2025 IEEE 36th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC): Track 3: Mobile and Wireless Networks

2025 IEEE 36th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) | 979-8-3503-6323-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/PIMRC62392.2025.11274653

ATLAS: AI-Native Receiver Test-and-Measurement by Leveraging AI-Guided Search

Mauro Belgiovine[‡], Suyash Pradhan[⋆], Johannes Lange[†], Michael Löhning[†], Kaushik Chowdhury[⋆]
[‡]*NVIDIA Corporation*, Santa Clara, CA, USA.
[⋆]*Chandra Dept. of Electrical and Computer Engineering*, The University of Texas at Austin
[†]*NI - Test and Measurement Group of Emerson*, Dresden, Germany
mbelgiovine@nvidia.com, {johannes.lange, michael.loehning}@emerson.com, {suyash.p, kaushik}@utexas.edu

***Abstract*—Industry adoption of Artificial Intelligence (AI)-native wireless receivers, or even modular, Machine Learning (ML)-aided wireless signal processing blocks, has been slow. The main concern is the lack of explainability of these trained ML models and the significant risks posed to network functionalities in case of failures, especially since (i) testing on every exhaustive case is infeasible and (ii) the data used for model training may not be available. This paper proposes ATLAS, an AI-guided approach that generates a battery of tests for pre-trained AI-native receiver models and benchmarks the performance against a classical receiver architecture. Using gradient-based optimization, it avoids spanning the exhaustive set of all environment and channel conditions; instead, it generates the next test in an online manner to further probe specific configurations that offer the highest risk of failure. We implement and validate our approach by adopting the well-known DeepRx AI-native receiver model as well as a classical receiver using differentiable tensors in NVIDIA's Sionna environment. ATLAS uncovers specific combinations of mobility, channel delay spread, and noise, where fully and partially trained variants of AI-native DeepRx perform suboptimally compared to the classical receivers. Our proposed method reduces the number of tests required per failure found by 19% compared to grid search for a 3-parameters input optimization problem, demonstrating greater efficiency. In contrast, the computational cost of the grid-based approach scales exponentially with the number of variables, making it increasingly impractical for high-dimensional problems.**

***Index Terms*—Test and Measurement, Deep Learning, AI-native, Cellular Networks**

I. Introduction

Next-generation wireless standards are actively exploring incorporating AI/ML approaches for sensing and channel-noise-resilient communication within their wireless transceivers [1], [2]. Although there have been prototypes and proof-of-concept deployments, there is still a wide gap between fully replacing traditional receivers with a fully AI-native counterpart. The root cause of this gap is the concern that ML models incorporated within an AI-native receiver may not generalize well in various practical environments, leading to catastrophic loss of performance of the network. Thus, without a rigorous and principled methodology for testing and measuring these native AI receivers, operators cannot be assured of their robustness under real-world conditions as e.g. observed on a real-time system-level prototype built with NI's USRP software-defined radio-based research platform [3].

• **Problem.** Traditional receiver designs have well-established performance characteristics based on proven theoretical frameworks. However, there is room for improvement, especially in ensuring low bit error rate (BER) in noisy conditions, where modular ML models have outperformed traditional designs [4], [5]. Replacing individual signal processing blocks with ML counterparts improves the explainability of the AI-native receiver, though the risk remains that ML "black-boxes" may behave unpredictably under certain conditions, especially when not fully exposed to all system variations during training. This risk increases when the entire receiver chain is integrated into a deep neural network, eliminating the modular breakdown for troubleshooting. An example is the DeepRx model for 5G cellular networks [6]. Neither a clear testing methodology exists for Deep learning (DL) based receivers due to the infeasibility of covering all possible wireless scenario conditions, nor have recent advances in eXplainable AI (XAI) [7] yet provided conclusive insights for models designed for the wireless physical layer.

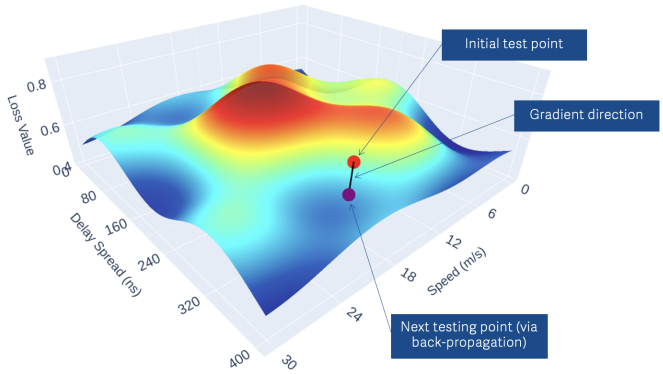


Fig. 1: Visualization of AI-native gradient-based scenario search for a simplified loss function with two variables (i.e., user terminal's speed and wireless channel delay spread).

• **Solution.** ATLAS addresses the challenge of testing AI-native receivers by using a gradient-based optimization approach to efficiently uncover failure cases that traditional grid-based methods or theoretical analyses cannot predict. In Fig. 1, we illustrate the procedure of initiating a random test point to validate a DL-based receiver, followed by computing the gradient direction with respect to key scenario variables such as user terminal's speed and wireless channel delay spread.

979-8-3503-6323-4/25/$31.00 ©2025 IEEE

Authorized licensed use limited to: University of Texas at Austin. Downloaded on May 17,2026 at 07:43:37 UTC from IEEE Xplore. Restrictions apply.

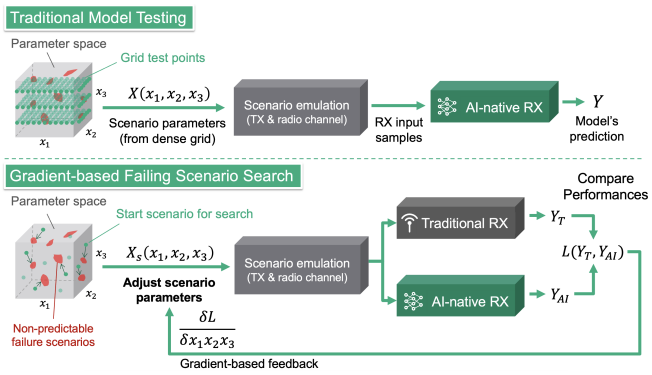


Fig. 2: Overview of ATLAS gradient-based failure scenario search (bottom) compared to classical grid-based exhaustive search (top). ATLAS employs an iterative optimization loop to fine-tune scenario simulation parameters, crafting input signals that feed into a dual processing pipeline—one path through a differentiable traditional receiver and the other through the AI-native receiver under test. A dedicated loss function then drives parameter updates by maximizing the error of the AI-native receiver and minimizing the traditional one.

This gradient can then be used to guide the selection of subsequent test points, allowing for a more efficient identification of critical scenarios compared to exhaustive search methods. As shown in Fig. 2, conventional testing relies on grid searches over system and environmental configurations (illustrated here along three dimensions), comparing AI-native and traditional receiver performance. Instead, ATLAS leverages the differentiability of DL models and a *differentiable traditional receiver* to compute gradients of key system parameters, guiding an adaptive search via back-propagation to pinpoint failure scenarios. This targeted approach minimizes test data collection and computational overhead while efficiently discovering critical test cases. Implemented using NVIDIA's Sionna [8], a differentiable wireless simulation library built on TensorFlow [9], ATLAS iteratively refines environmental conditions to generate adversarial inputs that expose weaknesses in AI-native receivers. By integrating feedback from both AI-native and traditional receivers, the framework optimizes signal processing modules and adjusts physical parameters, quickly converging to challenging deployment conditions. This AI-guided test selection not only validates AI-native models but also aids in dataset generation, mitigating the curse of dimensionality with multiple system parameters.

- **Contributions.** Our key contributions are as follows:
  - We propose a gradient-based approach to efficiently explore the test scenario parameter space for scenario configurations and the derived input signals that degrade the performance of AI-native receiver models.
  - Our approach leverages the differentiability of DL models and a fully differentiable traditional receiver pipeline, implemented via Sionna, to (i) compare AI-native and traditional receiver performance and (ii) guide parameter exploration to identify unpredictable failure scenarios.
  - We evaluate ATLAS using three modified versions of the DeepRx model [6] to assess its effectiveness across different training and parameter configurations.
  - We demonstrate improved efficiency of the proposed approach for an optimization problem based on 3 channel-dependent variables, achieving a 19% reduction in the number of tests per number of failures found, compared to grid-based tests. Moreover, we analyze the cost of the two approaches for a varying number of system variables, demonstrating better scalability of proposed solution.

II. Related Work

- **Testing Deep Learning models**: The reliability of DL models has been widely studied, with confidence calibration ensuring alignment between predicted scores and actual accuracy [10], [11]. While effective on image datasets, these methods adjust confidence rather than systematically identifying failure cases, which is crucial for AI-native receivers. XAI techniques like SHAP [12] help interpret unexpected behaviors by highlighting influential input features, aiding structured testing. However, this computes individual contribution of each feature in the input space of a tested model, which scales poorly with the large complex samples of time and frequency-domain wireless signals. Controlled perturbations to input data are frequently used to examine and understand the behavior of complex AI systems. The one-pixel attack [13] highlights the vulnerability of deep neural networks to low-dimensional adversarial perturbations by using differential evolution to alter just a single pixel in a black-box setting. Automated white-box testing [14], [15] refines test data samples via back-propagation to expose model weaknesses. Our approach is inspired by these gradient-based methods, which modify the input data of image processing networks to generate additional critical test cases. However, rather than directly perturbing the frequency-domain input data of the neural receiver under test—which could result in unrealistic or non-interpretable samples, we search for critical parameters in the wireless transmission scenario. These parameters correspond to meaningful physical properties of the wireless channel and environment, thereby enabling the generation of realistic and interpretable test cases, enabling efficient testing of AI-native receivers.
- **Reliability of AI-native wireless devices**: Authors in [16] highlight the challenges of deploying AI-native receivers in real-world wireless environments. [17] improves adaptability using transfer and meta-learning for beamforming but do not identify failure scenarios. Bayesian approaches, such as those in [18] and [19], model uncertainty to handle outliers and improve demodulator calibration, though they are computationally intensive and rely on accurate priors. However, these methods lack active exploration of the test space for adversarial test case generation.

III. Proposed Gradient-Based Approach

Our proposed method leverages gradient-based optimization to search the wireless scenario parameter space for configurations that constitute failure cases in AI-native wireless receivers. Typical applications of gradient-based optimization

Authorized licensed use limited to: University of Texas at Austin. Downloaded on May 17,2026 at 07:43:37 UTC from IEEE Xplore. Restrictions apply.

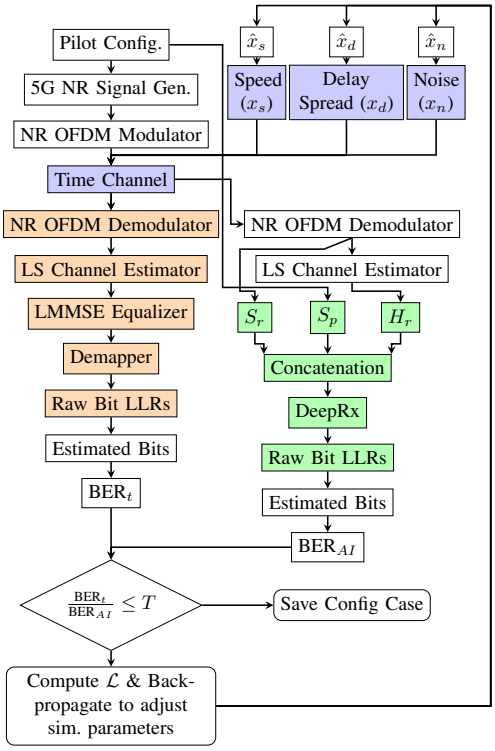


Fig. 3: Flow diagram of proposed optimization system, with differentiable traditional 5G transmitter/receiver pipeline (left) and AI-native receiver as the device under test (right). Symbols $\hat{x}_s$, $\hat{x}_d$, $\hat{x}_n$ denote re-parametrized *speed* (x_s), *delay spread* (x_d), and *noise* (x_n) inputs. Color-coded blocks represent (1) wireless channel generation (blue), (2) traditional receiver (orange), and (3) DeepRx-based DUT (green). $\mathcal{L}$ is the loss function defined in Eq. 2.

consist in defining a differentiable loss function used to compute the gradient of a set of variables and using this information to adjust them toward achieving a specific goal. Unlike the traditional application of gradient-based optimization in DL, our approach does not aim to optimize the parameters of the model, but rather the (differentiable) *system parameters* that produce configurations leading to higher output error of the AI-native receivers when compared to traditional architectures.

A. Device Under Test (DUT) Description

To illustrate our approach, we consider as the device under test (DUT) an AI-based receiver based on the popular DeepRx [6] architecture, which consists in a state-of-the-art fully-convolutional neural wireless receiver trained to take in input frequency-domain information, namely (i) received signal S_r, (ii) known pilot sequence S_p and (iii) Least-Square coarse channel estimation H_r, and produce raw Log-Likelihood Ratio (LLR) estimation of the received bits.The UpLink (UL) signals used for model training and failure search experiments follow 5G NR specifications [20], with 14 OFDM symbols (1 ms TTI), 15 kHz sub-carrier spacing, 2-symbol DM-RS pilot configuration, 6 PRBs, and random bit payloads. These frequency-domain signals are modulated to the time domain, transmitted through a wireless channel, and then demodulated back to the frequency domain at the receiver. The DUT performance is measured in terms of Bit-Error-Rate (BER) for a number of different system configurations, which for this work are limited to real-valued wireless channel parameters (i) UE's *speed* x_s (m/s), (ii) *delay spread* x_d (ns) and (iii) channel's *noise power* x_n (dBm), with the latter determining the Signal-to-Noise Ratio (SNR) of simulated transmissions. Note that, while we focus on channel parameters due to their direct impact on communication performance and desirable differentiable properties, this approach allows to study the impact of any other continuous variables involved in the system. The BER$_{AI}$ of DeepRx is compared with the BER$_t$ obtained via a traditional wireless receiver under the same channel conditions, in order to study the differences in their behavior for a given system configuration and receiver input. In the proposed application, we want to identify scenarios where BER$_{AI}$ exceeds BER$_t$ by a pre-decided tolerance ratio T, defining the following failure scenario trigger condition:

$$\frac{\text{BER}_t}{\text{BER}_{AI}} \leq T \tag{1}$$

B. Gradient-based Failing Configuration Search

ATLAS follows an iterative approach, exploring multiple wireless scenarios through search *episodes*. In each episode, we uniformly sample an initial wireless system configuration from all possible configurations expected during deployment. The performance of traditional and AI-native receivers is evaluated using a differentiable dual-receiver pipeline (Fig. 3) to compute the BERs and gradients of the system variables. The system parameters are then adjusted iteratively using the gradients until a failure scenario is identified. Each episode consists of one or more iterations of the following steps:

1) **Wireless Scenario Simulation**: For each scenario configuration, we use Sionna simulation blocks to generate a *batch* of 5G NR signals, each passed through an independent simulated wireless channel realization. The respective common scenario parameters are either initially sampled from the scenario parameter space or provided by the gradient-based update during optimization.
2) **Parallel Receiver Pipelines**: The resulting batch of receiver input signals is processed by two parallel receiver pipelines: (i) a traditional differentiable pipeline and (ii) a DL-based receiver pipeline. Both pipelines produce a BER value averaged over all the simulated transmissions that serves as a proxy to evaluate the performance gain of AI-based receiver compared to the traditional counterpart.
3) **Failure Case Identification**: Once the BERs are obtained, we evaluate the search stop criteria in Eq. 1. If satisfied, the current scenario configuration is considered to be failing and stored, stopping the current search. Otherwise, the search continues to the next step.
4) **Gradient-Based Optimization**: We compute the gradients of studied system-level variables through a loss function that compares the BERs of the two receivers,

Authorized licensed use limited to: University of Texas at Austin. Downloaded on May 17,2026 at 07:43:37 UTC from IEEE Xplore. Restrictions apply.

Parameter	PTLC	FTLC	FTHC
Modulation	16 QAM	64 QAM	QPSK, 16 QAM, 64 QAM
Eb/N0 (dB)	$\{0, 1, 2, 3, 18, 19, 20\}$	$\{0, 2, 4, ..., 20, 22\}$	$\{0, 2, 4, ..., 20, 22\}$
Delay Spread (ns)	$\{0, 10, 20, 300, 350, 400\}$	$[0, 400]$	$[10, 400]$
Speed (m/s)	$\{0, 1, 2, 20, 25, 30\}$	$[0, 30]$	$[0, 30]$
Channel Profile	TDL-D	TDL-D	TDL-B, TDL-C, TDL-D
Model Config.			
# ResNet Layers	5	5	11
# Parameters	58k	58k	700k

TABLE I: Training parameters for DeepRx-based wireless receiver models studied in this work. Values in curly brackets indicates sets of discrete values, while values in square brackets indicate random sampling in a continuous interval.

capturing the impact of system variations on performance differences between models. We then back-propagate the gradients to adjust the channel configurations, iteratively refining the scenario parameters based on the optimization objective. The process returns to Step 1 and repeats with the updated configuration.

In order to relate the traditional and AI-based receiver performances and guide the gradient-based exploration in the desired direction, we define a loss function $\mathcal{L}$ as follows:

$$\mathcal{L} = \text{BER}_t - \text{BER}_{AI} \tag{2}$$

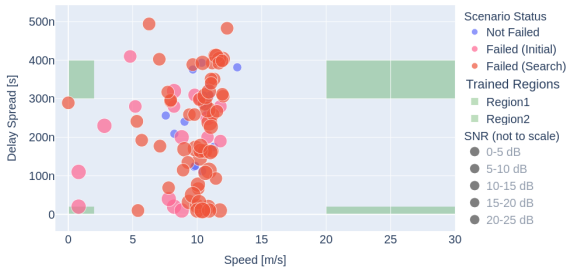
Our objective is to *minimize* $\mathcal{L}$ via gradient descent, where BER_t and BER_{AI} are *differentiable* functions representing the BER for the traditional and DeepRx-based receivers, respectively, given the same input samples from the differentiable wireless simulator, Sionna. By minimizing $\mathcal{L}$, we force the DeepRx receiver to perform worse than the traditional one ($\text{BER}_{AI} > \text{BER}_t$). For effective performance, we expect $\mathcal{L} < 0$, as $-1 < \text{BER}_t - \text{BER}_{AI} < 1$. This method systematically uncovers weaknesses in ML-based wireless receivers and ensures reliable performance across diverse conditions. Each episode is capped at a maximum of I optimization iterations, with *early-stopping* used to avoid non-productive directions (i.e., when the loss stops decreasing), prompting a new search in a different region. This approach identifies cases where the AI receiver underperforms, highlighting pitfalls and providing configurations to refine future AI solutions.

C. Re-parametrization of Optimized System Variables

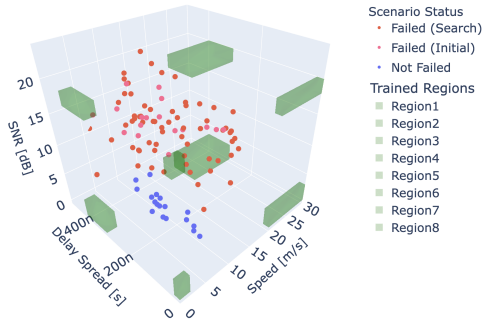
Optimizing variables with significantly different scales (e.g., speed and delay spread) can cause numerical instability and varying gradient magnitudes, complicating the selection of an appropriate learning rate for smooth updates. To mitigate this issue, we normalize each optimized variable x to a well-conditioned range, typically $[0, 1]$, using MinMax normalization: $\hat{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$ where $x_{\min}$ and $x_{\max}$ are the known lower and upper bounds of the variable. The optimizer computes gradients and updates $\hat{x}$ instead of x, ensuring balanced updates. Before using the optimized values, we map them back to the original scale: $x = \hat{x}(x_{\max} - x_{\min}) + x_{\min}$. This ensures stability in a normalized space while maintaining x_{norm} within $[0, 1]$ to prevent out-of-bounds updates.

IV. Results from Under-Trained Model

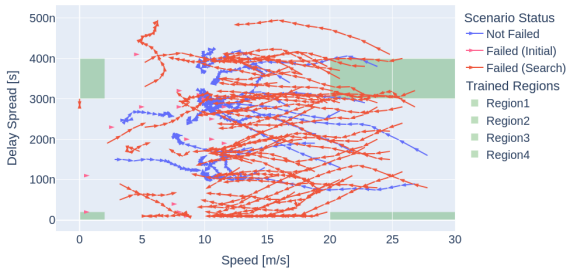
This section demonstrates how ATLAS uncovers weaknesses of AI-native receivers trained under varying conditions.



(a) Scenario configurations and test results after gradient-descent failing scenario search across selected channel parameters.



(b) Scenario search results mapped to 3D parameter space



(c) Search trajectories (2D) for each episode across parameter space (arrows indicate the search direction per optimization step)

Fig. 4: Partially Trained, Low Complexity (PTLC) model gradient-based scenario search results. Green areas indicate limited wireless channel parameter value ranges used to train PTLC model sub-optimally.

All models were trained using synthetically generated transmissions and bit labels for BER computation with Sionna. We begin with an under-trained version of DeepRx as the DUT, highlighting how ATLAS identifies wireless scenario configurations underrepresented during training. This version, termed *partially trained, low complexity* (PTLC), follows the original model but uses only 5 out of the 11 ResNet layers.

A. PTLC Model Training

In this experiment, we under-train the PTLC DeepRx model by generating a dataset with only extreme high and low values for respective wireless channel parameters. The training procedure starts with data generation using Sionna, simulating end-to-end transmission of a 5G NR signal, followed by model training using the generated data. Table I details the

Authorized licensed use limited to: University of Texas at Austin. Downloaded on May 17,2026 at 07:43:37 UTC from IEEE Xplore. Restrictions apply.

model configuration and parameter ranges for the wireless environment used to generate input samples for model training.

B. Failing Scenario Search Results

After training the PTLC model, we perform gradient-based search over $N_e = 100$ episodes to identify failure scenarios. Each episode allows up to $I = 100$ iterations, starting with an initial learning rate $l_r = 0.01$, which is halved if the loss does not improve for 5 epochs. An early stopping criterion is triggered after two such reductions. At each iteration we simulate a batch of 25 scenario realizations (signals and channel instances). The start scenario parameters for each episode are sampled from a discrete set evenly distributed across the training ranges, specifically:

- Speed (m/s): $\{0, 1, ..., 29, 30\}$;
- Delay Spread (ns): $\{10, 20, ..., 380, 400\}$
- SNR (dB): $\{5, 10, 15, 20\}$

We process signals via traditional and AI pipelines to compute BER for each receiver. If the failure criterion is not met, we use back-propagation with the loss function $\mathcal{L}$ (see Eq. 2) to compute gradients and update the channel configuration. The batch size is selected based on the available GPU memory (in our case, an NVIDIA A100 with 80GB of VRAM), albeit larger batch sizes generally leading to smoother loss behavior and more stable gradient estimates. Throughout our experiments, we use the highest Modulation and Coding Scheme (MCS) available during training (i.e., 16 QAM) as it is more sensitive to decoding errors. We set a stopping threshold $T = 0.9$ (see Eq. 1), indicating that we terminate the search when the AI-native receiver's BER is at least 10% higher than that of the traditional receiver. This threshold enables us to identify configurations that are more likely to lead to decoding failures, even though a well-trained AI model is expected to outperform traditional methods across most scenarios. Thus, the choice of T reflects the desired level of conservativeness in the failure scenario analysis. Fig. 4 summarizes the scenario configurations found via gradient-based search, including 65 failure cases, 17 immediate failures[1], and 18 non-failing configurations. As shown in Fig. 4a, all identified failure cases lie outside the trained channel parameter space as intuitively expected, validating that the search method works. Notably, failures occur only at SNRs > 10 dB, indicating that the under-trained AI receiver still outperforms the traditional one in low-SNR regimes. Furthermore, failures cluster around speeds of 10–15 m/s at higher SNRs. These findings confirm that ATLAS not only locates the model's weak points but also highlights regions where such vulnerabilities concentrate. To illustrate how the gradient-based search traverses the parameter space, Fig. 4c shows 2D trajectories over delay spread and speed, demonstrating convergence toward "blind spot" regions omitted during training.

C. Search Results Validation

After identifying failure points via gradient-based exploration, we validate them through an independent run of tests to assess whether these scenarios consistently yield worse BER for the DeepRx model compared to the traditional pipeline. For each identified configuration, we average the BER across 1500 scenario realizations, as memory constraints no longer apply in this offline phase. The same decision criterion in Eq. 1 is used, with results shown for two thresholds: $T = 1.0$ and $T = 0.9$. The higher threshold captures scenarios where DeepRx offers no clear performance gain despite its computational overhead. Such cases are problematic, as DL-based models should justify their cost by consistently outperforming traditional receivers: any configuration in which these failures occur are critical and must be addressed during training.

[1]Scenarios where the stopping criterion is met before or within a single optimization step.

	PTLC		**FTLC**		**FTHC**	
Grad. Based Search						
Fail (search)	65		47		51	
Fail (init)	17		4		0	
Not fail	18		49		49	
Validation - $T = 0.9$	True	False	True	False	True	False
Fail (search)	63	2	42	5	34	17
Fail (init)	17	0	4	0	0	0
Not Fail	18	0	49	0	49	0
Search Accuracy*	0.98		0.95		0.83	
Search Precision†	0.976		0.902		0.667	
Recall‡	1.0		1.0		1.0	
Validation - $T = 1.0$	True	False	True	False	True	False
Fail (search)	65	0	46	1	42	9
Fail (init)	17	0	4	0	0	0
Not Fail	18	0	49	0	49	0
Search Accuracy*	1.0		0.99		0.91	
Search Precision†	1.0		0.980		0.823	
Recall‡	1.0		1.0		1.0	

TABLE II: Performance of proposed gradient-based failure scenario search on PTLC, FTLC and FTHC models with 100 search episodes applied per model and results validated over 1500 scenario realizations. * = Ratio of all correct predictions over all predictions. † = $\frac{\text{TP}}{\text{TP+FP}}$ (TP = True Positives, FP = False Positives). ‡ = $\frac{\text{TP}}{\text{TP+FN}}$ (FN = False Negatives).

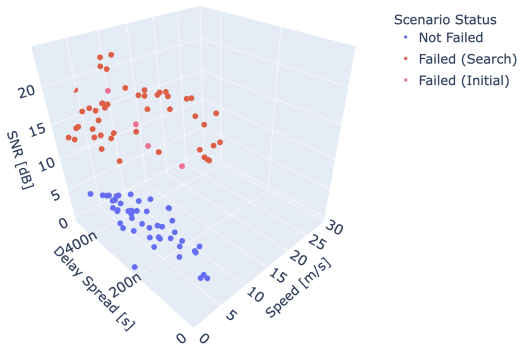
V. Results from Fully-Trained Models

Once we established the ability of our model to identify configurations of system parameters that affect poorly trained models, we want to test its efficiency on fully trained models with varying complexity, here referenced again in terms of the number of parameters/layers in the convolutional network.

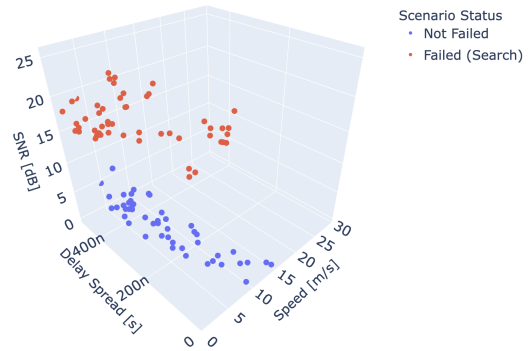
A. FTLC and FTHC Models Training

The Fully Trained, Low Complexity (FTLC) and Fully Trained, High Complexity (FTHC) models share the same training procedure but differ in the number of parameters. FTLC has 5 ResNet blocks (like PTLC), while FTHC uses 11 ResNet blocks. The aim is to evaluate whether the failure configuration search method can identify failing scenarios in well-trained models and assess the impact of model parameterization (i.e., number of layers and trainable weights). Both models were trained using the procedure outlined for PTLC, with training parameters in Table I. Similar to PTLC, we test both models at the highest modulation scheme they were trained on (i.e., 64 QAM).

Authorized licensed use limited to: University of Texas at Austin. Downloaded on May 17,2026 at 07:43:37 UTC from IEEE Xplore. Restrictions apply.



(a) FTLC model



(b) FTHC model

Fig. 5: Search results projected in 3D parameter space for FTLC and FTHC models using gradient-based search.

B. Failing Scenario Search Results

Similarly to the PTLC model, we run 100 search episodes with a maximum of $I = 100$ iterations and early stopping. Fig. 5 shows the search results for failing scenarios in FTLC and FTHC models. Both FTLC and FTHC, like PTLC, exhibit a concentration of failing scenarios in higher SNR regions, i.e. SNR $>$ 10 dB for FTLC and SNR $>$ 15 dB for FTHC. This highlights the denoising capabilities of DL models, which outperform traditional receivers in noisy conditions. However, even fully trained models are affected in higher SNR regions, where the channel effects dominate. This suggests that training datasets for DL-based channel receivers should incorporate more channel samples in higher SNR regions. In terms of failures, both FTLC and FTHC exhibit fewer failure scenarios than PTLC for the same number of episodes (Table II). This is expected, as both models were trained over a broader range of channel parameters, resulting in improved generalization of the model. Whereas, FTHC has more trainable weights, allowing it to better adapt to diverse scenarios and reducing rate of failures under unexpected conditions.

C. Search Results Validation

While both models exhibit fewer failure points overall, they still show several failures found by the gradient-based search. The results validation shows that the search precision varies with model complexity. For the FTLC model it achieves a precision of 90.2% with $T = 0.9$ and 98% with $T = 1.0$. For the FTHC we see lower precision values of 66.7% ($T = 0.9$) and 82.3% ($T = 1.0$). Larger training datasets and increased model parameters improve AI-based receiver robustness but make model verification more challenging. Nevertheless, ATLAS effectively identifies regions in the training space where fully trained models unexpectedly underperform compared to traditional receivers. This helps analyze model weaknesses and refine the bias-variance trade-off for better reliability.

VI. Comparison with Grid-Based Testing

We compare the efficiency of gradient-based search in ATLAS with conventional grid-based testing using the FTHC model as the DUT. Grid-based testing evaluates uniformly spaced configurations, but its cost grows exponentially with dimension, i.e. $\mathcal{O}(k^d)$ for k points over d parameters, making it impractical in high-dimensional spaces. Although domain knowledge can guide resolution of tests, DL models often react sharply to small environmental changes, making fixed grids unreliable for failure discovery. In contrast, ATLAS samples initial configurations and uses gradients to iteratively navigate toward failure regions without full parameter space coverage. Early-stopping further boosts efficiency by halting unproductive searches. Gradient-based methods like Adam compute only d gradients per step, leading to a linear complexity of $\mathcal{O}(N_e I d)$ across N_e episodes and I iterations each. Table III compares the cost-efficiency of both approaches, using $\frac{\#\text{Tests}}{\#\text{Failures}}$ as the metric. Here, each optimization step is treated as a single test per the setup in Section V-B. For the grid-based approach, we define a uniform sampling space spanning the parameter ranges used during model training, evaluating all points within this discretized configuration space:

- Speed (m/s): $[0, 30]$ divided in 25 intervals;
- Delay Spread (ns): $[0, 400]$ divided in 25 intervals;
- SNR (dB): $[0, 22]$ divided in 16 intervals;

for a total of 10k grid test points. After validating with $T = 1.0$, we observe that ATLAS identifies 42 failing configurations with an average of 59.8 tests per failure. In contrast, the grid-based search discovers 135 failures, but requires an average of 74.1 tests per failure. This reflects a 19% reduction in the total number of tested configurations using ATLAS, emphasizing the efficiency of a guided search that circumvents exhaustive evaluation of all parameter combinations. Fig. 6 shows the number of tests required by ATLAS versus grid-based search as the number of parameters d increases. ATLAS scales linearly with d and early-stopping, approximated as an average of $I/2$ and $I/4$ iterations across all search episodes, further reduces cost. In contrast, grid-based methods suffer from exponential growth with d due to the curse of dimensionality. Higher grid granularity (k) exacerbates this, making it increasingly difficult for traditional methods to capture DL receivers' sensitivity to subtle system variations.

VII. Conclusions and Future Work

This work highlights ATLAS's advantage over traditional grid-based testing by actively targeting failure scenarios instead of exhaustively testing all parameter combinations. Its gradient-based search, enabled by a differentiable traditional

Authorized licensed use limited to: University of Texas at Austin. Downloaded on May 17,2026 at 07:43:37 UTC from IEEE Xplore. Restrictions apply.

	Gradient-based search	Grid search
# Episodes	100	-
# Optim. step	100 (max)	-
# Tested scenarios	2511	10000
# Validated fails ($T = 1.0$)	42	135
# Tests per failure	59.8	74.1

TABLE III: Comparison of failing scenario search test efficiency (i.e., ratio of total tests executed over the number of found failing configurations) between proposed gradient-based approach and grid-based approach for FTHC model.

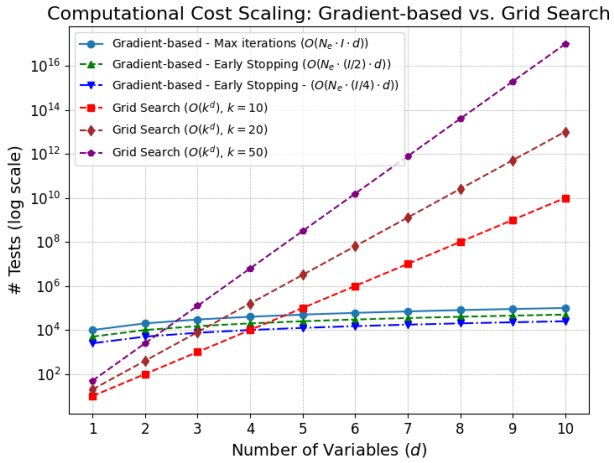


Fig. 6: Cost scaling comparison of ATLAS vs. grid-based testing as the number of variables d increases. Grid search grows exponentially with d, while ATLAS scales linearly.

and DL-based receiver pipeline, scales efficiently in high-dimensional spaces and uncovers nuanced failure conditions beyond fixed test grids. While simulation results are promising, future work should focus on improving search efficiency and extend applicability to real-world, non-differentiable radio channels via gradient approximation and richer digital twin simulations. Algorithms such as Differential Evolution, as used in [13], or Bayesian Optimization, can offer promising avenues for real system implementation due to their derivative-free approach and better flexibility due to their model-agnostic nature. Building on this, we propose validating ATLAS on a physical testbed, which could be created, e.g., using NI's USRP-based research platform in combination with protocol stacks from Open Air Interface as described in [3].

VIII. Acknowledgment

This work is partially supported by U.S. National Science Foundation under 2112471.

References

[1] Linglong Dai, Ruicheng Jiao, Fumiyuki Adachi, H. Vincent Poor, and Lajos Hanzo. Deep learning for wireless communications: An emerging interdisciplinary paradigm. *IEEE Wireless Communications*, 27(4).

[2] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(3):2224–2287, 2019.

[3] National Instruments. Prototyping a Real-Time Neural Receiver with USRP and OpenAirInterface. https://www.ni.com/en/solutions/electronics/5g-6g-wireless-research-prototyping/prototyping-real-time-neural-receiver-usrp-openairint.html, 2024.

[4] Nasim Soltani, Hai Cheng, Mauro Belgiovine, Yanyu Li, Haoqing Li, Bahar Azari, Salvatore D'Oro, Tales Imbiriba, Tommaso Melodia, Pau Closas, Yanzhi Wang, Deniz Erdogmus, and Kaushik Chowdhury. Neural network-based ofdm receiver for resource constrained iot devices. *IEEE Internet of Things Magazine*, 5(3):158–164, 2022.

[5] Bahar Azari, Hai Cheng, Nasim Soltani, Haoqing Li, Yanyu Li, Mauro Belgiovine, Tales Imbiriba, Salvatore D'Oro, Tommaso Melodia, Yanzhi Wang, Pau Closas, Kaushik Chowdhury, and Deniz Erdoğmuş. Automated deep learning-based wide-band receiver. *Computer Networks*.

[6] Mikko Honkala, Dani Korpi, and Janne M. J. Huttunen. Deeprx: Fully convolutional deep learning receiver. *IEEE Transactions on Wireless Communications*, 20(6):3925–3940, 2021.

[7] Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, et al. Explainable ai (xai): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 55(9):1–33, 2023.

[8] Jakob Hoydis, Sebastian Cammerer, Fayçal Ait Aoudia, Avinash Vem, Nikolaus Binder, Guillermo Marcus, and Alexander Keller. Sionna: An open-source library for next-generation physical layer research. *arXiv preprint arXiv:2203.11854*, 2022.

[9] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[11] Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *CVPR workshops*, volume 2, 2019.

[12] Wanying Ge, Jose Patino, Massimiliano Todisco, and Nicholas Evans. Explaining deep learning models for spoofing and deepfake detection with shapley additive explanations. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6387–6391, 2022.

[13] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[14] Jing Yu, Yao Fu, Yanan Zheng, Zheng Wang, and Xiaojun Ye. Test4deep: an effective white-box testing for deep neural networks. In *2019 IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC)*, pages 16–23. IEEE, 2019.

[15] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, 2017.

[16] Jakob Hoydis, Fayçal Ait Aoudia, Alvaro Valcarce, and Harish Viswanathan. Toward a 6g ai-native air interface. *IEEE Communications Magazine*, 59(5):76–81, 2021.

[17] Yi Yuan, Gan Zheng, Kai-Kit Wong, Björn Ottersten, and Zhi-Quan Luo. Transfer learning and meta learning-based fast downlink beamforming adaptation. *IEEE Transactions on Wireless Communications*, 2020.

[18] Matteo Zecchin, Sangwoo Park, Osvaldo Simeone, Marios Kountouris, and David Gesbert. Robust bayesian learning for reliable wireless ai: Framework and applications. *IEEE Transactions on Cognitive Communications and Networking*, 9(4):897–912, 2023.

[19] Kfir M. Cohen, Sangwoo Park, Osvaldo Simeone, and Shlomo Shamai. Learning to learn to demodulate with uncertainty quantification via bayesian meta-learning. In *WSA 2021; 25th International ITG Workshop on Smart Antennas*, 2021.

[20] 3GPP Technical Specification Group Radio Access Network. 5G; NR; Physical channels and modulation. Release 18. https://www.etsi.org/deliver/etsi_ts/138200_138299/138211/18.02.00_60/ts_138211v180200p.pdf, 2024. [Online; accessed Mar 6, 2025].

Authorized licensed use limited to: University of Texas at Austin. Downloaded on May 17,2026 at 07:43:37 UTC from IEEE Xplore. Restrictions apply.