Asynchronous Unsupervised Online Learning of Bayesian Deep Receivers

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

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

Abstract

Deep learning-aided receivers have shown promising performance in challenging channels, particularly when allowed to adapt online, which leads to notable computational burden. To reduce this burden, we propose an *unsupervised asynchronous online learning* framework based on Bayesian deep learning. Our approach leverages uncertainty estimates inherently produced by Bayesian neural networks to construct lightweight statistical tests that monitor the receiver, without requiring access to transmitted labels, that autonomously trigger retraining only when the receiver is no longer adequate. This results in a receiver that adapts efficiently while significantly reducing retraining frequency. Our numerical studies demonstrate that the proposed framework enables timely adaptation while achieving symbol error rates comparable to costly synchronous retraining at every coherence block.

1 Introduction

The growing demand for high-throughput wireless connectivity has led to an interest in integrating deep learning tools into the receiver processing chain (1). Such *deep receivers* are data-driven architectures that learn to reliably decode signals in environments where classical model-based techniques struggle (2). However, unlike traditional deep learning domains such as vision or language processing, wireless receivers must operate in highly dynamic environments and under strict latency and energy constraints, posing fundamental challenges on deployment of deep receivers (3).

To tackle these challenges, several approaches have been proposed to enable pre-trained deep receivers to maintain reliable performance under channel variations. A straight-forward approach is joint learning (4), where a deep neural network (DNN) is trained over a large set of measured or simulated channel conditions (5; 6) to form a non-coherent receiver, at the cost of degraded performance compared to channel-specific training and while requiring complex architectures. Another strategy augments the input with estimated channel features (7; 8), typically relying on linear channel models. More recently, hypernetworks were proposed to provide adaptation to some forms of channel variations (9; 10). However, pre-trained deep receivers are often restricted to complex DNNs, and typically lack the sample-efficiency and flexibility required for adaptive operation.

An alternative for addressing temporal variability is *online learning*, where the receiver continuously adapts its internal parameters. Online learning enables deep receivers to stay aligned with the current channel conditions, and has been shown to yield notable performance improvements when retraining occurs at each coherence interval (11; 12). However, due to the high computational and latency overhead associated with frequent retraining, numerous efforts have focused on reducing its burden. A popular approach designs deep receivers by converting model-based receivers into machine learning (ML) models (13; 14), obtaining compact and structured models that are more

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI and ML for Next-Generation Wireless Communications and Networking (AI4NextG).

amenable to rapid adaptation with limited data (15; 16; 17). Alternative approaches generate data for online learning via self-supervision (18) and data augmentation (19; 20), or design enhanced learning methodologies using meta-learning (21; 22) and Bayesian learning (23; 24; 25), where the latter was recently shown to enable online learning with notable latency reduction (26). Still, as deep receivers often generalize reasonably well to mild channel variations (27), it is inefficient to retrain after every block. This has motivated the design of asynchronous online learning schemes (28), where retraining is triggered only when the receiver is no longer adequate. However, these schemes require labeled data, which typically exists only for specific pilot transmissions, motivating the development of *unsupervised* asynchronous learning techniques, capable of autonomously initiate on-device retraining.

In this work, we focus on deep receivers based on *Bayesian deep learning*. While prior works on Bayesian learning for receivers (23; 25; 24; 26) considered improving soft-symbol estimation through calibration (24), learning from scarce data via Bayesian prior regularization (23; 25), and continual adaptation using Bayesian tracking (26), we propose to harness a different yet complementary aspect of Bayesian learning: its ability to provide statistical uncertainty estimates as means of enabling *unsupervised asynchronous online learning*. We propose mechanisms that monitor the uncertainty levels of Bayesian neural network to autonomously detect when the receiver becomes outdated, triggering retraining only when necessary. We adopt the ML framework of concept drift detection (29), and extend it to the Bayesian setting by designing tests that operate directly on the posterior uncertainty measures of Bayesian deep receivers (30). Our detectors exploit the uncertainty measures to identify drift without requiring transmitted pilots. This allows deep receivers to be adapted online while minimizing retraining frequency. Through a numerical study using dynamic channels modeled by COST2100 (31) and QuadRiGa (32), we demonstrate that the proposed uncertainty-aware drift detection framework can reliably identify when adaptation is necessary and lead to timely retraining.

2 System Model and Problem Formulation

Channel Model We consider a discrete-time block-fading communication model. Each block indexed by t spans B time instances, during which the channel parameters, denoted by $\boldsymbol{h}[t]$, remain fixed. At time index i within block t, the transmitter sends a symbol $\boldsymbol{s}_i[t] \in \mathcal{S}$ drawn from a finite constellation $\mathcal{S}$. The corresponding channel output is denoted $\boldsymbol{y}_i[t] \in \mathcal{Y}$, and the received block is collected as $\boldsymbol{y}^{\mathrm{rec}}[t] := \{\boldsymbol{y}_i[t]\}_{i=1}^{B}$. The channel behavior is modeled by a conditional distribution governed by the current channel state $\boldsymbol{h}[t]$, such that

$$\boldsymbol{y}_i[t] \sim P_{\boldsymbol{h}[t]}(\boldsymbol{y}_i[t]|\boldsymbol{s}_i[t]). \tag{1}$$

This formulation accommodates a broad class of channels, including both linear and nonlinear impairments, where $P_{\boldsymbol{h}[t]}(\cdot|\cdot)$ represents a (possibly unknown) parametric family that characterizes the stochastic relationship between the transmitted and received signals under the channel $\boldsymbol{h}[t]$.

Bayesian Deep Receivers Receivers based on *Bayesian deep learning* (23; 25; 24; 26) incorporate probabilistic modeling into neural network-based inference. In this framework, the receiver is aided by a DNN, whose weights, denoted by $\boldsymbol{\varphi}[t]$, are a random vector with distribution $q_{\boldsymbol{\theta}[t]}$, parameterized by $\boldsymbol{\theta}$. Such parameterized distribution is obtained from an explicit formulation, as in variational inference (33), or via alternative induced trainable stochasticity, as in Monte Carlo dropout (34).

For a given realization of $\boldsymbol{\varphi}[t]$, the DNN-aided receiver maps a channel output $\boldsymbol{y}$ into an estimate of the conditional distribution of $\boldsymbol{s}$, denoted $\hat{P}_{\boldsymbol{\varphi}[t]}(\boldsymbol{s}|\boldsymbol{y})$. Specifically, inference is carried using Monte-Carlo sampling, where J i.i.d. realizations $\boldsymbol{\varphi}_j[t] \sim q_{\boldsymbol{\theta}[t]}$ are generated, and the soft estimate is obtained as

$$\hat{P}_{q_{\boldsymbol{\theta}[t]}}(\boldsymbol{s}|\boldsymbol{y}) = \frac{1}{J}\sum_{j=1}^{J} \hat{P}_{\boldsymbol{\varphi}_j[t]}(\boldsymbol{s}|\boldsymbol{y}). \tag{2}$$

The soft estimate is used to provide a hard estimate denoted $\hat{\boldsymbol{s}}_i[t]$, either by downstream processing (as in, e.g., (11; 35)), or by taking the maximum a-posteriori probability.

Online Learning When applying *online learning* on block t, the receiver updates the distribution parameters $\boldsymbol{\theta}[t]$ using data corresponding to the current block, denoted $\mathcal{Q}[t]$. This block represents a set of channel outputs and their corresponding inputs, obtained from pilots or via self-supervision (18).

Training the Bayesian deep receiver is based on the evidence lower bound (ELBO) loss, given by

$$\mathcal{L}_{\mathcal{Q}[t]}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{Q}[t]|} \sum_{i=1}^{|\mathcal{Q}[t]|} \mathbb{E}_{\boldsymbol{\varphi}\sim q_{\boldsymbol{\theta}}} \left[\log \hat{P}_{\boldsymbol{\varphi}}\left(\boldsymbol{s}_i|\boldsymbol{y}_i\right)\right] - \mathrm{KL}(q_{\boldsymbol{\theta}}(\boldsymbol{\varphi})||p_t(\boldsymbol{\varphi})), \tag{3}$$

where $\mathrm{KL}(\cdot||\cdot)$ is the Kullback-Leibler (KL) divergence, and $p_t(\cdot)$ is some prior distribution, whose setting can be either fixed or based on the previous distribution (36).

Problem Formulation Conventional online learning of deep receivers typically re-trains at every channel variation. This setting, often referred to as *synchronous online learning*, allows the receiver to track dynamic channel variations in real-time. However, it incurs substantial complexity. To address this challenge, we follow the recently proposed strategy of *asynchronous online learning* (28), where training is triggered only when necessary, i.e., when the receiver's performance degrades.

Formally, we define an indicator $\mathcal{M}^{\mathrm{tr}}[t] \in \{0, 1\}$ that determines whether the receiver trains on block t. Our goal is to set $\mathcal{M}^{\mathrm{tr}}[t]$ based *solely on channel outputs*, namely, based on observations of the form $\mathcal{D}[t] \subset \{\boldsymbol{y}_i[t]\}_{i=1}^{B}$. The channel outputs $\mathcal{D}[t]$ used for setting $\mathcal{M}^{\mathrm{tr}}[t]$ are typically a subset of the overall channel outputs, as, e.g., one may prefer to detect whether to adapt after receiving the first few channel outputs. Such unsupervised monitoring can enable asynchronous online learning in settings with sparse pilots and/or self-supervised adaptation. Over a total horizon of T blocks, the receiver should be adapted so as to minimize the overall symbol error rate (SER), while constraining the number of retraining operations to a fixed budget $C \ll T$, namely

$$\min_{\{\mathcal{M}^{\mathrm{tr}}[t]\}} \quad \frac{1}{T \cdot B} \sum_{t=1}^{T} \sum_{i=1}^{B} \Pr\left(\hat{\boldsymbol{s}}_i[t] \neq \boldsymbol{s}_i[t]\right) \quad \text{subject to} \sum_{t=1}^{T} \mathcal{M}^{\mathrm{tr}}[t] \leq C. \tag{4}$$

3 Unsupervised Bayesian Drift Detection

We henceforth focus on deep receivers based on Bayesian deep learning, and leverage their uncertainty-related features as a basis for *unsupervised drift detection*. The underlying rationale is that, rather than monitoring only for variations in the underlying channel statistics, we identify when the receiver itself becomes inadequate. We do this by exploiting the uncertainty measures inherently produced by Bayesian DNNs (Subsection 3.1), based on which we design statistical tests that autonomously indicate when the receiver is no longer suitable (Subsection 3.2). These tests enable unsupervised asynchronous online learning without labeled data, as formulated in Subsection 3.3 and discussed in Subsection 3.4.

3.1 Uncertainty Features for Drift Detection

A key advantage of Bayesian deep receivers is their ability to quantify predictive uncertainty during inference. Recall from (2) that the receiver estimates the posterior distribution of $\boldsymbol{s}$ given $\boldsymbol{y}$ by averaging over J Monte Carlo samples of the stochastic weights $\boldsymbol{\varphi}_j[t] \sim q_{\boldsymbol{\theta}[t]}$. From these samples, one can extract features that are informative on the DNN uncertainty (30):

Let $H(P(\boldsymbol{s})) = -\sum_{\boldsymbol{s}_c \in \mathcal{S}} P(\boldsymbol{s}_c) \log P(\boldsymbol{s}_c)$ be the entropy operator. The *total predictive uncertainty* is then captured by the entropy of the marginal predictive distribution, which is computed as

$$U_{\mathrm{tot}}(\boldsymbol{y}) = H\Big(\frac{1}{J} \sum_{j=1}^{J} \hat{P}_{\boldsymbol{\varphi}_j[t]}(\boldsymbol{s}|\boldsymbol{y})\Big). \tag{5}$$

The *aleatoric uncertainty*, which captures the intrinsic stochasticity of the wireless channel, is quantified by the expected entropy of the predictive distribution,

$$U_{\mathrm{alea}}(\boldsymbol{y}) = \frac{1}{J} \sum_{j=1}^{J} H\left(\hat{P}_{\boldsymbol{\varphi}_j[t]}(\boldsymbol{s}|\boldsymbol{y})\right). \tag{6}$$

The *epistemic uncertainty*, which reflects uncertainty in the receiver model itself, is obtained from the variability across different Monte Carlo samples, and is expressed as

$$U_{\mathrm{epis}}(\boldsymbol{y}) = U_{\mathrm{tot}}(\boldsymbol{y}) - U_{\mathrm{alea}}(\boldsymbol{y}). \tag{7}$$

Algorithm 1: Bayesian Unsupervised Drift Detection

Input : Channel outputs $\mathcal{D}[t]$; Bayesian DNN $\boldsymbol{\theta}[t]$;
Previous quantities $\{n[t-1], \mu[t-1], \sigma[t-1]\}$; Threshold λ

Sample DNN $\boldsymbol{\varphi}_j[t] \sim q_{\boldsymbol{\theta}[t]}$ for $j \in 1, \ldots, J$

Compute $\{\{\hat{P}_{\boldsymbol{\varphi}_j[t]}(\boldsymbol{s}|\boldsymbol{y})\}_{j=1}^{J}\}_{\boldsymbol{y}\in\mathcal{D}[t]}$

Calculate $n[t]$ (Q1), $\mu[t]$ (Q2), and $\sigma[t]$ (Q3)

Calculate $\mathcal{T}[t]$ from (9) or (10)

if $\mathcal{T}[t] > \lambda$ **then**

  return $\mathcal{M}^{\mathrm{tr}}[t] = 1$

else

  return $\mathcal{M}^{\mathrm{tr}}[t] = 0$

These uncertainty features provide informative indicators for asynchronous retraining. Specifically, The epistemic uncertainty U_{epis} is expected to grow when the receiver's internal representation no longer matches the current channel distribution, indicating a need for retraining. The total uncertainty U_{tot} encapsulates both stochasticity of the channel and mismatch in the receiver. As we wish to primarily monitor drift in the receiver, while also supporting detection of drifts due to overall high uncertainty, we define the following instantaneous features

$$f_i[t] = \alpha \cdot U_{\mathrm{tot}}(\boldsymbol{y}_i[t]) + \beta \cdot U_{\mathrm{epis}}(\boldsymbol{y}_i[t]), \tag{8}$$

where α, β are hyperparameters. By monitoring (8) over time, one can design statistical tests that reliably detect when the receiver becomes invalid, without requiring ground-truth transmitted symbols.

3.2 Statistical Tests for Drift Detection

Having defined the instantaneous uncertainty features in (8), our next goal is to determine whether the receiver is no longer suitable for the current channel. The underlying rationale is that when the channel distribution drifts beyond the generalization region of the receiver, then $f_i[t]$ will experience a statistically significant increase. To capture such changes, we monitor the following quantities:

Q1 *Sample size* $n[t] = |\mathcal{D}[t]|$;

Q2 *Empirical mean* $\mu[t] = \frac{1}{n[t]} \sum_{i\in\mathcal{D}[t]} f_i[t]$;

Q3 *Empirical variance* $\sigma[t] = \frac{1}{n[t]-1} \sum_{i\in\mathcal{D}[t]} (f_i[t] - \mu[t])^2$;

Based on Q1-Q3, which are aggregated between blocks in which no adaptation is carried out, we construct a test statistic $\mathcal{T}[t]$ used for drift detection. Specifically, we propose two statistical tests:

1) Two-Sample t-Test generalized to unequal variances (37) computes the statistic via the Student's t cumulative distribution function with ν degrees of freedom, denoted $F_\nu(\cdot)$, as

$$\mathcal{T}[t] = F_\nu \left(\frac{|\mu[t] - \mu[t-1]|}{\sqrt{\frac{\sigma[t]}{n[t]} + \frac{\sigma[t-1]}{n[t-1]}}} \right). \tag{9}$$

We specifically set ν via (37, Eq. (26)).

2) Effect-Size Test (38), setting $\mathcal{T}[t]$ to be

$$\mathcal{T}[t] = \frac{|\mu[t] - \mu[t-1]|}{\sqrt{\frac{(n[t]-1)\sigma[t] + (n[t-1]-1)\sigma[t-1]}{n[t]+n[t-1]-2}}}. \tag{10}$$

The resulting unsupervised Bayesian drift detection operation is summarized as Algorithm 1. Although similar, the t-statistic and the effect size differ by a factor of $\sqrt{n(t)}$. The t-test serves as a test of statistical significance and increases with sample size, such that even small effect sizes can yield small $\mathcal{T}[t]$ values when $n(t)$ is sufficiently large. On the other hand, the effect size quantifies the magnitude of the observed difference, independent of sample size.

3.3 Asynchronous Online Learning Framework

The proposed uncertainty-aware drift detectors form the basis for an *asynchronous online learning* framework for Bayesian deep receivers. The procedure operates sequentially over channel blocks as follows. At the beginning of block t, the receiver collects the set of channel outputs $\mathcal{D}[t] = \{\boldsymbol{y}_i[t]\}_{i=1}^{B}$ observed in that block. For each $\boldsymbol{y}_i[t] \in \mathcal{D}[t]$, the receiver computes the instantaneous feature $f_i[t]$ as in (8), and the aggregated statistics $\{n[t], \mu[t], \sigma[t]\}$ are updated. These statistics are then used to evaluate the test statistic $\mathcal{T}[t]$ (via (9) or (10)).

If $\mathcal{T}[t]$ exceeds the detection threshold λ, the framework declares that adaptation is required, i.e., sets $\mathcal{M}^{\mathrm{tr}}[t] = 1$. In this case, the block $\{\boldsymbol{y}_i[t]\}_{i=1}^{B}$ is augmented with label information obtained from scarce pilot transmissions or via self-supervision (18), forming the dataset $\mathcal{Q}[t]$. The receiver then updates the parameters $\boldsymbol{\theta}[t]$ of the Bayesian distribution $q_{\boldsymbol{\theta}[t]}$ by minimizing the ELBO loss in (3). The prior distribution $p_t(\boldsymbol{\varphi})$ in (3) can now be set to the posterior obtained at the previous retraining instance, i.e., $q_{\boldsymbol{\theta}[t-1]}$ (39), thereby enabling continual adaptation with memory of past channel states.

The threshold λ thus governs the trade-off between performance and computational cost: by adjusting λ, one can enforce the budget constraint on the total number of retraining operations $\sum_{t=1}^{T} \mathcal{M}^{\mathrm{tr}}[t] \leq C$ as posed in (4). In practice, λ may be chosen either analytically, e.g., via statistical significance levels, or empirically, e.g., through calibration on a validation set.

3.4 Discussion

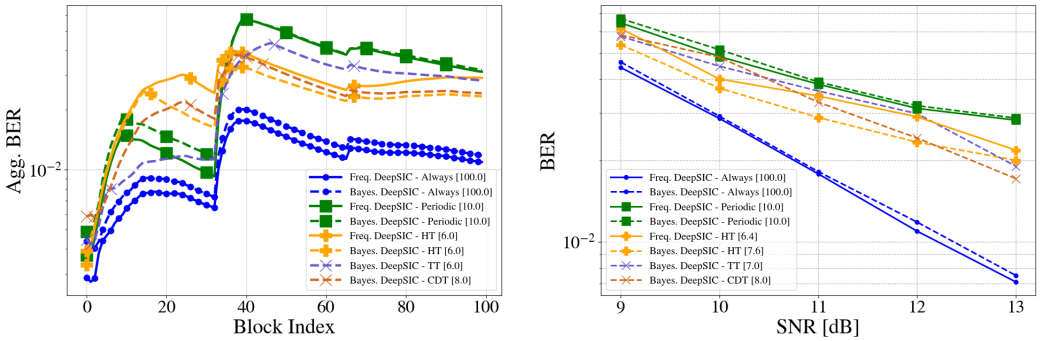
The proposed mechanism directly addresses the problem formulated in (4) by enabling unsupervised asynchronous online learning of deep receivers. By capitalizing on Bayesian deep learning, which has already been shown to enhance calibration (24) and facilitate online adaptation (26), we introduce a novel form of model-specific drift detection. The use of predictive uncertainty as a monitoring signal allows the receiver to autonomously identify when its internal representation becomes outdated. Importantly, the computational burden of the proposed mechanism is minimal: the cost of evaluating the statistical tests depends only on a fixed number of Monte Carlo samples and does not scale with the number of trainable parameters. In contrast, retraining complexity grows at least linearly with the model size when using variational inference methods such as Bayes by backprop (40), and can be substantially higher for more sophisticated training schemes (36). Thus, Algorithm 1 provides an efficient, statistically principled means of triggering retraining only when necessary.

4 Numerical Evaluations

Setup In this section, we numerically evaluate our asynchronous unsupervised online learning framework. We consider an uplink multiple-input multiple-output (MIMO) setting comprised of four users transmitting quadrature phase shift keying (QPSK) signals to a receiver equipped with four antennas. We transmit $T = 100$ blocks of $B = 8 \cdot 10^3$ symbols, from which $|\mathcal{D}(t)| = 6 \cdot 10^3$ are used for drift detection. We consider two channels: (i) COST2100 (31) using the 5 GHz indoor hall setting for each per user channel; and (ii) QuaDRiGa (32) simulating user movement in a 3GPP Urban Micro LOS environment. For the DNN-based receiver, we use the DeepSIC architecture of (12) with 5 iterations. We use both a standard frequentist model (as in (12)) as well as the Bayesian version proposed in (24), which is based on Monte Carlo dropout (34).

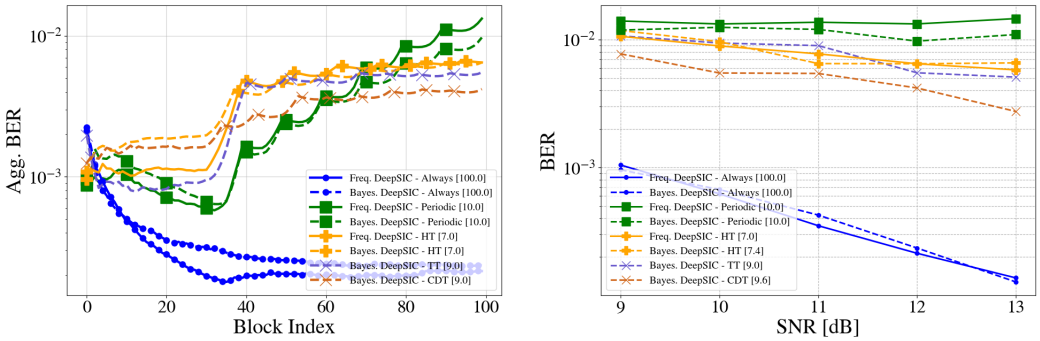
Training Methods We compare the performance achieved with our unsupervised Algorithm 1 using both the t-test (9) and the Effect size test (10) to the following alternatives: (i) *Always*: synchronous online training on each consecutive block. The performance of this approach serves as a lower bound on the achievable error-rate; (ii) *Periodic*: synchronous adaptation once every 10 blocks, suffers from the inability to choose when to issue re-training as in our suggested asynchronous; (iii) *Asynchronous Supervised*: using the Hotteling detector of (28). To ensure a fair comparison with the synchronous periodic setting, we set the threshold λ such that the number of re-trainings allowed for all asynchronous online learning frameworks (the average number of retrainings is indicated in brackets). Each re-training involves 250 epochs, and we focus on total uncertainty, setting $(\alpha, \beta) = (1, 0)$ in (8). Performance is evaluated as the bit error rate (BER).

Results Fig. 1 compares the training methods for COST2100. For the unsupervised methods, we see a BER decrease versus the synchronous periodic method with even less retrainings. We



(a) Aggregated BER versus block (b) Average BER versus SNR

Figure 1: COST2100 results



(a) Aggregated BER versus block (b) Average BER versus SNR

Figure 2: QuaDRiGa Results

also observe that both unsupervised methods performed similarly to the supervised Hotelling-test drift event detector, with similar number of retrainings. Similar trends are noted when evaluating overall average performance for different signal-to-noise ratios (SNRs) in Fig.1(b), In the QuaDRiGa channel, the users are stationary in the beginning and start moving around t 30. As shown in Fig. 2, the unsupervised methods avoid unnecessary retrainings during the stationary phase and promptly detect the transition when mobility begins. Among the detectors, the effect-size test identifies the change most rapidly. Additionally, while the asynchronous method displays a steeper gradient after the users start to move, all asynchronous methods maintain a shallower slope resulting in a decrease in the aggregated BER in line with the COST2100 results.

5 Conclusions

We proposed unsupervised drift detection methods for asynchronous online learning deep receivers by leveraging the uncetratiny measures of Bayesian neural networks. Our results highlight that the mechanisms provide robust adaptation to channel dynamics. These findings highlight the potential of unsupervised asynchronous online learning as a practical alternative to supervised approaches.

References

[1] L. Dai, R. Jiao, F. Adachi, H. V. Poor, and L. Hanzo, "Deep learning for wireless communications: An emerging interdisciplinary paradigm," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 133–139, 2020.

[2] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. on Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, 2017.

[3] W. Tong and G. Y. Li, "Nine challenges in artificial intelligence and wireless communications for 6G," *IEEE Wireless Commun.*, vol. 29, no. 4, pp. 140–145, 2022.

[4] J. Xia, D. Deng, and D. Fan, "A note on implementation methodologies of deep learning-based signal detection for conventional MIMO transmitters," *IEEE Trans. Broadcast.*, vol. 66, no. 3, pp. 744–745, 2020.

[5] J. Hoydis *et al.*, "Learning radio environments by differentiable ray tracing," *IEEE Trans. Mach. Learn. Commun. Netw.*, vol. 2, pp. 1527–1539, 2024.

[6] R. Faqiri *et al.*, "PhysFad: Physics-based end-to-end channel modeling of RIS-parametrized environments with adjustable fading," *IEEE Trans. Wireless Commun.*, vol. 22, no. 1, pp. 580–595, 2022.

[7] M. Honkala, D. Korpi, and J. M. Huttunen, "DeepRx: Fully convolutional deep learning receiver," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3925–3940, 2021.

[8] M. Goutay, F. A. Aoudia, J. Hoydis, and J.-M. Gorce, "Machine learning for MU-MIMO receive processing in OFDM systems," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2318–2332, 2021.

[9] G. Liu *et al.*, "A hypernetwork based framework for non-stationary channel prediction," *IEEE Trans. Veh. Technol.*, vol. 73, no. 6, pp. 8338–8351, 2024.

[10] T. Raviv and N. Shlezinger, "Modular hypernetworks for scalable and adaptive deep MIMO receivers," *IEEE Open Journal of Signal Processing*, vol. 6, pp. 256–265, 2025.

[11] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. Goldsmith, "Viterbinet: A deep learning based Viterbi algorithm for symbol detection," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3319–3331, 2020.

[12] N. Shlezinger, R. Fu, and Y. C. Eldar, "DeepSIC: Deep soft interference cancellation for multiuser MIMO detection," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1349–1362, 2021.

[13] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," *Proc. IEEE*, vol. 111, no. 5, pp. 465–499, 2023.

[14] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Model-based machine learning for communications," *Machine Learning and Wireless Communications*, p. 145, 2022.

[15] N. Farsad, N. Shlezinger, A. J. Goldsmith, and Y. C. Eldar, "Data-driven symbol detection via model-based machine learning," in *IEEE Statistical Signal Processing Workshop (SSP)*, 2021, pp. 571–575.

[16] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless networks design in the era of deep learning: Model-based, AI-based, or both?" *IEEE Trans. Commun.*, vol. 67, no. 10, pp. 7331–7376, 2019.

[17] 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 Commun.*, vol. 31, no. 4, pp. 163–169, 2024.

[18] M. B. Fischer, S. Dörner, S. Cammerer, T. Shimizu, H. Lu, and S. Ten Brink, "Adaptive neural network-based OFDM receivers," in *Proc. IEEE SPAWC*, 2022.

[19] T. Raviv and N. Shlezinger, "Data augmentation for deep receivers," *IEEE Trans. Wireless Commun.*, vol. 22, no. 11, pp. 8259–8274, 2023.

[20] L. Huang, W. Pan, Y. Zhang, L. Qian, N. Gao, and Y. Wu, "Data augmentation for deep learning-based radio modulation classification," *IEEE Access*, vol. 8, pp. 1498–1506, 2019.

[21] T. Raviv, S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger, "Online meta-learning for hybrid model-based deep receivers," *IEEE Trans. Wireless Commun.*, vol. 22, no. 10, pp. 6415–6431, 2023.

[22] S. Park, H. Jang, O. Simeone, and J. Kang, "Learning to demodulate from few pilots via offline and online meta-learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 226 – 239, 2020.

[23] M. Zecchin, S. Park, O. Simeone, M. Kountouris, and D. Gesbert, "Robust Bayesian learning for reliable wireless AI: Framework and applications," *IEEE Trans. on Cogn. Commun. Netw.*, vol. 9, no. 4, pp. 897–912, 2023.

[24] T. Raviv, S. Park, O. Simeone, and N. Shlezinger, "Uncertainty-aware and reliable neural MIMO receivers via modular Bayesian deep learning," *IEEE Trans. Veh. Technol.*, 2025.

[25] K. M. Cohen, S. Park, O. Simeone, and S. Shamai, "Bayesian active meta-learning for reliable and efficient AI-based demodulation," *IEEE Trans. Signal Process.*, vol. 70, pp. 5366–5380, 2022.

[26] Y. Gusakov, O. Simeone, T. Routtenberg, and N. Shlezinger, "Rapid online Bayesian learning for deep receivers," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025.

[27] N. Farsad and A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5663–5678, 2018.

[28] N. Uzlaner, T. Raviv, N. Shlezinger, and K. Todros, "Asynchronous online adaptation via modular drift detection for deep receivers," *IEEE Trans. Wireless Commun.*, vol. 24, no. 5, pp. 4454–4468, 2025.

[29] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, 2018.

[30] J. Gawlikowski *et al.*, "A survey of uncertainty in deep neural networks," *Artificial Intelligence Review*, vol. 56, no. Suppl 1, pp. 1513–1589, 2023.

[31] L. Liu *et al.*, "The COST 2100 MIMO channel model," *IEEE Wireless Commun.*, vol. 19, no. 6, pp. 92–99, 2012.

[32] 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 Trans. Antennas Propag.*, vol. 62, no. 6, pp. 3242–3256, 2014.

[33] V. Fortuin, "Priors in Bayesian deep learning: A review," *International Statistical Review*, vol. 90, no. 3, pp. 563–591, 2022.

[34] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," *Advances in neural information processing systems*, vol. 30, 2017.

[35] T. Van Luong *et al.*, "Deep learning based successive interference cancellation for the non-orthogonal downlink," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 11 876–11 888, 2022.

[36] 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.

[37] B. L. Welch, "The generalization of "student's" problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1-2, p. 28–35, 1947.

[38] J. Cohen, *Statistical power analysis for the behavioral sciences*. Routledge, 2013.

[39] M. E. Khan and H. Rue, "The Bayesian learning rule," *Journal of Machine Learning Research*, vol. 24, no. 281, pp. 1–46, 2023.

[40] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1613–1622.