Open-Source RF Environment Generator for Procedurally-Generated Over-the-Air Datasets Using GNU Radio

Samuel Brosh*, Kathleen De Key*, Charlotte Lecomte, and Ryan Westafer
Information and Communications Laboratory
Georgia Tech Research Institute, Atlanta, GA, USA
{samuel.brosh, kathleen.dekey, charlotte.lecomte, ryan.westafer}@gtri.gatech.edu

***Abstract*—The development of robust Radio Frequency Machine Learning (RFML) systems has been constrained by the suitability of realistic training data and the lack of models trained against real-world channel effects. While recent efforts focus on precise mathematical models for RF propagation, the most accurate approach is to operate directly in the real-world electromagnetic environment. This work presents an open-source RF Environment Generator (RFEG) that enables procedurally-generated over-the-air (OTA) datasets for online machine learning training for a variety of downstream tasks. The system employs distributed USRPs controlled via GNU Radio and general purpose processor (GPP) nodes, orchestrated by a central node generating JSON-formatted signal parameter files to emulate a user-specified environment.**

The RFEG implements matched filtering and time-synchronization protocols enabling precise alignment between received signals and their reference copies, which is essential for downstream RFML tasks including symbol-level information recovery, channel estimation, equalization, interference mitigation and demodulation. We demonstrate the system through OTA validation using 3 distributed USRP emitters. By providing fully open-source, reconfigurable infrastructure, the RFEG enables research groups to perform cognitive sensing research embedded in their electromagnetic environment, rather than being constrained to generic datasets that may not reflect their operational scenarios.

***Index Terms*—wireless communications, automatic modulation classification, dataset generation, channel impairment, machine learning, artificial intelligence**

I. Introduction

Radio Frequency Machine Learning (RFML) — the application of data-driven models to tasks such as modulation recognition, signal detection, specific-emitter identification, and interference mitigation — has advanced rapidly, but its progress remains bounded by the data on which models are trained. Unlike vision or language, where massive, diverse, real-world corpora are readily available, the RF community depends heavily on either (i) small, curated collections tied to specific hardware and environments, or (ii) synthetic datasets generated from mathematical channel models. Both approaches limit the generalization of trained models to the operational electromagnetic (EM) environments in which they are ultimately deployed.

Synthetic generation is attractive because it is cheap, perfectly labeled, and arbitrarily large. Widely used publicly available datasets such as the RadioML variants [1] have enabled a generation of modulation-classification research, and large-scale frameworks such as TorchSig [2] now provide standardized, programmatic synthesis and training pipelines at scale. Increasingly sophisticated differentiable and ray-traced channel simulators [3] [4] now model propagation, fading, and hardware impairments with impressive fidelity. Yet however precise these models become, they remain models: there exists nothing that models the real-world like the real-world itself. Real deployments exhibit site-specific multipath, non-stationary interference, hardware nonidealities, and incidental emitters that are difficult to enumerate *a priori* and therefore difficult to simulate faithfully. A model trained exclusively on simulated data is limited to the simulator's parameter space.

The most faithful training data is data collected over-the-air (OTA) in the same class of environment in which a system will operate. The obstacle has historically been effort: OTA collection is labor-intensive, hard to label precisely, and difficult to reconfigure across the range of scenarios needed to train robust models. What the field lacks is not the ability to capture RF, but the ability to procedurally generate labeled RF environments over-the-air — to specify a scenario parametrically and have real radios realize it in the physical environment, with ground-truth labels attached.

This paper presents the RF Environment Generator (RFEG), an open-source system that closes that gap. RFEG procedurally generates OTA datasets by orchestrating a set of distributed software-defined radios (SDRs) to realize a user-specified environment in the real EM environment. A central coordinating node compiles an environment into JSON-formatted signal-parameter files that are distributed to emitter nodes — Ettus USRPs driven by GNU Radio and hosted on GPP controllers — while an arbitrary sensing node captures the resulting signal. Because the central node specifies every emitted waveform, the system produces datasets with precise ground-truth labels, and because emission occurs over-the-air, those datasets carry the real channel effects that simulation cannot fully reproduce.

This work was funded by the GTRI Independent Research and Development (IRAD) program.

[a]These authors contributed equally to this work.

RFEG operates in two modes. Narrowband mode implements matched-filtering and time-synchronization protocols that align each received signal with its transmitted reference copy. This alignment is a prerequisite for supervised, symbol-level RFML tasks including information recovery, channel estimation, equalization, interference mitigation, and demodulation. Wideband mode coordinates a multi-node event in which several SDRs transmit distinct modulation schemes, sample rates, and center frequencies simultaneously, composing a populated spectral environment suitable for wideband detection and classification tasks.

We validate the system through OTA experiments on the Georgia Tech campus, using distributed USRP emitters and a sensing receiver node. We report time-synchronization performance, present representative spectrograms and I/Q captures from generated environments, and demonstrate a downstream model trained on an RFEG-generated narrowband dataset.

The contributions of this work are:

- An open-source, reconfigurable OTA environment-generation system built on GNU Radio, USRPs, and GPP nodes, orchestrated by a central node via declarative JSON environment descriptions.
- A narrowband collection mode with matched-filtering and time-synchronization protocols that align received signals to their reference copies, enabling labeled, symbol-level RFML datasets.
- A wideband collection mode that composes multi-emitter spectral environments across frequency for detection- and classification-oriented datasets.
- OTA validation on the Georgia Tech campus with three emitters and a radio receiver, including synchronization results, example datasets, and a downstream model trained on RFEG data.

By providing fully open-source, reconfigurable infrastructure, RFEG enables research groups to conduct cognitive-sensing research embedded in their own electromagnetic environment, rather than being constrained to generic datasets that may not reflect their operational scenarios. We view procedural OTA generation as a step toward online RFML — training against live, labeled data as it is collected — and we outline that trajectory as future work.

The remainder of this paper is organized as follows. §II reviews related work in synthetic RFML datasets, channel and propagation simulators, over-the-air testbeds, and GNU Radio–based experimentation, and positions RFEG among them. §III describes the methods, detailing the narrowband and wideband implementations of the RF Environment Generator, the modulation classes and parameter randomization, the matched-filtering and synchronization pipeline, and the ResNet-Transformer automatic modulation classification (AMC) model used to exercise the generated data. §IV presents our experimental setup and results, including within-environment validation across two indoor environments and a cross-dataset comparison against RadioML 2018.01a that quantifies the synthetic-to-over-the-air generalization gap. §V concludes and outlines directions for future work, including online training against live data and multi-element array-based sensing.

II. Related Work

A. Synthetic RFML datasets

The most widely adopted RFML benchmarks are synthetically generated. The RadioML datasets [1] popularized deep modulation classification by pairing many modulation types with simulated channel impairments, and remain a common baseline. More recently, TorchSig [2] provides a large-scale, PyTorch-native framework for programmatic RF signal synthesis, transforms, and model training (e.g., the Sig53 dataset), representing the current state of the art in standardized, at-scale synthetic RFML data. Such datasets are valuable for reproducibility and scale, but their channel and impairment models are fixed at generation time and do not reflect any specific deployment environment. Models trained on them frequently degrade when confronted with real hardware and propagation, motivating data that carries genuine channel effects. Importantly, these synthetic pipelines and RFEG are complementary: RFEG produces labeled OTA captures in standard I/Q formats that can be consumed directly by TorchSig-style training and evaluation workflows, supplying real-channel data to pipelines otherwise fed by synthesis.

B. Channel and propagation simulators

A parallel line of work improves the realism of simulated RF. Ray-tracing and geometry-based simulators such as DeepMIMO [3] and differentiable link-level simulators such as NVIDIA Sionna [4] model site-specific propagation, MIMO channels, and hardware impairments with increasing fidelity, and are invaluable for controlled, repeatable experimentation.

C. Over-the-air testbeds and SDR collection

Large shared testbeds such as POWDER and the broader PAWR program [5], and platforms like Colosseum [6], provide access to real or emulated RF at scale. These are powerful but heavyweight, shared, and often remote from a given group's operational environment. At the other extreme, ad-hoc single-radio captures are lightweight but hard to label and reconfigure.

D. GNU Radio and USRP-based experimentation

RFEG builds on the GNU Radio framework [7] and Ettus USRP hardware [8], which together form the open-source SDR stack. GNU Radio flowgraphs define the per-node transmit and receive chains, while the USRPs provide the RF front-ends. A distinguishing aspect of RFEG is its orchestration layer: rather than a single flowgraph, it coordinates multiple nodes from a central controller using declarative environment files, and — for narrowband collection — enforces time synchronization across nodes so that received signals can be aligned to their references.

E. Summary

Prior work tends to focus on one of three directions: scale and labels (synthetic datasets such as RadioML and TorchSig), channel realism (using RF channel and propagation simulators), or raw OTA access (large testbeds). RFEG's contribution is to combine procedural, labeled environment specification with real OTA propagation in lightweight, open-source, group-owned infrastructure. Software architecture also facilitates online training against live data. We believe this contribution is necessary to allow groups to pivot away from ungeneralizable synthetic training and testing, while mitigating the usual costs, time, labor, and subject matter expertise associated with creating large manually labeled over-the-air datasets.

III. Methods

A. Narrowband Implementation

In the narrowband implementation of the RFEG, a single node connects two Ettus USRP devices, one serving as the transmitter and the other as the receiver. Upon initiating the data collection routine, the program randomly selects from 15 modulation classes as shown in Table I. Additional parameters—including symbol rate, samples per symbol, random seed, root-raised cosine (RRC) filter span and rolloff factor, and transmit gain—can also be randomly selected from user-configured ranges. These randomly generated parameters are passed to a GNU Radio flowgraph to generate modulated signals for over-the-air transmission. The receiver operates simultaneously with the transmitter to record the transmitted data.

TABLE I: Modulation Classes

Modulation Type	Orders	Number of Classes
ASK	2, 4, 8, 16	4
PSK	2, 4, 8, 16	4
FSK	2, 4, 8, 16	4
QAM	16, 64, 256	3

The recorded data requires processing and labeling to be suitable for machine learning training. Specifically, the received signal must be aligned with the clean generated signal, which is corrupted by channel noise and impairments. To achieve this alignment, we prepend each data packet with a 13-bit Barker code interpolated by a factor of 70. This preamble enables reliable packet detection even at SNR levels below -10 dB due to its excellent autocorrelation properties.

The received signal undergoes several processing steps. First, matched filtering is applied to maximize the SNR. Next, coarse carrier frequency offset (CFO) correction is performed by sweeping frequencies within ±30 ppm of the nominal carrier frequency. The frequency yielding maximum correlation with the interpolated Barker code is selected as the estimated CFO. Since each data packet is preceded by the Barker code, we can reliably identify packet boundaries and align them with the corresponding clean reference signals. The Barker code is subsequently removed from the data, as it does not provide meaningful information for machine learning training. The aligned signals with original unfiltered received signal are then segmented into sequences of 1024 samples.

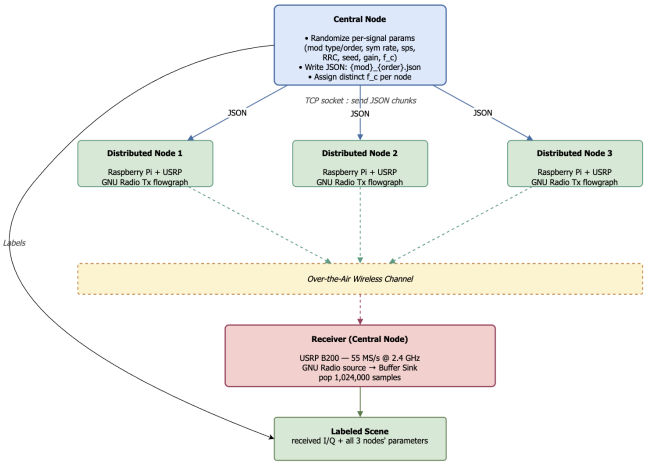

Fig. 1: Data flow diagram of wideband radio frequency environment generator.

The aligned clean and received signal pairs are stored along with their modulation class and associated metadata labels, including:

- Modulation parameters: type and order
- Signal parameters: symbol rate, samples per symbol (SPS), and TX/RX sample rates
- Pulse shaping parameters: RRC filter span and rolloff factor
- Hardware parameters: transmit gain and carrier frequency offset
- Random seed for reproducibility

The processed data and labels are saved as Python pickle files, enabling straightforward integration into machine learning pipelines.

B. Wideband Implementation

In the wideband implementation of the RFEG, a central node connects to the receiver while n distributed nodes connect to individual transmitters, as in Figure 1. The wideband RFEG generates the same modulation classes and supports the same user-configurable parameters as the narrowband implementation. However, a crucial difference is that only the received signal is saved for machine learning training. Consequently, coarse carrier frequency offset correction and Barker code preambles are omitted in signal generation and post-processing.

Upon initiating the data collection routine, the central node randomly selects the modulation type and associated parameters to generate signals for the n distributed nodes. Unlike the narrowband case, where the carrier frequency is fixed, the wideband implementation randomly selects carrier frequencies from a specified range. Each parameter set is stored in an individual JSON file for transmission to the corresponding distributed node.

```
{
  "mod_order": 8,
  "mod_type": "ASK",
  "sym_rate": 200000,
  "sps": 3,
  "samp_rate": 600000,
  "seed": 850,
  "rrc_rolloff": 0.26,
  "rrc_span": 4,
  "fc": 2415000000.0,
  "gain": 53,
}
```


Fig. 2: Example JSON configuration file sent to distributed transmitter nodes. Parameters include modulation scheme, signal characteristics, pulse shaping, carrier frequency, and hardware settings.

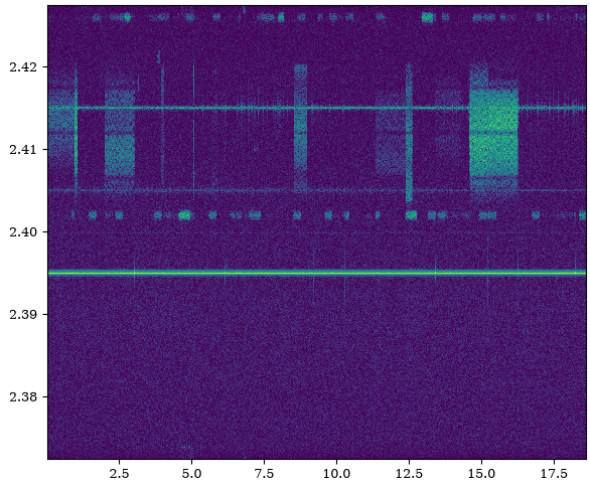

Fig. 3: A sequence captured by the sensing node for the wideband implementation. Note the three narrowband signals received at 2.395, 2.405, and 2.415 GHz are the signals that are automatically labeled by RFEG. These three signals are embedded in the ambient electromagnetic spectrum of an indoor lab environment.

The central and distributed nodes communicate via Python sockets using TCP to ensure reliable data transfer. Upon receiving a JSON file, each distributed node parses its contents, generates the corresponding waveform, and transmits at the specified carrier frequency. The receiver waits 5 seconds before initiating data capture to ensure all distributed nodes have begun transmission. For each unique environment, the receiver collects 1,024,000 samples and stores the corresponding parameters for all nodes in Python pickle files. We show an example spectrogram of a received wideband sequence in Figure 3.

C. AMC Training

We propose using a hybrid deep learning architecture that combines residual convolutional neural networks (ResNet) with transformer-based attention mechanisms for automatic modulation classification. This ResNet-Transformer model processes raw in-phase and quadrature (I/Q) samples to extract both local temporal features and long-range dependencies in the received signal.

The architecture consists of three main components: a convolutional feature extractor, a transformer encoder with positional encoding, and a classification head. The input to the network is a complex-valued signal represented as a two-channel tensor of dimension (B,2,1024), where B is the batch size, 2 represents the I and Q channels, and 1024 is the sequence length.

The ResNet performs feature extraction and begins with a stem layer comprising a 1D convolutional layer. This is followed by four residual blocks with progressively increasing channel dimensions (64, 128, 256, and 512) and spatial downsampling via strided convolutions. This residual architecture facilitates gradient flow during training and enables the learning of hierarchical feature representations. After the four residual layers, the feature maps have dimensions (B,512,128).

The output of the convolutional layers is permuted to (B,128,512) to match the transformer's expected input format (batch, sequence length, feature dimension). Sinusoidal positional encodings are added for temporal position information. The positional encoding uses sine and cosine functions of different frequencies:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{1}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{2}$$

Pos is the position, i is the dimension index, and dmodel=512. The transformer encoder consists of two layers, each with 8 attention heads, a feedforward dimension of 1024, and a dropout rate of 0.2. The multi-head self-attention mechanism enables the model to capture long-range dependencies and relationships between different temporal positions in the signal.

The transformer output is averaged across the sequence dimension via mean pooling, producing a fixed-size feature vector of dimension 512. A dropout layer (rate 0.4) is applied for regularization, followed by a fully connected layer that maps to the 15 modulation classes.

D. Experimental Setup

The ResNet-Transformer AMC model was trained using narrowband data collected over multiple days in an indoor laboratory environment. Data were acquired in the 2.4 GHz ISM band and subjected to real-world channel effects and in-band interference. To evaluate the model's generalization capability, we validated its performance on two separate test datasets: one collected in the same indoor environment as the training data (Env 1) and another collected in a slightly different indoor environment (Env 2) with different multipath characteristics and interference profiles. This validation strategy assesses both the model's ability to classify modulation schemes under familiar conditions and its robustness to environmental

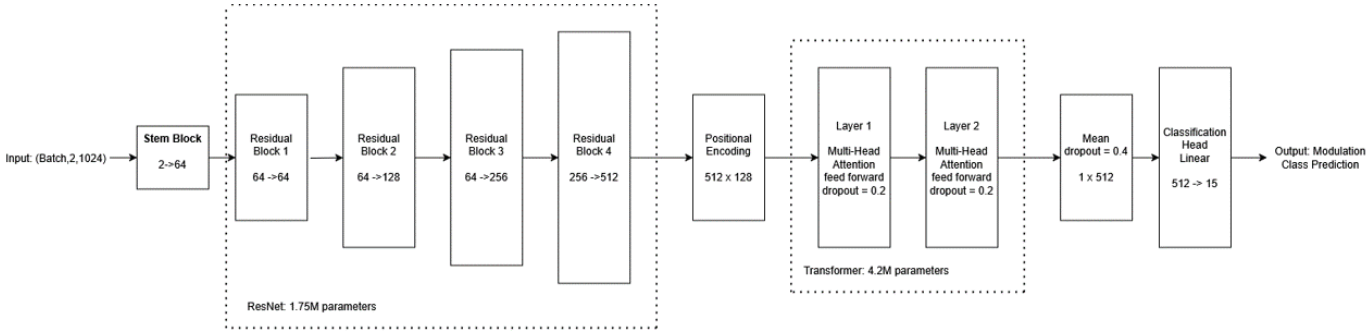

Fig. 4: ResNet-Transformer architecture for automatic modulation classification. The model consists of three main stages: (1) a convolutional feature extractor , (2) a transformer with positional encoding, and (3) a classification head. The input I/Q samples are transformed through the network to produce predictions over 15 modulation classes.

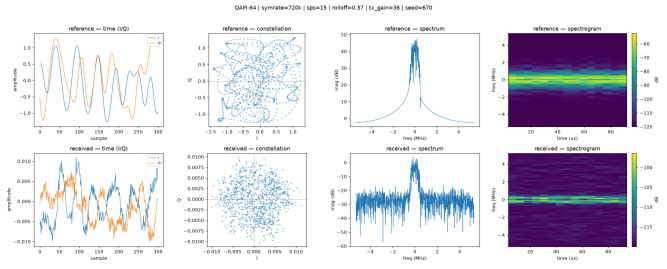

Fig. 5: An example received signal from the Narrowband Implementation, with its associated reference signal. USRP B210 SDRs were used to transmit and receive the I/Q.

variations. We show an example narrowband sequence, used as an ML sample, in Figure 5.

IV. Results and Discussion

The model achieved an overall classification accuracy of 87.08% on Env 1 and 85.34% on Env 2, demonstrating strong generalization across different propagation environments. The modest 1.74 percentage point degradation when transitioning to Env 2 indicates that the model has learned robust features that are relatively invariant to environmental changes, rather than overfitting to specific channel characteristics.

Figures 6 and 7 present the normalized confusion matrices for both validation environments.The model exhibits excellent performance on low-order schemes of 2 and 4, with accuracities over 88% in both environments. This strong performance is expected, as these modulations have more distinct constellation patterns and greater Euclidean distances between symbols. The most significant performance degradation occurs in high-order QAM schemes. QAM-64 and QAM-256 exhibit the lowest classification accuracies, with QAM-256 achieving only 60% accuracy in both environments since it requires higher SNR to work reliably.

Misclassifications predominantly occur within the same modulation family. This suggests that the model successfully learns to distinguish between modulation types but faces challenges in discriminating between different orders of the same type and tends to confuse adjacent modulations order.

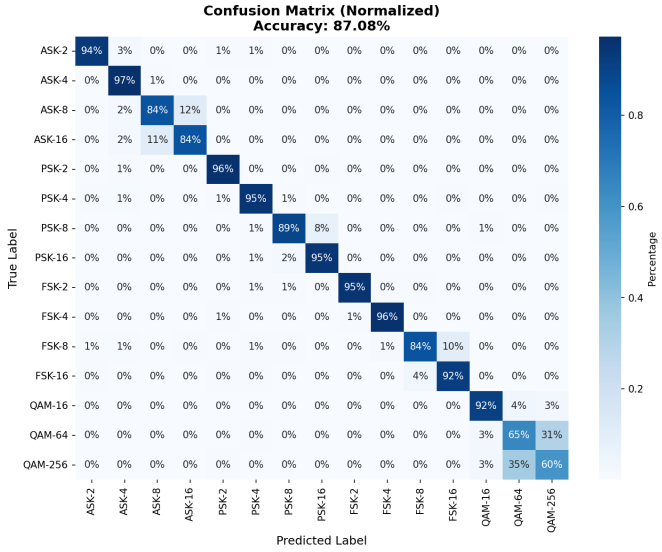

Fig. 6: Normalized confusion matrix for validation on Environment 1 (same as training environment). The model achieves 87.08% overall accuracy.

There is minimal confusion between different modulation families, which shows that the hybrid ResNet-Transformer architecture effectively captures the fundamental differences in modulation characteristics.

The strong overall performance (>85% accuracy) across both environments validates the effectiveness of the hybrid ResNet-Transformer architecture for AMC tasks. Comparing the two confusion matrices, we observe that most modulation classes maintain similar performance across environments. There are notable drops in performance in ASK-2, QAM-256, and FSK-8. These results suggest that the model has learned generalizable representations for most modulation types, though certain schemes (particularly ASK and medium-order FSK) are more sensitive to environmental variations.

To assess the generalization capability and domain adaptation characteristics of the ResNet-Transformer AMC model, we conducted cross-dataset evaluations between our custom

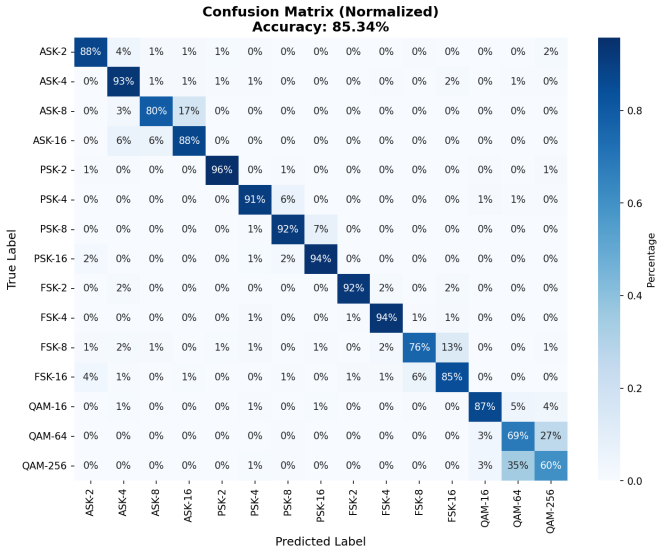

Fig. 7: Normalized confusion matrix for validation on Environment 2 (different indoor environment). The model maintains 85.34% accuracy.

dataset and the widely-used RadioML 2018.01a benchmark dataset [1]. We trained two separate models: one on our custom dataset and another on RadioML 2018.01a, then evaluated each model on all three test sets (Custom Env 1, Custom Env 2, and RadioML 2018.01a).

Table II summarizes the cross-dataset evaluation results. When trained and tested on the same dataset (in-domain evaluation), both models achieve strong performance: 87.08% on Custom Env 1 and 76.95% on RadioML 2018.01a. The slightly lower performance on RadioML can be attributed to a broader range of modulation schemes (24 classes vs. 15 in our dataset).

TABLE II: Cross-Dataset Evaluation Results

Training Dataset	Test Dataset Accuracy		
	Custom Env 1	Custom Env 2	RadioML
Custom Dataset	**87.08%**	**85.34%**	40.15%
RadioML 2018.01a	27.55%	24.21%	**76.95%**
Performance Gap	59.53%	61.13%	36.80%

The cross-dataset evaluation reveals a significant difference between the two datasets. The model trained on our custom dataset achieves only 40.15% accuracy on RadioML 2018.01a, representing a 46.93 percentage point drop from its in-domain performance. Conversely, the RadioML-trained model achieves only 27.55% and 24.21% on Custom Env 1 and Env 2, respectively, a dramatic 49.40-52.74 percentage point degradation. This substantial performance gap indicates fundamental differences in the signal characteristics between synthetic (RadioML) and over-the-air (custom) datasets.

Figure 8 and 9 illustrates the classification accuracy as a function of SNR for both training scenarios. When the

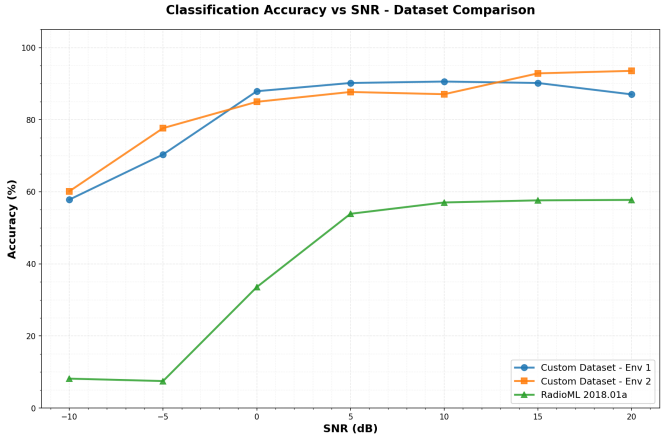

Fig. 8: SNR-dependent accuracy for model trained on custom dataset. The model maintains >60% accuracy on both custom environments but achieves only 40-58% on RadioML 2018.01a, indicating poor cross-domain generalization.

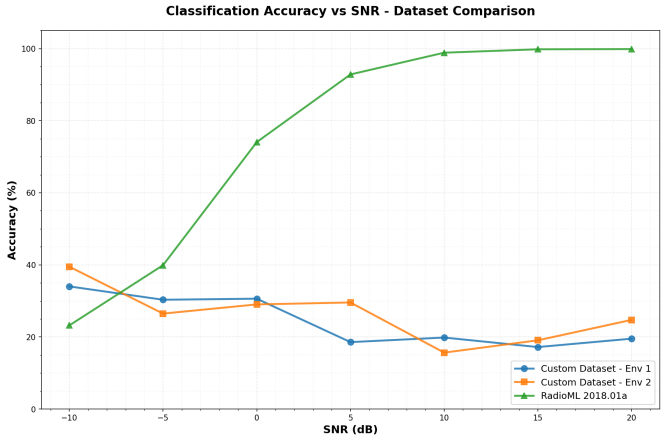

Fig. 9: SNR-dependent accuracy for model trained on RadioML 2018.01a. While achieving near-perfect accuracy on RadioML at high SNR, the model fails on custom datasets with accuracy decreasing at higher SNR—suggesting learned features are specific to synthetic signal characteristics.

custom-trained model is evaluated on RadioML, accuracy exhibits clear SNR-dependent degradation, indicating that the model has learned some generalizable SNR-related characteristics. In contrast, when the RadioML-trained model is evaluated on custom datasets, accuracy remains relatively flat across SNR levels, suggesting that the model failed to learn meaningful modulation features that transfer to real-world signals. This asymmetric generalization behavior can be attributed to the following factors:

1. Signal Generation Methodology: RadioML 2018.01a uses synthetic signal generation with simulated channel models, while our custom dataset captures real over-the-air transmissions with authentic hardware impairments, multipath propagation, and ambient interference. The model learns dataset-specific artifacts rather than modulation-invariant fea-

tures.

2. Hardware Impairments: Real USRP transmissions introduce carrier frequency offset, phase noise, I/Q imbalance, and nonlinear distortions that are absent or differently modeled in synthetic datasets. The RadioML-trained model has not encountered these real-world imperfections during training.

3. Channel Characteristics: Our custom dataset experiences frequency-selective fading and time-varying multipath in the 2.4 GHz ISM band, while RadioML uses simplified channel models. The learned features are thus optimized for different propagation environments.

4. Preprocessing Differences: The two datasets may employ different normalization schemes, sample rates, or filtering approaches, leading to distribution shifts in the input space.

5. Class Distribution: While both datasets include common modulation types, the specific implementation details (pulse shaping, symbol timing, etc.) differ, creating subtle but significant feature distribution mismatches.

These results highlight a critical challenge in deep learning-based AMC: models trained on synthetic datasets may not generalize to real-world deployments. The 40-60 percentage point performance gaps observed in our experiments suggest that:

- Benchmark performance on synthetic datasets (e.g., RadioML) may significantly overestimate real-world capability
- Domain adaptation techniques are essential for transferring models between synthetic and real environments
- Training on diverse, real-world data is crucial for robust AMC systems
- The AMC community should prioritize the development of realistic datasets that capture authentic hardware and propagation effects

Our custom dataset, collected with real USRP hardware in operational environments, provides a more realistic testbed for evaluating AMC algorithms intended for practical deployment.

V. Conclusions

We presented the RF Environment Generator (RFEG), an open-source system for procedurally generating labeled over-the-air (OTA) datasets for radio frequency machine learning. Rather than approximating the wireless channel with mathematical models, RFEG realizes user-specified environments directly in the physical electromagnetic environment: a controller parameterizes each emitted waveform, distributed USRP emitters driven by GNU Radio transmit those waveforms, and a receiver captures the aggregate signal while the known transmit parameters serve as ground-truth labels. The system operates in a narrowband mode, which uses a Barker-code preamble, matched filtering, and a carrier-frequency-offset search to align each received signal with its exact transmitted reference, and a wideband mode, which generates multi-emitter environments across frequency for detection- and classification-oriented tasks.

We exercised the narrowband datasets with a ResNet-Transformer automatic modulation classifier spanning 15 modulation classes. The model achieved over 85% accuracy across two distinct indoor environments, indicating that it learns features that are largely robust to changes in multipath and interference. By making realistic, labeled data inexpensive to generate, RFEG directly addresses this gap while avoiding the cost, time, and manual-labeling effort that OTA collection has traditionally required.

Several directions remain for future work. The wideband mode enables spectral object-detection and classification tasks in which multiple emitters are localized and identified in the time–frequency plane; in the future, there is a need to synchronize the distributed nodes such that freuqency hopping, or bursty time patterns can be realized in the multi-band setup. The receiver architecture also generalizes naturally to multi-element arrays, opening the door to array-based sensing and spatial processing. Finally, because RFEG produces labeled data as it is collected, it provides a foundation for online RFML—training models against a live, evolving electromagnetic environment rather than a fixed corpus. We release RFEG as open-source infrastructure so that other research groups can generate datasets embedded in their own operational environments.

References

[1] T.J. O'Shea, "RadioML2016.10A/10B/10C", IEEE Dataport, July 21, 2025, doi:10.21227/wz93-h307

[2] L. Boegner et al., 'Large Scale Radio Frequency Wideband Signal Detection & Recognition', arXiv [eess.SP]. 2022.

[3] J. Hoydis et al., 'Sionna: An Open-Source Library for Next-Generation Physical Layer Research', arXiv [cs.IT]. 2023.

[4] A. Alkhateeb, 'DeepMIMO: A Generic Deep Learning Dataset for Millimeter Wave and Massive MIMO Applications', arXiv [cs.IT]. 2019.

[5] J. Breen et al., 'Powder: Platform for Open Wireless Data-driven Experimental Research', Computer Networks, vol. 197, p. 108281, 2021.

[6] L. Bonati et al., 'Colosseum: Large-Scale Wireless Experimentation Through Hardware-in-the-Loop Network Emulation', in Proceedings of IEEE DySPAN, Virtual Conference, December 2021.

[7] GNU Radio Development Team, 'GNU Radio'.

[8] Ettus Research, 'USRP Hardware Driver (UHD)', GitHub Repository. GitHub.