 0$, the corresponding Lagrangian is

$$\mathcal{L}(\pi, \beta) = J_{\text{MaxEnt}}(\pi) - \beta\left(\mathbb{E}_{\mathbf{s},\mathbf{z}}[D_{\text{KL}}] - D_{\text{target}}\right). \tag{22}$$

The associated saddle-point problem is

$$\min_{\beta\ge 0} \max_\pi \mathcal{L}(\pi, \beta). \tag{23}$$

For policy optimization, the term βD_{target} is constant with respect to the policy parameters and can be omitted, yielding the KL-regularized objective in Eq. 11. For multiplier optimization, the policy is fixed and β is updated by minimizing

$$J(\beta) = -\beta \left(\mathbb{E}_{\mathbf{s},\mathbf{z}}[D_{\text{KL}}] - D_{\text{target}} \right). \tag{24}$$

In practice, the expectation is estimated on a mini-batch $\mathcal{B}$:

$$\widehat{D}_{\text{KL}} = \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{s},\mathbf{z})\in\mathcal{B}} D_{\text{KL}}\big(\pi_{\phi,\psi}(\cdot \mid \mathbf{s},\mathbf{z}) \parallel \pi_{\phi}(\cdot \mid \mathbf{s})\big), \tag{25}$$

and the coefficient is updated using

$$J(\beta) = -\beta \left(\widehat{D}_{\text{KL}} - D_{\text{target}} \right), \qquad \beta \geq 0. \tag{26}$$

Equivalently, one may parameterize $\beta = \exp(\lambda)$ and optimize λ to enforce non-negativity. When $\widehat{D}_{\text{KL}} > D_{\text{target}}$, the gradient step increases β, strengthening the penalty and pulling the final policy closer to the base policy. When $\widehat{D}_{\text{KL}} < D_{\text{target}}$, the update decreases β, allowing more context-specific residual adaptation.

In general, the optimal value of D_{target} depends on the extent to which the environment dynamics are affected by the context. In our implementation, we usually set

$$D_{\text{target}} = 0.5|\mathcal{A}|. \tag{27}$$

This choice has a simple interpretation. If the final policy and the base policy have matched variances, i.e., $\Delta\sigma_{\psi,k} = 0$ for all action dimensions, the KL in Eq. 10 reduces to

$$D_{\text{KL}} = \frac{1}{2} \sum_{k=1}^{|\mathcal{A}|} \frac{\mu_{\psi,k}^2}{\sigma_{\phi,k}^2}. \tag{28}$$

Thus, $D_{\text{target}} = 0.5|\mathcal{A}|$ corresponds to allowing the residual mean to differ from the base mean by approximately one base-policy standard deviation per action dimension when the variances are matched.

B.2 Sensitivity to the Target KL

The target KL controls the stability–plasticity trade-off between the base policy and the residual modulator. A smaller target KL enforces stronger anchoring to the base policy, which can improve stability but may restrict context-specific adaptation. A larger target KL gives the residual modulator more flexibility, but may weaken the anchoring effect and allow the residual branch to dominate the shared base behavior. Therefore, it is important to verify that DAPHNI does not rely on a narrowly tuned value of D_{target}.

To evaluate this sensitivity, we set $D_{\text{target}} = c|\mathcal{A}|$ and vary the multiplier c from 0.3 to 0.7 on Half Cheetah under the four wind dynamics. Figure 7 reports the resulting returns, with points and error bars denoting the mean and standard deviation over random seeds. Across this range, DAPHNI maintains strong performance under all dynamic forms, indicating that the method is not highly sensitive to the exact target-KL value. The default choice $c = 0.5$ provides a stable middle point: it preserves sufficient anchoring to keep the base policy meaningful, while still allowing the residual modulator to adapt to context-dependent dynamics.

The sensitivity analysis also supports the role of the automatic dual update. Rather than fixing a static KL coefficient β, the target-KL formulation adjusts the effective regularization strength according to the empirical KL observed during training. This makes the residual constraint adaptive to different stages of learning and reduces the need for manual coefficient tuning. We therefore use $D_{\text{target}} = 0.5|\mathcal{A}|$ for all main experiments unless otherwise specified.

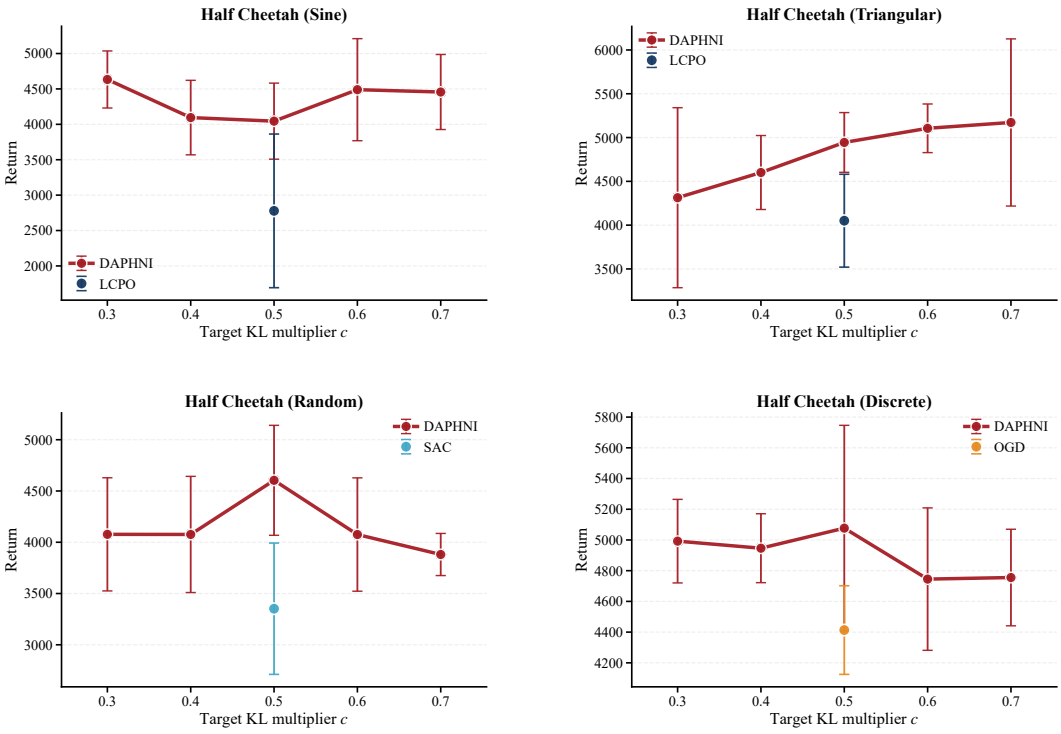


Figure 7: Sensitivity analysis of the target KL value on Half Cheetah under four wind dynamics. Points and error bars denote the mean and standard deviation over random seeds. The baseline result in each subplot corresponds to the best-performing baseline under the corresponding dynamic form.

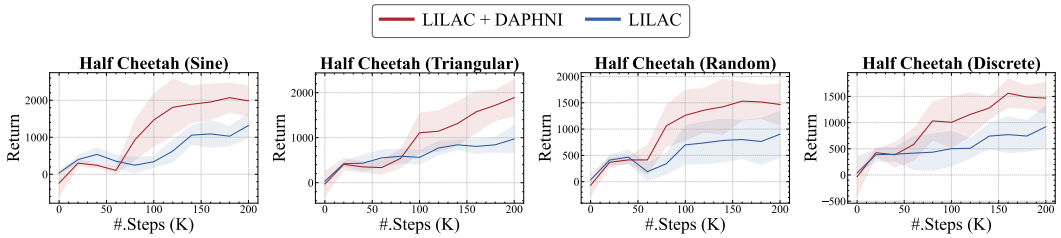


Figure 8: Learning curves under unobservable contexts on Half Cheetah. DAPHNI is combined with LILAC by replacing the original policy with the proposed decoupled policy architecture.

C Experiments under Unobservable Contexts

DAPHNI is a policy-side forgetting-mitigation mechanism and is therefore orthogonal to the choice of RL backbone or context-acquisition method. To test whether its benefit persists when the current context is unavailable, we combine DAPHNI with LILAC (Xie et al., 2021), a representative context-inference method for non-stationary RL that has been shown to infer informative latent contexts in Half Cheetah. We retain LILAC's latent-context inference module and replace its original policy with the proposed decoupled architecture, so that DAPHNI receives an inferred context rather than the ground-truth context.

Figure 8 compares LILAC and LILAC+DAPHNI on Half Cheetah under four wind dynamics. LILAC+DAPHNI exhibits a short initial adaptation phase, which is expected when the context inference module and the decoupled policy must be optimized jointly. After this phase, it achieves higher and more stable returns across the shown dynamics. The complete numerical results, including the multivariate dynamic form, are provided in Appendix D.4.

The returns of LILAC+DAPHNI are clearly below those of DAPHNI with directly observed contexts in the main experiments. This gap is expected: observed-context DAPHNI receives the current context, whereas LILAC+DAPHNI must infer a latent representation from recent trajectories, introducing estimation uncertainty and delay before the policy can adapt. The comparison therefore reflects the additional difficulty of context acquisition rather than a failure of DAPHNI's policy-side mechanism. Within the unobservable-context setting, LILAC+DAPHNI consistently improves over LILAC, indicating that the same decoupled and similarity-aware adaptation mechanism remains effective when supplied with informative inferred contexts.

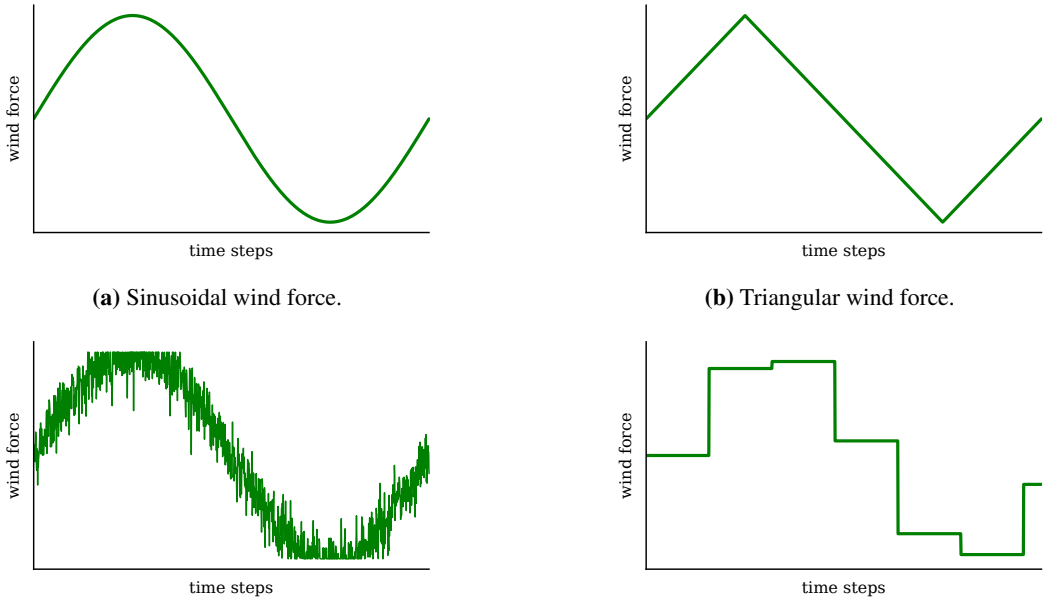
D Experimental Details

D.1 Environment Information

Benchmark Information. In Section 5, we evaluate all methods on five continuous-control benchmarks. Four are standard environments from the Gymnasium MuJoCo suite: Half Cheetah-v4, Hopper-v4, Walker2d-v4, and Ant-v4. These environments are widely used in deep reinforcement learning and non-stationary reinforcement learning because they cover a broad spectrum of locomotion morphologies, state–action dimensionalities, and contact dynamics. Half Cheetah is a planar locomotion task with relatively stable forward-running dynamics. Hopper and Walker2d involve underactuated bipedal locomotion with substantially higher sensitivity to balance and contact perturbations. Ant requires control of a higher-dimensional quadruped body with more complex multi-limb coordination. We also evaluate a non-stationary continuous-action extension of Gymnasium LunarLander-v3. Lunar Lander tests goal-directed landing under changing physical conditions and therefore provides a complementary setting to sustained locomotion. Its observation retains the original eight state coordinates and appends a two-dimensional context. All other physics and environment settings are retained.

Contextual Dynamic Information. Following the settings used in prior non-stationary MuJoCo studies (Xie et al., 2021; Feng et al., 2022; Hamadanian et al., 2025), we induce non-stationarity with a time-varying external wind force. The four scalar wind profiles are sinusoidal, triangular, stochastic-noise-perturbed, and piecewise discrete. The sinusoidal profile gives smooth periodic changes, the triangular profile gives smooth linear variation, the stochastic-noise-perturbed profile combines periodic variation with random fluctuations, and the piecewise discrete profile produces abrupt changes between persistent values. These profiles therefore cover smooth predictable changes, noisy fluctuations, and abrupt regime-level shifts, testing both continuous adaptation and robustness to heterogeneous non-stationarity. Figures 9a–9d visualize these profiles. Considering the different physical properties of the agents, we use benchmark-specific wind-force ranges. Half Cheetah and Ant have elongated or multi-legged morphologies with relatively stronger actuation and are commonly evaluated under positive external wind forces in prior non-stationary MuJoCo settings, so their ranges are $[0, 20]$. Hopper and Walker2d are balance-sensitive bipedal systems that can be destabilized by much smaller perturbations. The perturbation direction also matters because forward and backward forces can induce qualitatively different balance-recovery behaviors, so their ranges are $[-4, 4]$. These ranges make the induced non-stationarity challenging while avoiding magnitudes that would make some benchmarks trivially unstable or physically incomparable. Since the scalar wind is one-dimensional, we map it to the context vector $\mathbf{z}_t$ with a standard Fourier feature mapping. This bounded encoding gives nearby wind values a smooth representation and comparable context norms.

The multivariate MuJoCo form changes three physical factors simultaneously: wind force, whole-body mass and inertia, and sliding friction. The corresponding ranges are $([0, 20], [0.8, 1.2], [0.7, 1.3])$ for Half Cheetah, $([-4, 4], [0.9, 1.1], [0.8, 1.2])$ for Hopper, $([-4, 4], [0.9, 1.1], [0.85, 1.15])$ for Walker2d, and $([-2, 2], [0.85, 1.15], [0.8, 1.2])$ for Ant. The three normalized context coordinates follow independent sinusoidal schedules. The observation appends these three coordinates to the physical state, and the coordinates are provided as raw inputs without Fourier mapping, norm normalization, phase input, or time input. The physical parameters are applied directly to the simulator. Gravity, actuation, rewards, time steps, termination conditions, and v4 observation choices otherwise retain their original settings. Thus, this form changes acceleration, body inertia, and contact response at the same time, rather than encoding a single wind variable in a higher-dimensional representation.



(a) Sinusoidal wind force.

(b) Triangular wind force.

(c) Wind force perturbed by stochastic noise.

(d) Piecewise discrete wind force.

Figure 9: Four wind dynamic forms used to construct non-stationary environments for each MuJoCo benchmark in Section 5.1.

For Lunar Lander, the two raw context coordinates vary gravity and fuel cost independently. They follow sinusoidal schedules with different periods, and determine gravity $G = 10 + 1.5z_1$ in $[8.5, 11.5]$ and the fuel-cost multiplier $\kappa = 1 + 0.5z_2$ in $[0.5, 1.5]$. Built-in wind is disabled. Gravity is updated before each transition, so it changes the transition function. The multiplier changes the nonterminal fuel penalty in the reward while preserving the original terminal-reward overrides, so it also changes the reward function. The two-dimensional context is supplied directly without Fourier mapping, norm normalization, phase input, or time input. This construction creates coupled transition and reward non-stationarity along a two-dimensional trajectory. It tests adaptation beyond locomotion dynamics and does not reduce to a single ordered context direction.

The clock for every dynamic schedule continues across episode resets, and each seed receives fixed independently sampled phases. Methods that use observed contexts receive the current context with the observation, while future context, phase, and time are withheld. The four scalar wind forms therefore test smooth predictable, noisy, and abrupt changes, whereas the multivariate MuJoCo and Lunar Lander forms additionally test simultaneous changes in multiple physical or task factors.

D.2 Baseline Information

Locally Constrained Policy Optimization (LCPO). LCPO (Hamadanian et al., 2025) is an on-policy method for online RL in non-stationary context-driven environments. It assumes an observed exogenous context process that may vary continuously and mitigates catastrophic forgetting by anchoring policy outputs on historical samples whose contexts differ from the current training batch. Specifically, LCPO maintains a bounded buffer of past state-context samples, identifies out-of-distribution contexts relative to the current batch, and imposes a KL constraint on the policy outputs at those samples. These historical samples are used as policy anchors rather than as off-policy return-optimization data, thereby limiting policy drift on previously encountered context regions while optimizing behavior under the current context.

We include LCPO as a primary baseline because it directly targets the closest problem setting to ours: catastrophic forgetting in online RL under continuous context-driven non-stationarity. Unlike approaches that rely on explicit task labels, change-point boundaries, or segmented task identities, LCPO is designed for context processes that may change smoothly, abruptly, stochastically, or along arbitrary traces. It also represents a recent and directly related approach to forgetting mitigation in

continuous-context non-stationary RL. Comparing against LCPO therefore allows us to evaluate DAPHNI against a strong method with the same high-level objective but a different mechanism: constrained policy optimization with OOD anchoring versus policy-side base–residual decomposition with similarity-driven hypernetwork interactions in a more sample-efficient manner.

For the comparison experiments in Section 5.2, LCPO is evaluated in its intended observable-context setting. We provide the observable context as input and keep its original on-policy OOD anchoring mechanism. Since LCPO and DAPHNI use different optimization paradigms, we allocate LCPO ten times the environment-step budget used by DAPHNI to match wall-clock training time, as described in Section 5.2. Apart from this protocol-level adjustment, LCPO is evaluated without modifying its objective, context input, or anchoring mechanism.

Sliding Orthogonal Gradient Descent (OGD). Sliding OGD (Woo et al., 2022) was proposed as part of a structure-learning-based framework for RL in non-stationary environments. The original framework first uses a task decomposition embedding module to infer time-varying latent factors from transitions, and then applies a sliding-window variant of OGD to mitigate catastrophic forgetting. Its core forgetting-mitigation mechanism is gradient projection: when a new RL loss gradient is computed, it is projected away from a queue of recent gradients using a Gram–Schmidt-style update,

$$\lambda(\theta) \leftarrow \nabla_\theta L(\theta) - \sum_{\lambda' \in \Lambda} \mathrm{proj}(\nabla_\theta L(\theta), \lambda'), \tag{29}$$

where $\mathrm{proj}(x, \lambda') = \frac{x^\top \lambda'}{\lambda'^\top \lambda'} \lambda'$. The projected gradient is then used for the parameter update and added to the sliding buffer, reducing interference with recently used update directions.

We include Sliding OGD as a representative regularization- and gradient-projection-based baseline for catastrophic-forgetting mitigation. It preserves prior knowledge through the optimization dynamics by restricting parameter-update directions, whereas DAPHNI addresses forgetting through policy structure by separating context-invariant behavior from context-specific residual adaptation. This comparison is therefore useful for assessing whether explicit policy-side decomposition and similarity-driven residual updates provide benefits beyond generic gradient-interference control.

For the comparison experiments in Section 5.2, we adapt Sliding OGD to the observable-context setting used in our main experiments. The original latent task-inference module is designed for settings where the non-stationary factor is hidden or only indirectly observable through transitions. Because the context is directly available in our protocol, we omit this inference module. We concatenate the observable context with the state as policy and critic input, and apply the Sliding OGD projection to the actor gradient before the optimizer step. This preserves the core OGD mechanism under the observable-context protocol.

Model-Based RL Context Detection (MBCD). MBCD (Alegre et al., 2021) is a model-based non-stationary RL method that mitigates forgetting through online context detection and context-specific policy reuse. It assumes that the agent encounters a sequence of latent MDP contexts with unknown boundaries. MBCD maintains a mixture of probabilistic dynamics models, each associated with an identified context, and uses CUSUM-style likelihood-ratio statistics to detect change-points. When a change-point is detected, the method either reuses a previously learned context model and policy or initializes a new context-specific model and policy. Its forgetting-mitigation mechanism is therefore based on segmentation and isolation: knowledge is preserved by detecting discrete regimes and reusing policies when similar regimes recur.

We include MBCD as a representative change-point-based baseline. It is a strong method in this category because it combines online high-confidence change-point detection with learned probabilistic dynamics models, supports continuous-state and continuous-action domains, and does not require a pre-training phase or prior knowledge of the number or sequence of contexts. It was also used as a change-point-based baseline in prior work on continuous context-driven non-stationary RL (Hamadanian et al., 2025). Since our work focuses on continuous context shifts, MBCD provides an informative contrast between methods that mitigate forgetting through discrete regime detection and DAPHNI, which adapts directly over continuous context variables without explicit segmentation.

For the comparison experiments in Section 5.2, we retain the defining components of MBCD: probabilistic dynamics models, likelihood-based change-point detection, and context-specific policy/model

selection. In the original algorithm, learned dynamics models can also be used to generate simulated rollouts for Dyna-style policy optimization. Our main comparison, however, evaluates adaptation and forgetting mitigation under the same real-environment interaction budget. We therefore train policies from real environment samples and use the learned models for context detection and policy/model selection. This keeps MBCD as a change-point-based forgetting-mitigation baseline while aligning its policy-learning data source with the other methods in the main evaluation protocol.

PT-DQN. PT-DQN (Anand & Precup, 2023) is an architecture-based continual RL method motivated by complementary learning systems. It decomposes the action-value function into two independently parameterized components: a permanent value function and a transient value function,

$$Q^{(PT)}(\mathbf{s}, \mathbf{a}) = Q_\theta^{(P)}(\mathbf{s}, \mathbf{a}) + Q_w^{(T)}(\mathbf{s}, \mathbf{a}). \tag{30}$$

The permanent component is updated more conservatively to accumulate long-term knowledge, while the transient component adapts more rapidly to recent experience. This design mitigates forgetting by separating stability and plasticity in value estimation.

We include PT-DQN as a representative architecture-based baseline. Compared with many modular continual-learning methods that require task identities, task-specific heads, or known task sequences, PT-DQN does not require explicit task boundaries as input. This property makes it more compatible with online non-stationary settings than many task-indexed architectural methods. It also provides a relevant comparison to DAPHNI: PT-DQN decomposes the value function into permanent and transient components, whereas DAPHNI decomposes the policy into context-invariant base behavior and context-dependent residual adaptation.

For the comparison experiments in Section 5.2, we adapt PT-DQN to the continuous-action benchmark suite using branching action discretization, since PT-DQN is originally formulated for discrete action spaces. For each action dimension, the continuous action interval is uniformly divided into a fixed number of bins, and the policy selects one bin independently per dimension. The resulting multi-discrete action is mapped back to the corresponding continuous control vector before being applied to the environment. This preserves the original action bounds and avoids changing the environment dynamics. Branching discretization also avoids enumerating the full Cartesian product of all action bins, making PT-DQN executable across the continuous-action benchmarks while preserving its permanent–transient value decomposition.

D.3 Experimental Designs

Context Retention Change. Context Retention Change (CRC) is a forgetting-resistance metric that is specifically designed for continuously varying contexts, and it is defined only for the four scalar wind dynamics. These settings provide a single ordered wind coordinate with a well-defined high-context region and an opposite endpoint. The multivariate MuJoCo form and Lunar Lander jointly vary several factors and, in the latter case, both transitions and rewards, so a single directional revisit would confound retention with the choice of a multivariate trajectory. For each seed and scalar dynamic form, we first evaluate a previously encountered high-context region, continue training toward the opposite context extreme, and then reevaluate the same high-context region along the same continuously drifting context trajectory. At each evaluation point, 10 fixed initial states are used with deterministic actions and at most 1,000 interaction steps per episode. If R_{before} and R_{after} denote the episode-averaged returns before and after the intervening training, respectively, then

$$\text{CRC} = 100 \times \frac{R_{\text{after}} - R_{\text{before}}}{R_{\text{before}}}. \tag{31}$$

The percentage is computed for each valid pair, averaged within seed, and then aggregated over 10 seeds using the mean and sample standard deviation. Larger values indicate better relative retention. The first 30,000 training interactions are excluded from pair selection to skip the potentially unstable initial phase of agent training, but remain part of the 100,000-step training budget. Evaluations continue to use the drifting context rather than a fixed-context test.

Base policy extraction. This experiment tests whether the state-only base policy learned by DAPHNI captures shared behavior across contexts. We first train the complete DAPHNI policy under

the standard protocol, freeze the resulting parameters, and extract its state-only base branch without any additional optimization. We then evaluate the full DAPHNI policy, the extracted base policy with the residual contribution removed, and an untrained random policy under the same evaluation protocol, using the four scalar wind dynamics and the multivariate form. All methods use the same context schedules, evaluation initializations, and 10 random seeds. The extracted base policy performs substantially above the random policy across all dynamic forms, while the full DAPHNI policy is stronger overall. This separates evidence that the base branch learns useful shared behavior from the complementary role of the residual modulator in adapting that behavior to the current context.

Hypernetwork interaction. This probe tests whether the residual modulator exhibits the similarity-driven update bias predicted by Proposition 1 during actual training. It is defined only for the four scalar wind dynamics because the probe partitions the one-dimensional wind range $[0, 20]$ into ordered source intervals. The multivariate and Lunar Lander contexts have no unique scalar distance, and their coupled transition–reward changes would make the same influence matrix dependent on an arbitrary coordinate or trajectory choice. For each replay batch, contexts are grouped into four source intervals: $[0, 5)$, $[5, 10)$, $[10, 15)$, and $[15, 20]$. An update is retained for a source interval only when that interval contains at least 40% of the batch. We keep the batch states $\mathbf{s}$ fixed before and after the update, replace the context $\mathbf{z}_A$ by the probe values 2.5, 7.5, 12.5, and 17.5, and compute

$$\Delta_A = \bar{\boldsymbol{\mu}}_{\text{after}}(\mathbf{s}, \mathbf{z}_A) - \bar{\boldsymbol{\mu}}_{\text{before}}(\mathbf{s}, \mathbf{z}_A), \qquad I(B, A) = \|\Delta_A\|_2, \tag{32}$$

where B is the corresponding source interval and $\bar{\boldsymbol{\mu}}(\cdot)$ denotes the mean residual-policy output over the fixed batch states. Averaging $I(B, A)$ over retained updates gives a 4×4 source–probe influence matrix, which is row-wise min–max normalized for visualization. We compare DAPHNI with concat and inverted modulators over the same protocol and 10 random seeds. Concentration on the same and nearby probe contexts supports a similarity-driven update pattern. The probe is empirical evidence about the trained interaction structure, rather than a strict verification of all finite-width NTK assumptions or a comparison of absolute influence magnitudes across methods.

D.4 Complete Experimental Results

Tables 4, 5, 6, 7, 8, and 9 provide the complete numerical results for the comparison experiments on the MuJoCo suite, the Lunar Lander benchmark, the ablation studies, CRC, base policy extraction, and the unobservable-context experiments, respectively. Tables 4, 5, 6, 8, and 9 report final mean returns over seeds, with the corresponding standard deviations in parentheses. Table 7 reports CRC values and their seed-level standard deviations. The best mean return in each environment–dynamic setting is highlighted in the return tables, and the best CRC in each dynamic setting is highlighted in the CRC table. To indicate statistical reliability, a black dot (•) marks a result significantly worse than its reference under a pairwise one-sided Welch's two-sample t-test at the 5% significance level. The reference is DAPHNI in Tables 4, 5, and 6. It is the best-performing variant, or DAPHNI when it is best, in Table 7. It is the extracted base policy in Table 8 and LILAC+DAPHNI in Table 9. These tables complement the normalized plots in the main text by reporting raw values and seed-level variability for all reported configurations.

D.5 Hyperparameter Settings

We report the hyperparameters used in our experiments in Tables 10–13. For clarity, we group the hyperparameters into four categories: network architecture, training configuration, SAC backbone, and DAPHNI-specific design. Unless otherwise specified, the same hyperparameter settings are used across the four MuJoCo benchmarks, the Lunar Lander benchmark, and all dynamic forms.

The hyperparameters used in our experiments largely follow standard or commonly adopted default choices for SAC-style continuous-control agents. As summarized in Table 10, all methods use sufficiently expressive networks: both the policy network and the value network use hidden dimension 256 with two fully connected hidden layers. This network scale is standard and relatively large for the five benchmarks considered in Section 5, including Half Cheetah, Hopper, Walker2d, Ant, and Lunar Lander. Therefore, the performance gains of DAPHNI are unlikely to be explained by insufficient function-approximation capacity in the comparison methods. Instead, the results are more directly attributable to the proposed decoupled policy structure, the hypernetwork-based residual modulation, and the KL anchoring mechanism. In terms of computational cost, DAPHNI

Table 4: Complete results of the comparison experiments in Section 5.2. A black dot indicates that the corresponding baseline is significantly worse than DAPHNI under a pairwise one-sided Welch's two-sample t-test at the 5% significance level.

		DAPHNI	LCPO	SAC	OGD	MBCD	PT-DQN
Half Cheetah	Sine	**4044.7** (±537.1)	2777.7• (±1085.1)	1635.1• (±527.1)	2271.1• (±1127.8)	1138.4• (±322.1)	695.6• (±252.5)
	Triangular	**4944.1** (±341.0)	4050.7• (±530.2)	3723.4• (±391.5)	3987.9• (±1445.2)	1132.1• (±328.1)	1816.6• (±306.7)
	Random	**4604.1** (±536.2)	2870.6• (±977.4)	3351.8• (±640.4)	3240.4• (±1412.5)	1704.7• (±557.9)	755.0• (±351.7)
	Discrete	**5076.9** (±669.4)	3417.6• (±1159.5)	4285.8• (±790.2)	4413.4• (±288.5)	2188.9• (±1136.2)	1303.8• (±643.6)
	Multi	**3452.8** (±594.6)	2038.7• (±570.5)	2314.0• (±686.7)	2278.4• (±802.7)	982.9• (±381.2)	617.2• (±224.4)
Hopper	Sine	**2780.3** (±347.1)	2499.4• (±125.3)	2367.9• (±627.4)	1938.4• (±1089.3)	1402.1• (±403.7)	895.4• (±144.0)
	Triangular	**1870.7** (±1024.0)	1698.5 (±356.7)	1618.6 (±1028.1)	1586.7 (±398.4)	988.6• (±269.4)	888.1• (±220.8)
	Random	**2170.0** (±808.9)	1788.4 (±127.1)	1836.7 (±819.7)	1930.3 (±574.2)	1128.4• (±339.6)	946.5• (±228.5)
	Discrete	**2659.4** (±579.4)	2409.4 (±327.3)	2022.1 (±1016.2)	2154.4 (±1091.8)	1652.2• (±418.1)	883.8• (±226.6)
	Multi	**2341.5** (±578.4)	1945.2• (±427.0)	1876.7 (±697.7)	1671.4• (±788.3)	1013.8• (±351.2)	763.5• (±173.7)
Walker	Sine	**1826.1** (±722.4)	1092.9• (±661.3)	1491.4 (±501.6)	1190.5• (±450.9)	959.1• (±207.7)	669.6• (±116.0)
	Triangular	**1443.5** (±308.0)	1168.1 (±556.9)	999.3• (±265.0)	1142.9• (±386.7)	748.0• (±241.4)	683.2• (±236.4)
	Random	**1812.2** (±504.4)	1292.9• (±536.5)	1433.8• (±412.2)	1502.0 (±270.0)	839.7• (±350.5)	640.1• (±226.8)
	Discrete	**1215.6** (±509.2)	984.6 (±648.3)	990.3 (±366.6)	1027.2 (±185.9)	727.3• (±157.4)	578.7• (±171.5)
	Multi	**1583.7** (±527.5)	1033.0• (±347.1)	1056.0• (±426.3)	920.5• (±401.0)	637.6• (±225.7)	418.6• (±135.4)
Ant	Sine	**1212.3** (±529.4)	819.6• (±345.2)	724.2• (±354.7)	840.8• (±341.4)	337.7• (±150.1)	251.1• (±86.4)
	Triangular	**958.5** (±342.3)	723.4 (±291.3)	710.9 (±299.5)	756.8 (±338.4)	309.7• (±155.6)	351.9• (±134.1)
	Random	**817.7** (±201.5)	621.7• (±215.4)	523.2• (±153.7)	604.4• (±186.0)	214.8• (±103.6)	323.0• (±96.5)
	Discrete	**1087.8** (±208.4)	741.9• (±170.4)	836.4• (±201.4)	849.3• (±259.9)	576.2• (±199.1)	286.2• (±100.3)
	Multi	**1107.7** (±395.5)	824.6 (±370.2)	708.2• (±258.0)	756.0• (±324.8)	361.5• (±180.0)	286.1• (±132.6)

Table 5: Results of the Lunar Lander experiments in Section 5.2. A black dot indicates that the corresponding baseline is significantly worse than DAPHNI under a pairwise one-sided Welch's two-sample t-test at the 5% significance level.

	DAPHNI	LCPO	SAC	OGD	MBCD	PT-DQN
Lunar Lander	**249.5** (±38.9)	212.4 (±73.3)	216.9• (±39.2)	190.4• (±46.9)	99.3• (±88.7)	159.5• (±80.1)

introduces moderate additional overhead due to the residual modulator and the KL-coefficient update. In our implementation, its wall-clock training time is approximately 1.3–$1.4\times$ that of the standard

Table 6: Complete results of the ablation studies in Section 5.3. A black dot indicates that the corresponding ablation variant is significantly worse than DAPHNI under a pairwise one-sided Welch's two-sample t-test at the 5% significance level.

		DAPHNI	w/o modulator	w/o KL	concat modulator	w/o base policy	inverted modulator
Half Cheetah	Sine	**4044.7** (±537.1)	1635.1• (±527.1)	3294.5 (±1336.3)	3051.7• (±1284.6)	2276.5• (±1088.1)	3410.8• (±309.7)
	Triangular	**4944.1** (±341.0)	3723.4• (±391.5)	4323.3• (±444.3)	4090.7• (±745.9)	4220.9• (±377.9)	4157.4• (±911.5)
	Random	**4604.1** (±536.2)	3351.8• (±640.4)	3541.6• (±597.2)	3391.2• (±1386.3)	2202.3• (±833.4)	4332.9 (±529.8)
	Discrete	**5076.9** (±669.4)	4285.8• (±790.2)	4056.6• (±642.5)	4619.3• (±207.3)	3913.1• (±262.8)	4604.6 (±651.8)
	Multi	**3452.8** (±594.6)	2314.0• (±686.7)	2809.6• (±978.7)	2709.8• (±890.3)	2082.8• (±831.9)	2933.6 (±789.1)

Table 7: Complete results of the Context Retention Change (CRC) (%) experiments in Section 5.3. A black dot indicates that the corresponding ablation variant is significantly worse than the best-performing variant (or DAPHNI) under a pairwise one-sided Welch's two-sample t-test at the 5% significance level.

		DAPHNI	w/o modulator	w/o KL	concat modulator	w/o base policy	inverted modulator
Half Cheetah	Sine	16.6% (±4.7%)	10.8%• (±3.6%)	6.5%• (±3.0%)	12.3%• (±3.3%)	**18.7%** (±4.0%)	7.7%• (±3.9%)
	Triangular	22.7% (±5.2%)	13.5%• (±5.1%)	8.8%• (±3.7%)	18.6%• (±4.6%)	**27.1%** (±6.2%)	13.1%• (±5.4%)
	Random	19.4% (±5.4%)	9.1%• (±3.4%)	13.4%• (±4.1%)	16.6%• (±4.9%)	**22.5%** (±8.0%)	13.9%• (±4.4%)
	Discrete	**22.6%** (±4.5%)	9.6%• (±4.0%)	13.8%• (±5.8%)	7.6%• (±4.6%)	20.4% (±6.9%)	5.5%• (±3.4%)

comparison methods, such as SAC in Section 5.2 and LILAC in Appendix C. All experiments were conducted on a cloud server with four NVIDIA GeForce RTX 3090 GPU instances.

E Limitation and Future Work

E.1 Limitation

DAPHNI is currently instantiated with Gaussian policies for continuous action spaces. Applying the same decomposition to discrete or hybrid action spaces would require corresponding policy heads and action distributions, while leaving the base–residual policy structure unchanged.

E.2 Future Work

Model-based extensions for improved sample efficiency. DAPHNI is implemented with an off-policy SAC backbone, but its policy-side decomposition is not restricted to model-free RL. A promising direction is to combine DAPHNI with context-aware model-based RL, where learned dynamics models generate short context-local rollouts to augment real interactions. In such a framework, model-based rollouts could improve data efficiency, while DAPHNI would continue to mitigate forgetting through base–residual decomposition, similarity-driven adaptation, and KL anchoring. This direction is particularly relevant for online non-stationary environments where real interactions are costly.

Table 8: Complete results of the base policy extraction experiments in Section 5.4. A black dot indicates that the random policy is significantly worse than the extracted base policy under a pairwise one-sided Welch's two-sample t-test at the 5% significance level.

		DAPHNI	extracted base policy	random policy
Half Cheetah	Sine	4044.7 ($\pm$537.1)	2606.4 ($\pm$360.9)	75.8• ($\pm$35.5)
	Triangular	4944.1 ($\pm$341.0)	2491.2 ($\pm$319.3)	66.5• ($\pm$27.1)
	Random	4604.1 ($\pm$536.2)	2357.6 ($\pm$526.6)	59.9• ($\pm$29.5)
	Discrete	5076.9 ($\pm$669.4)	2830.4 ($\pm$319.5)	58.8• ($\pm$37.4)
	Multi	3452.8 ($\pm$594.6)	1311.5 ($\pm$246.5)	43.7• ($\pm$31.4)

Table 9: Complete results of the experiments under unobservable contexts in Appendix C. A black dot indicates that LILAC is significantly worse than LILAC + DAPHNI under a pairwise one-sided Welch's two-sample t-test at the 5% significance level.

		LILAC + DAPHNI	LILAC
Half Cheetah	Sine	**1982.0** ($\pm$416.1)	1318.3• ($\pm$339.4)
	Triangular	**1888.6** ($\pm$419.2)	972.4• ($\pm$319.7)
	Random	**1467.5** ($\pm$399.9)	902.7• ($\pm$447.5)
	Discrete	**1467.2** ($\pm$308.4)	920.9• ($\pm$416.6)
	Multi	**1705.5** ($\pm$472.1)	1135.6• ($\pm$378.2)

Separate optimization schedules for the base policy and residual modulator. Another direction is to optimize the base policy and residual modulator on different timescales. The base policy could be updated more conservatively to consolidate context-invariant behavior, while the residual modulator could adapt more rapidly to context-specific changes. Such asymmetric schedules would make the stability–plasticity trade-off more explicit and may further improve DAPHNI's robustness and adaptation speed under continuous context shifts.

Table 10: Network-related hyperparameters.

Hyperparameter	**Value / Description**
Policy network type	Gaussian policy with $\tanh$ squashing
Policy output	Mean and log standard deviation of the Gaussian action distribution
Policy hidden dimension	256
Number of policy hidden layers	2 fully connected hidden layers
Policy activation function	ReLU
Action squashing	$\tanh(\cdot)$
Action rescaling	Rescaled to the original environment action-space bounds
Log standard deviation range	$[-20, 2]$
Value network type	Twin Q-value networks
Value network output	Scalar Q-value
Value hidden dimension	256
Number of value hidden layers	2 fully connected hidden layers per Q-value network
Value activation function	ReLU
Target value network	Exponential moving average copy of the twin value networks
Weight initialization	Orthogonal initialization with gain 1.0
Bias initialization	Constant 0.0
Numerical stability constant	$\epsilon = 10^{-6}$

Table 11: Training-related hyperparameters.

Hyperparameter	**Value / Description**
Replay buffer capacity	10^6 transitions
Replay buffer storage	State, action, reward, next state, and terminal indicator
Replay sampling strategy	Uniform random sampling from stored transitions
Minimum replay size before policy updates	3% of the total environment steps
Action selection before minimum replay size	Uniform random action sampled from the environment action space
Action selection after minimum replay size	Stochastic action sampled from the learned policy
Batch size	512
Update frequency	One gradient update per environment step
Evaluation policy	Deterministic policy using the mean action
Environment reset condition	Reset when either termination or truncation occurs
Replay terminal flag	Environment termination indicator
Random seeds	Python, NumPy, PyTorch, CUDA, and environment action spaces are seeded

Table 12: SAC-backbone-related hyperparameters.

Hyperparameter	**Value / Description**
Discount factor γ	0.99
Target-network update coefficient τ	0.005
Entropy regularization	Automatic entropy tuning
Initial entropy coefficient	$\alpha = 1.0$, since $\log \alpha$ is initialized to 0
Target entropy	$-\dim(\mathcal{A})$
Policy learning rate	3×10^{-4}
Value learning rate	3×10^{-4}
Entropy coefficient learning rate	3×10^{-4}
Policy optimizer	Adam
Value optimizer	Adam
Entropy coefficient optimizer	Adam
Soft value target	$\min(Q_1, Q_2) - \alpha \log \pi(\mathbf{a} \mid \mathbf{s})$
Value loss	Sum of two mean-squared Bellman errors
Policy objective	$\mathbb{E}_{\mathbf{s},\mathbf{a}\sim\pi}[\alpha \log \pi(\mathbf{a} \mid \mathbf{s}) - \min(Q_1, Q_2)]$
Entropy coefficient objective	$-\log \alpha \, \mathbb{E}_{\mathbf{a}\sim\pi}[\log \pi(\mathbf{a} \mid \mathbf{s}) + \mathcal{H}_{\text{target}}]$
Target update frequency	Every gradient update
Target update rule	$\theta_{\text{target}} \leftarrow \tau\theta + (1-\tau)\theta_{\text{target}}$
Value-gradient clipping	Global norm clipped to 1.0
Action sampling	Reparameterized Gaussian sampling followed by $\tanh$ correction
Evaluation action	Deterministic mean action after $\tanh$ squashing and action rescaling

Table 13: DAPHNI-specific hyperparameters.

Hyperparameter	**Value / Description**
Base-policy input	State only
Residual-modulator input	State and context representation
Hypernetwork input	State only
Hypernetwork hidden dimension	512
Number of hypernetwork hidden layers	1 fully connected hidden layer
Hypernetwork activation function	ReLU
Residual output dimension	$2\dim(\mathcal{A})$
Hypernetwork final-layer initialization	Uniform initialization in $[-3\times10^{-3}, 3\times10^{-3}]$ for the final generated layer parameters
Target KL	$0.5\dim(\mathcal{A})$
Initial KL coefficient	$\exp(-0.7)$
KL coefficient learning rate	3×10^{-4}
KL coefficient optimizer	Adam
KL coefficient update	Automatic target-KL dual update
Base variance in KL penalty	Detached from the computation graph when computing the residual mean penalty