RadioShift: A Framework Measuring the Robustness of Lightweight Neural Networks over RF Signals

Anagh Mishra[†], Phu Le[†], Ryan Evans[†], Nirnimesh Ghose[§], Boyang Wang[†]
[†]University of Cincinnati, [§]University of Nebraska-Lincoln
{*mishraag, lepq, evans2ra*}*@mail.uc.edu, nghose@unl.edu, boyang.wang@uc.edu*

***Abstract*—Modulation classification classifies modulation schemes on RF signals without prior coordination between a transmitter and a receiver. Although current lightweight neural networks perform well on standard simulated datasets, their robustness given domain shifts as well as in real-world environments remains unverified. This study investigates the robustness of lightweight neural networks running on FPGAs when exposed to severe domain shifts in RF signals. We produce simulated RF signals with various channel fading and hardware imperfections and collect real-world I/Q frames from USRPs. We generate hardware implementations of quantized neural networks with the FINN framework and perform cross-domain testing with these quantized neural networks running on FPGAs. Our findings demonstrate that (1) domain shifts between training RF data and test RF data *indeed* cause significant accuracy degradation; (2) discrepancy in *hardware imperfections* can lead to more severe accuracy degradation compared to *channel fading* discrepancy; (3) accuracy degradation is not even across modulation types, where predictions on a subset of modulations may still perform reasonably well; (4) the accuracy reported in FPGAs is *not consistent* with the one observed in GPUs, highlighting the importance of reporting results in FPGAs rather than using GPUs only for RF classifications.**

I. Introduction

Modulation classification can assist a receiver identify the specific modulation used by a transmitter on Radio Frequency (RF) signals in wireless communications [1], [2], [3]. It has a wide range of applications, such as in military, spectrum sharing, cognitive radio, and signal decoding in non-cooperative scenarios [4], [5]. Existing research [1], [6] shows that deep neural networks can efficiently perform modulation classification over raw I/Q samples in RF signals.

However, neural networks are often computationally intensive and require Graphics Processing Units (GPUs), raising significant challenges for their deployment on wireless devices to process RF signals in real time. Several recent studies [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] have investigated approaches to optimize and obtain *lightweight neural networks* for modulation classifications. Specifically, methods, including pruning [14], [17], quantization [7], [11], and hardware-level optimization on Field-Programmable Gate Array (FPGA) [9], [8], [12], [13], [18], have been proposed for deep learning modulation classification. These studies generally aim to optimize a neural network by reducing its size while minimizing performance degradation in accuracy.

Despite promising results in optimizing neural networks, these studies often report accuracy within a single dataset only but lack rigorous evaluations and insights on the *robustness of lightweight neural networks* given domain shifts between training and test RF signals across different datasets. Due to the diverse nature of wireless communications, signals are distorted by multiple factors, including wireless channels (e.g., fading and noise) and impairments (e.g., carrier frequency offset, phase offset, Doppler shift, etc.) [4], [2], [6]. Establishing a dataset of RF signals to include all the possible distortions is nearly infeasible. In other words, facing unknown domain shifts between training and test data is inevitable for a neural network processing RF signals. Therefore, understanding the robustness of lightweight neural networks given domain shifts is also critical to design decisions in addition to other factors, such as power consumption and throughput.

Our Contributions. In this paper, to fill the research gap, we empirically investigate the robustness of lightweight neural networks for modulation classification. We answer the following open research question

- ***RQ1: To what extent do domain shifts in RF signals cause accuracy degradation on lightweight neural networks for modulation classification?***
- ***RQ2: Which factor in wireless communication can cause more severe accuracy degradation?***

To answer the research question, we build an *end-to-end framework*, named *RadioShift*, which can automatically generate simulated I/Q frames with various domain shifts, capture real-world I/Q frames, train lightweight neural networks, and test them on FPGAs given the RF signals we collect.

Specifically, we examine domain shifts on RF signals caused by two common factors, *channel fading* (Rician v.s. Rayleigh) and *hardware imperfections* (minor v.s. severe in parts-per-million). We generate our simulated RF signals with Matlab including different domain shifts and capture real-world RF signals with two Ettus Research Universal Software Radio Peripherals (USRPs). We examine lightweight neural networks that can be obtained by *quantization*, which is one of the common neural network compression techniques. We use a lightweight VGG-10 with 160,079 parameters as a baseline neural network and obtain quantized models. We implement all the lightweight neural networks with AMD/Xilinx Fast, Scalable Quantized Neural Network Inference (FINN) framework [19] and deploy and report results of these neural networks on an AMD Zynq ZCU104 FPGA Evaluation Board.

Our Efforts and Findings. We summarize our efforts and

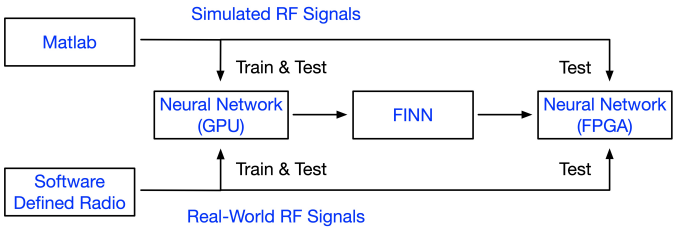


Fig. 1: Overview of Our RadioShift Framework

findings below

- We generate a dataset, named RadioShift, which contains 3,932,160 simulated I/Q frames from 15 modulation classes across multiple Signal-to-Noise Ratios (SNRs), from 0db to 30db. In addition, our dataset covers hardware imperfections and channel conditions that were not included in existing simulated datasets and offers new opportunities for the research community to investigate the robustness of neural networks for modulation classification. Moreover, 16,384 real-world I/Q frames from 4 modulation schemes were also included in our dataset.
- Our experimental results suggest that ① domain shifts between training RF data and test RF data *indeed* cause significant accuracy degradation (e.g. dropping from 82.3% to 29.4%) for lightweight neural networks that perform modulation classification. ② The discrepancy in *hardware imperfections* can lead to significant accuracy degradation while the *channel fading* discrepancy causes less accuracy degradation. ③ The accuracy degradation is *not even across all modulation classes*, where a neural network can still predict relatively well over a small set of modulations, including FM and BPSK. ④ The accuracy reported in FPGAs is *not consistent* with the one recorded in GPUs, which highlights the importance of reporting results in FPGAs to fairly understand the performance of a neural network in future research.
- Our framework can also be expanded, tuned, and utilized to study the robustness of lightweight neural networks for other classification tasks over RF signals.

Reproducibility. Our source code and datasets can be found at `https://github.com/UCdasec/RadioShift`

II. Related Work

Lightweight Neural Networks for Modulation Classification. Several studies [7], [8], [9], [10], [11], [12], [13], [18] focus on optimizing the size and performance of neural networks by applying quantization and deploying compressed neural networks on FPGAs. Tridgell et al. [7] apply ternary-weight neural networks for modulation classification, where weights are represented with {1, 0, -1} only. Their experimental results show that a compressed VGG10 with 490,000 parameters in ternary weights can achieve up to 82% accuracy given traces of 30 dB from RadioML 2018 dataset on Xilinx ZCU111 board. Den Boer et al. [8] leverage High-Level Synthesis to optimize the implementation of neural networks on FPGAs. Their model carries 13,840 parameters and can achieve 71.49% accuracy at 30dB. Kumar et al. [11] first apply quantization and then use unstructured pruning to set near zero weights as zero weights. They utilize the FINN framework to evaluate tiny neural networks on a Xlinix ZCU111 board.

Jentzsch et al. [9] leverage the Xlinix FINN framework to quantize the weights and activations of a VGG10 to 4 bits and still achieve 94% accuracy given traces of 30 dB in RadioML 2018 dataset. The quantized VGG10 consists of 72,000 parameters only. Jung et al. [12] leverage Short-Time Fourier transform to convert I/Q samples in the time domain into spectrograms, and then pass them to an on-chip customized neural network synthesized in 28nm CMOS. Woo et al. [13] also adopt ternary weights in a lightweight neural network (MobileNetV3) and implement compressed neural networks on a Xilinx ZCU102 board. The quantized MobileNetV3 includes 0.4 million parameters only. MacLellan et al. [18] build a Radio Frequency System on Chip, which integrates RF signal receiver with a quantized neural network for modulation classification.

Besides leveraging quantization, pruning has also been proposed in recent studies [14], [15], [16], [17] to optimize neural networks for modulation classification. Chen et al. [16] utilize single-short pre-defined structured pruning, which measures channel importance based on the cosine similarity of weights on convolutional layers and batch normalization layers. Ninan et al. [17] adopt single-short automatic structured pruning, which derives a customized pruning rate per layer for neural networks processing RF signals. Their evaluation show that a pruned neural network with 12,505 parameters can achieve 79% accuracy on RadioML 2018 dataset for RF signals with 6dB and higher on a Xlinix ZCU104 board.

However, none of these studies comprehensively evaluate the robustness of lightweight neural networks given domain shifts in RF signals.

Robustness of Neural Networks for Modulation Classification. Given unknown domain shifts in RF signals, one typical approach for enhancing the robustness of a neural network is to transform raw I/Q samples in the time domain to other domains and leverage data represented in other domains as inputs to one or multiple neural networks. For instance, Liu et al. [6] propose to transform I/Q samples into phase, angular, and frequency representations and pass all the three representations into a single CNN based on ResNet-26. They examine the robustness of the ResNet-26 with 10 different datasets, where each dataset has unique RF domain shifts based on a potential real-world communication scenario, and demonstrate that the proposed ResNet-26 has a lower accuracy drop compared to other methods given domain shifts across datasets.

Another approach is to pre-process RF signals to mitigate distortions before passing them to a neural network for modulation classification. For example, Yashashwi et al. [20] propose to remove carrier frequency offsets and phase noise from RF signals with a neural network for pre-processing. Shi et al. [21] focus on removing distortion caused by phase offsets. Transfer learning has also been examined as an alternative approach to address domain shifts in modulation classification [22].

Unfortunately, none of these studies investigate the robustness of neural networks when neural networks are significantly compressed and deployed on FPGAs.

III. Background

Problem Formulation. An I/Q frame consists of k I/Q samples generated from a modulation scheme, where each I/Q sample contains a pair of numbers (I for in-phase, Q for quadrature). There are m modulation schemes. Given a dataset of $n \times m$ I/Q frames with n I/Q frames per modulation, we divide the dataset into two sets, one training set and one test set. A classifier F, e.g., a neural network, is trained with the training set of I/Q frames. The performance of this classifier is then evaluated over the test set of I/Q frames.

Metric. We leverage *accuracy* as the metric to measure the performance of a neural network in modulation classification. During the testing, given x I/Q frames, if y I/Q frames are predicted correctly by a trained classifier, the accuracy is calculated as y/x. A higher accuracy indicates a stronger capability in modulation classification.

IV. RF Signal Generation and Capture

Simulated RF Signal. To rigorously investigate model robustness against domain shifts, we develop a dataset generation framework that provides granular control over RF channel configurations by using Matlab[1]. Specifically, it allows for the parameterization of multipath fading, including Line-of-Sight and Non-Line-of-Sight environments, and hardware imperfections in oscillators, which lead to carrier frequency offset and sampling rate offset.

The signal generation process is bifurcated into digital and analog data paths. For digital modulation schemes, the pipeline starts by generating random integers in the range of $[0, M-1]$, corresponding to the specific modulation order M (e.g., $M = 8$ for 8PSK). These integers are mapped to complex symbols, upsampled to achieve the target samples-per-symbol rate, and then passed through a root-raised-cosine filter for pulse shaping.

In contrast, analog modulation schemes utilize continuous-time audio sequences (e.g., .wav files) as the underlying data source. Because these signals are inherently analog, they are directly modulated using the target technique without passing through the digital pulse-shaping stage.

Following baseband generation, the signals undergo over-the-air channel modeling, where environmental and hardware impairments are injected. To simulate multipath fading, the framework applies a Rician channel model for direct Line-of-Sight communications (utilizing a defined K-factor for the dominant path) or a Rayleigh channel model for strictly Non-Line-of-Sight environments. This stage incorporates physical variables such as path delays, path gains, and Doppler shifts.

To emulate *hardware imperfections*, we model *oscillator stability* based on standard radio specifications, quantified in parts-per-million (PPM). The PPM stability metric is mathematically translated into precise carrier frequency offset and sampling rate offset values. Following the injection of these impairments, Additive White Gaussian Noise is added to the signal to achieve the targeted SNR. Finally, the continuous signal stream is segmented into frames of 1,024 I/Q samples, which are subsequently annotated with their corresponding modulation and SNR labels.

[1]https://www.mathworks.com/help/comm/ug/modulation-classification-with-deep-learning.html

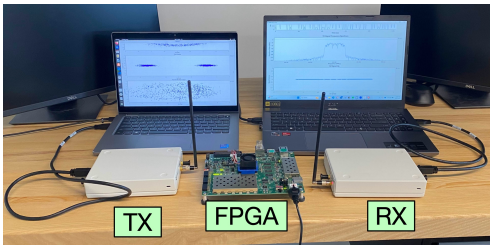


Fig. 2: Our USRP Data Collection Setup (The FPGA is not required for data collection but runs classification later.

Real-World RF Signal Collection. To validate model robustness against physical RF impairments, we deploy an Over-the-Air testbed utilizing two NI USRP-2900 Software Defined Radios driven by a custom GNU Radio DSP pipeline. Operations were conducted in the 915 MHz ISM (Industrial, Scientific and Medical) band at a 1 MHz sampling rate. Our testbed supports four modulation schemes: including BPSK, QPSK, 8PSK, and FM. To capture analog FM signals, we implement a dedicated receiver designed to tune into local broadcast stations, utilizing a rational resampler and a wideband FM receiver block routed to an audio sink.

For the digital PSK schemes, we establish a complete transmitter and receiver architecture. The TX maps input integers to complex symbols based on the specified PSK scheme, applies a Root Raised Cosine filter for pulse shaping, and transmits the signal at 915 MHz via a USRP sink. The RX front-end utilizes physical offset tuning (915 MHz) coupled with a digital DC blocker to mitigate Local Oscillator leakage. An Automatic Gain Control block normalizes the signal amplitude before matched root-raised-cosine filtering. To combat hardware clock drift, timing recovery is achieved via a Gardner Timing Error Detection algorithm, which continuously interpolates optimal sampling instants. Subsequently, a 2nd-order Costas Loop provides fine frequency correction and phase tracking, feeding into a hard-decision constellation decoder. This unified DSP architecture is dynamically configurable, supporting multiple modulation schemes and operational modes, such as discrete message reception or continuous signal monitoring.

To verify link reliability, we transmit a formatted payload with strict header protocols over multiple repetitions. After confirming successful packet reception for BPSK, QPSK, and 8PSK, we initiate the real-world dataset capture by transmitting random symbol sequences. We only capture I/Q samples after automatic gain control and direct current blocking (prior to timing recovery).

V. Our Dataset

RadioShift Dataset. Our RadioShift dataset contains simulated I/Q frames from 15 modulation classes, including 12

digital modulation classes {BPSK, QPSK, 8PSK, 16QAM, 32QAM, 64QAM, 128QAM, 256QAM, 16APSK, 32APSK, 64APSK, 128APSK} and 3 analog modulation classes {FM, AM-DSB-SC, and AM-SSB-SC}. Each I/Q frame consists of 1,024 samples, and there are 8 samples per symbol. The sampling rate is 200 kHz. We adopt two center frequencies, 902 MHz (digital modulations) and 100 Mhz (analog modulations) respectively. We apply 16 SNR levels ranging from 0dB to 30dB with an 2dB increment per SNR level (i.e., {0dB, 2dB, 4dB, ... 30dB}).

RadioShift dataset consists of four subsets, Ric-PPM1, Ric-PPM20, Ray-PPM1, and Ray-PPM20 (Ray: Rayleigh, Ric: Rician) based on the selection of channel condition and hardware imperfections (in ppm) when we generate I/Q frames. For instance, the I/Q frames in Ray-PPM20 are generated by using Rayleigh as the channel and a ppm (parts per million) value randomly generated from 0 to 20 per I/Q frame. While all the I/Q frames in Ray-PPM1 are generated with a ppm value randomly generated from 0 to 1. In each subset, we generate 4,096 I/Q frames per modulation per SNR level, which leads to 983,040 I/Q frames in each subset and a total of 3,932,160 I/Q frames in our RadioShift dataset.

Our I/Q samples are saved in both float32 format and int8 format. The samples in int8 format are normalized from the ones in float32 format with min-max normalization.

It is worth mentioning that the RadioML 2018 dataset [23] is one of the widely used datasets for modulation classification, which contains over 2.5 million simulated I/Q frames from 24 modulations. However, it does not provide I/Q frames with domain shifts caused by different factors. As a result, it cannot serve the purpose of our research investigation.

RadioShift-USRP Dataset. We also collect I/Q samples from the two USRPs in our lab setup described in the previous section. Specifically, we collect I/Q samples from 4 modulations, including BPSK, QPSK, 8PSK, and FM. We collect 4,096 frames from each modulation and each frame contains 1,024 samples, which leads to 16,384 frames. These frames are saved in float32 and normalized to int8 with min-max normalized (min: -3.2 and max: 3.2). We denote this dataset as RadioShift-USRP.

VI. Evaluation

Neural Networks. We choose a lightweight VGG model as our baseline architecture for modulation classification. Specifically, we use a VGG10 model, which is a CNN architecture. The model consists of 7 convolutional blocks, where each block includes 1 convolutional layer, 1 batch normalization layer, 1 Relu layer, and 1 maxpool layer. Afterwards, there are 2 linear blocks, where each block includes 1 linear layer, 1 batch normalization layer, and 1 ReLU layer. There is 1 final linear layer at the end which maps the prediction value to each class. The hyperparameters of the baseline architecture are summarized in Table I.This neural network contains 160,079 parameters.

Experimental Setting. We run all our experiments on a server with Ubuntu 22.04, Intel i9 CPU, 128 GB memory, and a NVIDIA GeForce RTX Titan GPU. For each (baseline) neural network, we train it for 150 epochs and apply an early stop with 10 epochs if the training is beyond 50 epochs. The batch size in training is 1024.

TABLE I: Hyperparameters of the Lightweight VGG10

Conv_Block_1	in channels: 2 ; out channels: 64; kernel: (3, 1); BatchNorm; Relu; MaxPool
Conv_Block_2	in channels: 64; out channels: 64; kernel: (3, 1); BatchNorm; Relu; MaxPool
Conv_Block_3	in channels: 64; out channels: 64; kernel: (3, 1); BatchNorm; Relu; MaxPool
Conv_Block_4	in channels: 64; out channels: 64; kernel: (3, 1); BatchNorm; Relu; MaxPool
Conv_Block_5	in channels: 64; out channels: 64; kernel: (3, 1); BatchNorm; Relu; MaxPool
Conv_Block_6	in channels: 64; out channels: 64; kernel: (3, 1); BatchNorm; Relu; MaxPool
Conv_Block_7	in channels: 64; out channels: 64; kernel: (3, 1); BatchNorm; Relu; MaxPool
Flatten	Applied after Conv_Block_7
Linear_Block_1	in features: 512 ; out features: 128; BatchNorm; Relu
Linear_Block_2	in features: 128 ; out features: 128; BatchNorm; Relu
Linear_Layer	in features: 128; out features: *No. of classes*;

We use PyTorch to implement neural networks. Unless specifically mentioned, all the neural networks with float32 parameters are trained and tested with I/Q samples in the int8 format on GPUs. All the quantized neural networks are trained and tested with I/Q samples in the int8 format to facilitate their testings on FPGAs. We always split each dataset into training, validation, and testing with a ratio of 8:1:1.

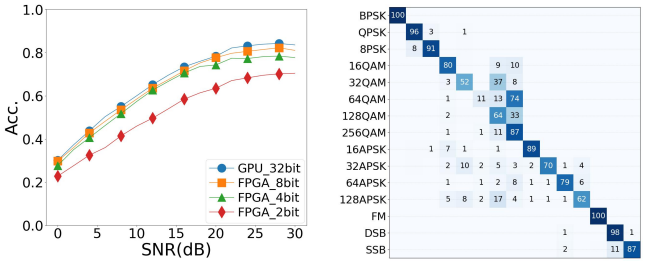
We adopt FINN as the framework to obtain each tiny neural network, where *quantize-aware training* was utilized to initially obtain a compressed neural network running on GPU and then transformed into a corresponding hardware implementation on FPGA by going through the FINN framework. In essence, FINN is a High-Level Synthesis Tool transforming the design of a neural network in high-level software (e.g., Python) to low-level hardware (e.g., Verilog). FINN also generates the synthesized hardware design code in Verilog and VHDL, which allows us to report the utilize and power consumption of a compressed neural network on a specific board. We utilize an AMD Zynq ZCU104 FPGA Evaluation Board to report the performance of neural networks on FPGAs.

Experiment 1. Same Dataset Testing. We first train the baseline VGG (weights in float32) with I/Q frames from Ric-PPM1, and then test this baseline with test frames from Ric-PPM1. Then, we train compressed neural networks (8-bit , 4-bit, and 2-bit) respectively, and perform the testing on GPUs and FPGAs. In addition, we also train neural networks with Ric-PPM20, Ray-PPM1, and Ray-PPM20 respectively, and then examine their performance given test traces from the same datasets. We summarize the results in Table II. We first find that the accuracy of a neural network increases as the SNR increases and maintains at a similar level after SNR reaches 20 dB or higher (as shown in Fig. 3), which is consistent as existing studies in modulation classification [2], [3].

Findings 1. Based on the results from Table II, we observe that, **(1.a)** *the accuracy of a neural network on GPUs*

TABLE II: Performance of Neural Networks (Quantization, FINN, Train & Test with Same Dataset)

Train & Test Dataset	Baseline (ft32)	8bit (GPU)	8bit (FPGA)	4bit (GPU)	4bit (FPGA)	2bit (GPU)	2bit (FPGA)
Ric-PPM1 (30dB)	83.4%	81.0%	84.0%	80.2%	82.2%	70.5%	70.5%
Ric-PPM1 (≥6dB)	63.6%	61.0%	66.6%	59.7%	64.5%	53.3%	54.9%
Ric-PPM20 (30dB)	80.7%	75.1%	77.9%	77.5%	81.0%	69.6%	69.4%
Ric-PPM20 (≥6dB)	60.3%	56.0%	61.5%	58.2%	63.7%	50.8%	52.9%
Ray-PPM1 (30dB)	85.1%	84.1%	86.5%	80.5%	82.3%	63.4%	63.0%
Ray-PPM1 (≥6dB)	66.2%	64.3%	71.0%	62.2%	67.2%	51.4%	53.1%
Ray-PPM20 (30dB)	83.6%	81.0%	85.1%	77.7%	79.5%	70.5%	71.7%
Ray-PPM20 (≥6dB)	70.6%	68.8%	75.5%	66.6%	71.2%	56.3%	58.7%



(a) Accuracy of Quantized Neural Networks (from 0dB to 30dB)

(b) 4-bit Nerual Network on FPGA, Confusion Matrix at 30dB

Fig. 3: Train and Test with Ray-ppm20

TABLE III: Utilization of Quantized VGG10 on AMD ZCU104 Board (part: xczu7ev-ffvc1156-2-e), Train with Ric-PPM20. Power is reported in the form of Dynamic‖Static‖Total.

	8bit (FPGA)	4bit (FPGA)	2bit (FPGA)
Power (W)	2.1‖0.6‖2.7	0.9‖0.6‖1.5	0.5‖0.6‖1.1
Max Frequency	300 MHz	300 MHz	300 MHz
LUTs	25,512 (11.1%)	15,484 (6.7%)	14,395 (6.2%)
LUTRAM	1,465 (1.4%)	979 (1.0%)	498 (0.5%)
Flip-Flops	45,380 (9.9%)	23,237 (5.0%)	21,802 (4.7%)
BRAM	208.5 (66.8%)	69 (22.1%)	50.5 (16.4%)
DSP	138 (8.0%)	86 (5.0%)	0 (0.0%)
IO	30 (8.3%)	30 (8.3%)	30 (8.3%)

and FPGAs, overall, decreases, when we further quantize the weights in a neural network from 8 bits to 2 bits. This is an expected tradeoff as the weights are more quantized in order to reduce power consumption of a neural network. In general, 4-bit neural networks achieve a promising balance between accuracy and power consumption, where these neural networks can still maintain a similar level of accuracy compared to their 8-bit models but can reduce the total power consumption by 44% (2.7W to 1.5W). On the other hand, 2-bit neural networks introduce significant accuracy decrease (11%∼23% drop) compared to 8-bit models.

(1.b) *The accuracy of 8-bit neural networks on FPGAs are almost the same or even slightly higher than the accuracy of their baseline models with float32 weights.* This discrepancy is expected because one uses float32 and one uses int8 for their weights. **(1.c)** *The accuracy of a neural network on the FPGA is similar but not exactly the same as the accuracy of its counterpart on the GPU.* This is due to the transformations of the FINN framework, such as internal rounding, where a quantized neural network on the FPGA is not exactly the same as its original version on the GPU in order to facilitate its implementation at the hardware level. These findings are consistent across all of our datasets.

TABLE IV: Performance of Neural Networks (Quantization, FINN, 30dB only, Train and Test with Different Datasets)

Train	Test	Baseline (ft32)	8bit (FPGA)	4bit (FPGA)	2bit (FPGA)
Ric-PPM1	Ric-PPM1	83.4%	84.0%	82.2%	70.5%
	Ric-PPM20	35.7%	35.0%	34.3%	31.3%
	Ray-PPM1	76.6%	73.7%	72.9%	63.1%
	Ray-PPM20	36.2%	36.0%	35.0%	32.1%
Ric-PPM20	Ric-PPM1	75.0%	71.5%	71.6%	62.7%
	Ric-PPM20	80.7%	77.9%	81.0%	69.4%
	Ray-PPM1	78.4%	74.2%	76.0%	65.2%
	Ray-PPM20	80.4%	76.5%	78.6%	70.1%
Ray-PPM1	Ric-PPM1	62.5%	60.3%	62.5%	47.0%
	Ric-PPM20	29.1%	29.2%	29.4%	26.6%
	Ray-PPM1	85.1%	86.5%	82.3%	63.0%
	Ray-PPM20	29.4%	30.2%	29.6%	26.8%
Ray-PPM20	Ric-PPM1	76.2%	75.7%	70.3%	63.0%
	Ric-PPM20	82.0%	80.2%	76.5%	69.3%
	Ray-PPM1	80.8%	78.8%	74.7%	66.6%
	Ray-PPM20	83.6%	85.1%	79.5%	71.7%

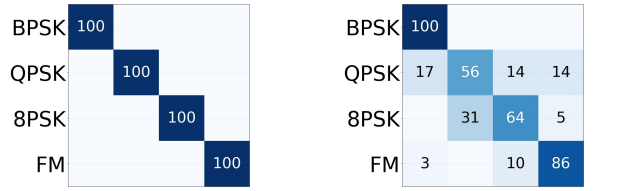
Experiment 2: Cross Dataset Testing. We examine the accuracy of quantized neural networks in the cross dataset scenario, where training and testing frames are not from the same dataset. Specifically, we take the trained neural networks from the previous experiment and test them by providing test frames from other datasets. For ease of presentation, we focus on frames from 30dB only in this experiment.

Findings 2. We find that **(2.a)** domain shifts between training and test data clearly affect the accuracy of neural networks (both the baselines on the GPU and compressed ones on the FPGA). **(2.b)** The discrepancy on channel fading alone introduces less severe accuracy drops when training frames are from one fading while the test frames are produced by another channel fading but with the same hardware imperfections (e.g., train with Ric-PPM1 and test with Ray-PPM1). On the other hand, **(2.c)** the discrepancy on hardware imperfections alone causes significant accuracy drops, especially when training data does not include severe ppms while test data has. For instance, when a 4-bit neural network is trained by Ray-PPM1, its accuracy achieves 82.5% given test frames from Ray-PPM1 but drops to 29.6% given test frames from Ray-PPM20.

(2.d) More importantly, we observe that, given the same domain shifts, the accuracy drop of a compressed neural network can be inconsistent compared to the drop of its baseline model. For instance, when we train with Ric-PPM20 and test with Ric-PPM1, the baseline model has an accuracy drop of 5.7% (from 80.7% to 75.0%) but its 4-bit model has an accuracy drop of 9.4% (from 81.0% to 71.6%). Therefore, *we suggest*

TABLE V: Performance of Neural Networks (Quantization, FINN, 4 Modulations only, Train with Simulated I/Q and Test with Real-World I/Q)

Train	Test	Baseline (ft32)	8bit (FPGA)	4bit (FPGA)	2bit (FPGA)
Ric-PPM20	Ric-PPM20	85.4%	85.3%	85.9%	83.1%
	Ray-PPM20	84.9%	84.4%	85.2%	82.2%
	USRPs	68.2%	65.7%	76.2%	63.3%
Ray-PPM20	Ric-PPM20	85.2%	85.1%	84.5%	83.2%
	Ray-PPM20	84.7%	84.4%	84.0%	82.6%
	USRPs	59.2%	70.7%	65.5%	61.9%
USRPs	USRPs	100.0%	100.0%	100.0%	100.0%



(a) Train: USRP, Test: USRP (b) Train: Ric-PPM20, Test: USRP

Fig. 4: Confusion Matrix (4-bit FPGA)

that future studies should always report results of lightweight neural networks on FPGAs rather than the ones on GPUs to derive more accurate research findings.

Experiment 3: Cross Dataset Testing - Real-World I/Q Samples. We further test the robustness of our lightweight nerual networks over real-world I/Q samples we collect from USRPs from four modulations, BPSK, QPSK, 8PSK, and FM. We first train neural networks from simulated I/Q frames (4 modulations only, 0dB to 30 dB) and then test with simulated I/Q frames and real-world I/Q frames from USRPs over these four modulations. In addition, we also train neural networks with real-world I/Q frames and test them with real-world I/Q frames.

Our results show that lightweight neural networks trained and tested on real-world I/Q frames perform extremely well achieving 100% accuracy. In addition, lightweight neural networks trained using simulated I/Q frames are still practical but face *mild accuracy degradation* when tested with real-world I/Q frames from USRPs. Confusion matrices show that *accuracy degradation is not even*, where neural networks still perform well over BPSK and FM.

VII. Conclusion

We build an end-to-end framework that can measure the robustness of lightweight neural networks over RF signals by being able to generate simulated I/Q frames with various domain shifts, capture real-world I/Q frames, and deploy neural networks on FPGAs for real-world evaluations. Our future work include expanding real-world I/Q samples collection with more modulation schemes and exploring various neural network architectures.

Acknowledgments

The authors thank reviewers for their comments and suggestions. This work was partially supported by National Science Foundation (CNS-2225160 and CNS-2225161).

References

[1] T. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE JOURNAL OF SELECTED TOPICS IN SIGNAL PROCESSING*, vol. 12, no. 1, 2018.

[2] B. Jdid, K. Hassan, I. Dayoub, W. H. Lim, and M. Mokayef, "Machine Learning Based Automatic Modulation Recognition for Wireless Communications: A Comprehensive Survey," *IEEE Access*, 2021.

[3] T. Huynh-The, Q. Pham, T. Nguyen, T. T. Nguyen, R. Ruby, M. Zeng, and D. Kim, "Automatic Modulation Classification: A Deep Architecture Survey," *IEEE Access*, 2021.

[4] T. Erpek, T. J. O'Shea, Y. E. Sagduyu, Y. Shi, and T. C. Clancy, "Deep Learning for Wireless Communications," https://arxiv.org/pdf/2005.06068.

[5] N. Hoque and H. Rahbari, "Circumventing the Defense against Modulation Classification Attacks," in *Proc. of ACM WiSec'23*, 2023.

[6] D. Liu, K. Ergun, and T. S. Rosing, "Torwards A Robust and Efficient Classifier for Real World Radio Signal Modulation Classification," in *Proc. of 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'23)*, 2023.

[7] S. Tridgell, D. Boland, P. H. Leong, R. Kastner, A. Khodamoradi, and Siddhartha, "Real-Time Automatic Modulation Classification using RF-SoC," in *Proc. of IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW 2020)*, 2020.

[8] H. den Boer, R. W. D. Muller, S. Wong, and V. Voogt, "FPGA-based Deep Learning Accelerator for RF Applications," in *Proc. of MILCOM'21*, 2021.

[9] F. Jentzsch, Y. Unuroglu, A. Pappalardo, M. Blott, and M. Platzner, "RadioML Meets FINN: Enabling Future RF Applications with FPGA Streaming Architectures," *IEEE Micro*, vol. 42, no. 6, 2022.

[10] J. Rosa, D. Granhao, G. Carvalho, T. Goncalves, M. Figueiredo, L. C. Bento, N. Paulino, and L. M. Pessoa, "BacalhauNet: A Tiny CNN for Lightning-Fast Modulation Classification," *ITU Journal on Future and Evoling Technologies*, vol. 3, no. 2, 2022.

[11] S. Kumar, R. Mahapatra, and A. Singh, "Automatic Modulation Recognition: An FPGA Implementation," *IEEE Communication Letter*, 2022.

[12] K. Jung, J. Woo, and S. Mukhopadhyay, "On-Chip Acceleration of RF Signal Modulation Classification with Short-Time Fourier Transform and Convolutional Neural Network," *IEEE Access*, vol. 11, 2023.

[13] J. Woo, K. Jung, and S. Mukhopadhyay, "Efficient Hardware Design of DNN for RF Signal Modulation Recognition Employing Ternary Weights," *IEEE Access*, vol. 12, 2024.

[14] Y. Lin, Y. Tu, and Z. Dou, "An Improved Neural Network Pruning Technology for Automatic Modulation Classification in Edge Devices," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, 2020.

[15] Y. Wang, J. Yang, M. Liu, and G. Gui, "LightAMC: Lightweight Automatic Modulation Classification via Deep Learning and Compressive Sensing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, 2020.

[16] Z. Chen, Z. Wang, X. Gao, J. Zhou, D. Xu, S. Zheng, Q. Xuan, and X. Yang, "Channel Pruning Method for Signal Modulation Recognition Deep Learning Models," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 2, 2024.

[17] M. Ninan, R. Evans, L. Reichling, N. Ghose, and B. Wang, "TinyRadio: Tiny Neural Networks for Fingerprinting Radio Frequency Signals," in *Proc. of IEEE Natonal Aerospace annd Electronics Conference (NAECON'25)*, 2025.

[18] A. MacLellan, L. Crockett, and R. Stewart, "RFSoC Modulation Classification with Streaming CNN: Data Set Generation and Quantized-Aware Training," *IEEE Open Journal of Circuits and Systems*, 2025.

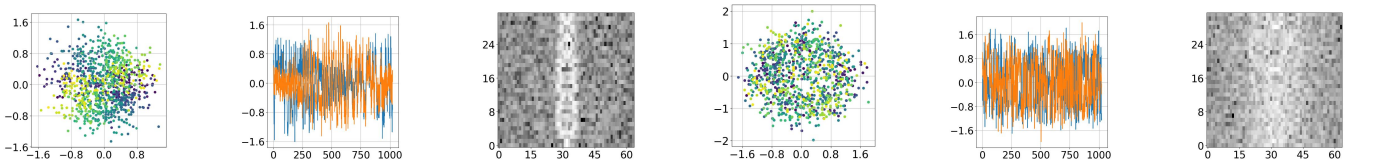
[19] "Fast, Scalable Quantized Neural Network Inference on FPGAs." [Online]. Available: https://github.com/Xilinx/FINN

[20] K. Yashashwi, A. Sethi, and P. Chaporkar, "A Leanable Distortion Correction Model for Modulation Recognition," *IEEE Wireless Communication Letter*, 2019.

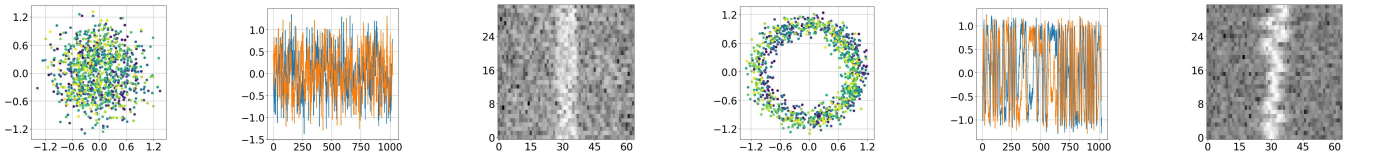
[21] J. Shi, S. Hong, C. Cai, Y. Wang, H. Wang, and G. Gui, "Deep Learning-Based Automatic Modulation Recognition Method in the Presence of Phase Offset," *IEEE Access*, 2020.

[22] X. Wang, Y. Zhao, and Z. Huang, "A Survey of Deep Transfer Learning in Automatic Modulation Classification," *IEEE Transactions on Congnitive Communications and Networking*, 2025.

[23] RadioML Datasets. [Online]. Available: https://www.deepsig.ai/datasets/

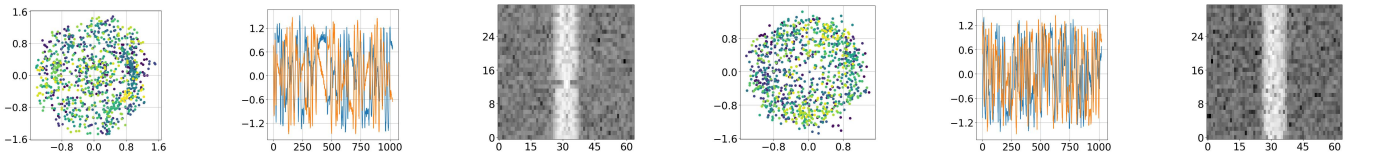


(a) BPSK (No. 200), IQ Samples (b) BPSK (No. 200), IQ Frame (c) BPSK (No. 200), Spectrogram (d) QPSK (No. 4296), IQ Samples (e) QPSK (No. 4296), IQ Frame (f) QPSK (No. 4296), Spectrogram

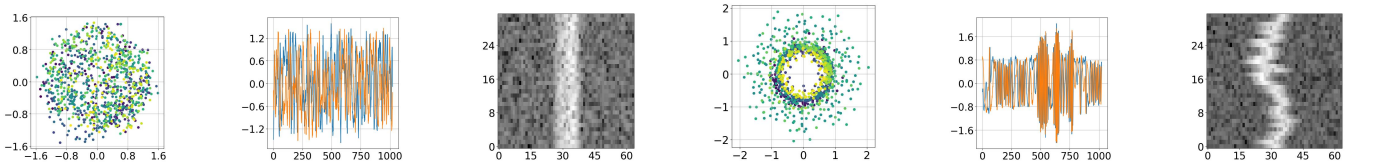


(g) 8PSK (No. 4296), IQ Samples (h) 8PSK (No. 8392), IQ Frame (i) 8PSK (No. 8392), Spectrogram (j) FM (No. 12488), IQ Samples (k) FM (No. 12488), IQ Frame (l) FM (No. 12488), Spectrogram

Fig. 5: I/Q Frame Plots (float32, **USRP, indoor, line-of-sight**, BPSK, QPSK, 8PSK, FM)

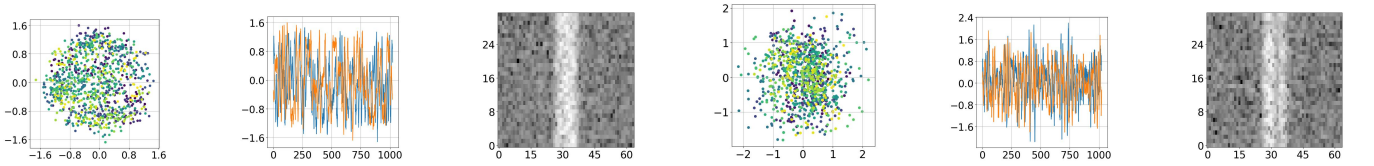


(a) BPSK (No. 61640), IQ Samples (b) BPSK (No. 61640), IQ Frame (c) BPSK (No. 61640, Spectrogram (d) QPSK (No. 127176), IQ Samples (e) QPSK (No. 127176), IQ Frame (f) QPSK (No. 127176), Spectrogram

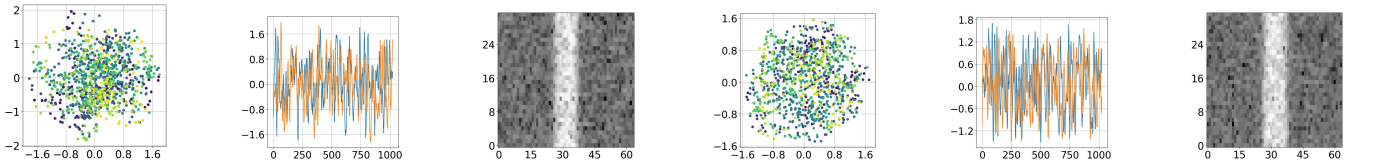


(g) 8PSK (No. 192712), IQ Samples (h) 8PSK (No. 192712), IQ Frame (i) 8PSK (No. 192712), Spectrogram (j) FM (No. 848072), IQ Samples (k) FM (No. 848072), IQ Frame (l) FM (No. 848072), Spectrogram

Fig. 6: I/Q Frame Plots (float32, **Rician-ppm1 (30db), simulated**, BPSK, QPSK, 8PSK, FM)

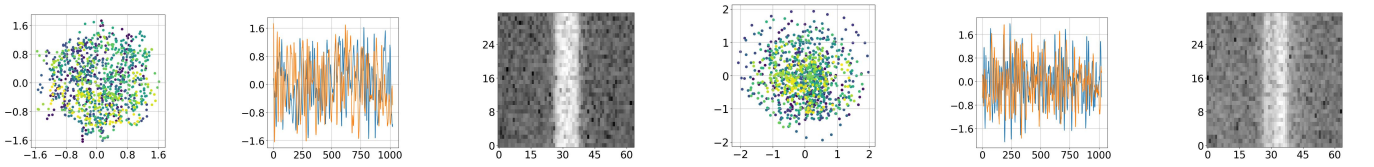


(a) 16APSK (No. 585928), IQ Samples (b) 16APSK (No. 585928), IQ Frame (c) 16APSK (No. 585928, Spectrogram (d) 32APSK (No. 651464), IQ Samples (e) 32APSK (No. 651464), IQ Frame (f) 32APSK (No. 651464), Spectrogram

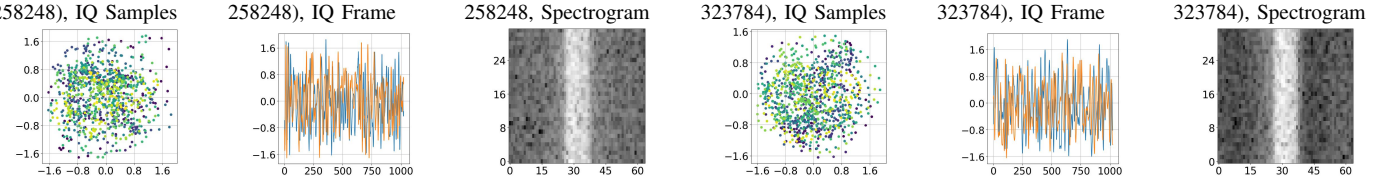


(g) 64APSK (No. 717000), IQ Samples (h) 64APSK (No. 717000), IQ Frame (i) 64PSK (No. 717000), Spectrogram (j) 128APSK (No. 782536), IQ Samples (k) 128APSK (No. 782536), IQ Frame (l) 128APSK (No. 782536), Spectrogram

Fig. 7: I/Q Frame Plots (float32, **Rician-ppm1 (30db), simulated**, 16APSK, 32APSK, 64APSK, 128APSK)

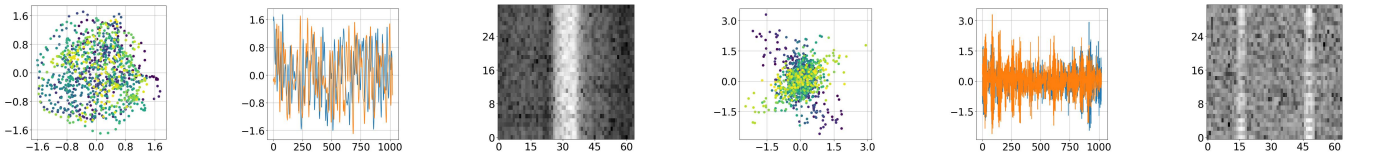


(a) 16QAM (No. 258248), IQ Samples (b) 16QAM (No. 258248), IQ Frame (c) 16QAM (No. 258248, Spectrogram (d) 32QAM (No. 323784), IQ Samples (e) 32QAM (No. 323784), IQ Frame (f) 32QAM (No. 323784), Spectrogram

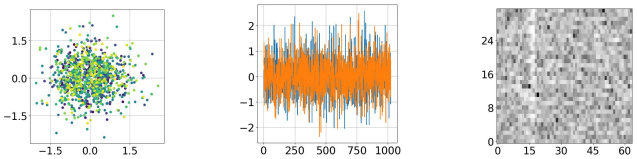


(g) 64QAM (No. 389320), IQ Samples (h) 64QAM (No. 389320), IQ Frame (i) 64QAM (No. 389320), Spectrogram (j) 128QAM (No. 454856), IQ Samples (k) 128QAM (No. 454856), IQ Frame (l) 128QAM (No. 454856), Spectrogram

Fig. 8: I/Q Frame Plots (float32, **Rician-ppm1 (30db), simulated**, 16QAM, 32QAM, 64QAM, 128QAM)

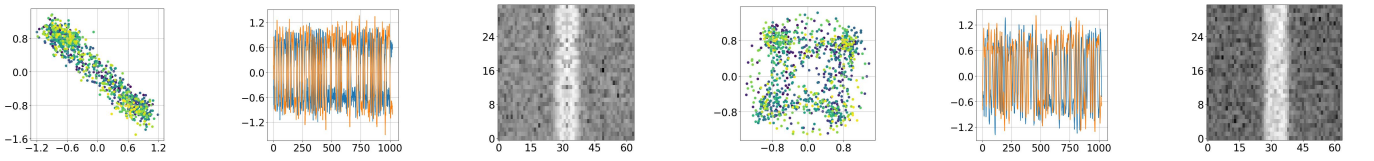


(a) 256QAM (No. 520392), IQ Samples (b) 256QAM (No. 520392), IQ Frame (c) 256QAM (No. 520392, Spectrogram (d) AM-DSB-SC (No. 913608), IQ Samples (e) AM-DSB-SC (No. 913608), IQ Frame (f) AM-DSB-SC (No. 913608), Spectrogram

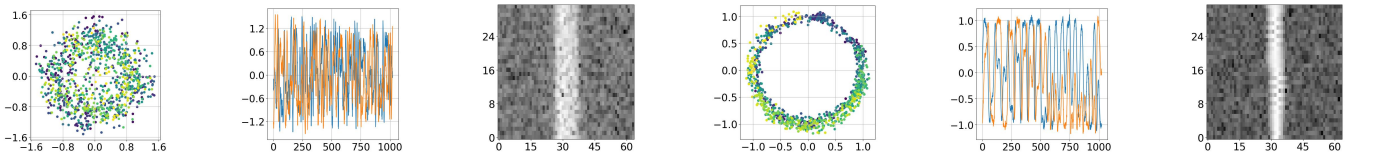


(g) AM-SSB-SC (No. 979144), IQ Samples (h) AM-SSB-SC (No. 979144), IQ Frame (i) AM-SSB-SC (No. 979144), Spectrogram

Fig. 9: I/Q Frame Plots (float32, **Rician-ppm1 (30db), simulated**, 256QAM, AM-SSB-SC, AM-DSB-SC)

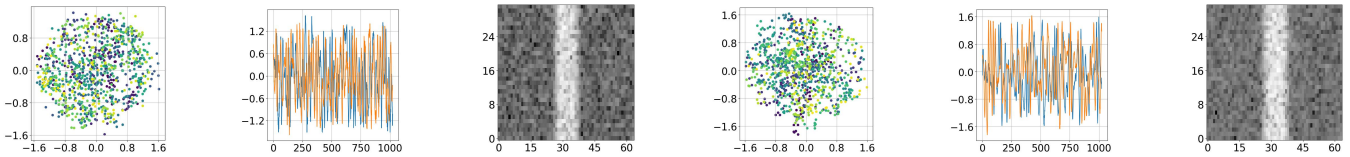


(a) BPSK (No. 61640), IQ Samples (b) BPSK (No. 61640), IQ Frame (c) BPSK (No. 61640, Spectrogram (d) QPSK (No. 127176), IQ Samples (e) QPSK (No. 127176), IQ Frame (f) QPSK (No. 127176), Spectrogram

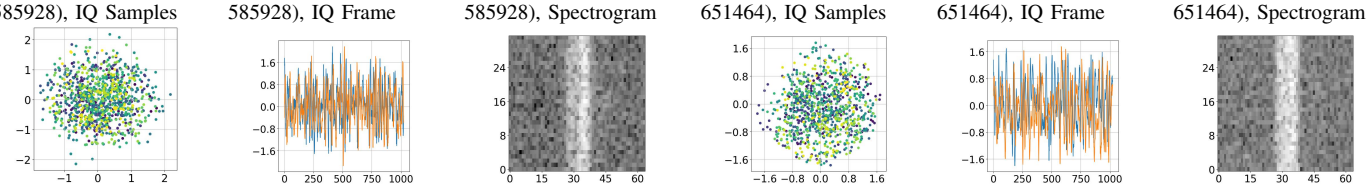


(g) 8PSK (No. 192712), IQ Samples (h) 8PSK (No. 192712), IQ Frame (i) 8PSK (No. 192712), Spectrogram (j) FM (No. 848072), IQ Samples (k) FM (No. 848072), IQ Frame (l) FM (No. 848072), Spectrogram

Fig. 10: I/Q Frame Plots (float32, **Rayleigh-ppm1 (30db), simulated**, BPSK, QPSK, 8PSK, FM)

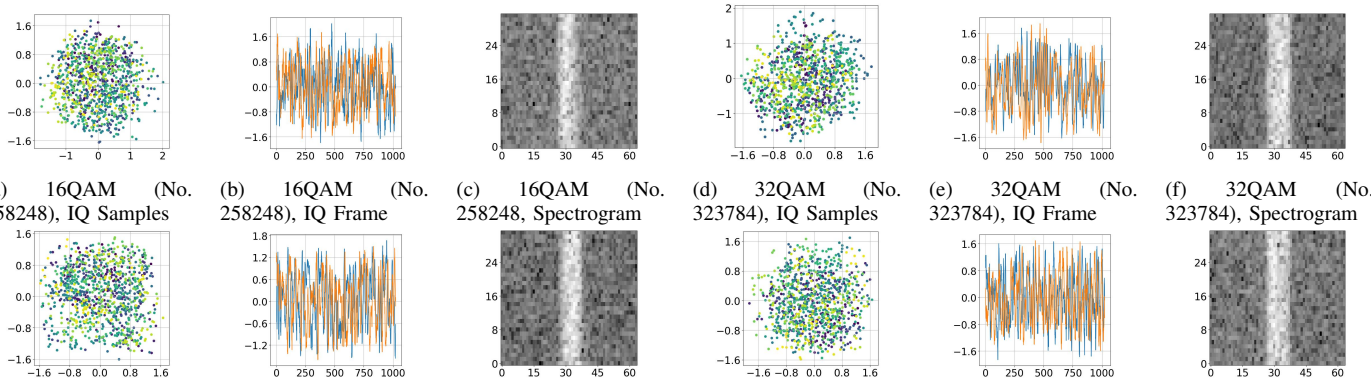


(a) 16APSK (No. 585928), IQ Samples (b) 16APSK (No. 585928), IQ Frame (c) 16APSK (No. 585928), Spectrogram (d) 32APSK (No. 651464), IQ Samples (e) 32APSK (No. 651464), IQ Frame (f) 32APSK (No. 651464), Spectrogram



(g) 64APSK (No. 717000), IQ Samples (h) 64APSK (No. 717000), IQ Frame (i) 64APSK (No. 717000), Spectrogram (j) 128APSK (No. 782536), IQ Samples (k) 128APSK (No. 782536), IQ Frame (l) 128APSK (No. 782536), Spectrogram

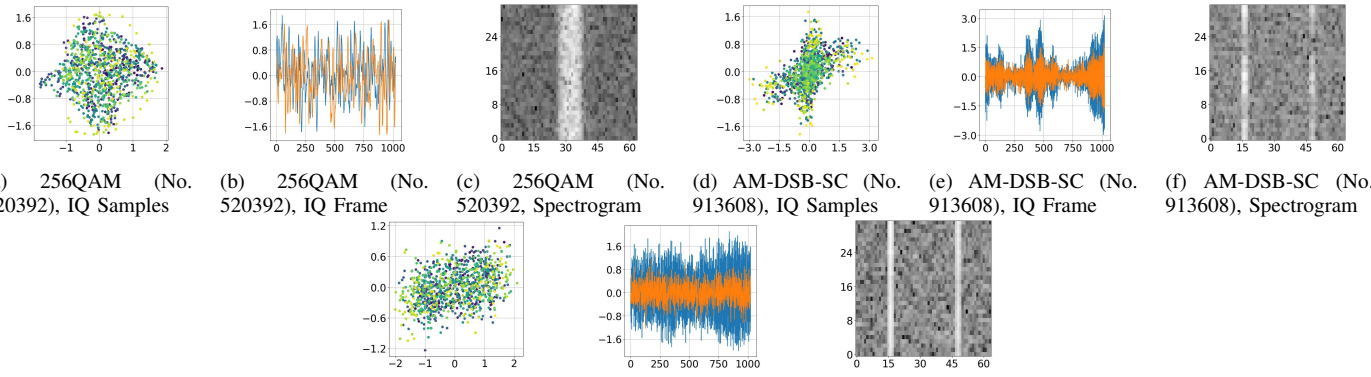
Fig. 11: I/Q Frame Plots (float32, **Rayleigh-ppm1 (30db), simulated**, 16APSK, 32APSK, 64APSK, 128APSK)



(a) 16QAM (No. 258248), IQ Samples (b) 16QAM (No. 258248), IQ Frame (c) 16QAM (No. 258248, Spectrogram (d) 32QAM (No. 323784), IQ Samples (e) 32QAM (No. 323784), IQ Frame (f) 32QAM (No. 323784), Spectrogram

(g) 64QAM (No. 389320), IQ Samples (h) 64QAM (No. 389320), IQ Frame (i) 64QAM (No. 389320), Spectrogram (j) 128QAM (No. 454856), IQ Samples (k) 128QAM (No. 454856), IQ Frame (l) 128QAM (No. 454856), Spectrogram

Fig. 12: I/Q Frame Plots (float32, **Rayleigh-ppm1 (30db), simulated**, 16QAM, 32QAM, 64QAM, 128QAM)



(a) 256QAM (No. 520392), IQ Samples (b) 256QAM (No. 520392), IQ Frame (c) 256QAM (No. 520392, Spectrogram (d) AM-DSB-SC (No. 913608), IQ Samples (e) AM-DSB-SC (No. 913608), IQ Frame (f) AM-DSB-SC (No. 913608), Spectrogram

(g) AM-SSB-SC (No. 979144), IQ Samples (h) AM-SSB-SC (No. 979144), IQ Frame (i) AM-SSB-SC (No. 979144), Spectrogram

Fig. 13: I/Q Frame Plots (float32, **Rayleigh-ppm1 (30db), simulated**, 256QAM, AM-SSB-SC, AM-DSB-SC)