DOI:10.1145/3547131

To view the accompanying Technical Perspective, visit doi.acm.org/10.1145/3547130

Polymorphic Wireless Receivers

By Francesco Restuccia and Tommaso Melodia

Abstract

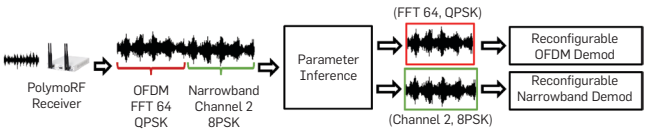
Today's wireless technologies are largely based on inflexible designs, which make them inefficient and prone to a variety of wireless attacks. To address this key issue, wireless receivers will need to (i) infer on-the-fly the physical layer parameters currently used by transmitters; and if needed, (ii) change their hardware and software structures to demodulate the incoming waveform. In this paper, we introduce *PolymoRF*, a deep learning-based polymorphic receiver able to reconfigure itself in real time based on the inferred waveform parameters. Our key technical innovations are (i) a novel embedded deep learning architecture, called *RFNet*, which enables the solution of key waveform inference problems, and (ii) a generalized hardware/software architecture that integrates *RFNet* with radio components and signal processing. We prototype *PolymoRF* on a custom software-defined radio platform and show through extensive over-the-air experiments that *PolymoRF* achieves throughput within 87% of a perfect-knowledge *Oracle* system, thus demonstrating for the first time that polymorphic receivers are feasible.

1. INTRODUCTION

It has been forecast that over 50 billion mobile devices will be soon connected to the Internet, creating the biggest network the world has ever seen.[3] However, only very recently has the community started to acknowledge that squeezing billions of devices into tiny spectrum portions will inevitably create disruptive levels of interference. Although Mitola and Maguire first envisioned the concept of "cognitive radios" 20 years ago,[8] today's commercial wireless devices still use inflexible wireless standards such as Wi-Fi and Bluetooth—and thus, are still very far from being truly real-time reconfigurable. Just to give an example of the seriousness of the spectrum inflexibility issue, DARPA has recently invested to launch the spectrum collaboration challenge (SC2), where the target is to design spectrum access schemes that "[...] best share spectrum with any network(s), in any environment, without prior knowledge, leveraging on machine-learning techniques."[25]

Intuitively, the issues of existing communication systems could be addressed by allowing transmitters to dynamically switch parameters such as carrier frequency, FFT size, and symbol modulation without coordination with the receiver. This will allow the transmitter efficient spectrum occupation using the most appropriate wireless scheme at any given moment. Figure 1 shows an example of a polymorphic receiver able to infer the current transmitter's physical layer scheme (e.g., OFDM vs. narrowband) and the scheme's parameters (e.g., FFT size, channel, modulation), and then demodulate each portion of the signal.

Figure 1. Example of a self-adaptive polymorphic receiver.



Doing away with explicit coordination and inflexible physical layers is the first step toward wireless receivers able to self-adapt to demodulate many waveform with a single radio interface.[15] Yet, despite their compelling necessity, these wireless receivers do not exist today. This manuscript aims to change the current state of affairs by proposing the first demonstration of *PolymoRF*, the first *polymorphic wireless receiver*. Achieving this goal required us to address a set of key research challenges summarized below:

(1) *Keeping up with the transmitter*. A crucial aspect is the real-time parameter inference. In practical systems, however, transmitters may choose to switch its parameter configuration in the order of milliseconds (e.g., frequency hopping, rate adaptation). For example, if the transmitter chooses to switch modulation every 100ms, the learning model should run in (much) less than 100 ms to predict the parameters and morph the receiver into a new configuration. To this end, we will show in Section 5.5 that CPU latency is several orders of magnitude greater than what is required to sustain realistic sampling rates from the RF interface. Thus, we need hardware-based designs to implement low-latency knowledge extraction techniques.

(2) *Creating learning architectures for the embedded RF domain*. Recent advances in RF deep learning[10-12, 14] have demonstrated that convolutional neural networks (ConvNets) may be applied to analyze RF data without feature extraction and selection algorithms.[4] Moreover, ConvNets present a number of characteristics (discussed in Section 3) that make them particularly desirable from a hardware implementation perspective. However, these solutions cannot be applied to implement real-time polymorphic wireless communications—as shown in Section 5.5, existing

The original version of this paper was published in *Proceedings of the 21st Int. Symp. on Theory, Algorithmic Foundations and Protocol Designs for Mobile Networks and Mobile Computing* (Oct. 2020), 271–280.

art[10, 12] utilizes general-purpose architectures with a very high number of parameters, requiring hardware resources and latency that go beyond what is acceptable in the embedded domain. This crucial issue calls for novel, RF-specific, real-time architectures. We are not aware of learning systems tested in a real-time wireless environment and used to implement inference-based wireless systems.

(3) *System-level feasibility of polymorphic platforms*. It is yet to be demonstrated whether polymorphic platforms are feasible and effective. This is not without a reason—from a system perspective, it required us to tightly interconnect traditionally separated components, such as CPU, RF front-end, and embedded operating system/kernel, to form a seamlessly running low-latency learning architecture closely interacting with the RF components and able to adapt at will its hardware and software based on RF-based inference. Furthermore, since polymorphic wireless systems are subject to inference errors, we need to test its performance against a perfect knowledge (thus, ideal and not implementable) system.

1.1. Technical contributions

This paper's key innovation is to finally bridge the gap between the extensive theoretical research on cognitive radios and the associated system-level challenges, by demonstrating that inference-based wireless communications are indeed feasible on off-the-shelf embedded devices. Beyond the examples and the evaluation conducted in Section 5, the main purpose of this work is to provide a *blueprint* for next-generation wireless receivers, where their radio hardware and software are not *protocol-specific*, but instead *spectrum-driven* and adaptable on-the-fly to different waveforms.

We summarize our main technical contributions as follows:

(1) We design a novel learning architecture called *RFNet*, specifically and carefully tailored for the embedded RF domain. Our key intuition in *RFNet* is to arrange I/Q samples to form an "image" that can be effectively analyzed by the ConvNet filters. This operation produces high-dimensional representations of small-scale transition in the I/Q complex plane, which can be leveraged to efficiently solve a wide variety of complex RF classification problems such as RF modulation classification. Extensive experimental evaluation indicates that a pipelined version of *RFNet* significantly reduces latency with respect to a CPU implementation;

(2) We propose a general-purpose hardware/software architecture for software-defined radios that enables the creation of custom polymorphic wireless systems through *RFNet*. Then, we implement a multipurpose library based on high-level synthesis (HLS) that translates an *RFNet* model implemented in software to a circuit implemented in the FPGA portion of the SoC. Moreover, we leverage key optimization strategies such as pipelining and unrolling to further reduce the latency of *RFNet* by more than 50% with respect to the unoptimized version, with only 7% increase of hardware resource consumption. Finally, we design and implement the device-tree entries and Linux drivers enabling the system to utilize *RFNet* and other key hardware peripherals.

(3) We prototype *PolymoRF* on a ZYNQ-7000 system-on-chip (SoC) and analyze its performance on a scheme where the transmitter can switch among three FFT sizes and three symbol modulation schemes without explicit notification to the receiver. A demo video of PolymoRF where the transmitter switches FFT size every 0.5s is available at https://youtu.be/5vf_pb0nvKk. We believe ours is the first demonstration of real-time OFDM reconfigurability without explicit transmitter/receiver coordination. Experiments on both line-of-sight (LOS) and non-line-of-sight (NLOS) channel conditions show that the system achieves at least 87% of the throughput of a perfect knowledge—and thus, unrealistic—*Oracle* OFDM system, thus proving the feasibility of polymorphic receivers.

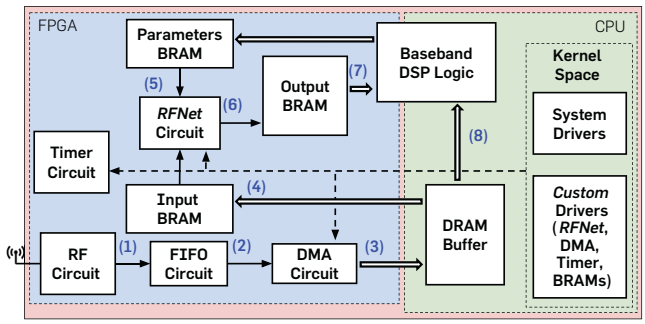
2. PolymoRF: AN OVERVIEW

The primary operations performed by the *PolymoRF* platform are summarized in Figure 2. In a nutshell, *PolymoRF* can be considered as a full-fledged learning-based software-defined radio architecture where both the inference system and the demodulation strategy can be morphed into new configurations at will.

We provide a walk-through of the main operations performed by *PolymoRF* with the help of Figure 2. Although for simplicity we refer to specific hardware equipment and circuits in our explanation, we point out that the building blocks of our platform design (BRAMs, DMA, FIFOs, etc.) can be implemented in any commercially available FPGA platform.

We assume the transmitter may transmit by choosing among a discrete set of physical layer parameters that are known at the receiver's side. We define as Y a tuple of such physical layer parameters, which may be changed at will by the transmitter but not before T_{sw} seconds between each change, which we refer to a *switching time*. For the sake of generality, in this paper we will not assume any particular strategy in the transmitter's parameter choice, which can be driven by a series of factors (including anti-jamming strategy, noise avoidance, throughput optimization, and so on)

Figure 2. Modules and operations of *PolymoRF*.



that will be considered as out of the scope of this paper, whose main focus is instead on the receiver's side.

(1) *Reconfigurable radio front-end*. The RF signal is received (step 1) through a reconfigurable RF front-end. In our prototype, we used an AD9361[1] radio interface, which supports frequency range between 70 MHz and 6.0 GHz and channel bandwidth between 200 kHz and 56 MHz. We chose the AD9361 because it is commonly used in software-defined radio systems—indeed, it is also used by USRPs such as the E310 and B210. Moreover, the AD9361 provides basic FPGA reference designs and kernel-space drivers to ease prototyping and extensions. Perhaps more importantly, the AD9361 local oscillator (LO) frequency and RF bandwidth can be reconfigured at will through CPU registers.

(2) *Conversion from RF to FPGA domain*. The AD9361 produces streams of I/Q samples of 200M samples/second—hence, it is clocked at 200 MHz. Since the AD9361 clock would be too fast for the other circuits in the FPGA, we implemented a FIFO to adapt the speed of samples from the AD9361 to the 100 MHz clock frequency used by the other circuits in the FPGA (step 2). We then use a direct memory access (DMA) core to store the stream of I/Q samples to a buffer in the DRAM (step 3). The use of DMA is crucial as the CPU cannot do the transfer itself, since it would be fully occupied for the entire duration of the read/write operation and thus unavailable to perform other work. Therefore, we wrote a custom DMA driver to periodically fill a buffer of size B residing in the DRAM with a subset of I/Q samples coming from the FIFO.

(3) *Learning and receiver polymorphism*. After the buffer has been replenished, its first X I/Q samples are sent to a BRAM (step 4) constituting the input to *RFNet*, a novel learning architecture based on *ConvNets*. This circuit is the fundamental core of the *PolymoRF* system; therefore, we will dedicate Sections 3 and 4 to discuss in detail its architecture and implementation, respectively. The parameters of *RFNet* are read by an additional BRAM (step 5), which in effect allows the reconfiguration of *RFNet* to address multiple RF problems according to the current platform need. As explained in Section 3, *RFNet* produces a probability distribution over the transmitter's parameter set Y. After *RFNet* has inferred the transmitter's parameters, it writes on a block-RAM (BRAM) its probability distribution (step 6). Then, the baseband DSP logic (which may be implemented in both hardware and software) reads the distribution from the BRAM (step 7) selects the parameter set with highest probability and "morphs" into a new configuration to demodulate the I/Q samples in B (step 8).

3. LEARNING SYSTEM: RFNet

We first motivate the use of convolutional neural networks for RFNet, we discuss some RF-specific learning challenges, and then we describe in details the RFNet input construction and its complete architecture.

(1) *Why using deep learning and not machine learning?* Deep learning relieves from the burden of finding the right "features" characterizing a given wireless phenomenon. At the physical layer, this is a key advantage for the following reasons. First, deep learning offers high-dimensional feature spaces. In particular, O'Shea et al.[12] have demonstrated that on the 24-modulation dataset considered, deep learning models achieve on the average about 20% higher classification accuracy than legacy learning models under noisy channel conditions. Second, automatic feature extraction allows to reuse the same hardware circuit to address different learning problems. Critically, this allows to keep both latency and energy consumption constant, which are particularly critical in wireless systems. Third, deep learning algorithms can be fine-tuned by performing batch gradient descent on fresh input data, avoiding manual re-tuning of the feature extraction algorithms.

(1) *Why using ConvNets for wireless deep learning?* There are several primary advantages that make the usage of ConvNet-based models particularly desirable for the embedded RF domain. First, *convolutional filters are designed to interact only with a very small portion of the input*. We show in Section 5.3 that this key property allows achieving significantly higher accuracy than traditional neural networks. Perhaps even more importantly, *ConvNets are scalable with the input size*. For example, for a 200×200 input and a DL with 10 neurons, a traditional neural network will have $200^2 \cdot 10 = 400$k weights, which implies a memory occupation of $4 \cdot 400\text{k} = 16$Mbytes to store the weights of a single layer (i.e., a float number for each weight). Clearly, this is unacceptable for the embedded domain, as the network memory consumption would become intractable as soon as several DLs are stacked on top of the other.

Moreover, *ConvNet filtering operations can be made low latency by parallelization*, which makes them particularly suitable to be optimized for the RF domain. Finally, we show in Section 5 that the same ConvNet architectures can be reused to address different RF classification problems (e.g., modulation classification in single- and multicarrier systems), as long as the ConvNet is provided appropriate weights through training. Our ConvNet hardware design (Section 4.1) has been specifically designed to allow seamless ConvNet reconfiguration and thus solving different RF problems according to the system's needs.

(2) *RF-specific learning challenges*. There are a number of key challenges in RF learning that are substantially absent in the CV domain. Among others, we know that RF signals are continuously subject to dynamic (and usually unpredictable) noise/interference coming from various sources. This may decrease the accuracy of the learning model. For example, portions of a QPSK

transmission could be mistaken for 8PSK transmissions since they share part of their constellations. We address the above core design issues with the following intuitions. First, although RF signals are affected by fading/noise, in most practical cases their effect can be considered as constant over small intervals. Second, though some constellations are similar to each other, the transitions between the symbols of the constellations are distinguishable when the waveform is sampled at a higher sampling rate than the one used by the transmitter. Third, convolution operations are equivariant to translation, so they can recognize I/Q patterns regardless of where they occur.

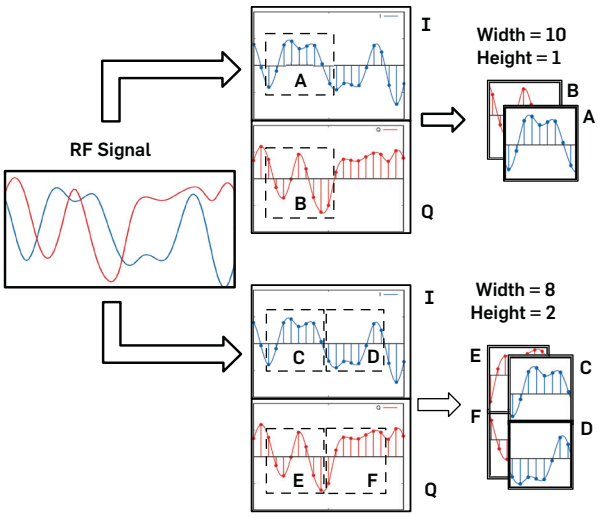
(3) *RFNet input construction*. By leveraging these key concepts, we can design a learning system that distinguishes waveforms by recognizing transitions in the I/Q complex plane regardless of where they happen, by leveraging the shift-invariance property of convolutional layers. More formally, let us consider a discrete-time complex-valued I/Q sequence $s[k]$, where $k \geq 0$. Let us consider $M = W \cdot H$ consecutive I/Q samples $s[j]$, $0 \leq j \leq W \cdot W$, where W and H are the *width* and *height* of the input tensor. The input tensor $\mathcal{T}$, of dimension $W \times H \times 2$, is constructed as follows:

$$\mathcal{T}[r, c, d] = \mathrm{Re}\{s(r \cdot W + c)\} \cdot (1-d) + \mathrm{Im}\{s[r \cdot W + c]\} \cdot d, \quad (1)$$

$$\text{where } d \in \{0, 1\},\ 0 \leq r \leq H,\ 0 \leq c \leq W$$

By construction, it follows that $\mathcal{T}[r+1, c] = s[(r+1) \cdot W + c] = s[r \cdot W + c + W]$, meaning that (i) I/Q samples in adjacent columns will be spaced in time by a factor of 1, and (ii) I/Q samples in adjacent rows will be spaced in time by a factor of W; moreover, (iii) our input tensors have depth equal to 2, corresponding to the I and Q data, respectively, which will allow the *RFNet* filters to examine each element of the input tensor *without decoupling the I and Q components of the RF waveform*. Figure 3 depicts an example of a 2×4 and 1×3 filters operating on a waveform.

Figure 3. How *RFNet* constructs tensors from I/Q samples.



4. PolymoRF: HW/SW ARCHITECTURE

This section presents the hardware and driver design and implementation of our *PolymoRF* system. We discuss the design, hardware implementation, and main operations of *RFNet* in Section 4.1 (Figure 4).

4.1. RFNet: Architecture and operations

(1) *Design constraints*. One of the core design issues to address is ensuring that *the same RFNet circuit can be reused for multiple learning problems and not just one architecture*. For example, the wireless node might want to classify only specific properties of an RF waveform, for example, classify only modulation since the FFT size is already known. This requires reconfigurability of the model parameters, as the device's hardware constraints may not be able to accommodate multiple learning architectures. In other words, we want RFNet to be able to operate with a different set of filters and weight parameters according to the circumstances. For this reason, we have used high-level synthesis (HLS) to design a library that translates a Keras-compliant *RFNet* into an FPGA-compliant circuit. HLS interprets an algorithmic description of a desired behavior (e.g., C/C++) and creates a model written in hardware description language (HDL) that can be executed by the FPGA.[20]

(2) *Circuit design*. Figure 5 shows a block scheme of our HLS-based *RFNet* circuit and its main interactions with the CPU and other FPGA components. We also provide an example with some numbers to ease presentation. The main feature of our RFNet implementation is its modularity-indeed, the circuits implementing each layer are *independent from each*

Figure 4. *RFNet* captures small-scale I/Q pattern sequences.

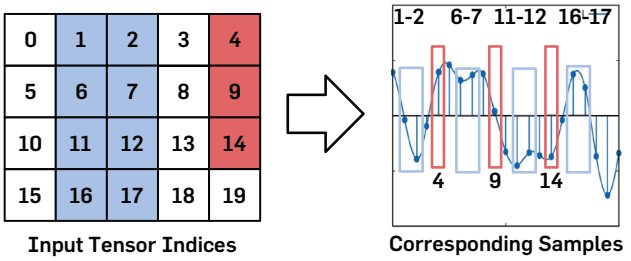
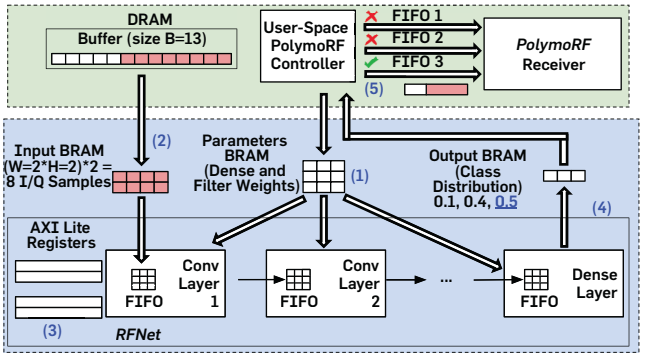


Figure 5. Block scheme of *PolymoRF*'s learning circuit.



other, which allows for ease of parallelization and transition from HLS to HDL. Consecutive layers in *RFNet* exchange data through high-speed AXI-Stream interfaces that then store the results of each layer in a FIFO, read by the next layer. Our architecture uses a 32-bit fixed point representation for real numbers, with 10 bits dedicated to the integer portion. We chose fixed point instead of floating point to decrease drastically computation and hardware architecture complexity, as we do not need the precision of floating point arithmetic. Another key advantage of our implementation is that it clearly separates the computation from the parameters, which allows for *seamless real-time reconfigurability*. This is achieved by writing the parameters in a BRAM accessible by the CPU and by the *RFNet* circuit.

(3) *Main operations*. The first operation is to write the *RFNet*'s parameters into a BRAM through the user-space PolymoRF controller (step 1). These parameters are the weights of the convolutional layer filters and the weights of the dense layers. Since we use fixed point architecture, each parameter is converted into fixed point representation before being written to the BRAM. As soon as a new input buffer *B* (of size 13 in our example) has been replenished, the controller writes the RFNet input (the first 8 I/Q samples in our example) into the input BRAM (step 2). *RFNet* operations are then started by writing into an AXI-Lite register (step 3) through a customized kernel-level Linux driver. Once the results have been written in the output BRAM (step 4), *RFNet* writes an acknowledgement bit into another AXI-Lite register, which signals the controller that the output is ready. Then, the controller reads the output (in our example, class 3 has the highest probability) and sends the entire buffer *B* through a Linux FIFO to the PolymoRF receiver (step 5), which is currently implemented in Gnuradio software. The receiver has different FIFOs, each for a parameter set. Whenever a FIFO gets replenished, the part of the flow graph corresponding to that parameter set activates and demodulates the I/Q samples contained in the buffer *B. Notice that for efficiency reasons the receiver chains do not run when the FIFO is empty*, therefore only one receiver chain can be active at time.

5. EXPERIMENTAL RESULTS

We first discuss details on our *PolymoRF* prototype in Section 5.1, and then discuss the data collection and training process in Section 5.2. We then investigate the performance of *RFNet* in Section 5.3 on a single-carrier system. Then, we implement and test the throughput performance on a multi-carrier polymorphic OFDM system in Section 5.4. Finally, we report the latency and hardware performance of *PolymoRF* in Section 5.5.

5.1. Protoype and experimental setup

Our prototype is entirely based on off-the-shelf equipment. Specifically, we use a Xilinx Zynq-7000 XC7Z045–2FFG900C system-on-chip (SoC), which is a circuit integrating CPU, FPGA, and I/O all on a single substrate.[9] We chose an SoC since it provides significant flexibility in the FPGA portion of the platform, thus allowing us to fully evaluate the trade-offs during system design. Moreover, the Zynq-7000 fully supports embedded Linux, which in effect makes the ZC706 a good prototype for a wireless platform. Our Zynq-7000 contains two ARM Cortex-A9 MPCore CPUs and a Kintex-7 FPGA,[21] running on top of a Xilinx ZC706 evaluation board.[22]

For both intra-FPGA and FPGA-CPU data exchange, we use the *Advanced eXtensible Interface* (AXI) bus specification.[23] In the AXI standard, the data is exchanged during *read* or *write transactions*. In each transaction, the *AXI master* is charged with initiating the transfer; the *AXI slave*, in turn, is tasked with responding to the AXI master with the result of the transaction (i.e., success/failure). An AXI master can have multiple AXI slaves and vice versa, according to the specific FPGA design. Multiple AXI masters/slaves can communicate with each other by using *AXI interconnects*. Specifically, *AXI-Lite* is used for register access and configures the circuits inside the FPGA, while AXI-Stream is used to transport high-bandwidth streaming data inside the FPGA. AXI-Full is instead used by the CPU to read/write consecutive memory locations from/to the FPGA.

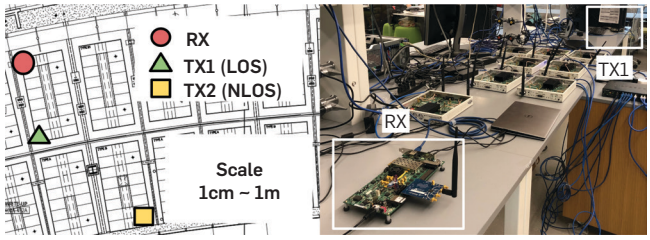
To study *PolymoRF* under realistic channel environments, we have used the experimental setup shown in Figure 6. These scenarios investigate a line-of-sight (LOS) configuration where the transmitter is placed approximately 3 m from the receiver, and a challenging non-line-of-sight (NLOS) channel condition where the transmitter is placed at 7 m from the receiver and in the presence of several obstacles between them. Thus, the experiments were performed in a contested wireless environment with severe interference from nearby Wi-Fi devices as well as multipath effect.

5.2. Data collection and training process

As far as the data collection and testing process is concerned, we first constructed a ~10GB dataset by collecting waveform data in the line-of-sight (LOS) configuration, and then used this data to train *RFNet* through Keras. Then, we tested our models on live-collected data in both LOS and NLOS conditions. The transmitter radio used was a Zedboard equipped with an AD9361 as RF front-end and using Gnuradio for baseband processing. Waveforms were transmitted at center frequency of 2.432 GHz (i.e., Wi-Fi's channel 5).

To train *RFNet*, we use an ℓ_2 regularization parameter $\lambda = 0.0001$. We also use an Adam optimizer with a learning

Figure 6. (left) Placement of the radios for experimental evaluation; (right) experimental setting.



rate of $l = 10^{-4}$ and categorical cross-entropy as a loss function. All architectures are implemented in Python, on top of the Keras framework and with Tensorflow as the backend engine.

5.3. Single-carrier evaluation

We consider the challenging problem of joint modulation and channel recognition in a single-carrier system where (i) modulation is chosen among BPSK, QPSK, 8PSK, 16-QAM, 32-QAM, and 64-QAM; (ii) spectrum is shifted of 0, 1 KHz, and 2 KHz from its center frequency. Due to space limitations, we only report results on the LOS scenario for the single-carrier scenario and report in Section 5.4 the performance of *RFNet* on the NLOS scenario with the multicarrier OFDM system.

(1) *Comparison with existing architectures*. We compare *RFNet* to,[10, 12] which is to the best of our knowledge[10, 12] the current state of the art in RF waveform classification using ConvNets. This approach, called for simplicity *Linear*, considers an input tensor of dimension $1 \times W \cdot H \times 2$ and convolutional layers with filters of dimension $1 \times F \times 2$. Thus, the filters in the first convolutional layer perform linear convolution over a set of F consecutive I/Q samples. We attempted to train the architecture in[12], which has M = 7 convolutional layers with C = 64 filters each and K = 2 dense layers with 128 neurons each. However, due to its huge dimensions, we were not able to synthesize this architecture on our test bed. Therefore, we compared *RFNet* with the architecture in,[10] that is, M = 2 convolutional layers with C = 25,680 and K = 1 with 256 neurons. For fair comparison with *Linear*, we selected the closest input size to ours (i.e., 1×128 vs. 10×10, 1×400 vs. 20×20, 1×900 vs. 30×30).

Figure 7 shows the test-set accuracy obtained for a subset of the considered architectures, where *RFNet* was trained with $M = 1$ convolutional layer with $C = 25$ filters, and no dense layer ($K = 0$). The obtained results indicate that traditional dense networks cannot recognize complex RF waveforms, as they attain slightly more accuracy (8%) than the random-guess accuracy (5.5%)—regardless of the number of layers. This is because dense layers are not able to capture localized, small-scale I/Q variations in the input data, which is instead done by convolutional layers. Moreover, Figure 7 indicates that *RFNet* has similar accuracy as obtained by *Linear*, despite using a much simpler architecture. This is due to the fundamental difference between how the convolutional layers in *PolymoRF* and *Linear* process I/Q samples.

(2) *Hyper-parameter evaluation*. We study the impact of the number of convolutional layers M and dense layers K, as well as the input size (W) and filter size (F) on the performance of *RFNet*. Figure 8 shows accuracy as a function of W and H, for hyper-parameters M = 1, 2 and C = 10, 25, 50. The results conclude that increasing C does improve the performance but up to a certain extent. Indeed, we notice that switching to $C = 50$ does not improve much the performance, especially when $M = 2$. This is because the number of distinguishing I/Q patterns is limited in number among different modulations, and thus, the filters in excess end up learning similar patterns. Furthermore, increasing W and H increases accuracy significantly, since a larger input size allows compensating for the adverse channels/noise conditions. Furthermore, Figure 9 illustrates the impact of K. Figure 9 suggests that the accuracy does not increase when adding a dense layer, regardless of its size, which indicates the correctness of our choice to exclude dense layers.

(3) *Impact of the sampling rate*. We investigate the impact of the transmitter's sampling rate in Figure 10, where we show the classification accuracy for differ-

Figure 7. Comparison among *RFNet*, *Dense*, and *Linear*.[10]

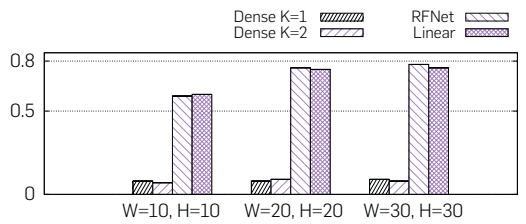


Figure 8. (top) Test set classification accuracy vs. input size *W/H* vs. *M*, with *K* = 0 (no dense layer); (bottom) confusion matrices as function of *M*, *W*, and *H*.

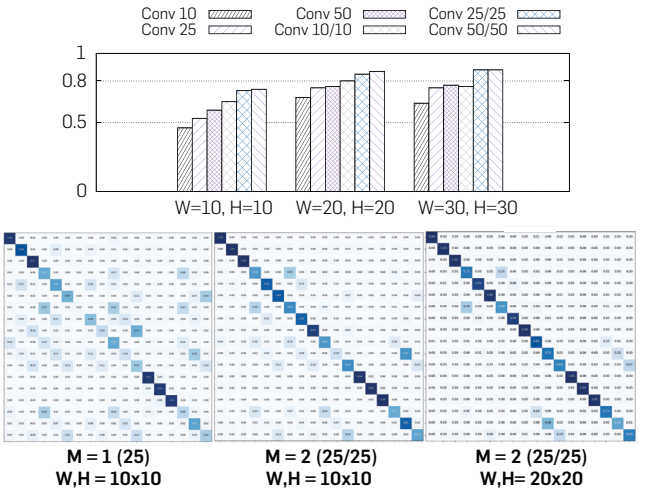
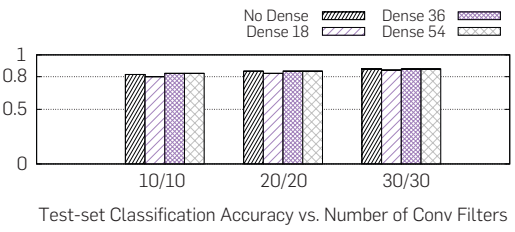


Figure 9. Accuracy vs. number of filters vs. dense layer size.



ent W, H, and C values. We also show the confusion matrices[a] for the W, H = 10, C = 50 architectures in Figure 11. As expected, these results confirm that the performance of *RFNet* decreases as the transmitter's sampling rate increases. This is because, as shown in Section 3 *RFNet* learns the I/Q transitions between the different modulations. Therefore, as the transmitter's sampling rate increases, the model will have fewer I/Q samples between the constellation points. Indeed, the confusion matrices show that with 5 MS/s the model becomes further confused with QAM constellations, and with 10 MS/s higher-order PSKs and QAMs "collapse" onto the lowest-order modulations.

(4) *Remarks*. The above results imply that oversampling the signal leads to better modulation classification accuracy. However, we would like to point out that oversampling does not mean that the physical layer has to process more data—indeed, the extra samples can be dropped when going through the demodulation chain, while the oversampled I/Q signal can be forwarded to *RFNet* for classification.

5.4. Multicarrier evaluation

We evaluated *PolymoRF* on an OFDM system (in short, *Poly-OFDM*) which supports three different FFT sizes (64, 128, and 256) and three different symbol modulations in the FFT bins (BPSK, QPSK, and 8PSK), creating in total a combination of nine different parameter sets that are switched pseudo-randomly by the transmitter. A demo video where the transmitter switches FFT size every 0.5s is available at https://youtu.be/5vf_pb0nvKk. In the following, we use the C = 25,25, 20 × 20, pipelined *RFNet* architecture, which presents latency of about 17 ms (see Section 5.5). In these experiments, we set (i) the transmitter's sampling rate to 5M samples/sec; *PolymoRF*'s buffer size *B* to 250k I/Q samples; (iii) the switching time of the transmitter to 250 ms. Thus, *RFNet* is run approximately five times during each switching time.

The most critical aspect to be evaluated is how *Poly-OFDM*, an inference-based system, compares with an ideal system that has perfect knowledge of the modulation and FFT size being used by the transmitter at each time, which we call for simplicity *Oracle*. Although *Oracle* cannot be implemented in practice, we believe this experiment is crucial to understand what is the throughput loss with respect to a system where the physical layer configuration is known a priori. In Figure 12, we show the comparison between *Oracle* and *Poly-OFDM* as a function of the FFT size and the symbol modulation. As we notice, the overall throughput results decrease in the NLOS scenario, which is expected given the impairments imposed by the challenging channel conditions. On the other hand, the results in Figure 12 confirm that Poly-OFDM is able to obtain similar throughput performance with that of a traditional OFDM system, obtaining on the average 90% and 87% throughput of that of the traditional system.

5.5. RFNet latency evaluation and comparison

Table 1 compares latency, the number of parameters, and BRAM occupation of *RFNet* vs. a C++ implementation running in the CPU of our test bed. As we can see, *RFNet* consumes at most 34% of the available BRAM of the platform. Moreover, Table 2 shows the comparison between the pipelined version of the ConvNet circuits and the CPU latency, as well as the look-up table (LUT) consumption increase with respect to the unpipelined version. Table 2 concludes that on the average, our parallelization strategies bring close to 60% and 100% latency reduction with respect to the unoptimized and CPU versions, respectively, with a LUT utilization increase of about 7% on the average.

a Class labels are ordered by modulation and frequency shift, that is, from "BPSK, 0 KHz", "BPSK, 1 KHz", ... to "64-QAM, 2 KHz".

Figure 10. Accuracy vs. transmitter's sampling rate.

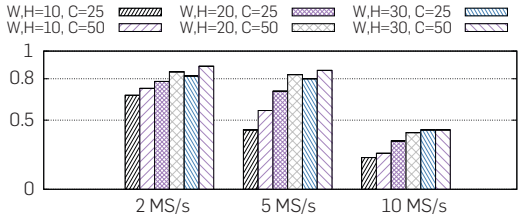


Figure 11. Confusion matrices for transmitter's sampling rate of 5 MS/s and 10 MS/s, W, H = 10, C = 50 model.

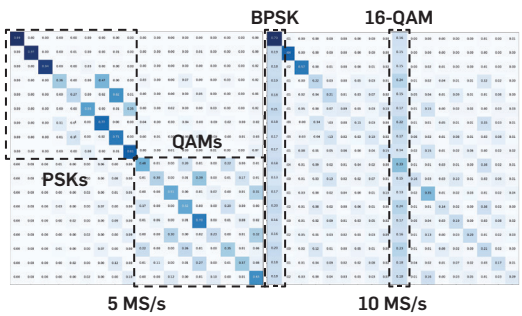


Figure 12. Comparison between *Oracle* and *Poly-OFDM*, (top) *LOS* and (bottom) *NLOS* scenarios.

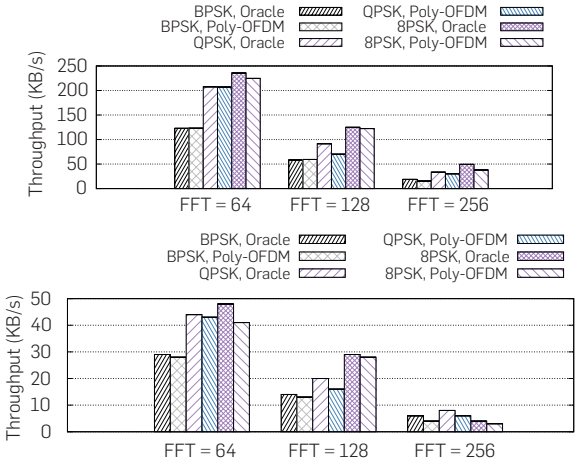


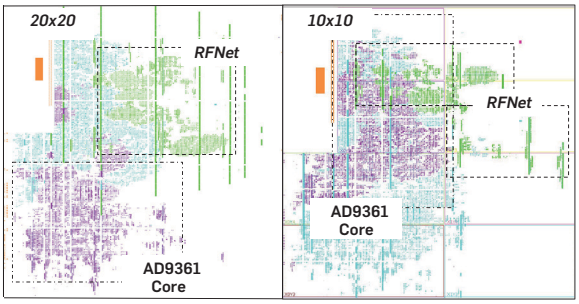
Table 1. Latency/hardware consumption evaluation.

Model	Input	Latency (ms)	Params (k)	BRAM (%)
RFNet C = 25	10 × 10	2.918	∼11	1
	20 × 20	26.55	∼45	7
	30 × 30	93.35	∼81	15
RFNet C = 50	10 × 10	5.835	∼23	3
	20 × 20	53.11	∼90	14
	30 × 30	233.3	∼203	29
RFNet C = 25,25	10 × 10	6.704	∼10	2
	20 × 20	38.41	∼17	8
	30 × 30	144.9	∼34	17
RFNet C = 50,50	10 × 10	21.41	∼31	4
	20 × 20	100.3	∼40	16
	30 × 30	336.9	∼81	34

Table 2. Pipelined vs. CPU latency.

Model	Input	CPU (ms)	Pipelined (ms)	LUT (%)
RFNet C = 25	10 × 10	49.31	1.19	+3
	20 × 20	478.4	8.077	+7
	30 × 30	1592	25.54	+9
RFNet C = 50	10 × 10	106.4	2.381	+6
	20 × 20	934.2	16.15	+12
	30 × 30	3844	63.81	+20
RFNet C = 25,25	10 × 10	122.1	3.959	+1
	20 × 20	677.9	16.29	+4
	30 × 30	2354	49.57	+7
RFNet C = 50,50	10 × 10	363.9	13.51	+2
	20 × 20	1826	48.87	+7
	30 × 30	5728	131.7	+11

Figure 13. *PolymoRF* FPGA implementations.



To give the reader a perspective of the amount of resources consumed on the FPGA, Figure 13 shows the FPGA implementation of respectively 10 × 10 and 20 × 20 *RFNet* model, both pipelined and with C = 25,25 architecture, where we highlight and color the resource consumption of *RFNet* with respect to the AD9361 circuitry. Figure 13 indicates that the resource consumption of the *RFNet* circuit is significantly lesser than the AD9361 one in the 10 × 10 case and becomes comparable with the 20 × 20 architecture. In any case, the overall resource consumption of our FPGA designs is approximately 50% of the total FPGA resources.

6. RELATED WORK

Learning-based radios are envisioned to be able to automatically infer the current spectrum status in terms of occupancy,[17] interference,[2] and malicious activities.[5] Most of the existing work is based on low-dimensional machine learning,[4, 16, 24] which requires the cumbersome manual extraction of very complex, *ad hoc* features from the waveforms. For this reason, deep learning has been proposed as a viable alternative to traditional learning techniques.[7] The key problem of RF modulation recognition through deep learning has been extensively investigated.[6, 11, 12, 18, 19] The seminal work by O'Shea et al.[12] proposed ConvNets-based to address the issue. However, the authors do not address the issue of what to do with the inferred RF information. Moreover, the aforesaid work proposes models leveraging a significant number of parameters, thus ultimately not applicable to real-time RF settings. Recently, Restuccia and Melodia[13] have demonstrated the need for real-time hardware-based RF deep learning. However, the main limitation of this study[13] is that it focused on the learning aspect only, ultimately not addressing the problem of connecting real-time inference with receiver reconfigurability.

7. CONCLUSION

This paper has proposed *PolymoRF*, a prototype that can be reused to develop and test novel polymorphic wireless communication systems. One of the key insights brought by our experimental evaluation is that the RF channel may impact the performance of *RFNet* to a significant extent. To this end, we can (i) train different learning models for different channels and reconfigure the weights of *RFNet* in the FPGA accordingly; and (ii) apply small, controlled modifications to the RF signal at the transmitter's side to compensate for the current RF channel condition. Another core aspect is the impact of polymorphism on the effectiveness of smart jamming attacks. We are conscious that the aforesaid issues are definitely worth investigating; however, they deserve separate papers and are the subject of our ongoing work.

Acknowledgments

This work is supported in part by the Office of Naval Research (ONR) under contracts N00014-18-9-0001. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements of the ONR or the U.S. government.

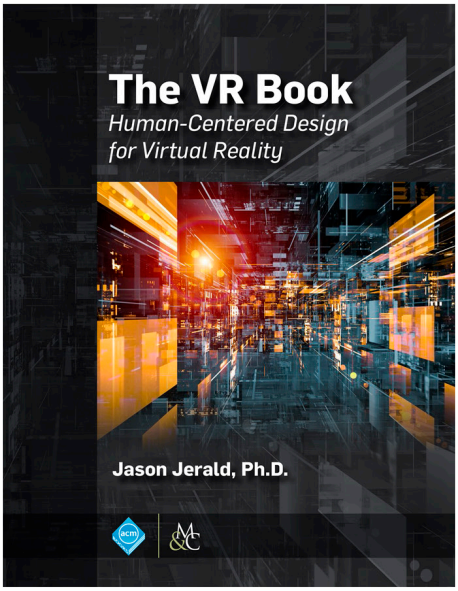
References

1. Analog Devices Incorporated. AD9361 RF agile transceiver data sheet, 2018. http://www.analog.com/media/en/technical-documentation/data-sheets/AD9361.pdf.
2. Chen, Y., Oh, H.-S. A survey of measurement-based spectrum occupancy modeling for cognitive radios. *IEEE Commun. Surv. Tutor 18*, 1 (2016), 848–859.
3. Cisco Systems. Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper, 2017. http://tinyurl.com/zzo6766.
4. Ghodeswar, S., Poonacha, P.G. An SNR estimation based adaptive hierarchical modulation classification method to recognize M-ary QAM and M-ary PSK signals. In *Proceedings of International Conference on Signal Processing, Communication and Networking (ICSCN)* (2015).
5. Jin, X., Sun, J., Zhang, R., Zhang, Y., Zhang, C. SpecGuard: Spectrum misuse detection in dynamic spectrum access systems. *IEEE Trans. Mob. Comput 17*, 12 (Dec 2018), 2925–2938.
6. Kulin, M., Kazaz, T., Moerman, I., Poorter, E.D. End-to-end learning

from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications. *IEEE Access 6*, 2018, 18484–18501.
7. Mao, Q., Hu, F., Hao, Q., Deep learning for intelligent wireless networks: A comprehensive survey. *IEEE Commun. Surv. Tutor 20*, 4 (2018), 2595–2621.
8. Mitola, J., Maguire, G.Q. Cognitive radio: Making software radios more personal. *IEEE Pers. Commun 6*, 4 (1999), 13–18.
9. Molanes, R.F., Rodríguez-Andina, J.J., Fariña, J. Performance characterization and design guidelines for efficient processor - FPGA communication in cyclone V FPSoCs. *IEEE Trans. Ind. Electron. 65*, 5 (2018), 4368–4377.
10. O'Shea, T.J., Corgan, J., Clancy, T.C. Convolutional radio modulation recognition networks. In *International Conference on Engineering Applications of Neural Networks* (2016), Springer, 213–226.
11. O'Shea, T.J., Hoydis, J. An introduction to deep learning for the physical layer. *IEEE Trans. Cogn. Commun. Netw 3*, 4 (2017), 563–575.
12. O'Shea, T.J., Roy, T., Clancy, T.C. Over-the-air deep learning based radio signal classification. *IEEE J. Sel. Top. Signal Process 12*, 1 (2018), 168–179.
13. Restuccia, F., Melodia, T. Big data goes small: Real-time spectrum-driven embedded wireless networking through deep learning in the RF loop. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, (2019).
14. Restuccia, F., Melodia, T. DeepWiERL: Bringing deep reinforcement learning to the internet of self-adaptive things. *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, 2020.
15. Restuccia, F., Melodia, T. Physical-layer deep learning: Challenges and applications to 5G and beyond. arXiv:2004.10113 (2020).
16. Shi, Q., Karasawa, Y. Automatic modulation identification based on the probability density function of signal phase/ *IEEE Trans. Commun 60*, 4 (April 2012), 1033–1044.
17. Subramaniam, S., Reyes, H., Kaabouch, N. Spectrum occupancy measurement: An autocorrelation based scanning technique using USR. In *Proceedings of IEEE Annual Wireless and Microwave Technology Conference (WAMICON)* (2015).
18. Wang, T., Wen, C.-K., Wang, H., Gao, F., Jiang, T., Jin, S. Deep learning for wireless physical layer: Opportunities and challenges. *China Commun 14*, 11 (2017), 92–111.
19. West, N.E., O'Shea, T.J. Deep architectures for modulation recognition. In *Proceedings of IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)* (2017).
20. Winterstein, F., Bayliss, S., Constantinides, G.A. High-level synthesis of dynamic data structures: A case study using Vivado HLS. In *Proceedings of International Conference on Field-Programmable Technology (FPT)* (2013).
21. Xilinx Inc. Zynq-7000 SoC data sheet: Overview, 2018. https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.
22. Xilinx Inc. ZC706 evaluation board for the Zynq-7000 XC7Z045 all programmable SoC user guide, 2018. https://www.xilinx.com/support/documentation/boards_and_kits/zc706/ug954-zc706-eval-board-xc7z045-ap-soc.pdf.
23. Xilinx Inc.. AXI reference guide, UG761 (v13.1) March 7, 2011 (2011). https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf0.0.0.0.
24. Xiong, W., Bogdanov, P., Zheleva, M. Robust and efficient modulation recognition based on local sequential IQ features. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)* (2019).
25. Yu, Y., Wang, T., Liew, S.C. Deep-reinforcement learning multiple access for heterogeneous wireless networks. *IEEE J. Sel. Areas Commun. 37*, 6 (2019), 1277–1290.

Francesco Restuccia and Tommaso Melodia({frestuc, melodia}@northeastern.edu), Institute for the Wireless Internet of Things, Northeastern University, Boston, MA, USA

© 2022 ACM 0001-0782/22/9 $15.00



Without a clear understanding of the human side of virtual reality, the experience will always fail.

"Dr. Jerald has recognized a great need in our community and filled it. The VR Book is a scholarly and comprehensive treatment of the user interface dynamics surrounding the development and application of virtual reality. I have made it a required reading for my students and research colleagues. Well done!"

- Professor Tom Furness, University of Washington VR Pioneer and Founder of HIT Lab International and the Virtual World Society

ISBN: 978-1-970001-12-9 DOI: 10.1145/2792790
http://books.acm.org
http://www.morganclaypoolpublishers.com/vr