
Online Learning via Learned Latent Bayesian Tracking

Guy Gerson

School of ECE
Ben-Gurion University
Be'er-Sheva, Israel
guygers@post.bgu.ac.il

Tomer Raviv

School of ECE
Ben-Gurion University
Be'er-Sheva, Israel
tomerraviv95@gmail.com

Nir Shlezinger

School of ECE
Ben-Gurion University
Be'er-Sheva, Israel
nirshl@bgu.ac.il
o.simeone@northeastern.edu

Tirza Routtenberg

School of ECE
Ben-Gurion University
Be'er-Sheva, Israel
tirzar@bgu.ac.il

Osvaldo Simeone

Intelligent Networked Systems Institute (INSI)
Northeastern University London
London, U.K

Abstract

Online learning in non-stationary environments requires models to adapt rapidly from streaming data under strict computational constraints. A principled approach casts online learning as Bayesian state tracking, where model parameters are updated sequentially via Bayesian filtering. However, applying Bayesian filters directly to modern deep models is computationally prohibitive due to the high dimensionality of parameter space, forcing existing methods to rely on restrictive approximations or manually designed low-dimensional subspaces. In this work, we identify the absence of a suitable low-dimensional dynamical representation as the core bottleneck in Bayesian filtering-based online learning. Accordingly, we propose *Adaptive Update through Representation Adaptation (AURA)*, a meta-learning framework that learns offline a low-dimensional latent state-space model governing the evolution of optimal model parameters under distribution shift. Online adaptation is then performed via extended Kalman filtering in this learned latent space followed by the reconstruction of the full model parameters through a learned lifting map, enabling efficient single-step online adaptation while preserving model expressiveness. Evaluated on online adaptation of neural wireless receivers under time-varying channels and on non-stationary image classification, AURA shows substantial improvements in adaptation speed, accuracy, and computational efficiency over existing online learning and Bayesian filtering baselines, demonstrating that an adaptation-aware latent geometry is beneficial for effective Bayesian online learning in high-dimensional models.

1 Introduction

Modern machine learning systems are increasingly deployed in environments where the data-generating distribution evolves over time [1]. Such non-stationarity arises in a wide range of settings, including adaptive signal processing, wireless communications, and vision systems under distribution shift [2]. In these scenarios, a learning model must continuously adapt to remain aligned with the

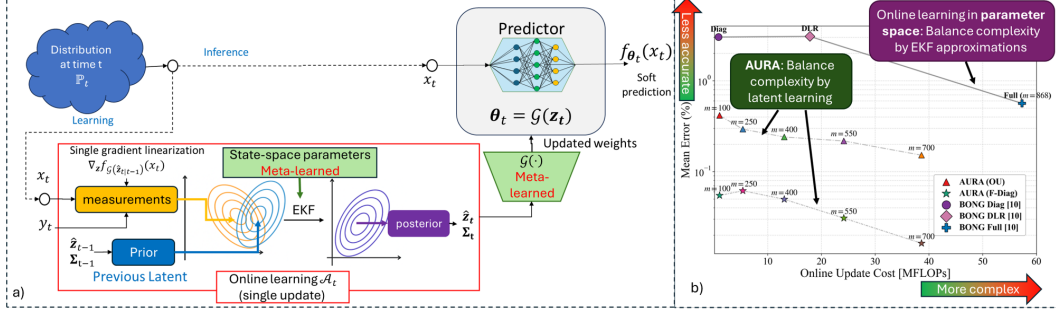


Figure 1: Proposed methodology: a) AURA pipeline via EKF in meta-learned latent state-space representation (marked in red fonts); b) Error-complexity trade-off for deep receiver adaptation, showing AURA’s enhanced tradeoff compared to altering the learning algorithm in parameter space.

current distribution, often under strict computational and latency constraints. These considerations pose the challenge of how to enable *rapid and reliable adaptation* from limited, streaming data [3].

A common approach to adaptation in non-stationary environments relies on online optimization, where model parameters are updated sequentially using stochastic gradient-based methods as new data arrives [4, 5]. While effective in many settings, such approaches lack a principled mechanism for uncertainty-aware updates and typically require multiple gradient steps and careful tuning to ensure stability. This renders them poorly suited to regimes where adaptation must occur within a small number of samples under stringent computational and latency constraints. An alternative paradigm, dating back over two decades [6], casts online learning as a form of Bayesian state estimation (tracking), where the model parameters are treated as a state vector evolving over time and updated recursively using Bayesian filters. This perspective offers an appealing route toward rapid online adaptation, as Bayesian filters, and particularly Kalman-type filters, naturally yield principled single-step prediction–correction updates that propagate uncertainty [7]. However, applying such methods directly to modern deep models is computationally prohibitive, since the complexity of Bayesian filtering scales at least quadratically (and often cubically) with the dimension of the state, i.e., the number of model parameters in our context. Consequently, existing Kalman-based online learning methods have resorted to two classes of approximations: 1) structural constraints on the covariance, such as diagonal [8] or low-rank parameterizations [9–11]; and 2) representational restrictions, such as confining adaptation to predefined parameter subsets or relying on manually specified dynamical models [12–14]. While these approximations reduce complexity, they substantially limit adaptation performance and often remain too costly for large-scale real-time learning.

Contributions. This work identifies the absence of a suitable *latent dynamical state-space representation* as the core bottleneck in Kalman-based online learning. We argue that the main challenge is not the Bayesian filter itself, but rather the lack of a low-dimensional dynamical representation in which filtering can be performed efficiently without sacrificing model expressiveness. Motivated by this observation, our contributions are as follows:

- **Latent Bayesian online adaptation.** We propose *Adaptive Update through Representation Adaptation (AURA)*, a framework for rapid online learning, which is performed in a learned low-dimensional latent space via single-step extended Kalman filter (EKF) [15], with the full model parameters recovered through a lifting map. This decouples the computational cost of filtering, which scales with the latent dimension, from the expressiveness of the underlying model. The EKF is selected for its simplicity and end-to-end differentiability [16, 17].
- **End-to-end meta-learning of the latent adaptation space.** Rather than manually setting the state-space representation governing the EKF operation [18], AURA learns the latent dynamics, and the lifting map from offline non-stationary trajectories, optimizing them end-to-end for online adaptation performance. Accordingly, AURA effectively meta-learns a geometry of parameter space tailored for rapid online adaptation. This yields the online learning paradigm illustrated in Fig. 1, where fast and reliable adaptation emerges from the interplay between structured Bayesian updates and a learned low-dimensional representation.

- **Empirical validation across two domains.** We demonstrate the effectiveness of AURA in settings involving non-stationary data. In particular, we consider online adaptation of neural wireless receivers under time-varying channels, where rapid adaptation is essential [19], as well as image classification with non-stationary distortion. Our results show that AURA consistently enables accurate adaptation using a single low-complexity update step per sample.

2 Related Work

Streaming and Online Learning. Learning from non-stationary data streams is an active area of research [1], where the primary objective is to rapidly adapt to evolving data distributions [5]. In such settings, data arrives sequentially and may exhibit concept drift [20], rendering previously learned models obsolete. A wide range of approaches have been proposed, including online gradient-based methods [21], adaptive optimizers [22], and proximal methods [23]. These approaches focus on asymptotic performance, aiming to track the evolving optimum up to bounded error rather than ensuring reliable performance at each time instance, which can lead to suboptimal behavior under rapidly changing distributions [24]. A related body of work considers adaptation settings in which the learner is provided with a small batch or task-specific dataset at each adaptation stage, rather than individual streaming samples. This includes incremental and continual learning methods, which update models sequentially across tasks or domains while mitigating catastrophic forgetting [25, 26], as well as test-time adaptation (TTA) and continual TTA approaches that adapt deployed models using unlabeled test batches under distribution shift [27–30]. Related paradigms further include gradual and temporal domain adaptation methods, which exploit temporally ordered shifts across adaptation domains [31, 32]. These approaches typically rely on batch-level statistics or iterative optimization over multiple samples, making them less suitable for streaming online with strict latency constraints.

Bayesian Filtering for Online Learning. A complementary line of work treats model parameters as a time-evolving state vector and performs adaptation via Bayesian filtering. Classical approaches based on Kalman filtering for training neural networks date back to early works on nonlinear Kalman extensions [6]. More recent works have extended this perspective to modern deep models, for instance, by reinterpreting stochastic gradient updates through Kalman-style uncertainty propagation [33], or by combining deep representations with state-space models for online classification [13]. In parallel, several works develop filtering-based approaches for *Bayesian neural networks*, including Kalman-based posterior tracking [11], subspace and last-layer Bayesian updates [12], and natural-gradient-based variants [10]. While these methods provide principled uncertainty-aware updates, their applicability to deep models is fundamentally limited by the computational complexity of Bayesian filtering, which scales at least quadratically with the number of parameters. As a result, proposed implementations rely on restrictive structural covariance approximations, such as diagonal [8] or diagonal plus low-rank (DLR) covariances [9], or confine adaptation to manually predefined subspaces [14]. In contrast, our approach explicitly learns a low-dimensional latent state-space and its transformation to parameter space tailored for online adaptation, enabling efficient Kalman-based updates while preserving the expressiveness of the underlying model.

3 System Model and Problem Formulation

Time-Varying Data Distribution. We consider a multi-class classification problem with an input space $\mathcal{X} \subseteq \mathbb{R}^D$ and a finite label space $\mathcal{Y} = \{1, \dots, C\}$, in a sequential setting with temporal distribution shifts. Let t denote a discrete time index, and assume that at each time step, an input-label pair $(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$ is drawn from a time-dependent distribution $\mathbb{P}_t(x, y)$. We consider a general form of non-stationarity in which the data-generating distribution may evolve continuously over time. This formulation captures a broad range of temporal variations, including gradual drifts as well as more abrupt changes in the statistics of the data-generating process.

Predictive Model. Let $f_\theta : \mathcal{X} \rightarrow \Delta^{C-1}$ denote a predictive model parameterized by $\theta \in \mathbb{R}^d$, where Δ^{C-1} is the probability simplex over the label space $\mathcal{Y}$. Given an input $x \in \mathcal{X}$, the model outputs predicted class probabilities represented by the probability mass function $f_\theta(x) \in \Delta^{C-1}$ over $\mathcal{Y}$. As the underlying distribution evolves over time, a fixed parameterization θ may become suboptimal, necessitating adaptation to maintain accurate predictions.

Problem Formulation. We consider a *streaming online learning* setting [1] where the predictive model must continuously adapt to a time-varying data-generating distribution. At each time step t , the learner observes a labeled sample $(x_t, y_t) \sim \mathbb{P}_t$ and updates the model parameters accordingly. Denoting by θ_t the model parameters after the update at time t , the adaptation is performed via an update rule

$$\theta_t = \mathcal{A}_t(x_t, y_t), \quad (1)$$

where $\mathcal{A}_t$ is an online update operator which can have memory, e.g., depend on the previous parameterization θ_{t-1} . The updated predictor f_{θ_t} is then used for inference under the current distribution $\mathbb{P}_t$. This formulation gives rise to two main challenges:

C1 Adaptation must be carried out sequentially with limited data.

C2 The update rule must be computationally efficient and capable of rapid adaptation.

To support the design of $\mathcal{A}_t$ under C1-C2, we assume access to an auxiliary labeled dataset comprising multiple non-stationary T -length trajectories. Specifically, this dataset, which reflects the non-stationary nature of the environment, is given by

$$\mathcal{D} = \left\{ \mathcal{D}^{(i)} \right\}_{i=1}^{|\mathcal{D}|}, \quad \mathcal{D}^{(i)} = \left\{ \left(x_{t,b}^{(i)}, y_{t,b}^{(i)} \right) \right\}_{t=1, b=1}^{T, B}, \quad (2)$$

where each trajectory $\mathcal{D}^{(i)}$ consists of T sequential time steps, and each time step t contains B labeled samples drawn from the corresponding time-varying distribution. These offline trajectories capture diverse distribution shifts and are used solely to design or learn the online adaptation mechanism prior to deployment. During deployment, the learner receives streaming samples sequentially and adapts online.

4 Proposed Method: AURA

We now present AURA, our proposed framework for rapid online learning in non-stationary environments. Our design is motivated by the observation that Bayesian filtering provides a principled mechanism for sequential uncertainty-aware adaptation from limited data (thus well suited to C1). However, the direct application of Bayesian filtering to modern predictive models is computationally prohibitive due to the high dimensionality of parameter space (thus failing to address C2). Accordingly, the key idea underlying AURA is to perform online adaptation via Bayesian filtering in a *learned low-dimensional latent space* rather than in the original parameter space. Specifically, we parameterize the evolving model parameters through a latent state-space representation, where online adaptation is conducted by tracking this latent state using a single-step EKF-based update, and the adapted full model parameters are recovered through a learned lifting map (Subsection 4.1). Since such a latent state-space representation is generally not available *a priori*, AURA jointly learns the latent parameterization and its governing dynamics from offline non-stationary training trajectories (Subsection 4.2). This results in a meta-learning procedure [34] in which the representation is optimized specifically to facilitate rapid and reliable online adaptation under the update rule induced by Bayesian filtering.

4.1 Latent Bayesian Online Adaptation

At the core of AURA lies the idea of performing online adaptation via Bayesian filtering in a learned low-dimensional latent space. Specifically, instead of trying to directly learn the desired high-dimensional model parameters for the current distribution $\mathbb{P}_t$, denoted by $\theta_t^* \in \mathbb{R}^d$, we represent them through a latent variable (“state”) $z_t \in \mathbb{R}^m$, where $m \ll d$, together with a lifting map $\mathcal{G} : \mathbb{R}^m \rightarrow \mathbb{R}^d$, such that the desired parameters at time t are given by

$$\theta_t^* = \mathcal{G}(z_t).$$

The latent state z_t is designed to capture the adaptation-relevant degrees of freedom of the predictor, enabling efficient online updates in a compact representation space.

Latent State-Space Representation. We model the evolution of the latent adaptation state via a stochastic linear dynamical system,

$$z_{t+1} = Fz_t + v_t, \quad (3)$$

where $F \in \mathbb{R}^{m \times m}$ is a diagonal state transition matrix, and v_t is process noise with covariance Q . This model provides a tractable surrogate for the temporal evolution of the adaptation state. In particular, the setting $F = \gamma I$ with $\gamma < 1$ is used in the common Ornstein–Uhlenbeck (OU)-type dynamics [35].

The learner observes the latent state only indirectly through streaming labeled samples $(x_t, y_t) \sim \mathbb{P}_t$. We model the observation process as

$$\text{OneHot}(y_t) = f_{\mathcal{G}(z_t)}(x_t) + \mathbf{w}_t, \quad (4)$$

where $\text{OneHot}(\cdot)$ denotes one-hot encoding and $\mathbf{w}_t$ captures the discrepancy between the predicted class probabilities of the desired model and the observed label. Motivated by the fact that, under cross-entropy training, the mismatch of the optimal probabilistic predictor is zero-mean and uncorrelated with the estimate, we model $\mathbf{w}_t$ as zero-mean observation noise with covariance $\mathbf{R}$.¹ is chosen as a regularized positive-definite covariance, which ensures numerical stability of the learning procedure despite the simplex structure of the observation representation.

EKF-Based Online Adaptation. Given the latent state-space representation (3)–(4), AURA performs online adaptation via extended Kalman filtering. The EKF is especially suited for the considered setting for several reasons: (i) It performs adaptation using a *single* local linearization and update per time step, relying on one gradient (Jacobian) computation; (ii) While the complexity of the EKF scales quadratically with the state dimension, here the state corresponds to the low-dimensional latent vector dimension (m) rather than the full parameter dimension (d), rendering the resulting updates feasible under C2; (iii) Compared with alternative Bayesian filters (e.g., unscented Kalman filters [6], cubature Kalman filters [36], or particle filters [37]), the EKF admits a particularly simple and efficient implementation, while retaining strong performance in smooth nonlinear tracking.

The EKF tracks the Gaussian approximation of the posterior distribution of the latent state via its mean $\hat{\mathbf{z}}_t$ and covariance Σ_t . At each time step, it first predicts the latent state according to [18]

$$\hat{\mathbf{z}}_{t|t-1} = \mathbf{F}\hat{\mathbf{z}}_{t-1}, \quad \Sigma_{t|t-1} = \mathbf{F}\Sigma_{t-1}\mathbf{F}^\top + \mathbf{Q}. \quad (5)$$

The predicted states are then corrected using the incoming labeled observation via

$$\hat{\mathbf{z}}_t = \hat{\mathbf{z}}_{t|t-1} + \mathbf{K}_t \left(\text{OneHot}(y_t) - f_{\mathcal{G}(\hat{\mathbf{z}}_{t|t-1})}(x_t) \right), \quad \Sigma_t = \Sigma_{t|t-1} - \mathbf{K}_t \mathbf{H}_t \Sigma_{t|t-1}, \quad (6)$$

where $\mathbf{H}_t := \partial f_{\mathcal{G}(z)}(x_t) / \partial \mathbf{z}^\top \big|_{\mathbf{z}=\hat{\mathbf{z}}_{t|t-1}}$ locally linearizes the observation model, and

$$\mathbf{K}_t := \Sigma_{t|t-1} \mathbf{H}_t^\top \left(\mathbf{H}_t \Sigma_{t|t-1} \mathbf{H}_t^\top + \mathbf{R} \right)^{-1}. \quad (7)$$

The adapted predictor parameters are finally recovered through the lifting map as $\theta_t = \mathcal{G}(\hat{\mathbf{z}}_t)$. We summarize the resulting online adaptation procedure in Algorithm 1. At each incoming sample, AURA performs a single tracking step in latent space, followed by reconstruction of the full model parameters.

Algorithm 1 AURA online learning at time t

Require: Prior latent state $\{\hat{\mathbf{z}}_{t-1}, \Sigma_{t-1}\}$, current data (x_t, y_t) ,
Require: Latent state-space parameters $\mathbf{F}, \mathbf{Q}, \mathbf{R}$; lifting map $\mathcal{G}(\cdot)$

- 1: Predict $\hat{\mathbf{z}}_{t|t-1}$ and $\Sigma_{t|t-1}$ via (5)
- 2: Compute the Jacobian: $\mathbf{H}_t = \partial f_{\mathcal{G}(z)}(x_t) / \partial \mathbf{z}^\top \big|_{\mathbf{z}=\hat{\mathbf{z}}_{t|t-1}}$
- 3: Compute Kalman gain $\mathbf{K}_t$ via (7)
- 4: Update $\hat{\mathbf{z}}_t$ and Σ_t via (6)
- 5: **return** updated parameters $\theta_t = \mathcal{G}(\hat{\mathbf{z}}_t)$

4.2 Offline Meta-Learning of the Latent Adaptation Space

The effectiveness of Algorithm 1 critically depends on the choice of the latent state-space representation, namely, the lifting map $\mathcal{G}(\cdot)$ and the latent dynamics parameters governing (3). In particular, while the latent Bayesian formulation enables efficient online adaptation, its performance hinges on identifying a representation in which the desired model evolution is well captured by the assumed low-dimensional state-space representation. To this end, AURA learns the latent adaptation space offline from non-stationary training trajectories in (2). Specifically, we treat the components of the latent state-space model as *meta-learnable hyperparameters* of the online adaptation process, and optimize them such that the EKF-based online learner performs effectively under distribution shift.

¹Formal justification of this modeling assumption is provided in Appendix B.

Parameterized Latent State-Space Model. We parameterize the lifting map $\mathcal{G} : \mathbb{R}^m \rightarrow \mathbb{R}^d$ as a neural network with learnable parameters $\psi_G \in \mathbb{R}^n$. In addition, we treat the diagonal state transition matrix $\mathbf{F}$ as a learnable parameter, and optionally also learn the noise covariance matrices $\mathbf{Q}, \mathbf{R}$. Collectively, we denote the meta-learned latent state-space parameters by $\phi := \{\psi_G, \mathbf{F}, \mathbf{Q}, \mathbf{R}\}$. Under this parameterization, the online adaptation rule induced by Algorithm 1 is written as $\theta_t = \mathcal{A}_t(x_t, y_t; \phi)$, highlighting its dependence on the learned latent state-space model.

Meta-Learning Objective. We employ trajectory-based meta-learning in which each non-stationary sequence in (2) is treated as an adaptation task. The goal is to meta-learn ϕ such that the induced Algorithm 1 achieves low cumulative loss along each trajectory, by treating the first sample at each time step as the support set and the remaining as the query set [34]. Accordingly, we optimize the meta-objective

$$\mathcal{L}_{\mathcal{D}}^{\text{meta}}(\phi) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \frac{1}{T} \sum_{t=1}^T \frac{1}{B-1} \sum_{b=2}^B \ell\left(f_{\mathcal{A}_t(x_{t,1}^{(i)}, y_{t,1}^{(i)}; \phi)}(x_{t,b}^{(i)}, y_{t,b}^{(i)})\right), \quad (8)$$

which minimizes the empirical online prediction loss incurred by the latent Bayesian adaptation.

End-to-End Optimization. Since both the EKF recursion and the lifting map $\mathcal{G}_{\psi_G}(\cdot)$ are differentiable [17], the meta-objective in (8) can be optimized end-to-end using stochastic gradient-based methods by differentiating through the online adaptation trajectory. This yields a training procedure in which the latent representation and its governing dynamics are learned jointly so as to maximize the effectiveness of the resulting online adaptation rule. The meta-learning process, summarized as Algorithm 2, can be interpreted as learning a latent geometry of parameter space specifically tailored for rapid adaptation.

Algorithm 2 Offline meta-learning of AURA

Require: Offline dataset $\mathcal{D} = \{\mathcal{D}^{(i)}\}$, learning rate η , initial state-space parameters $\phi, \hat{z}_0, \Sigma_0$

- 1: **repeat**
- 2: Sample mini-batch of trajectories $\{\mathcal{D}^{(i)}\}$
- 3: **for** each trajectory $\mathcal{D}^{(i)} = \{(x_{t,b}^{(i)}, y_{t,b}^{(i)})\}_{t,b=1,1}^{T,B}$ **do**
- 4: **for** $t = 1, \dots, T$ **do**
- 5: Run Algorithm 1 on $(x_{t,1}^{(i)}, y_{t,1}^{(i)})$ to obtain $\theta_t^{(i)} = \mathcal{A}_t(x_{t,1}^{(i)}, y_{t,1}^{(i)}; \phi)$
- 6: **end for**
- 7: **end for**
- 8: Update latent state-space parameters: $\phi \leftarrow \phi - \eta \nabla_{\phi} \mathcal{L}_{\{\mathcal{D}^{(i)}\}_{i=1}^B}^{\text{meta}}(\phi)$
- 9: **until** convergence
- 10: **return** Learned latent state-space parameters ϕ

4.3 Discussion and Practical Considerations

Complexity Analysis. We next characterize the per-step computational complexity of Algorithm 1. The dominant computational steps are as follows: (i) *Jacobian computation*: evaluating the Jacobian $\mathbf{H}_t$ requires differentiating through the composition $\mathbf{z} \mapsto \mathcal{G}(\mathbf{z}) \mapsto f_{\mathcal{G}(\mathbf{z})}(x_t)$, incurring complexity $\mathcal{O}(n)$ (as $\mathcal{G}$ is implemented via an n -parameter neural network); (ii) *Prediction step*: propagating the latent covariance in (5) requires $\mathcal{O}(m^2)$ operations (as $\mathbf{F}$ is constrained to be diagonal); (iii) *Kalman update*: forming the Kalman gain and updating the posterior moments requires $\mathcal{O}(m^2)$ operations, since the output dimension C is typically small and independent of the model size; (iv) *Parameter reconstruction*: evaluating $\theta_t = \mathcal{G}(\hat{z}_t)$ requires a forward pass through the lifting map, with complexity $\mathcal{O}(n)$. In general, the complexity per-step of AURA is of the order of $\mathcal{O}(n + m^2)$. Since $m \ll d$ by design, this is substantially more efficient than applying Bayesian filtering directly in the ambient parameter space, whose complexity scales at least quadratically in the number of model parameters d .

Design Trade-Offs. AURA exposes several design parameters that govern the trade-off between performance and complexity: (i) The latent dimension m controls the expressiveness of the adaptation space: larger latent spaces can capture richer parameter variations but incur higher complexity; (ii) Constraining the structure of the state transition $\mathbf{F}$ (e.g., to be a scaled identity matrix) provides a mechanism for reducing complexity; (iii) The architecture of the lifting map $\mathcal{G}(\cdot)$ controls the expressive power of

the parameter reconstruction, with larger parameterizations increasing both representational flexibility and computational cost. These design choices allow AURA to be tailored to a broad range of constraints.

Interpretation and Theoretical Motivation. Although AURA employs Bayesian filtering as its online adaptation mechanism, it is not intended as a Bayesian neural network method and does not seek to maintain a posterior distribution over the full predictive model, as is common in Kalman-based Bayesian learning [10, 14, 11]. Rather, the Bayesian posterior maintained by AURA pertains solely to the low-dimensional latent state, which is treated as a stochastic tracking variable used to guide online updates. The predictive model parameters themselves are deterministically instantiated via the lifting map $\theta_t = \mathcal{G}(\hat{z}_t)$. AURA thus leverages Bayesian filtering as a structured mechanism for efficient online optimization, rather than for uncertainty quantification over network weights [38].

The use of latent Bayesian tracking in AURA is motivated by the hypothesis that the evolution of optimal model parameters often admits a structured low-dimensional representation. Such a hypothesis is consistent with empirical observations that overparameterized neural networks often admit substantial redundancy and can be effectively adapted within low-dimensional subspaces or low intrinsic-dimensional parameterizations [39, 40]. It is further aligned with classical dimensionality reduction results, such as the Johnson–Lindenstrauss lemma [41], which imply that structured high-dimensional point sets can often be embedded into substantially lower-dimensional spaces while approximately preserving their geometry. Under suitable regularity assumptions, one can further show that gradual changes in the data-generating distribution induce correspondingly smooth variations in compressed representations of the optimal model parameters, providing support for the latent dynamical systems viewpoint adopted by AURA. Formal statements and proofs of these observations are provided in Appendix B.

5 Experimental Study

We evaluate AURA on two complementary settings, assessing its generality and practical relevance: (i) a non-stationary image classification task under gradually evolving corruptions [42, 28]; and (ii) online adaptation of neural wireless receivers, a motivating real-world application where rapid millisecond-level adaptation is essential for practical deployment under time-varying channels [43–46]. Additional ablations, robustness studies, and proof-of-concept hardware experiments are provided in the appendix. The source code is available at <https://github.com/aura-online-adaptation>.

5.1 General-Purpose Online Adaptation for Non-stationary Image Classification

We first evaluate AURA on non-stationary image classification under evolving distribution shifts [42, 28]. We compare AURA not only with streaming online learning methods that update from labeled samples as they arrive [13, 47], but also with batch-wise adaptation methods commonly studied in the TTA literature [48–52]. Such methods accumulate multiple test samples for adaptation, and often rely on batch statistics or batch-level optimization objectives rather than single-sample streaming updates.

Experimental Setup. We consider pretrained image classifiers deployed under temporally evolving corruption streams. We generate non-stationary input streams by gradually varying the mixture of multiple corruption types over time [42, 28], inducing smooth distribution drift across consecutive frames. We evaluate on MNIST-C [53], CIFAR-10-C, and CIFAR-100-C [54], which are corrupted variants of MNIST [55] and CIFAR-10/100 [56], using standard pretrained backbone architectures for each benchmark based on ResNet-18 [57], wide ResNet-28-10 [58], and ResNet-29 [59], respectively.

We compare against three classes of baselines. First, we consider the original model with no adaptation. Second, among *streaming supervised online learning* methods, we compare (i) AURA; (ii) supervised online gradient descent (GD) [47]; and (iii) parameter-space Bayesian tracking of the final classification layer (EKF-FC) [13]. EKF-FC is applied with a full covariance on MNIST-C and CIFAR-10-C, where the corresponding $d \times d$ matrix is small enough to enable parameter-space learning; on CIFAR-100-C the classifier layer has $d \approx 10^5$ parameters and the full covariance becomes infeasible, so we set the EKF-FC there with a diagonal covariance approximation. Applying Bayesian online learning to the *entire* image-classifier parameter space, even under aggressive covariance approximations (e.g., diagonal [8] or DLR [9]), is computationally prohibitive in this setting and is therefore omitted. Third, we compare with four representative *batch-wise unsupervised adaptation* methods: three normalization-based TTA approaches—BN-Test, BN-EMA [48, 49], and BN-Alpha [50, 51]—and

Table 1: Mean classification error (% , 3 seeds per cell) under gradual non-stationary corruptions. Methods are grouped by adaptation regime. Best per dataset in **bold**; second best underlined

Regime	Method	# Train Samples	MNIST-C	CIFAR-10-C	CIFAR-100-C
Pre-training	Source	0	47.93 $\pm$ 19.83	41.81 $\pm$ 5.62	37.41 $\pm$ 4.82
		5	7.50 $\pm$ 0.50	29.46 $\pm$ 5.55	37.93 $\pm$ 1.83
		10	5.84 $\pm$ 0.24	25.89 $\pm$ 0.75	34.00 $\pm$ 2.37
		15	4.53 $\pm$ 0.39	22.37 $\pm$ 1.38	33.22 $\pm$ 3.35
Streaming	EKF-FC [13]	5	19.74 $\pm$ 5.09	30.62 $\pm$ 1.34	31.27 $\pm$ 1.98
		10	17.00 $\pm$ 4.96	28.69 $\pm$ 4.61	33.38 $\pm$ 2.65
		15	12.16 $\pm$ 3.06	26.99 $\pm$ 5.26	32.43 $\pm$ 1.55
	AURA [†]	5	2.76 $\pm$ 0.63	16.13 $\pm$ 1.50	29.47 $\pm$ 0.62
		10	2.67 $\pm$ 0.93	16.84 $\pm$ 0.51	29.65 $\pm$ 1.97
		15	1.63 $\pm$ 0.09	14.28 $\pm$ 2.87	26.66 $\pm$ 3.89
Batch-wise	BN-Test [48]	100	3.97 $\pm$ 2.58	19.46 $\pm$ 3.34	31.32 $\pm$ 2.08
	BN-Alpha [50]	100	38.24 $\pm$ 14.23	28.05 $\pm$ 2.48	33.06 $\pm$ 1.15
	BN-EMA [49]	100	3.17 $\pm$ 0.99	18.57 $\pm$ 2.64	33.19 $\pm$ 2.15
	ROID [52]	100	<u>2.64 $\pm$ 1.37</u>	<u>16.00 $\pm$ 1.29</u>	26.50 $\pm$ 1.37

[†] Latent dimension $m = 256, 500, 500$ for MNIST-C, CIFAR-10-C, CIFAR-100-C.

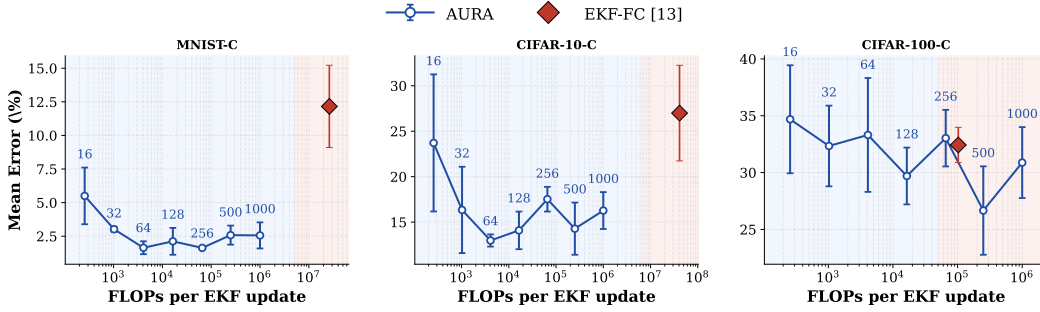


Figure 2: Complexity–performance trade-off in streaming online learning. The blue curve sweeps AURA’s latent dimension; the number above each marker is that m . The red diamond is EKF-FC. Shaded regions separate the latent and parameter-space regimes. Error bars are ± 1 standard deviation.

ROID [52], a self-supervised continual TTA method that adapts from unlabeled inputs via diversity-weighted soft-likelihood-ratio and consistency losses. Each test stream consists of 1000 time steps, each delivering a batch of 100 corrupted inputs. Within a step, up to 15 inputs come with ground-truth labels and are used by the *streaming supervised* methods (Online GD, EKF-FC, AURA) for adaptation.

Results. Table 1 reports classification error and the per-method learning-sample budget. AURA performs competitively across all benchmarks and achieves the best performance among streaming-feasible adaptation methods. In particular, AURA attains the lowest overall error on both MNIST-C and CIFAR-10-C, outperforming every batch-wise baseline despite using as few as five adaptation samples per frame. On CIFAR-100-C it ranks second overall and substantially improves over the streaming baselines, Online GD [47] and EKF-FC [13]. ROID [52] attains the lowest error on CIFAR-100-C, but it is structurally incompatible with the streaming setting. Its loss combines a soft-likelihood-ratio term with diversity- and certainty-based weights and a class-prior EMA, all computed from the *batch statistics* of the current frame and therefore requiring multiple unlabeled samples to be well-defined. Streaming ROID is consequently feasible only through external buffering, which induces a per-batch latency penalty proportional to the buffer size. In contrast, AURA adapts through a single structured latent Bayesian update on the current sample and emits a prediction before the next arrival.

Complexity–Performance Trade-off. To assess the computational benefits of latent-space Bayesian adaptation, Fig. 2 compares the streaming online learning complexity of AURA and EKF-FC in

floating-point operations (FLOPs) per update. This experiment serves both as a comparison against parameter-space adaptation and as an ablation of the complexity–performance trade-off controlled by m . The results show that AURA consistently dominates parameter-space adaptation in the complexity–accuracy plane: for substantially lower computational cost, latent-space adaptation yields markedly better prediction accuracy than parameter-space filtering on the classifier layer. Moreover, varying m reveals the expected trade-off between expressiveness and efficiency: increasing the latent dimension initially improves performance by enriching the adaptation space, after which gains saturate and may slightly deteriorate due to over-parameterization or increased estimation noise. The selected operating points used in Table 1 are highlighted in Fig. 2, and are seen to lie near the empirical Pareto frontier for each dataset. These results indicate that learning a compact latent adaptation geometry makes Bayesian online learning substantially more computationally efficient without sacrificing adaptation quality.

5.2 Rapid Online Adaptation for Wireless Receivers

Next, we examine wireless communications, a practically important benchmark for evaluating rapid online adaptation [19]. In particular, time-varying wireless channels constitute a prominent application where the inability to adapt neural receivers at millisecond timescales remains a major obstacle to practical deployment of neural physical layer communication systems [45].

Experimental Setup. We consider uplink multi-user detection with three single-antenna users and five receive antennas, using the DeepSIC neural receiver of [60] as the underlying predictive model. Channel realizations are obtained from QuaDRiGa [61] under the 3GPP Indoor Office scenario. We evaluate both linear and nonlinear channel models, where the latter incorporates hardware-induced distortions via a $\tanh$ nonlinearity (modeling nonlinearities at the RF front-end [62]). We compare AURA against representative streaming learning with online GD and EKF variants in full parameter space which we term Bayesian online natural gradient (BONG) following [10], with full, diagonal, and DLR covariance parameterizations [9]. Offline meta-training and online evaluation are conducted on disjoint channel trajectories. During online adaptation, the receiver is evaluated over temporally evolving channel sequences of 150 frames. The first four frames serve for synchronization, during which 64 labeled pilot samples per frame are provided for initialization. Subsequently, the receiver enters a tracking regime in which only six labeled samples (pilots) are available per frame for adaptation, after which the adapted detector is used for inference on the remaining symbols, and evaluated in its bit error rate (BER). Appendix C contains full details of the training setup.

Results. The results in Figs. 3–4 demonstrate that AURA consistently outperforms all considered baselines across both linear and nonlinear QuaDRiGa scenarios. In particular, AURA achieves the lowest BER across the signal-to-noise ratio (SNR) sweep while maintaining substantially more stable temporal tracking throughout the channel evolution. These gains are especially pronounced relative to full-parameter Bayesian filtering methods, highlighting that the benefit of AURA stems not merely from employing Bayesian online updates, but from performing them in a learned low-dimensional latent adaptation space tailored for efficient tracking. This supports our central hypothesis that identifying an adaptation-aware latent geometry is critical for enabling effective Bayesian online learning in high-dimensional models. Additional ablation studies analyzing the impact of latent dimension, latent dynamics parameterization, and robustness across channel models are provided in Appendix D. Furthermore, Appendix E presents a proof-of-concept hardware demonstration showing that AURA enables millisecond-order self-adaptive neural WiFi receivers on software-defined radio platforms, validating the practical deployability of the proposed framework in real-time embedded systems.

6 Conclusion

We introduced AURA, a framework for rapid online adaptation in non-stationary environments. By learning a low-dimensional state-space representation tailored for online adaptation, AURA enables an efficient single-step Bayesian update. This yields a principled online learning mechanism that combines Kalman-based filtering with flexible learned representations. Experimental results demonstrate that AURA achieves strong adaptation performance across diverse domains while operating under stringent computational constraints.

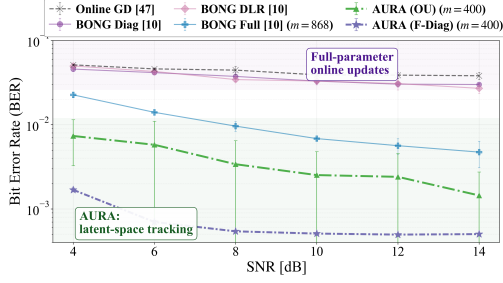


Figure 3: BER versus SNR, linear channel

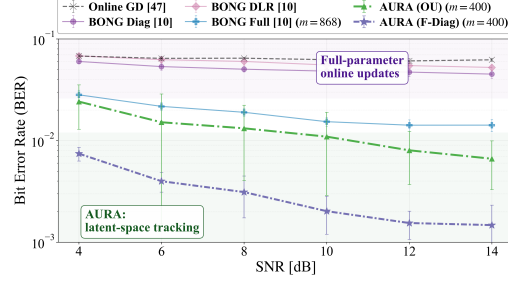


Figure 4: BER versus SNR, nonlinear channel

References

- [1] A. Cossu, F. Giannini, G. Ziffer, A. Bernardo, A. Gepperth, E. Della Valle, B. Hammer, and D. Bacciu, “A practical guide to streaming continual learning,” *Neurocomputing*, p. 132951, 2026.
- [2] A. Galashov, M. Titsias, A. György, C. Lyle, R. Pascanu, Y. W. Teh, and M. Sahani, “Non-stationary learning of neural networks with automatic soft parameter reset,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 83 197–83 234, 2024.
- [3] F. Giannini, G. Ziffer, A. Cossu, and V. Lomonaco, “Streaming continual learning for unified adaptive intelligence in dynamic environments,” *IEEE Intelligent Systems*, vol. 39, no. 6, pp. 81–85, 2024.
- [4] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.
- [5] S. C. Hoi, D. Sahoo, J. Lu, and P. Zhao, “Online learning: A comprehensive survey,” *Neurocomputing*, vol. 459, pp. 249–289, 2021.
- [6] E. A. Wan and R. Van Der Merwe, “The unscented Kalman filter for nonlinear estimation,” in *IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000, pp. 153–158.
- [7] S. Särkkä and L. Svensson, *Bayesian filtering and smoothing*. Cambridge university press, 2023, vol. 17.
- [8] P. G. Chang, K. P. Murphy, and M. Jones, “On diagonal approximations to the extended Kalman filter for online training of Bayesian neural networks,” in *Continual Lifelong Learning Workshop at ACML 2022*, 2022.
- [9] P. G. Chang, G. Durán-Martín, A. Y. Shestopaloff, M. Jones, and K. Murphy, “Low-rank extended Kalman filtering for online learning of neural networks from streaming data,” *arXiv preprint arXiv:2305.19535*, 2023.
- [10] M. Jones, P. Chang, and K. Murphy, “Bayesian online natural gradient (BONG),” *Advances in Neural Information Processing Systems*, vol. 37, pp. 131 104–131 153, 2024.
- [11] Y. Gusakov, O. Simeone, T. Rottunberg, and N. Shlezinger, “Rapid online Bayesian learning for deep receivers,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.
- [12] Á. Cartea, G. Duran-Martin, and L. Sánchez-Betancourt, “Detecting toxic flow,” *Quantitative Finance*, pp. 1–21, 2026.
- [13] M. Titsias, A. Galashov, A. Rannen-Triki, R. Pascanu, Y. W. Teh, and J. Bornschein, “Kalman filter for online classification of non-stationary data,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [14] G. Duran-Martin, L. Sánchez-Betancourt, A. Shestopaloff, and K. P. Murphy, “A unifying framework for generalised Bayesian online learning in non-stationary environments,” *Transactions on Machine Learning Research*, 2025.

- [15] S. F. Schmidt, “The Kalman filter-its recognition and development for aerospace applications,” *Journal of Guidance and Control*, vol. 4, no. 1, pp. 4–7, 1981.
- [16] I. Greenberg, N. Yannay, and S. Mannor, “Optimization or architecture: How to hack Kalman filtering,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 50 482–50 505, 2023.
- [17] L. Xu and R. Niu, “EKFNet: Learning system noise covariance parameters for nonlinear tracking,” *IEEE Transactions on Signal Processing*, vol. 72, pp. 3139–3152, 2024.
- [18] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*. OUP Oxford, 2012, vol. 38.
- [19] T. Raviv, S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger, “Adaptive and flexible model-based AI for deep receivers in dynamic channels,” *IEEE Wireless Communications*, vol. 31, no. 4, pp. 163–169, 2024.
- [20] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, “Learning under concept drift: A review,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.
- [21] M. Biehl and H. Schwarze, “Learning by on-line gradient descent,” *Journal of Physics A: Mathematical and general*, vol. 28, no. 3, pp. 643–656, 1995.
- [22] P. Bartlett, E. Hazan, and A. Rakhlin, “Adaptive online gradient descent,” *Advances in Neural Information Processing Systems*, vol. 20, 2007.
- [23] R. Dixit, A. S. Bedi, R. Tripathi, and K. Rajawat, “Online learning with inexact proximal online gradient descent algorithms,” *IEEE Transactions on Signal Processing*, vol. 67, no. 5, pp. 1338–1352, 2019.
- [24] A. Simonetto, E. Dall’Anese, S. Paternain, G. Leus, and G. B. Giannakis, “Time-varying convex optimization: Time-structured algorithms and applications,” *Proceedings of the IEEE*, vol. 108, no. 11, pp. 2032–2048, 2020.
- [25] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, “Continual lifelong learning with neural networks: A review,” *Neural networks*, vol. 113, pp. 54–71, 2019.
- [26] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, “A continual learning survey: Defying forgetting in classification tasks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.
- [27] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, “Tent: Fully test-time adaptation by entropy minimization,” in *International Conference on Learning Representations (ICLR)*, 2021.
- [28] Q. Wang, O. Fink, L. Van Gool, and D. Dai, “Continual test-time domain adaptation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7201–7211.
- [29] S. Niu, J. Wu, Y. Zhang, Z. Wen, Y. Chen, P. Zhao, and M. Tan, “Towards stable test-time adaptation in dynamic wild world,” in *International Conference on Learning Representations (ICLR)*, 2023.
- [30] M. Schirmer, D. Zhang, and E. Nalisnick, “Temporal test-time adaptation with state-space models,” *Transactions on Machine Learning Research*, 2025.
- [31] A. Kumar, T. Ma, and P. Liang, “Understanding self-training for gradual domain adaptation,” in *International Conference on Machine Learning*. PMLR, 2020, pp. 5468–5479.
- [32] G. Bai, C. Ling, and L. Zhao, “Temporal domain generalization with drift-aware dynamic neural networks,” in *International Conference on Learning Representations (ICLR)*, 2023.
- [33] Z. Xia, A. Davtyan, and P. Favaro, “KOALA++: Efficient Kalman-based optimization of neural networks with gradient-covariance products,” *arXiv preprint arXiv:2506.04432*, 2025.

- [34] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, “Meta-learning in neural networks: A survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.
- [35] R. Kurlle, B. Cseke, A. Klushyn, P. van der Smagt, and S. Günnemann, “Continual learning with bayesian neural networks for non-stationary data,” in *International Conference on Learning Representations (ICLR)*, 2020.
- [36] I. Arasaratnam and S. Haykin, “Cubature Kalman filters,” *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.
- [37] P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez, “Particle filtering,” *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, 2003.
- [38] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, and R. Roscher, “A survey of uncertainty in deep neural networks,” *Artificial Intelligence Review*, vol. 56, no. Suppl 1, pp. 1513–1589, 2023.
- [39] C. Li, H. Farkhoor, R. Liu, and J. Yosinski, “Measuring the intrinsic dimension of objective landscapes,” in *International Conference on Learning Representations (ICLR)*, 2018.
- [40] A. Aghajanyan, S. Gupta, and L. Zettlemoyer, “Intrinsic dimensionality explains the effectiveness of language model fine-tuning,” in *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2021, pp. 7319–7328.
- [41] W. B. Johnson, J. Lindenstrauss *et al.*, “Extensions of Lipschitz mappings into a Hilbert space,” *Contemporary mathematics*, vol. 26, no. 189-206, p. 1, 1984.
- [42] O. Press, S. Schneider, M. Kümmerer, and M. Bethge, “Rdumb: A simple approach that questions our progress in continual test-time adaptation,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 39 915–39 935, 2023.
- [43] X. Lin, “An overview of the 3gpp study on artificial intelligence for 5g new radio,” *arXiv preprint arXiv:2308.05315*, 2023.
- [44] R. Wiesmayr, S. Cammerer, F. Aït Aoudia, J. Hoydis, J. Zakrzewski, and A. Keller, “Design of a standard-compliant real-time neural receiver for 5g nr,” in *2025 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN)*. IEEE, 2025, pp. 1–6.
- [45] L. Dai, R. Jiao, F. Adachi, H. V. Poor, and L. Hanzo, “Deep learning for wireless communications: An emerging interdisciplinary paradigm,” *IEEE Wireless Communications*, vol. 27, no. 4, pp. 133–139, 2020.
- [46] Qualcomm Technologies, “Breaking the wireless barriers to mobilize 5G NR mmWave,” Qualcomm Technologies, Tech. Rep., 2018.
- [47] L. Bottou *et al.*, “Online learning and stochastic approximations,” *On-line learning in neural networks*, vol. 17, no. 9, p. 142, 1998.
- [48] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, “Improving robustness against common corruptions by covariate shift adaptation,” in *Advances in Neural Information Processing Systems*, 2020.
- [49] Z. Nado, S. Padhy, D. Sculley, A. D’Amour, B. Lakshminarayanan, and J. Snoek, “Evaluating prediction-time batch normalization for robustness under covariate shift,” *arXiv preprint arXiv:2006.10963*, 2020.
- [50] M. J. Mirza, J. Micorek, H. Possegger, and H. Bischof, “The norm must go on: Dynamic unsupervised domain adaptation by normalization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [51] H. Lim, B. Kim, J. Choo, and S. Choi, “Ttn: A domain-shift aware batch normalization in test-time adaptation,” in *International Conference on Learning Representations (ICLR)*, 2023.

- [52] R. A. Marsden, M. Döbler, and B. Yang, “Universal test-time adaptation through weight ensembling, diversity weighting, and prior correction,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2024, pp. 2543–2553.
- [53] N. Mu and J. Gilmer, “MNIST-C: A robustness benchmark for computer vision,” *arXiv preprint arXiv:1906.02337*, 2019. [Online]. Available: <https://arxiv.org/abs/1906.02337>
- [54] D. Hendrycks and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” in *International Conference on Learning Representations*, 2019. [Online]. Available: <https://openreview.net/forum?id=HJz6tiCqYm>
- [55] Y. LeCun, C. Cortes, and C. J. C. Burges, “MNIST handwritten digit database,” <https://yann.lecun.com/exdb/mnist/>, 2010.
- [56] A. Krizhevsky, V. Nair, and G. Hinton, “CIFAR-10 and CIFAR-100 datasets,” <https://www.cs.toronto.edu/~kriz/cifar.html>, 2009.
- [57] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [58] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016.
- [59] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated residual transformations for deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.
- [60] N. Shlezinger, R. Fu, and Y. C. Eldar, “DeepSIC: Deep soft interference cancellation for multiuser MIMO detection,” *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1349–1362, 2021.
- [61] S. Jaeckel, L. Raschkowski, K. Börner, and L. Thiele, “Quadriga: A 3-D multi-cell channel model with time evolution for enabling virtual field trials,” *IEEE Transactions on Antennas and Propagation*, vol. 62, no. 6, pp. 3242–3256, 2014.
- [62] L. Smaini, *RF analog impairments modeling for communication systems simulation: application to OFDM-based transceivers*. John Wiley & Sons, 2012.
- [63] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, “Variational continual learning,” *arXiv preprint arXiv:1710.10628*, 2017.
- [64] T. B. Schön, A. Wills, and B. Ninness, “System identification of nonlinear state-space models,” *Automatica*, vol. 47, no. 1, pp. 39–49, 2011.
- [65] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, “Deep state space models for nonlinear system identification,” *IFAC-PapersOnLine*, vol. 54, no. 7, pp. 481–486, 2021.
- [66] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, “Backprop KF: Learning discriminative deterministic state estimators,” *Advances in Neural Information Processing Systems*, vol. 29, 2016.
- [67] A. Kloss, G. Martius, and J. Bohg, “How to train your differentiable filter,” *Autonomous Robots*, vol. 45, no. 4, pp. 561–578, 2021.
- [68] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. Van Sloun, and Y. C. Eldar, “KalmanNet: Neural network aided Kalman filtering for partially known dynamics,” *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022.
- [69] V. G. Satorras, Z. Akata, and M. Welling, “Combining generative and discriminative models for hybrid inference,” in *Advances in Neural Information Processing Systems*, 2019, pp. 13 802–13 812.
- [70] A. Ghosh, A. Honoré, and S. Chatterjee, “DANSE: Data-driven non-linear state estimation of model-free process in unsupervised learning setup,” *IEEE Transactions on Signal Processing*, vol. 72, pp. 1824–1838, 2024.

- [71] P. Becker, H. Pandya, G. Gebhardt, C. Zhao, C. J. Taylor, and G. Neumann, “Recurrent Kalman networks: Factorized inference in high-dimensional deep feature spaces,” in *International Conference on Machine Learning*, 2019, pp. 544–552.
- [72] N. Shlezinger, G. Revach, A. Ghosh, S. Chatterjee, S. Tang, T. Imbiriba, J. Dunik, O. Straka, P. Closas, and Y. C. Eldar, “Artificial intelligence-aided Kalman filters: AI-augmented designs for Kalman-type algorithms,” *IEEE Signal Processing Magazine*, vol. 42, no. 3, pp. 52–76, 2025.
- [73] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, “Parameter-efficient transfer learning for NLP,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 2790–2799.
- [74] R. Karimi Mahabadi, J. Henderson, and S. Ruder, “Compacter: Efficient low-rank hypercomplex adapter layers,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 1022–1035, 2021.
- [75] X. L. Li and P. Liang, “Prefix-tuning: Optimizing continuous prompts for generation,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2021, pp. 4582–4597.
- [76] B. Lester, R. Al-Rfou, and N. Constant, “The power of scale for parameter-efficient prompt tuning,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 3045–3059.
- [77] E. Ben Zaken, Y. Goldberg, and S. Ravfogel, “Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models,” in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2022.
- [78] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “LoRA: Low-rank adaptation of large language models,” in *International Conference on Learning Representations (ICLR)*, 2022.
- [79] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “QLoRA: Efficient finetuning of quantized LLMs,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 10 088–10 115, 2023.
- [80] L. Wang, S. Chen, L. Jiang, S. Pan, R. Cai, S. Yang, and F. Yang, “Parameter-efficient fine-tuning in large language models: a survey of methodologies,” *Artificial Intelligence Review*, vol. 58, no. 8, p. 227, 2025.
- [81] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- [82] M. E. Khan and H. Rue, “The Bayesian learning rule,” *Journal of Machine Learning Research*, vol. 24, no. 281, pp. 1–46, 2023.
- [83] G. Bonnet, “Transformations des signaux aléatoires à travers les systèmes non linéaires sans mémoire,” *Annales des Télécommunications*, vol. 19, pp. 203–220, 1964.
- [84] R. Price, “A useful theorem for nonlinear devices having gaussian inputs,” *IRE Transactions on Information Theory*, vol. 4, no. 2, pp. 69–72, 1958.
- [85] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, “Augmix: A simple data processing method to improve robustness and uncertainty,” *arXiv preprint arXiv:1912.02781*, 2019.
- [86] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, “Robustbench: a standardized adversarial robustness benchmark,” *arXiv preprint arXiv:2010.09670*, 2020.
- [87] G. Duran-Martin, L. Sánchez-Betancourt, Á. Cartea, and K. Murphy, “Martingale posterior neural networks for fast sequential decision making,” in *Advances in Neural Information Processing Systems*, vol. 38, 2025.

- [88] L. Liu, C. Oestges, J. Poutanen, K. Haneda, P. Vainikainen, F. Quitin, F. Tufvesson, and P. De Doncker, “The COST 2100 MIMO channel model,” *IEEE Wireless Communications*, vol. 19, no. 6, pp. 92–99, 2012.
- [89] J. Song, J. Lee, I. S. Kweon, and S. Choi, “EcoTTA: Memory-efficient continual test-time adaptation via self-distilled regularization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 920–11 929.
- [90] X. Luo, J. Liu, K. Chen, J. Yang, B. Ding, A. Basu, and H. Li, “Efficient test-time adaptation through latent subspace coefficients search,” 2025.

Appendix

Appendix Overview

Additional Related Work	17
Theoretical Motivation for the Latent State-Space Formulation	17
Existence of Low-Dimensional Model Representations	18
Observation Model as Additive Noise Channel	18
Relation to Bayesian Learning	19
Proof of Theorem B.1	19
Proof of Lemma B.2	20
Proof of Lemma B.3	21
Implementation Details of the Experiments	23
Non-stationary Wireless Receiver Adaptation	23
Non-stationary Image Classification	24
Additional Experiments	25
Additional Wireless Communication Experiments	25
Additional Non-stationary Image Classification Experiments	30
Proof-of-Concept: Adaptive Deep Software-Defined Radio	32
Hardware Setup and Adaptive Receiver Architecture	33
Hardware Performance and Runtime Characterization	33
Limitations	34

A Additional Related Work

Online Continual Learning. Online continual learning (OCL) studies learning from sequential, potentially non-stationary data while retaining useful knowledge acquired from previous observations [26, 1]. A central challenge is the stability-plasticity trade-off: adapting rapidly to new data without catastrophically overwriting previously acquired knowledge. Existing approaches address this challenge through mechanisms such as replay, regularization, and Bayesian posterior updates [63, 35]. AURA shares with OCL the sequential, non-stationary setting and the need for rapid adaptation from limited observations, but differs in its primary objective. Rather than explicitly preserving performance on previously encountered distributions, AURA aims to track the predictor that is best aligned with the current distribution. Accordingly, it does not rely on episodic replay or explicit consolidation. Instead, it meta-learns offline a latent parameterization and dynamical model that capture how effective predictors evolve, and uses them for uncertainty-aware Bayesian tracking at deployment. This distinction is particularly relevant in settings where past operating conditions become obsolete rather than tasks that must later be revisited.

Differentiable and Learned Kalman Filters. The offline meta-learning stage of AURA, which learns latent adaptation dynamics by differentiating through Kalman-style recursions, is related to prior works that cast Kalman filtering as a differentiable machine learning model. Classical learning of state-space model parameters is typically studied separately from filtering within the framework of system identification, where the underlying dynamics are estimated from data [64, 65]. More recent works instead optimize Kalman-type filters end-to-end by backpropagating through the filtering recursion to learn state-space parameters or noise statistics directly from task losses [16, 17, 66, 67]. In parallel, a growing literature converts Kalman-type filtering itself into trainable neural architectures via learned gains [68], learned corrections [69], augmentations of prior [70] and posterior updates [71], and broader AI-aided Kalman filtering frameworks [72]. While these works primarily learn *state estimators* for tracking latent physical processes, AURA differs fundamentally in its objective: it does not learn a Kalman filter for state estimation, but rather leverages differentiable Bayesian filtering as a *meta-learning mechanism* for constructing an online adaptation rule, where the tracked latent state represents a learned low-dimensional parameterization of the evolving optimal model parameters in a non-stationary learning problem.

Parameter-Efficient Fine-Tuning and Adaptation. The learned low-dimensional adaptation space of AURA is also conceptually related to the literature on parameter-efficient fine-tuning (PEFT), which seeks to adapt large pretrained models (typically language models) by updating only a restricted subset or structured parameterization of the model weights rather than the full parameter vector. Representative PEFT approaches include adapter-based methods [73, 74], prompt- and prefix-based tuning [75, 76], selective parameter updates such as BitFit [77], and low-rank or reparameterized adaptation mechanisms including LoRA and its variants [78, 79]; see also the recent comprehensive overview [80]. Despite the shared motivation of reducing adaptation dimensionality, the setting considered in this work is fundamentally different. PEFT methods typically address *offline* adaptation to a fixed downstream task or domain, where substantial task-specific data and multiple optimization iterations are available. In contrast, AURA addresses *streaming online adaptation* in continuously evolving non-stationary environments, where only a handful of labeled samples are observed per update and adaptation must be performed under strict latency constraints. Consequently, while PEFT focuses on reducing the cost of conventional fine-tuning, AURA instead learns a latent adaptation geometry specifically tailored for real-time Bayesian online tracking of drifting optima.

B Theoretical Motivation for the Latent State-Space Formulation

AURA is based on the principle that rapid adaptation can be achieved by *tracking the evolution of optimal model parameters in a suitably learned latent space*. Concretely, we posit that, although θ_t may be high-dimensional, it admits a low-dimensional representation whose temporal evolution is governed by smooth dynamics. When such a representation is available, online learning can be cast as a state estimation problem, where adaptation is carried out efficiently via Bayesian filtering in the latent space. In this appendix we establish the foundations of this approach. This includes: (i) providing a theoretical argument for the existence of low-dimensional representations in which the optimal parameters evolve smoothly over time; (ii) identify conditions for which the representation of the

observation corruption as additive uncorrelated noise in (4) holds; and (iii) provide a rigorous relation between Kalman tracking in latent space and conventional Bayesian learning based on evidence lower bound (ELBO) minimization. All proofs are delegated to the end of this appendix.

B.1 Existence of Low-Dimensional Model Representations

We begin by formalizing the learning objective of deep neural network (DNN) predictors and establishing the modeling viewpoint that motivates our formulation of a latent dynamical system.

Learning Framework. Following standard statistical learning terminology [81], we consider learning guided by a loss function $\ell : \Delta^C \times \mathcal{Y} \mapsto \mathbb{R}$. The instantaneous risk at time t is defined as

$$\mathcal{L}_t(\theta) := \mathbb{E}_{(x,y) \sim \mathbb{P}_t} [\ell(f_\theta(x), y)]. \quad (9)$$

The learning objective at time t is thus to minimize the risk induced by the current distribution. Accordingly, we define the instantaneous optimal parameter vector as

$$\theta_t^* \in \arg \min_{\theta} \mathcal{L}_t(\theta). \quad (10)$$

Motivating Learning Model. To ground the casting of online learning as the tracking of a dynamic low-dimensional system, we make use of the following assumption:

ASI (Lipschitz Compressibility) There exists an L -Lipschitz compression mapping $\mathcal{C} : \mathbb{R}^d \mapsto \mathbb{R}^m$ with $m \ll d$ such that $\theta_t^* \in \mathbb{R}^d$ can be recovered from $z_t = \mathcal{C}(\theta_t^*) \in \mathbb{R}^m$ for each t .

Assumption *ASI* reflects the hypothesis that, although the ambient parameter space is high-dimensional, the set of optimal solutions induced by varying data distributions lies on a low-dimensional manifold. In our setting, this suggests that, provided the sequence $\{\theta_t^*\}$ remains sufficiently structured, there exists a low-dimensional representation z_t that captures the relevant variations across time. Importantly, we do not require explicit knowledge of the mapping $\mathcal{C}$, but only posit its existence as a modeling assumption that will later be relaxed through learning. Under *ASI*, we can identify regimes in which the optimal parameters evolve along a low-dimensional manifold and vary similarly to the distribution variations. This is formulated in the following theorem:

Proposition B.1. *When the risk (9) is differentiable and μ -strongly quasi-convex, and the loss ℓ is uniformly bounded by some $B > 0$, then, the latent representations z_t and z_{t+1} satisfy*

$$\|z_t - z_{t+1}\|_2 \leq 2L \sqrt{\frac{2}{\mu} B \cdot D_{\text{TV}}(\mathbb{P}_t; \mathbb{P}_{t+1})}, \quad \forall t, \quad (11)$$

where $D_{\text{TV}}(\cdot; \cdot)$ is the total variation distance.

While Proposition B.1 is derived under restrictive conditions (e.g., strong quasi-convexity of the risk), its main implication is qualitative rather than prescriptive. Specifically, it suggests that the variation in the optimal parameters, when viewed through a suitable low-dimensional representation, is governed by the magnitude of the underlying distribution shift. This observation provides support for modeling the latent sequence $\{z_t\}$ as a *smoothly evolving dynamical system*. Although the assumptions required for the theorem may not hold exactly in practical deep learning settings, they serve to motivate the broader hypothesis that non-stationary learning problems often admit low-dimensional structures with temporally coherent dynamics, which can be exploited for efficient online adaptation.

B.2 Observations Model as an Additive Noise Channel

The observation model (4) reflects the mismatch between the probabilistic output of the predictive model and the label, with w_t being the estimation error. Specifically, when the loss ℓ guiding learning is the standard cross-entropy loss, then w_t holds the following property:

Lemma B.2. *When f_θ is sufficiently expressive such that $\exists \theta_t \in \mathbb{R}^d$ for which $f_{\theta_t}(x) = [\mathbb{P}_t(y = 1|x), \dots, \mathbb{P}_t(y = C|x)]$ for all $x \in \text{supp}(\mathbb{P}_t)$, then w_t is zero-mean and uncorrelated with $f_{\mathcal{G}(z_t)}(x_t)$.*

Recall that we design our online learning algorithm assuming that w_t is zero-mean and uncorrelated with the model output. This modeling decision is thus further motivated by Lemma B.2.

B.3 Relation to Bayesian Learning

AURA implements online learning by casting this task as the tracking of the latent representation of the optimal model parameters, rather than by using standard risk-minimization based learning formulation. Still, it admits a principled interpretation from a Bayesian learning perspective. In particular, under a linearized Gaussian approximation, EKF updates coincide with Bayesian online learning rules based on the ELBO. This is stated in the following lemma, adapted from [10]:

Lemma B.3. *When (i) the observation model induced by the predictor is replaced with its first-order Taylor approximation around $\hat{\mathbf{z}}_{t|t-1}$, i.e., $f_{\mathcal{G}(\mathbf{z}_t)}(x_t) = f_{\mathcal{G}(\hat{\mathbf{z}}_{t|t-1})}(x_t) + \mathbf{H}_t(\mathbf{z}_t - \hat{\mathbf{z}}_{t|t-1})$; and (ii) the noise terms $\mathbf{w}_t$ and $\mathbf{v}_t$ are Gaussian, then (5)-(6) are equivalent to a natural gradient minimization of the ELBO loss with Gaussian prior [82].*

Lemma B.3 shows that while our learning methodology is based on Kalman-style adaptation, designed for tracking dynamic systems, it can be viewed as specialization of existing Bayesian machine learning frameworks derived from conventional training objectives.

B.4 Proofs

B.4.1 Proof of Proposition B.1

We first state the additional assumption introduced in the statement of the theorem:

AS2 (Strong quasi-convexity) The risk (9) is differentiable and μ -strongly quasi-convex for some constant $\mu > 0$, i.e., for any $\boldsymbol{\theta} \in \mathbb{R}^d$ and t it holds that

$$f_{\mathbb{P}_t}(\boldsymbol{\theta}^*) \geq f_{\mathbb{P}_t}(\boldsymbol{\theta}) + \langle \nabla f_{\mathbb{P}_t}(\boldsymbol{\theta}), \boldsymbol{\theta}^* - \boldsymbol{\theta} \rangle + \frac{\mu}{2} \|\boldsymbol{\theta}^* - \boldsymbol{\theta}\|^2.$$

AS3 (Bounded loss) The loss ℓ is bounded by $B > 0$.

We first bound the change in the risk induced by the variation in the data-generating distribution. Fix $\boldsymbol{\theta}$, and define

$$g_{\boldsymbol{\theta}}(x, y) \triangleq \ell(f_{\boldsymbol{\theta}}(x), y).$$

By Assumption AS3, $g_{\boldsymbol{\theta}}$ is uniformly bounded, namely $|g_{\boldsymbol{\theta}}(x, y)| \leq B$ for all (x, y) . It is a standard property of total variation distance that, for any bounded measurable function g ,

$$|\mathbb{E}_{\mathbb{P}_t}[g(x, y)] - \mathbb{E}_{\mathbb{P}_{t+1}}[g(x, y)]| \leq 2\|g\|_{\infty} D_{\text{TV}}(\mathbb{P}_t; \mathbb{P}_{t+1}).$$

Applying this with $g = g_{\boldsymbol{\theta}}$ yields

$$\begin{aligned} |f_t(\boldsymbol{\theta}) - f_{t+1}(\boldsymbol{\theta})| &= |\mathbb{E}_{(x,y) \sim \mathbb{P}_t}[g_{\boldsymbol{\theta}}(x, y)] - \mathbb{E}_{(x,y) \sim \mathbb{P}_{t+1}}[g_{\boldsymbol{\theta}}(x, y)]| \\ &\leq 2B \cdot D_{\text{TV}}(\mathbb{P}_t; \mathbb{P}_{t+1}). \end{aligned} \quad (12)$$

Next, by Assumption AS2 and the optimality of $\boldsymbol{\theta}_t^*$ for f_t , we have

$$f_t(\boldsymbol{\theta}_{t+1}^*) \geq f_t(\boldsymbol{\theta}_t^*) + \frac{\mu}{2} \|\boldsymbol{\theta}_{t+1}^* - \boldsymbol{\theta}_t^*\|_2^2, \quad (13)$$

where we used that $\nabla f_t(\boldsymbol{\theta}_t^*) = 0$.

On the other hand, since $\boldsymbol{\theta}_{t+1}^*$ minimizes f_{t+1} ,

$$f_{t+1}(\boldsymbol{\theta}_{t+1}^*) \leq f_{t+1}(\boldsymbol{\theta}_t^*).$$

Combining this inequality with (12) yields

$$\begin{aligned} f_t(\boldsymbol{\theta}_{t+1}^*) &\leq f_{t+1}(\boldsymbol{\theta}_{t+1}^*) + 2B \cdot D_{\text{TV}}(\mathbb{P}_t; \mathbb{P}_{t+1}) \\ &\leq f_{t+1}(\boldsymbol{\theta}_t^*) + 2B \cdot D_{\text{TV}}(\mathbb{P}_t; \mathbb{P}_{t+1}) \\ &\leq f_t(\boldsymbol{\theta}_t^*) + 4B \cdot D_{\text{TV}}(\mathbb{P}_t; \mathbb{P}_{t+1}). \end{aligned} \quad (14)$$

Substituting (14) into (13) gives

$$f_t(\boldsymbol{\theta}_t^*) + \frac{\mu}{2} \|\boldsymbol{\theta}_{t+1}^* - \boldsymbol{\theta}_t^*\|_2^2 \leq f_t(\boldsymbol{\theta}_t^*) + 4B \cdot D_{\text{TV}}(\mathbb{P}_t; \mathbb{P}_{t+1}),$$

and therefore

$$\|\boldsymbol{\theta}_{t+1}^* - \boldsymbol{\theta}_t^*\|_2 \leq 2\sqrt{\frac{2B \cdot D_{\text{TV}}(\mathbb{P}_t; \mathbb{P}_{t+1})}{\mu}}. \quad (15)$$

Finally, by the definition of the compressed representations,

$$z_{t+1} - z_t = \mathcal{C}(\boldsymbol{\theta}_{t+1}^* - \boldsymbol{\theta}_t^*).$$

Hence, by continuity of the operator,

$$\|z_{t+1} - z_t\|_2 = \|\mathcal{C}(\boldsymbol{\theta}_{t+1}^* - \boldsymbol{\theta}_t^*)\|_2 \leq \|A\|_2 \|\boldsymbol{\theta}_{t+1}^* - \boldsymbol{\theta}_t^*\|_2.$$

Combining this with (15), we obtain

$$\|z_{t+1} - z_t\|_2 \leq 2L\sqrt{\frac{2B \cdot D_{\text{TV}}(\mathbb{P}_t; \mathbb{P}_{t+1})}{\mu}},$$

which proves the claim.

B.4.2 Proof of Lemma B.2

Proof. We prove the lemma in two steps. We first show that, under the stated expressivity assumption, the parameter vector realizing the true posterior probabilities is an optimizer of the instantaneous risk under the cross-entropy loss. We then show that the corresponding observation error is zero-mean and uncorrelated with the model output.

Step 1. Consider the standard cross-entropy loss

$$\ell(f_{\boldsymbol{\theta}}(x), y) = -\sum_{c=1}^C \mathbf{1}\{y = c\} \log [f_{\boldsymbol{\theta}}(x)]_c, \quad (16)$$

where $[f_{\boldsymbol{\theta}}(x)]_c$ denotes the c th component of the predictive probability vector $f_{\boldsymbol{\theta}}(x) \in \Delta^C$. Substituting (16) into the instantaneous risk (9), we obtain

$$\begin{aligned} \mathcal{L}_t(\boldsymbol{\theta}) &= \mathbb{E}_{(x,y) \sim \mathbb{P}_t} \left[-\sum_{c=1}^C \mathbf{1}\{y = c\} \log [f_{\boldsymbol{\theta}}(x)]_c \right] \\ &= \mathbb{E}_{x \sim \mathbb{P}_t} \left[\mathbb{E}_{y|x \sim \mathbb{P}_t} \left[-\sum_{c=1}^C \mathbf{1}\{y = c\} \log [f_{\boldsymbol{\theta}}(x)]_c \mid x \right] \right] \\ &= \mathbb{E}_{x \sim \mathbb{P}_t} \left[-\sum_{c=1}^C \mathbb{P}_t(y = c|x) \log [f_{\boldsymbol{\theta}}(x)]_c \right]. \end{aligned} \quad (17)$$

Now let

$$\mathbf{p}_t(x) := [\mathbb{P}_t(y = 1|x), \dots, \mathbb{P}_t(y = C|x)] \in \Delta^C.$$

By assumption, there exists $\boldsymbol{\theta}_t \in \mathbb{R}^d$ such that $f_{\boldsymbol{\theta}_t}(x) = \mathbf{p}_t(x)$ for all $x \in \text{supp}(\mathbb{P}_t)$. For any fixed x , the inner term in (17) is the cross-entropy between the true conditional distribution $\mathbf{p}_t(x)$ and the model prediction $f_{\boldsymbol{\theta}}(x)$. Using the identity

$$-\sum_{c=1}^C p_c \log q_c = -\sum_{c=1}^C p_c \log p_c + \sum_{c=1}^C p_c \log \frac{p_c}{q_c}, \quad (18)$$

valid for any probability vectors $\mathbf{p}, \mathbf{q} \in \Delta^C$, we may rewrite (17) as

$$\begin{aligned} \mathcal{L}_t(\boldsymbol{\theta}) &= \mathbb{E}_{x \sim \mathbb{P}_t} \left[-\sum_{c=1}^C \mathbb{P}_t(y = c|x) \log \mathbb{P}_t(y = c|x) \right] \\ &\quad + \mathbb{E}_{x \sim \mathbb{P}_t} \left[\sum_{c=1}^C \mathbb{P}_t(y = c|x) \log \frac{\mathbb{P}_t(y = c|x)}{[f_{\boldsymbol{\theta}}(x)]_c} \right]. \end{aligned} \quad (19)$$

The first term in (19) does not depend on θ , while the second term is a conditional Kullback–Leibler divergence and is therefore nonnegative. Consequently,

$$\mathcal{L}_t(\theta) \geq \mathbb{E}_{x \sim \mathbb{P}_t} \left[- \sum_{c=1}^C \mathbb{P}_t(y = c|x) \log \mathbb{P}_t(y = c|x) \right], \quad (20)$$

with equality if and only if $f_\theta(x) \equiv [\mathbb{P}_t(y = 1|x), \dots, \mathbb{P}_t(y = C|x)]$. Hence, the parameter vector θ_t minimizing (20) is an optimizer of (9), namely

$$\theta_t \in \arg \min_{\theta} \mathcal{L}_t(\theta).$$

By the definition of the instantaneous optimum in (10), we may therefore identify this optimizer with θ_t^* . Since, by construction of the latent representation, $\mathcal{G}(z_t) = \theta_t^*$, it follows that the observation model (4) can be written as

$$\text{OneHot}(y_t) = f_{\theta_t^*}(x_t) + w_t. \quad (21)$$

Step 2. Define the observation error as

$$w_t := \text{OneHot}(y_t) - f_{\theta_t^*}(x_t). \quad (22)$$

Since $f_{\theta_t^*}(x)$ realizes the true posterior probabilities, we have

$$f_{\theta_t^*}(x) = \mathbb{E}_{y|x \sim \mathbb{P}_t} [\text{OneHot}(y) | x]. \quad (23)$$

Therefore,

$$\begin{aligned} \mathbb{E}[w_t | x_t] &= \mathbb{E}[\text{OneHot}(y_t) - f_{\theta_t^*}(x_t) | x_t] \\ &= \mathbb{E}[\text{OneHot}(y_t) | x_t] - f_{\theta_t^*}(x_t) \\ &= \mathbf{0}, \end{aligned} \quad (24)$$

where the last equality follows from (23). Taking expectation over x_t yields

$$\mathbb{E}[w_t] = \mathbf{0}, \quad (25)$$

and thus w_t is zero-mean.

Moreover, since $f_{\theta_t^*}(x_t)$ is a deterministic function of x_t , (24) implies the orthogonality relation

$$\begin{aligned} \mathbb{E}[w_t f_{\theta_t^*}(x_t)^\top] &= \mathbb{E}[\mathbb{E}[w_t f_{\theta_t^*}(x_t)^\top | x_t]] \\ &= \mathbb{E}[f_{\theta_t^*}(x_t)^\top \mathbb{E}[w_t | x_t]] \\ &= \mathbf{0}. \end{aligned} \quad (26)$$

Hence, the estimation error is orthogonal to the model output. $\square$

B.4.3 Proof of Lemma B.3

The lemma extends the corresponding characterization in [10] to Bayesian learning in latent space. Accordingly, the proof follows similar arguments as in [10, Appendix E]. We begin by writing the ELBO objective associated with the Gaussian variational posterior based on the Bayesian learning rule of [82]. Under a Gaussian modeling of the parameters z_t as $z_t \sim \mathcal{N}(\hat{z}_t, \Sigma_t) := q_{\phi_t}$ where $\phi_t := \{\Sigma_t^{-1} \hat{z}_t, -\frac{1}{2} \Sigma_t^{-1}\}$ encapsulates the distribution parameters of z_t . The online ELBO can be written as

$$\mathcal{L}(\phi_t) = -\mathbb{E}_{z_t \sim q_{\phi_t}} [\log p(y_t | x_t, \mathcal{G}(z_t))] + D_{\text{KL}}(q_{\phi_t} \| q_{\phi_{t-1}}), \quad (27)$$

where we write $p(y | x, \theta)$ as the element of the model output $f_\theta(x)$ corresponding to $y \in \{1, \dots, C\}$.

For this Gaussian exponential-family representation, the corresponding dual, or expectation, parameters are

$$\rho_t^{(1)} = \hat{z}_t, \quad \rho_t^{(2)} = \hat{z}_t \hat{z}_t^\top + \Sigma_t. \quad (28)$$

Equivalently, the mean and covariance can be recovered from the natural or dual parameters as

$$\hat{\mathbf{z}}_t = -\frac{1}{2} \left(\boldsymbol{\phi}_t^{(2)} \right)^{-1} \boldsymbol{\phi}_t^{(1)} = \boldsymbol{\rho}_t^{(1)}, \quad (29)$$

$$\boldsymbol{\Sigma}_t = -\frac{1}{2} \left(\boldsymbol{\phi}_t^{(2)} \right)^{-1} = \boldsymbol{\rho}_t^{(2)} - \boldsymbol{\rho}_t^{(1)} \left(\boldsymbol{\rho}_t^{(1)} \right)^\top. \quad (30)$$

Starting from the prior predictive natural parameter $\boldsymbol{\phi}_{t|t-1}$, a single natural-gradient step on (27) yields

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t|t-1} + \mathbf{F}_{\boldsymbol{\phi}_{t|t-1}}^{-1} \nabla_{\boldsymbol{\phi}_{t|t-1}} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_{t|t-1}}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))], \quad (31)$$

where $\mathbf{F}_\phi$ is the Fisher information matrix with respect to the natural parameters ϕ . Since the Gaussian variational posterior belongs to the exponential family, the natural gradient with respect to the natural parameters coincides with the ordinary gradient with respect to the corresponding dual parameters $\boldsymbol{\rho}$. Therefore, (31) can be equivalently written as

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t|t-1} + \nabla_{\boldsymbol{\rho}_{t|t-1}} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_{t|t-1}}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))]. \quad (32)$$

Using the relations between the Gaussian natural, dual, and moment parameters, the chain rule yields, for any scalar function ℓ ,

$$\begin{aligned} \nabla_{\boldsymbol{\rho}_t^{(1)}} \ell &= \frac{\partial \hat{\mathbf{z}}_t}{\partial \boldsymbol{\rho}_t^{(1)}} \nabla_{\hat{\mathbf{z}}_t} \ell + \frac{\partial \boldsymbol{\Sigma}_t}{\partial \boldsymbol{\rho}_t^{(1)}} \nabla_{\boldsymbol{\Sigma}_t} \ell = \nabla_{\hat{\mathbf{z}}_t} \ell - 2(\nabla_{\boldsymbol{\Sigma}_t} \ell) \hat{\mathbf{z}}_t, \\ \nabla_{\boldsymbol{\rho}_t^{(2)}} \ell &= \frac{\partial \hat{\mathbf{z}}_t}{\partial \boldsymbol{\rho}_t^{(2)}} \nabla_{\hat{\mathbf{z}}_t} \ell + \frac{\partial \boldsymbol{\Sigma}_t}{\partial \boldsymbol{\rho}_t^{(2)}} \nabla_{\boldsymbol{\Sigma}_t} \ell = \nabla_{\boldsymbol{\Sigma}_t} \ell. \end{aligned} \quad (33)$$

Substituting $\ell = \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))]$ into the chain-rule identities, and using Bonnet's and Price's identities [83, 84], we obtain

$$\nabla_{\hat{\mathbf{z}}_t} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))] = \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\nabla_{\mathbf{z}_t} \log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))], \quad (34)$$

$$\nabla_{\boldsymbol{\Sigma}_t} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))] = \frac{1}{2} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\nabla_{\mathbf{z}_t}^2 \log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))]. \quad (35)$$

Substituting the above identities into (33) yields

$$\begin{aligned} \nabla_{\boldsymbol{\rho}_t^{(1)}} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))] &= \nabla_{\hat{\mathbf{z}}_t} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))] \\ &\quad - 2(\nabla_{\boldsymbol{\Sigma}_t} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))]) \hat{\mathbf{z}}_t \\ &= \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\nabla_{\mathbf{z}_t} \log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))] \\ &\quad - \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\nabla_{\mathbf{z}_t}^2 \log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))] \hat{\mathbf{z}}_t, \end{aligned} \quad (36)$$

$$\begin{aligned} \nabla_{\boldsymbol{\rho}_t^{(2)}} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))] &= \nabla_{\boldsymbol{\Sigma}_t} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))] \\ &= \frac{1}{2} \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_t}} [\nabla_{\mathbf{z}_t}^2 \log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))]. \end{aligned} \quad (37)$$

Equivalently, in terms of the posterior mean and covariance, the update becomes

$$\begin{aligned} \hat{\mathbf{z}}_t &= \hat{\mathbf{z}}_{t|t-1} + \boldsymbol{\Sigma}_t \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_{t|t-1}}} [\nabla_{\mathbf{z}_t} \log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))], \\ \boldsymbol{\Sigma}_t^{-1} &= \boldsymbol{\Sigma}_{t|t-1}^{-1} - \mathbb{E}_{\mathbf{z}_t \sim q_{\boldsymbol{\phi}_{t|t-1}}} [\nabla_{\mathbf{z}_t}^2 \log p(y_t | x_t, \mathcal{G}(\mathbf{z}_t))]. \end{aligned} \quad (38)$$

Next, recall that the state-space model assumes Gaussian process and observation noises, namely, $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ and $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. Consequently, the prior predictive distribution of the latent state is Gaussian and can be written as

$$q_{\boldsymbol{\phi}_{t|t-1}} \sim \mathcal{N}(\hat{\mathbf{z}}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1}).$$

Moreover, under the Gaussian observation model

$$p(y_t | x_t, \mathcal{G}(\mathbf{z}_t)) \sim \mathcal{N}(f_{\mathcal{G}(\mathbf{z}_t)}(\mathbf{x}_t), \mathbf{R}),$$

we replace the nonlinear predictor with its first-order Taylor approximation around $\hat{\mathbf{z}}_{t|t-1}$, namely,

$$f_{\mathcal{G}(\mathbf{z}_t)}(\mathbf{x}_t) = f_{\mathcal{G}(\hat{\mathbf{z}}_{t|t-1})}(\mathbf{x}_t) + \mathbf{H}_t(\mathbf{z}_t - \hat{\mathbf{z}}_{t|t-1}).$$

Under these assumptions, the expected gradient and Hessian admit the closed-form expressions

$$\begin{aligned} \mathbb{E}_{\mathbf{z}_t \sim q_{\phi_t|t-1}} [\nabla_{\mathbf{z}_t} \log p_t(\mathbf{y}_t | \mathbf{z}_t)] &= \mathbb{E}_{\mathbf{z}_t \sim q_{\phi_t|t-1}} \left[\mathbf{H}_t^\top \mathbf{R}^{-1} \left(\text{OneHot}(y_t) - \hat{\mathbf{y}}_t - \mathbf{H}_t(\mathbf{z}_t - \hat{\mathbf{z}}_{t|t-1}) \right) \right] \\ &= \mathbf{H}_t^\top \mathbf{R}^{-1} (\text{OneHot}(y_t) - \hat{\mathbf{y}}_t), \\ \mathbb{E}_{\mathbf{z}_t \sim q_{\phi_t|t-1}} [\nabla_{\mathbf{z}_t}^2 \log p_t(\mathbf{y}_t | \mathbf{z}_t)] &= \mathbb{E}_{\mathbf{z}_t \sim q_{\phi_t|t-1}} \left[-\mathbf{H}_t^\top \mathbf{R}^{-1} \mathbf{H}_t \right] \\ &= -\mathbf{H}_t^\top \mathbf{R}^{-1} \mathbf{H}_t. \end{aligned} \tag{39}$$

Applying (39) to (38), and rearranging the resulting expressions, yields

$$\begin{aligned} \mathbf{K}_t &= \Sigma_{t|t-1} \mathbf{H}_t^\top (\mathbf{H}_t \Sigma_{t|t-1} \mathbf{H}_t^\top + \mathbf{R})^{-1}, \\ \hat{\mathbf{z}}_t &= \hat{\mathbf{z}}_{t|t-1} + \mathbf{K}_t (\text{OneHot}(y_t) - f_{\mathcal{G}(\hat{\mathbf{z}}_{t|t-1})}(\mathbf{x}_t)), \\ \Sigma_t &= \Sigma_{t|t-1} - \mathbf{K}_t \mathbf{H}_t \Sigma_{t|t-1}. \end{aligned}$$

Hence, under the Gaussian noise assumption and the first-order linearization of the observation model, minimizing the ELBO via a single natural-gradient step is equivalent to a Kalman-style single-step adaptation update.

C Implementation Details of the Experiments in Section 5

C.1 Non-stationary Wireless Receiver Adaptation

Time-Varying MIMO Detection Scenario. We consider an uplink multiuser MIMO system with K single-antenna transmitters and N_r receive antennas. At each time frame, the users transmit a vector of information symbols, and the receiver observes a noisy superposition governed by a time-varying wireless channel. Specifically, for a transmitted QPSK symbol vector $\mathbf{s}_t \in \mathbb{R}^K$, which modulates the $2K$ bits, i.e., $C = 2^{2K}$, the received signal is modeled as

$$\mathbf{x}_t = \mathbf{H}_t \mathbf{s}_t + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_t), \tag{40}$$

where $\mathbf{H}_t \in \mathbb{R}^{N_r \times K}$ is the channel matrix and $\mathbf{P}_t$ denotes the noise covariance. The variation of $\mathbf{H}_t$ across time frames makes the MIMO detection rule time dependent, thereby giving rise to a non-stationary online detection setting.

The receiver architecture is based on DeepSIC [60], a learned implementation of iterative soft interference cancellation for multiuser detection. We evaluate this receiver under both linear and nonlinear observation models. In the linear case, the received signal follows (40). In the nonlinear case, we apply a component-wise tanh distortion to the received signal, resulting in a more challenging setting that departs from the nominal linear MIMO model and tests the robustness of the adaptation mechanism.

Protocol and Channel Generation. We instantiate the system with $K = 3$ users and $N_r = 5$ receive antennas. Time-varying channel trajectories are generated using the QuaDRiGa simulator under the 3GPP Indoor Office LOS scenario, yielding temporally correlated channel realizations that emulate realistic propagation dynamics. Each evaluation sequence contains 150 consecutive frames. The channel realization remains fixed over the duration of each frame and varies across frames, thereby inducing a sequence of temporally correlated detection problems.

The online protocol consists of two phases. The first four frames serve as an initial synchronization stage, during which the receiver is provided with 64 labeled pilot samples per frame. These samples are used to perform consecutive online updates and initialize the adaptive receiver state. The remaining 146 frames constitute the tracking stage. In each tracking frame, only 6 labeled pilot samples are available for adaptation, corresponding to 6 online updates. The adapted receiver is then evaluated on 1000 additional QPSK symbol vectors generated under the same channel realization, i.e., 2000

information bits per user. This setup reflects the operational structure of practical communication links: a short synchronization period enables reliable initialization, whereas subsequent operation must track channel variations using only limited pilot overhead. To avoid leakage between phases, all offline training is performed on channel trajectories disjoint from those used for online evaluation. During online operation, each method may update only from the labeled pilots revealed in the current stream.

The neural receiver is based on the DeepSIC multi-user detection network [60]. It is composed of four multi-layer perceptron modules, each containing one detector block per transmitted user. Each block produces a soft estimate of its associated user symbol from the received signal and the current soft estimates of the interfering users; these estimates are propagated across layers to progressively refine the multi-user decision. In our implementation, each block is a compact two-layer neural network with a nonlinear activation and 868 trainable parameters. Online adaptation is therefore performed at the block-detector level, with the proposed method replacing direct full-space block updates by updates in a learned low-dimensional latent representation from which the block parameters are reconstructed.

Compared Methods. We compare AURA against representative online adaptation baselines. As a gradient-based reference, we include *Online GD*, which updates the full set of receiver parameters directly using the labeled pilots available at each frame and perform five consecutive gradient updates per labeled sample. We further consider parameter-space EKF-based learning, using BONG [10], with full, diagonal [8], and DLR covariance parameterizations [9]. These baselines adapt directly in the original parameter space of the DeepSIC detector.

Implementation and Compute. All wireless receiver experiments were implemented in PyTorch and executed on CPU using a 13th Gen Intel(R) Core(TM) i7-13620H processor with 10 physical cores and 16 logical threads. All methods were evaluated on the same hardware configuration and with the same evaluation pipeline to ensure a fair comparison.

C.2 Non-stationary Image Classification

Gradual Distribution Shift Scenario. We consider a pretrained image classification model trained on a clean source dataset, such as CIFAR-10, CIFAR-100, or ImageNet. Let $\{\tilde{\mathbf{x}}_{t,i}\}_{i=1}^V$ denote the set of corrupted versions of the same underlying image at time step t , where V is the number of corruption types under consideration. To model a gradually evolving test distribution, we define a mixing vector $\boldsymbol{\alpha}_t \in \mathbb{R}^V$, satisfying $\sum_{i=1}^V \alpha_{t,i} = 1$, whose entries determine the contribution of each corruption type at time t .

Initially, the mixing process is set such that all mass is assigned to the clean component, namely $\boldsymbol{\alpha}_0 = [1, 0, \dots, 0]$. At each time step t , the observed test sample is generated as a convex combination of the corrupted versions of the same image, weighted by the current mixing vector:

$$\mathbf{x}_t = \sum_{i=1}^V \alpha_{t,i} \tilde{\mathbf{x}}_{t,i}. \quad (41)$$

Thus, as $\boldsymbol{\alpha}_t$ evolves over time according to a stochastic process, the induced test distribution undergoes a gradual and continuous shift. In the considered online adaptation setting, only a small subset of samples is labeled at each time step, enabling the adaptation methods to update using a limited amount of reliable supervision.

Protocol and Datasets. Concretely, each evaluation run consists of a stream of $T = 1000$ time steps (frames), each containing a batch of 100 images, for a total of 10^5 test samples per run. We draw corruptions from a pool of $V = 6$ types covering two qualitatively distinct regimes: three noise corruptions (Gaussian/shot/impulse on CIFAR; shot/impulse/fog on MNIST) and three blur/photometric corruptions (defocus, glass, motion on CIFAR; motion blur, glass blur, brightness on MNIST). The mixing vector $\boldsymbol{\alpha}_t$ evolves as a random walk on the simplex, so that consecutive frames induce similar but non-identical mixtures and the induced shift is both gradual and non-stationary. We evaluate on three standard corruption benchmarks, MNIST-C, CIFAR-10-C, and CIFAR-100-C, applying the corruptions at severity 5 on CIFAR-10-C and CIFAR-100-C. Each backbone is a standard image-classification architecture pretrained on the corresponding clean source dataset, and is kept fixed throughout the test-time stream: on MNIST-C we use a ResNet-18 [57] adapted to single-channel inputs, trained in-house on clean MNIST to 99.34% test accuracy; on CIFAR-10-C we use a Wide

Residual Network [58] with 28 layers and a widen factor of 10, pretrained on clean CIFAR-10 with standard supervised training; and on CIFAR-100-C we use a 29-layer ResNeXt [59] pretrained on clean CIFAR-100 with AugMix [85]. The CIFAR-100 backbone is obtained from RobustBench [86].

Compared Methods. We benchmark the proposed method against representative online adaptation strategies spanning non-adaptive, gradient-based, normalization-based, self-supervised, and Bayesian approaches. As a reference point, we include the *Source* model, which applies the pretrained classifier without any test-time parameter updates. To capture supervised gradient-based adaptation, we consider an *Online GD* baseline that updates the model directly using the labeled samples obtained on each new frame and perform five consecutive gradient update per labeled sample. We also evaluate a family of normalization-based baselines that adapt the batch-normalization statistics at test time, including a standard BN adaptation rule [48, 49] (*BN-Test*), an exponential-moving-average variant (*BN-EMA*), and an interpolation-based variant that combines source and target statistics [50, 51] (*BN-Alpha*). As a self-supervised alternative, we evaluate *ROID* [52]. We further consider a Bayesian filtering baseline based on the Kalman-filter framework of [13], adapted in our setting to operate on the final fully-connected classification layer. Finally, we compare these methods with the proposed approach, which performs Bayesian online adaptation in a learned low-dimensional latent space (with dimension size $m = 128$), while reconstructing the full parameter vector for prediction. In all image-classification experiments, online adaptation is applied to the final fully-connected classification layer together with selected batch-normalization parameters throughout the backbone. To keep the Bayesian update tractable, we assume independence between the adapted parameter groups, corresponding to a block-diagonal covariance structure across groups. This type of block-structured approximation has also been used in scalable Kalman-style neural-network updates to avoid modeling all cross-parameter dependencies [87], thereby reducing complexity.

Implementation and Compute. All image adaptation experiments were implemented in PyTorch and executed on an NVIDIA RTX 3060 Ti GPU. All methods were evaluated using the same hardware configuration and experimental pipeline to ensure a fair comparison.

D Additional Experiments

This appendix provides supplementary experimental studies that further analyze the behavior and robustness of AURA beyond the main-paper results of Section 5. These experiments are designed to provide deeper insight into the properties of the proposed latent Bayesian adaptation framework, including its sensitivity to latent dimensionality, robustness to train-test mismatch, and dependence on the offline meta-learning stage. We report additional studies for both application domains considered in this work: wireless receiver adaptation and non-stationary image classification. Together, these results further validate the generality, robustness, and practical design trade-offs of the proposed approach.

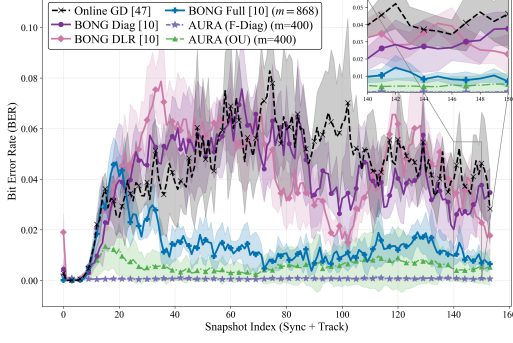
D.1 Additional Wireless Communication Experiments

D.1.1 Temporal Tracking Behavior

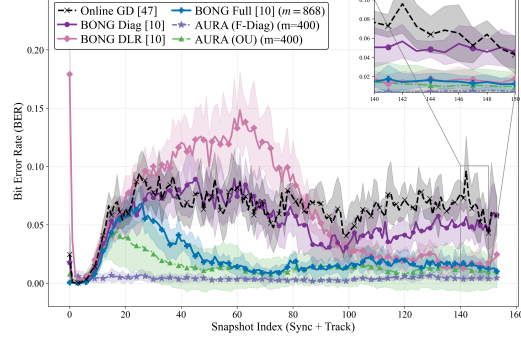
We complement the BER-SNR results in Section 5.2 by reporting the temporal tracking behavior over the online deployment horizon. Fig. 5 shows the BER as a function of the snapshot index for the linear and nonlinear QuaDRiGa channels. The results indicate that AURA maintains stable tracking throughout the channel evolution, supporting the aggregate gains reported in 5.

D.1.2 Effect of Latent Compression and Dynamics Parameterization

While Subsection 5.1 studies the effect of the latent dimension m in the non-stationary image classification setting, we here provide a complementary analysis in the wireless receiver domain that additionally examines the interaction between latent compression and the choice of temporal dynamics model. Specifically, we assess how the dimensionality of the learned latent adaptation space affects performance under two alternative state evolution parameterizations: (i) OU-based dynamics, corresponding to the scaled-identity transition model $\mathbf{F} = \gamma \mathbf{I}$, and (ii) fully learned linear dynamics, where $\mathbf{F}$ is meta-learned from data. To this end, we vary the latent compression ratio used by AURA, corresponding to different latent dimensions m , and evaluate the resulting online adaptation



(a) Linear channel.



(b) Nonlinear channel.

Figure 5: BER vs. Snapshot Index

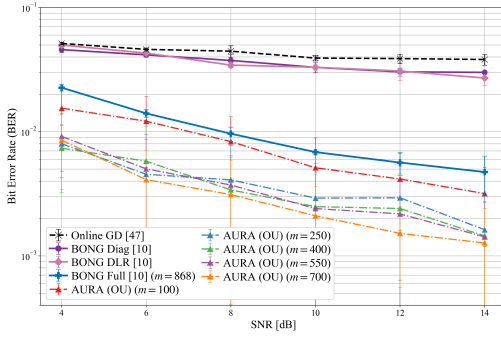


Figure 6: OU Ablation: BER vs. SNR.

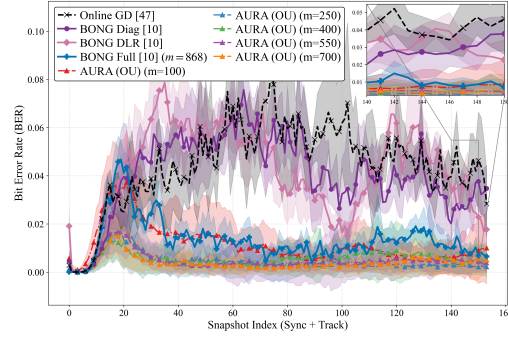


Figure 7: OU Ablation: BER vs. Snapshot Index.

performance under the linear QuaDRiGa channel model. This experiment aims to characterize the trade-off between latent compression, adaptation fidelity, and temporal modeling flexibility.

The results in Figs. 6–9 reveal several consistent trends. First, across all tested compression levels and for both temporal dynamics models, AURA maintains a clear advantage over the corresponding full-parameter Bayesian baselines. This indicates that the gains of the proposed latent Bayesian adaptation framework do not rely on a narrowly tuned latent dimensionality, but persist across a broad range of compression levels. Second, the results exhibit the expected compression–performance trade-off: increasing the latent dimension consistently improves detection accuracy, both in aggregate BER across the SNR sweep and in temporal BER throughout the adaptation trajectory. This behavior reflects the fact that larger latent spaces preserve a richer set of adaptation-relevant degrees of freedom, whereas excessive compression limits the representational capacity of the latent adaptation mechanism. Comparing the two temporal dynamics parameterizations reveals that the learned $\mathbf{F}$ -based dynamics generally achieve stronger performance than the OU-based model when sufficient latent capacity is available, owing to their increased flexibility in modeling temporal evolution. However, both parameterizations display similar qualitative compression trends, suggesting that the benefits of latent Bayesian adaptation are robust to the particular state evolution model employed.

D.1.3 Cross-Dataset Generalization

The wireless communication experiments in Section 5 evaluate AURA under matched train-test conditions, where both offline meta-training and online deployment are conducted using channel trajectories generated by the QuaDRiGa simulator. While this setting demonstrates the effectiveness of the proposed framework under realistic non-stationary fading, it does not assess the robustness of the learned latent adaptation space to distributional mismatch between offline and online environments. To study this aspect, we consider a cross-dataset transfer setting in which AURA is meta-trained offline on QuaDRiGa-generated channel trajectories and then deployed online on channels generated by COST2100 [88] (an alternative standardized stochastic wireless channel generator). This experiment

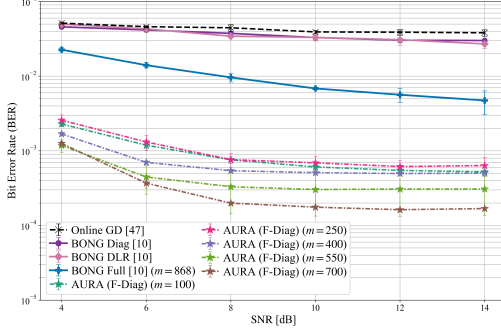


Figure 8: F -Diagonal Ablation: BER vs. SNR.

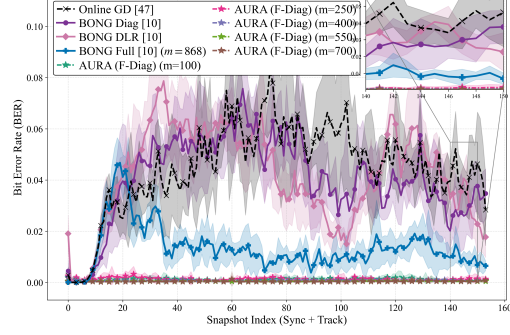


Figure 9: F -Diagonal Ablation: BER vs. Snapshot Index.

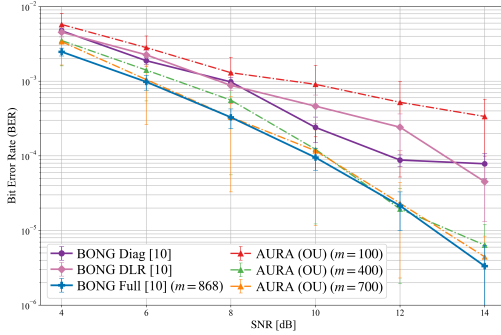


Figure 10: Cross-learning evaluation: BER vs. SNR.

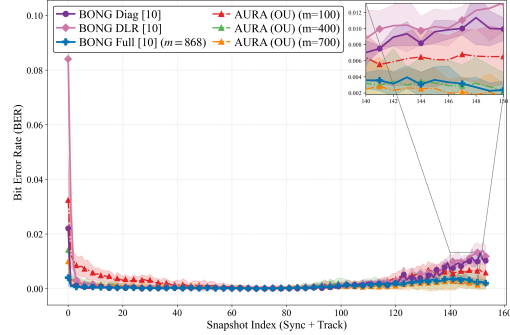


Figure 11: Cross-learning evaluation: BER vs. Snapshot Index.

evaluates whether the learned latent adaptation space captures transferable adaptation structure beyond the statistics of a single simulator.

The results reported in Figs. 10 and 11 show that AURA remains effective under a pronounced train-test mismatch. In this cross-simulator setting, AURA achieves BER performance that is broadly comparable to the leading full-parameter Bayesian baselines, while performing the online update in a substantially compressed latent space. Thus, although it does not provide a uniform BER advantage over the strongest full-space methods, the results indicate that the learned latent representation captures transferable adaptation structure beyond the simulator used for offline meta-training.

D.1.4 Effect of Offline Meta-Learning.

We further isolate the contribution of the offline meta-learning stage in the wireless receiver setting. To this end, we compare the full AURA framework against a cold-start variant in which the same latent dimensionality and prediction model are used, but the latent representation is not learned offline. This comparison is carried out for both OU and diagonal- F prediction dynamics, while keeping the receiver architecture and online Bayesian update protocol fixed. Table 2 reports the tracking BER for two SNR values and different latent dimensions m , where each warm-start/cold-start pair uses the same prediction model and latent dimension.

The results demonstrate that offline meta-learning is critical for reliable online tracking in the considered non-stationary communication scenario. Without offline learning, the latent-space update remains ineffective and yields high BER across prediction models and latent dimensions. In contrast, the warm-start AURA variants consistently reduce the BER by orders of magnitude under the same online adaptation budget. This confirms that the observed gains do not stem merely from restricting the update to a lower-dimensional parameterization, but from learning an adaptation-aware latent geometry that aligns the online Bayesian updates with the dominant channel-induced variations.

Table 2: Effect of offline learning in the QuaDRiGa communication scenario. Each pair compares the same prediction method and latent dimension with and without offline learning. BER in (%)

Prediction method	Latent dim.	Mode	SNR = 8 dB	SNR = 10 dB
OU	$m = 100$	warm-start	0.832 ± 0.485	0.512 ± 0.379
		cold-start	14.399 ± 3.024	11.542 ± 1.054
OU	$m = 250$	warm-start	0.409 ± 0.514	0.292 ± 0.418
		cold-start	14.662 ± 1.541	12.225 ± 1.259
OU	$m = 400$	warm-start	0.340 ± 0.496	0.249 ± 0.353
		cold-start	14.828 ± 2.880	12.320 ± 0.643
OU	$m = 700$	warm-start	0.311 ± 0.367	0.210 ± 0.152
		cold-start	14.870 ± 1.626	12.980 ± 1.487
F-Diag	$m = 100$	warm-start	0.0757 ± 0.0079	0.0608 ± 0.0045
		cold-start	18.471 ± 3.024	17.464 ± 1.622
F-Diag	$m = 250$	warm-start	0.0763 ± 0.0160	0.0690 ± 0.0148
		cold-start	18.677 ± 1.541	17.508 ± 1.810
F-Diag	$m = 400$	warm-start	0.0542 ± 0.0014	0.0510 ± 0.0010
		cold-start	18.321 ± 2.880	16.562 ± 0.670
F-Diag	$m = 700$	warm-start	0.0199 ± 0.0009	0.0176 ± 0.0004
		cold-start	18.225 ± 1.626	17.027 ± 1.815

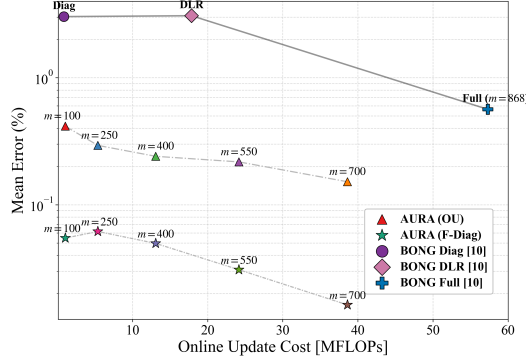


Figure 12: FLOPs-error trade-off on the linear QuaDRiGa channel at SNR = 12 dB

D.1.5 Computational Cost versus Adaptation Accuracy.

We complement the accuracy evaluation with a direct cost–accuracy analysis of the different adaptation rules. Specifically, we compare the FLOPs required by each method against the resulting mean error at a fixed operating point of SNR = 12 dB. The error is measured on the linear QuaDRiGa channel, while the FLOPs are obtained using a dedicated cost-evaluation script tailored to the operations of each adaptation rule. Diagonal covariance variants are evaluated using an element-wise implementation that exploits their structure, and BONG-DLR is evaluated with rank 30, as used in the reported experiments in 3 and 4.

Figure 12 reports the resulting trade-off. The AURA variants occupy a favorable region of the cost–accuracy plane, achieving substantially lower error than the full-space BONG baseline while requiring fewer FLOPs. Increasing the latent dimension provides a smooth trade-off between computational budget and adaptation quality. The diagonal- F variant achieves the lowest error among the AURA configurations, with a moderate increase in cost relative to the OU dynamics. In contrast, the diagonal and DLR BONG approximations reduce the cost of full-space Bayesian tracking but remain substantially less accurate in this setting. These results indicate that adapting in a learned latent space preserves the relevant tracking directions while avoiding the computational burden of full-parameter Bayesian updates.

Table 3: BER under intermittent supervision (% , mean $\pm$ standard deviation over three seeds).

I	K	AURA	BONG Full	BONG Diag
1	2	1.04 $\pm$ 0.21	12.09 $\pm$ 1.32	17.02 $\pm$ 1.48
1	4	0.10 $\pm$ 0.02	3.15 $\pm$ 0.35	8.54 $\pm$ 0.83
1	6	0.07 $\pm$ 0.01	1.11 $\pm$ 0.04	4.14 $\pm$ 0.23
2	2	6.19 $\pm$ 0.61	25.00 $\pm$ 1.25	29.38 $\pm$ 0.65
2	4	0.70 $\pm$ 0.07	12.17 $\pm$ 0.38	17.56 $\pm$ 0.68
2	6	0.20 $\pm$ 0.07	6.11 $\pm$ 0.17	12.18 $\pm$ 0.28
3	2	13.16 $\pm$ 1.91	34.03 $\pm$ 1.66	37.60 $\pm$ 1.22
3	4	3.07 $\pm$ 0.37	19.55 $\pm$ 1.17	24.12 $\pm$ 2.07
3	6	1.04 $\pm$ 0.15	12.52 $\pm$ 1.20	18.46 $\pm$ 1.88
4	2	18.31 $\pm$ 1.25	39.25 $\pm$ 1.55	39.29 $\pm$ 0.38
4	4	5.28 $\pm$ 0.27	25.21 $\pm$ 0.40	29.97 $\pm$ 0.88
4	6	1.99 $\pm$ 0.19	17.93 $\pm$ 0.63	22.11 $\pm$ 0.94
5	2	24.00 $\pm$ 0.88	42.47 $\pm$ 0.46	43.21 $\pm$ 1.26
5	4	10.60 $\pm$ 1.08	29.15 $\pm$ 0.39	34.20 $\pm$ 1.47
5	6	4.54 $\pm$ 0.96	22.04 $\pm$ 0.43	27.48 $\pm$ 2.05

D.1.6 Intermittent Supervision

The wireless experiments in Section 5.2 assume that a small number of labeled pilot symbols is available at every frame. We next examine a more restrictive setting in which reliable supervision is available only intermittently. Specifically, we provide K labeled pilots once every I frames, with $I \in \{1, \dots, 5\}$ and $K \in \{2, 4, 6\}$. At supervised frames, the receiver performs the standard prediction–correction update using the available pilots. At intervening unlabeled frames, the correction step in (6) is omitted, while the latent mean and covariance continue to evolve according to the prediction step in (5). Importantly, the channel realization continues to vary at every frame; hence, increasing I requires the receiver to track multiple successive channel variations without receiving an observation correction.

Table 3 reports the resulting BER in % over all 15 combinations of supervision interval and pilot budget. The results reveal two consistent trends. First, performance degrades as supervision becomes less frequent or fewer pilots are available, as expected when the receiver must rely more heavily on prediction between observation corrections. Second, AURA achieves the lowest BER in all 15 evaluated configurations, with its advantage over parameter-space Bayesian tracking persisting as supervision becomes increasingly sparse.

These findings show that AURA does not require an observation correction after every channel variation. The learned latent dynamics can propagate the adaptation state across several distribution changes, while subsequent labeled observations correct accumulated prediction errors. At the same time, the degradation observed for longer supervision intervals highlights the growing dependence on the accuracy of the learned dynamics. We therefore interpret these results as evidence that AURA remains effective under moderate intermittent supervision, rather than as evidence of stability over arbitrarily long periods without feedback.

D.1.7 Full Transition Matrix Ablation.

We additionally evaluate an ablation in which the latent dynamics are parameterized by a fully learned transition matrix $\mathbf{F}$, using the same linear QuadRiGa setting considered in (40). This variant provides a more expressive temporal model than the OU and diagonal- $\mathbf{F}$ parameterizations, since it allows arbitrary linear coupling between latent coordinates. As shown in Fig. 13, the full- $\mathbf{F}$ model can achieve strong tracking performance, particularly for sufficiently large latent dimensions.

However, this expressiveness comes with a substantially higher computational cost. In particular, the covariance prediction step in (5) involves the multiplication $\mathbf{F}\Sigma\mathbf{F}^\top$, whose complexity scales as $\mathcal{O}(m^3)$ for a dense transition matrix and covariance. Consequently, although the full- $\mathbf{F}$ variant is informative as an ablation and can be competitive in terms of BER, it is only attractive from a runtime perspective for small latent dimensions, roughly $m \leq 250$ in our implementation. For larger latent states, the cubic prediction cost becomes inconsistent with the goal of fast online adaptation. This moti-

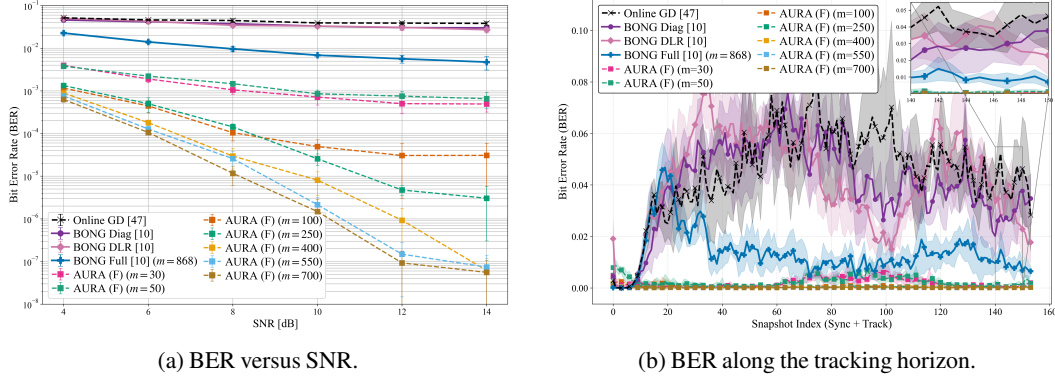


Figure 13: Ablation of full- $\mathbf{F}$ latent dynamics on linear QuaDRiGa channel.

Table 4: Effect of the lifting-map architecture on classification error (% , mean $\pm$ standard deviation over three seeds).

Dataset	# Labeled Samples	Affine	Nonlinear
CIFAR-10-C	5	16.13 $\pm$ 1.50	10.21 $\pm$ 0.40
	10	16.84 $\pm$ 0.51	10.15 $\pm$ 0.81
	15	14.28 $\pm$ 2.87	9.75 $\pm$ 0.54
CIFAR-100-C	5	29.47 $\pm$ 0.62	26.34 $\pm$ 1.77
	10	29.65 $\pm$ 1.97	25.21 $\pm$ 1.04
	15	26.66 $\pm$ 3.89	23.00 $\pm$ 0.79

vates our focus in the main experiments on the OU and diagonal- $\mathbf{F}$ dynamics, which retain a structured temporal prior while reducing the covariance prediction cost relative to a full transition matrix.

D.2 Additional Non-stationary Image Classification Experiments

D.2.1 Lifting Map Architecture

A central component of AURA is the learned lifting map $\mathcal{G} : \mathbb{R}^m \rightarrow \mathbb{R}^d$, which determines how the low-dimensional latent state is mapped to the adapted model parameters. We next examine whether the effectiveness of the learned adaptation space depends on restricting $\mathcal{G}$ to an affine transformation, or whether additional nonlinear expressiveness is beneficial.

We compare two lifting-map parameterizations while keeping the latent dimension and online adaptation protocol fixed. The affine variant corresponds to

$$\mathcal{G}_{\text{aff}}(\mathbf{z}) = \phi + \mathbf{A}\mathbf{z},$$

where ϕ denotes the source-model parameters and $\mathbf{A}$ is learned during offline meta-training. We additionally consider a residual nonlinear lifting map of the form

$$\mathcal{G}_{\text{nonlin}}(\mathbf{z}) = \phi + \mathbf{A}\mathbf{z} + \psi(\mathbf{z}),$$

where $\psi(\cdot)$ is a learned MLP with a single hidden layer of width 500 and GELU activation. Both variants are subsequently meta-learned end-to-end through 2

Table 4 reports the classification error on CIFAR-10-C and CIFAR-100-C for different numbers of labeled adaptation samples. The results highlight the importance of both learning and appropriately parameterizing the lifting map. As shown by the communication ablation in Table 2, merely restricting adaptation to a low-dimensional parameterization is not sufficient: without offline learning of the latent representation, Bayesian tracking fails to provide effective adaptation. Once the lifting geometry is learned, however, even an affine map yields strong performance. Increasing the expressiveness of $\mathcal{G}(\cdot)$ through a nonlinear parameterization further improves accuracy consistently across both CIFAR-10-C and CIFAR-100-C and across all considered supervision budgets. These results suggest that the learned latent geometry captures the dominant adaptation directions, while nonlinear lifting provides additional flexibility for representing the relation between the latent state and the evolving model parameters.

Table 5: Cross-corruption generalization under mixed-gradual CIFAR-10.

Method	ROID	AURA	BN-EMA	BN-Test	EKF-FC	BN-Alpha	Source	Online GD
Classification Error (%)	9.88 ± 1.01	9.99 ± 1.30	11.50 ± 2.64	11.62 ± 1.29	12.31 ± 1.02	13.55 ± 2.47	17.07 ± 2.67	16.21 ± 2.04

Table 6: Classification error (% , mean $\pm$ standard deviation over three seeds) for LoRA and AURA.

Dataset	LoRA	AURA, Affine	AURA, Nonlinear
CIFAR-10-C	15.73 ± 1.23	14.28 ± 2.87	9.75 ± 0.54
CIFAR-100-C	28.90 ± 1.63	26.66 ± 3.89	23.00 ± 0.79

This gain in expressiveness introduces a corresponding computational trade-off. Each EKF correction requires evaluating the observation Jacobian and therefore differentiating through $\mathcal{G}(\cdot)$, while parameter reconstruction additionally requires its forward evaluation. Consequently, deeper or more expressive lifting maps increase both forward and backward computational costs during online adaptation. The lifting-map architecture therefore provides a direct accuracy-latency design choice: affine mappings offer a simpler and more efficient realization, whereas nonlinear mappings can provide improved adaptation accuracy at higher computational cost. This overhead could potentially be reduced through dedicated hardware or compilation and optimized execution of the lifting and Jacobian computations; such implementation-level optimizations are not investigated in this work.

D.2.2 Cross-Dataset Generalization

The non-stationary image classification experiments in Section 5 evaluate AURA under gradual corruption streams generated from the same corruption pool used during offline meta-training. Although the offline and online phases remain distinct due to different temporal mixing trajectories, this setting does not test whether the learned latent adaptation space generalizes beyond the corruption families observed during training. To assess this aspect, we introduce an explicit corruption-family split between offline and online phases. Specifically, AURA is meta-trained offline using one subset of corruption types and subsequently deployed online on a disjoint set of previously unseen corruptions. This setting evaluates whether the learned latent adaptation mechanism captures transferable adaptation structure beyond the specific degradation families encountered during offline training.

The results in Table 5 show that AURA remains highly effective under this transfer setting, achieving strong performance while remaining competitive with the best-performing baseline and clearly outperforming the remaining alternatives. This indicates that the learned latent adaptation space captures adaptation-relevant structure that transfers across distinct corruption families and temporal drift patterns. These findings complement the main-paper image classification results by demonstrating that the adaptation mechanisms learned by AURA generalize beyond the precise distribution shifts observed during offline meta-training, further supporting the robustness and transferability of the proposed meta-learned latent adaptation framework.

D.2.3 Comparison with Efficient Adaptation Methods

A central motivation of AURA is to make rapid online adaptation computationally tractable by operating in a compact learned representation. We therefore compare against two complementary classes of efficient adaptation methods. First, we consider LoRA [78] as a parameter-efficient supervised adaptation baseline, asking whether restricting updates to a compact low-rank parameterization can reproduce the gains of the learned latent representation. Second, we compare against the efficiency-oriented TTA methods EcoTTA [89] and ELaTTA [90], which operate without ground-truth supervision and explicitly target low-cost deployment.

Comparison with LoRA. We evaluate LoRA [78] under the same supervised streaming protocol as AURA. Following a hyperparameter sweep, we use rank $r = 8$ and adapt the classification head together with selected convolutional layers: the first convolutional stage for CIFAR-10-C and the second stage for CIFAR-100-C. We compare LoRA with both affine and nonlinear realizations of the learned lifting map.

The comparison with LoRA isolates the effect of the adaptation parameterization under supervised streaming conditions. Both AURA variants improve upon LoRA, while the nonlinear lifting map provides the largest gain. This indicates that restricting adaptation to a compact parameterization

Table 7: Classification error (%) and adaptation latency for efficient test-time adaptation methods.

Method	Supervision	CIFAR-10-C	CIFAR-100-C	Latency (ms/sample) for CIFAR-100-C)
Source	None	41.81	37.41	–
AURA	5 labeled	16.13	29.47	2.01
EcoTTA [89]	Unlabeled	20.18	41.20	1.15
ELaTTA [90]	Unlabeled	44.82	39.98	0.41

Table 8: Recovery from abrupt distribution shifts on CIFAR-10-C. Mean classification error (%) across five corruption shifts.

Lifting map	After shift	10 updates	20 updates	40 updates
Affine	45.96	27.68	21.74	19.44
Nonlinear	42.44	23.30	19.05	17.66

alone does not account for the performance of AURA; learning the adaptation representation from non-stationary trajectories provides an additional benefit.

Comparison with Efficient TTA. We further compare against EcoTTA [89] and ELaTTA [90] on the gradual mixed-corruption streams used in the main image-classification experiments. These methods retain their respective unsupervised adaptation procedures, whereas AURA uses five labeled samples per update. The comparison therefore reflects distinct supervision regimes and is intended to characterize the resulting accuracy–efficiency trade-off rather than provide a supervision-matched benchmark.

EcoTTA and ELaTTA avoid labeled feedback and achieve lower adaptation latency, but incur higher classification error on the considered streams. EcoTTA remains competitive on CIFAR-10-C but does not improve over the source model on CIFAR-100-C, while ELaTTA does not improve upon the source model in either setting. These results place AURA at a different operating point in the supervision–accuracy–efficiency trade-off: it uses sparse reliable supervision to obtain substantially stronger adaptation accuracy while retaining low online computational cost.

D.2.4 Robustness to Abrupt Distribution Shifts

The main image-classification experiments consider gradually evolving distribution shifts, which are naturally aligned with the temporal model learned by AURA. We therefore evaluate whether this learned dynamical prior remains useful when the deployment distribution changes abruptly, despite offline meta-training being performed exclusively on gradual mixed-corruption trajectories.

We evaluate CIFAR-10-C at severity 5 under five abrupt corruption shifts: pixelation, contrast, zoom blur, fog, and JPEG compression. Following each shift, the new corruption is maintained for 40 online update steps, with 15 labeled adaptation samples available at each step. Performance is measured immediately after the transition and after 10, 20, and 40 updates. We evaluate both affine and nonlinear lifting maps using the same latent dimension, $m = 500$.

As shown in Table 8, both variants exhibit substantial recovery following the abrupt transition, despite the mismatch between the deployment dynamics and those used for offline meta-training. The nonlinear lifting map consistently attains lower error throughout the recovery horizon, while both parameterizations reduce the initial post-shift error by approximately 58% after 40 updates.

These results indicate that the learned temporal dynamics act as a predictive prior rather than imposing a rigid smoothness constraint on the adaptation trajectory. When an abrupt shift invalidates the prediction, subsequent innovation-based EKF corrections can progressively move the latent state toward the new regime. Thus, while gradual variation remains the setting most naturally matched to AURA, the learned latent dynamics do not prevent effective recovery from abrupt distribution changes.

E Proof-of-Concept: Adaptive Deep Software-Defined Radio

To complement the simulation-based results in Section 5, we provide a proof-of-concept hardware demonstration validating the practical deployability of AURA in a real-time communication system.

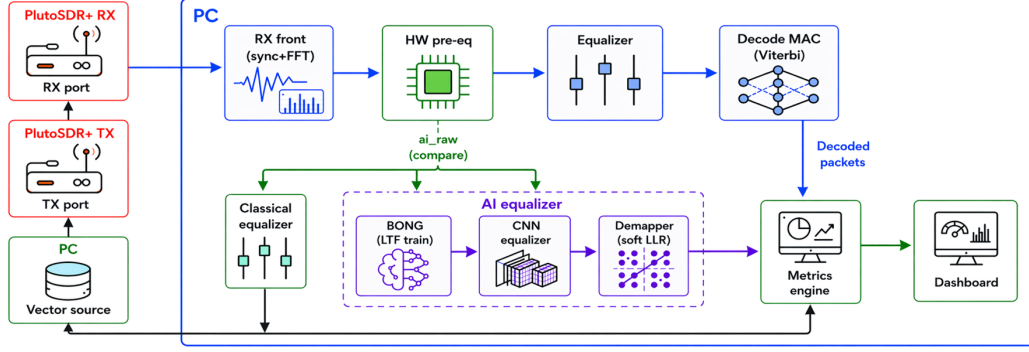


Figure 14: Hardware proof-of-concept setup for the self-adaptive neural WiFi receiver. The adaptive receiver is deployed inline with a conventional classical receiver within an over-the-air IEEE 802.11a communication link implemented using PlutoSDR+ RF front-ends, enabling simultaneous evaluation under identical channel realizations.

Specifically, we implement a self-adaptive neural WiFi receiver using AURA-style online adaptation within an over-the-air IEEE 802.11a communication link realized on software-defined radio hardware. This demonstration is intended to assess whether the computational efficiency of the proposed latent Bayesian adaptation framework translates into practical millisecond-order online adaptation under realistic hardware and latency constraints. As shown below, the resulting prototype confirms that AURA enables real-time adaptive neural reception on commodity RF platforms using CPU-only inference and adaptation.

E.1 Hardware Setup and Adaptive Receiver Architecture

The proof-of-concept system implements an over-the-air IEEE 802.11a OFDM WiFi link using PlutoSDR+ devices as RF front-ends. The deployed communication setup follows a standard packetized WiFi transmission structure with 64 OFDM subcarriers, of which 52 are active (48 data and 4 pilot subcarriers). We consider a block-fading regime in which the wireless channel is approximately constant within each packet but varies continuously across packets, inducing the non-stationary adaptation setting targeted by AURA.

A block diagram of the experimental platform is shown in Fig. 14. The adaptive receiver consists of a lightweight learnable 1D-CNN equalizer containing approximately 1.8K trainable parameters, followed by an analytical modulation-aware demapper that produces soft bit log-likelihood ratios for downstream decoding. Online adaptation is performed using the known WiFi preamble (long-training-field symbols) available at the beginning of each packet, enabling the receiver to refine its parameters prior to processing the data-bearing payload symbols.

To enable controlled comparison, the adaptive neural receiver is executed inline alongside a conventional classical receiver based on LMMSE equalization, with both receivers processing the same received packets simultaneously. This allows direct measurement of the corresponding bit-error rates under identical channel realizations. Importantly, the online adaptation pipeline is implemented entirely in NumPy and executed on CPU without GPU acceleration.

E.2 Hardware Performance and Runtime Characterization

The resulting hardware performance is illustrated in Fig. 15, which compares the measured bit-error rate of the adaptive neural receiver against that of the conventional classical receiver across varying SNR and carrier frequency offset (CFO) conditions. The results show that augmenting the receiver with online neural adaptation substantially enlarges the reliable operating regime, particularly under challenging low-SNR and high-CFO conditions where the classical receiver begins to degrade significantly. In several operating regions, the adaptive neural receiver achieves markedly improved detection performance, demonstrating that the learned adaptive equalization mechanism provides tangible robustness benefits in real over-the-air deployments.

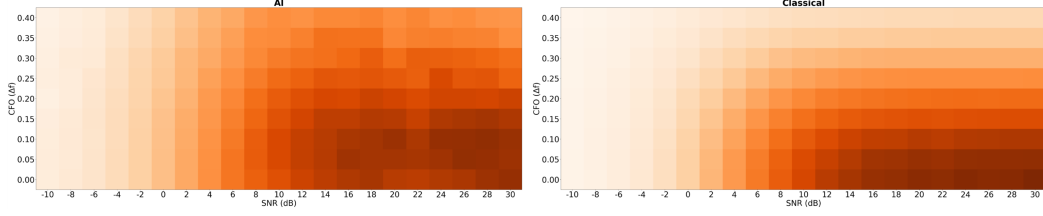


Figure 15: Average BER heatmaps versus SNR and CFO level for neural receiver adapted with AURA (left) compared to classical channel estimation + LMMSE equalization receiver (right)

Table 9: Measured per-packet runtime of the adaptive neural receiver for different CNN equalizer architectures. All runtimes are reported in milliseconds and measured using CPU-only execution.

Architecture	Inference Runtime [ms]	Adaptation Runtime [ms]
1.8K-Parameter CNN	0.084	10.573
6.6k-Parameter CNN	0.113	40.414
8.2k-Parameter CNN	0.221	84.393

To assess deployment feasibility, Table 9 reports the measured per-packet runtime of the adaptive receiver pipeline for multiple neural equalizer architectures of increasing complexity. Specifically, we report both inference latency and online adaptation latency for the lightweight 1.8K-parameter CNN used in the proof-of-concept, as well as for two deeper CNN variants. All runtimes are measured on CPU without GPU acceleration.

The measured runtimes confirm that AURA-style online adaptation remains feasible within millisecond-order latency budgets even when deployed on modest CPU hardware. In particular, the lightweight neural receiver used in the proof-of-concept performs full online adaptation in tens of milliseconds without hardware acceleration, while larger models remain deployable at increased but still practical latency. These findings validate that the computational structure underlying AURA enables real-world deployment under realistic hardware constraints, and suggest that further latency reductions are attainable through dedicated optimized implementations or edge-accelerated inference.

F Limitations

While AURA provides an efficient framework for Bayesian online adaptation via latent-space filtering, several limitations should be acknowledged:

- Although AURA substantially reduces the dimensionality of the online adaptation problem, scaling to extremely large models or broad parameter blocks remains challenging. Compressing very high-dimensional parameter spaces into a single latent state introduces memory and optimization bottlenecks, and requires balancing latent compactness against sufficient expressiveness for accurate parameter reconstruction and adaptation.
- The effectiveness of AURA depends not only on learning a suitable latent representation, but also on accurately modeling the temporal evolution of the latent state. Since the prediction step propagates the latent estimate according to an assumed or learned transition model, performance may degrade when deployment-time dynamics differ substantially from those observed during offline training. While the correction step can compensate using new observations, inaccurate predictions may reduce adaptation quality and stability.
- The proposed state-space formulation is naturally best suited for gradually varying environments, where the previous latent state provides an informative prior for the next update. Under abrupt distribution shifts or regime changes, the prediction model may become temporarily misleading, potentially requiring complementary mechanisms such as change-point detection, covariance inflation, or latent-state reset strategies.
- In its current form, AURA assumes access to labeled samples during online adaptation. While this is natural in several streaming settings (e.g., pilot-aided communications), it may limit applicability in fully unsupervised deployment scenarios. Extending the framework to

unlabeled adaptation via pseudo-labeling or self-supervised objectives is possible in principle, but may introduce sensitivity to label noise and error accumulation.

- **Approximate Filtering Assumptions.** The online adaptation rule relies on first-order linearization of the nonlinear observation model and Gaussian approximations inherent to the EKF. As with other Kalman-style methods, significant deviations from these assumptions may reduce the fidelity of the Bayesian update approximation.

These limitations suggest that AURA is most suitable for applications involving structured and moderately smooth non-stationarity with available online supervision, while improving robustness to abrupt shifts, unsupervised deployment, and larger-scale models remains an important direction for future work.