Real-Time and Embedded Deep Learning on FPGA for RF Signal Classification

Sohraab Soltani, Yalin E. Sagduyu, Raqibul Hasan, Kemal Davaslioglu, Hongmei Deng, and Tugba Erpek
Intelligent Automation, Inc., Rockville, MD, USA
Email: {ssoltani, ysagduyu, rhasan, kdavaslioglu, hdeng, terpek}@i-a-i.com

***Abstract*—We designed and implemented a deep learning based RF signal classifier on the Field Programmable Gate Array (FPGA) of an embedded software-defined radio platform, DeepRadio$^{\text{TM}}$, that classifies the signals received through the RF front end to different modulation types in real time and with low power. This classifier implementation successfully captures complex characteristics of wireless signals to serve critical applications in wireless security and communications systems such as identifying spoofing signals in signal authentication systems, detecting target emitters and jammers in electronic warfare (EW) applications, discriminating primary and secondary users in cognitive radio networks, interference hunting, and adaptive modulation. Empowered by low-power and low-latency embedded computing, the deep neural network runs directly on the FPGA fabric of DeepRadio$^{\text{TM}}$, while maintaining classifier accuracy close to the software performance. We evaluated the performance when another SDR (USRP) transmits signals with different modulation types at different power levels and DeepRadio$^{\text{TM}}$receives the signals and classifies them in real time on its FPGA. A smartphone with a mobile app is connected to DeepRadio$^{\text{TM}}$to initiate the experiment and visualize the classification results. With real radio transmissions over the air, we show that the classifier implemented on DeepRadio$^{\text{TM}}$achieves high accuracy with low latency (microsecond per sample) and low energy consumption (microJoule per sample), and this performance is not matched by other embedded platforms such as embedded graphics processing unit (GPU).**

***Index Terms*—Software-defined radio, deep learning, modulation classification, FPGA.**

I. Introduction

The wireless communication environment is often characterized by high mobility, channel uncertainty, growth in traffic demands, and vulnerability to jamming. New agile technology solutions are much needed to characterize the spectrum and ensure reliable communications in such a fast-paced dynamic environment. Furthermore, spectrum resources are scarce across time, frequency, and space dimensions, and often shared among a variety of users and applications such as sensing, communications, and electronic warfare (EW). Cognitive radio has emerged as the enabling technology to make efficient use of spectrum resources and support adaptation of wireless communications in highly dynamic environments [1]. Supported by flexible software-defined radio (SDR) designs and implementations [2], cognitive radio finds both commercial and tactical applications in conventional and emerging communications systems.

Cognitive radio performs various tasks such as spectrum sensing and dynamic spectrum access (DSA) for situational awareness and spectrum agility. To support these tasks, machine learning provides automated means to learn from and adapt to spectrum dynamics [3], [4]. In particular, deep learning can process raw spectrum data and effectively operate on the latent representations by capturing and analyzing high-dimensional and dynamic spectrum data that conventional feature-based machine learning algorithms fail to grasp.

Various waveform, channel, traffic, and interference effects, each with its own complex structures that quickly change over time, shape the spectrum [5]. One critical step to assess the spectrum in terms of resources or vulnerabilities is to classify wireless signals. Signal classification is a key step of various tasks in wireless security and communications such as jamming/anti-jamming, device/RF fingerprinting, signal authentication, perimeter security, and interference hunting. Therefore, there is a growing interest in applying deep learning to signal classification such as modulation recognition [6]. Deep learning was applied to signals collected over the air [7]. However, the processing was confined to a host computer, which results in longer processing delay and limits it portability to embedded platforms.

Wireless systems operate with high data rates, ranging from MHz in LTE to GHz in 5G systems. Therefore, high-volume data samples (in I/Q form) arrive at RF receivers to be processed. The latency to move the data to a host processor for deep learning is not feasible in real-time operations such as those that need to make a quick decision in microseconds time frame. While missing, a real-time embedded implementation of deep neural networks on the SDR is ultimately needed to support the embedded applications that run on stand-alone SDRs.

To fill this gap, we designed and implemented a deep neural network based classifier for signal classification on the Field Programmable Gate Array (FPGA) of DeepRadio$^{\text{TM}}$. Without using any other host processor or deep learning accelerator (such as [8]), the deep neural network runs directly on the FPGA fabric. This embedded solution provides high accuracy, low latency, and low power. In particular, we show the following novel capabilities for RF signal classification:

- low-latency (microseconds), matching the speeds of RF spectrum dynamics in terms of channel, interference, and

This effort is partially supported by the U.S. Army Research Office under contract W911NF-17-C-0090. The content of the information does not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

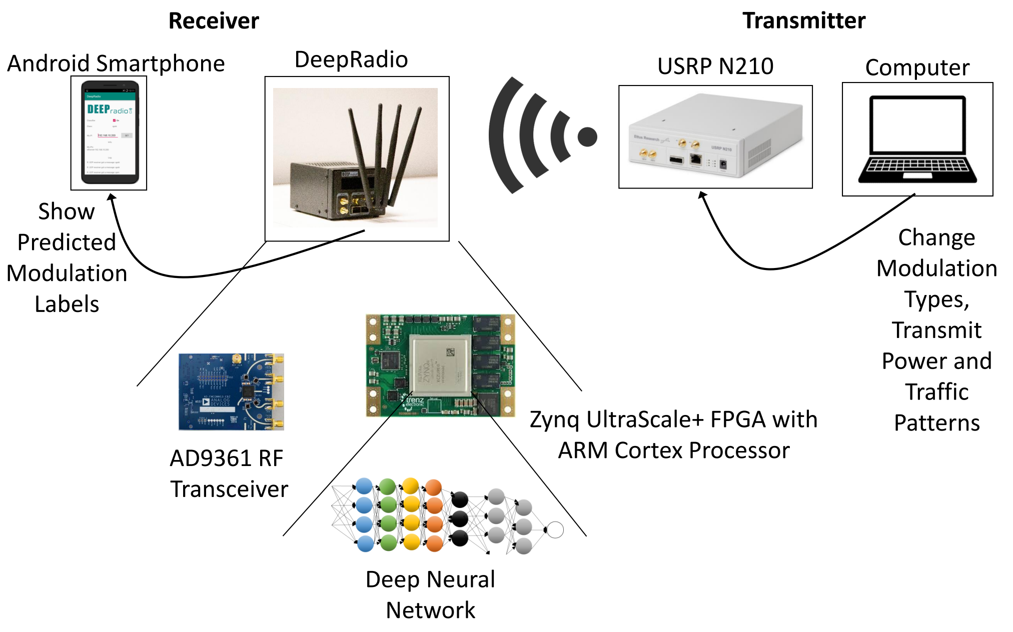

Receiver
Transmitter
Android Smartphone
DeepRadio
USRP N210
Computer
Show Predicted Modulation Labels
Change Modulation Types, Transmit Power and Traffic Patterns
AD9361 RF Transceiver
Zynq UltraScale+ FPGA with ARM Cortex Processor
Deep Neural Network

Fig. 1: System setup for RF signal classification.

traffic,

- low energy consumption (microJoule), resulting in a longer battery lifetime (translated to longer network lifetime) and high portability, and
- high accuracy, approaching the limits of floating-point software operation.

Note that our goal is not to introduce another SDR. We use DeepRadio^{TM}as an FPGA-based RF platform and provide a flexible design that can be ported to any other SDR that is equipped with FPGA. Our paper has three major discriminators compared to the state of the art that typically considers offline software implementation of simulated or prerecorded data:

- We consider hardware generated and over the air transmitted physical data.
- We consider algorithm implementation on FPGA in an embedded hardware platform for low latency and low power consumption.
- We consider real-time operation for both data collection and algorithm run.

The rest of the paper is organized as follows. Section II discusses related work. Section III introduces the system setup. Section IV presents implementation for deep neural network on FPGA. Section V presents the results in terms of classification accuracy, latency, and power efficiency.

II. Related Work

Deep learning finds rich application in wireless communications. Examples include spectrum sensing [10], MIMO detection [11], channel estimation and signal detection [12], physical layer communications [13], jammer detection [14], stealth jamming [15], [16], power control [17], signal spoofing [18], and transmitter-receiver scheduling [19].

RF signal classification can support different applications such as radio fingerprinting [28] that can be ultimately used in cognitive radio systems [29] subject to dynamic and unknown interference and jamming effects [30]. Modulation classification has been extensively studied with deep neural networks [6], [7], [20]–[27], where the goal is to classify a given signal to a known modulation type. Different types of datasets have been used to train deep neural network for modulation classification. For example, [6] provided a training dataset collected from GNU Radio without any real channel or hardware impairments. On the other hand, [7] provided a training dataset collected from over-the-air measurements of USRP radio transmissions. However, those studies have performed modulation classification offline in software environments. Our goal is to run modulation classification in an embedded platform in real time while accounting for latency and power efficiency requirements.

III. System Setup

The system setup is shown in Fig. 1. There are two major components, a transmitter and a receiver.

- There is one USRP N210 SDR that is controlled by a computer as an RF front end. This USRP either waits or transmits narrowband signals at 2.4 GHz frequency. Different transmit powers will be employed to generate different signal-to-noise-ratio (SNR) effects. The transmissions can be made over the air or over cables depending on spectrum management restrictions. Each transmitted signal is modulated with one of six different modulation types, namely

1) Binary Phase Shift Keying (BPSK),
2) Quadrature Phase Shift Keying (QPSK),
3) Continuous Phase Modulation (CPM),

4) Gaussian Frequency Shift Keying (GFSK),
5) Quadrature Amplitude Modulation (QAM) 16, and
6) Gaussian Minimum Shift Keying (GMSK).

- As receiver and classifier, DeepRadio™runs a signal classifier by taking the received signals (I/Q samples) as input data and determines whether the received signal is noise (label 0) or it is a signal transmitted with one of the six modulation types (labels 1-6).

Note that the uniqueness of the system set up is at the receiver side that makes real-time decisions by running a deep learning classifier on the FPGA of DeepRadio™.

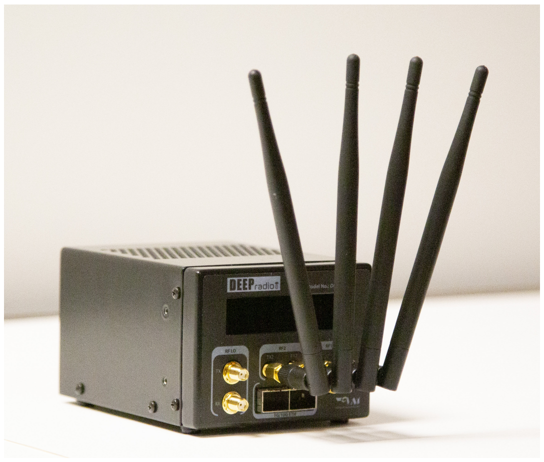


Fig. 2: DeepRadio™used for the RF classifier implementation.

The received signals are passed to the FPGA and run through a deep neural network that returns the signal labels (noise or one of six modulation types). DeepRadio™shown in Figure 2 is equipped with Analog Devices AD9361 RF transceiver and Xilinx Zynq UltraScale+ XCZU9EG FPGA with ARM Cortex Processor (Quad-core ARM Cortex-A53 1.2 GHz, 64 bit). The FPGA has 600 K System Logic Cells, 32.1 Mb Memory, $2,520$ DSP Slices, and 328 Maximum I/O Slices. The MIMO-capable RF transceiver covers 70 MHz to 6 GHz with tunable instantaneous bandwidth of between 200 kHz to 56 MHz.

Performance with the FPGA is compared to an embedded graphics processing unit (GPU) that connects to DeepRadio™to receive I/Q data and runs the RF classifier. For that purpose, we use two types of embedded GPU from NVIDIA, namely Jetson AGX Xavier Developer Kit (512-core Volta GPU with Tensor Cores) and Jetson Nano Developer Kit (128-core Maxwell).

IV. Implementation of Deep Learning based RF Signal Classifier

The FPGA implementation focuses on inference (test) time. Hence, the training data was collected offline and a deep neural network was trained offline. The trained model is a feedforward neural network (FNN). The input layer receives 900 I/Q signal samples that constitute one data sample for which a modulation label is assigned.

The deep neural network is a function $F(\mathbf{x}) = \mathbf{y}$ that takes an n-dimensional input $\mathbf{x} \in \mathrm{R}^n$ (in our case $\mathbf{x}$ is the received signal vector) and produces an m-dimensional output $\mathbf{y} \in \mathrm{R}^m$ (in our case $\mathbf{y}$ is the vector of likelihood scores corresponding to modulation types). For an m-class classifier, the output vector $\mathbf{y}$ satisfies $y_1 + \ldots + y_m = 1$ and $1 \geq y_i \geq 0$. The classifier assigns a label $C(\mathbf{x}) = \mathrm{argmax}_i F(\mathbf{x})$. An activation function, denoted by $\sigma(\cdot)$ is applied at layer i for weights $\boldsymbol{\theta}_i$ and biases b_i to perform $\sigma(\boldsymbol{\theta}_i \mathbf{x} + b_i)$ operation. Let F_i denote such operation at each layer, $\sigma(\boldsymbol{\theta}_i \mathbf{x} + b_i)$, then a k-layer neural network can be represented as $F = \mathrm{softmax} \circ F_k \circ F_{k-1} \circ \ldots \circ F_1$ [31]. The neural network training tries to minimize a cost function $J(\boldsymbol{\theta})$ using the backpropagation algorithm by computing the gradient of the cost function with respect to neural network parameters, i.e., $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$. After hyperparameter optimization, the deep neural network architecture consists of four layers. The number of neurons is 1800 at the input layer, 100 and 20 at the first and second hidden layers, respectively, and 7 at the output layer. The Rectified linear unit (ReLU) activation function is used at hidden layers. ReLU performs the $f(x) = \max(0, x)$ operation on input x. Softmax activation function is used at the output layer. Softmax performs $f(\boldsymbol{x})_i = e^{x_i} / \sum_j e^{x_j}$ on input $\boldsymbol{x}$. The deep neural network is trained with the backpropagation algorithm [32] in TensorFlow [33] using crossentropy as the loss function. Cross-entropy function is given by $\mathcal{L} = -\sum_{i=1}^{m} \beta_i \log(y_i)$, where $\boldsymbol{\beta} = \{\beta_i\}_{i=1}^{m}$ is a binary indicator of ground truth such that only the index corresponding to correct label in $\boldsymbol{\beta}$ is 1 and others are 0. The predicted outputs by the neural network are denoted by y_i's. Adam optimizer [34] is used to update network weights iteratively based on training data.

Figure 3 shows the block diagram of the system implementing a layer of neurons in the FNN on the FPGA. Each layer of the FNN performs essentially same operations except with different numbers of neurons and synapses. Each neuron, in a FNN evaluates the dot-product of the inputs and its weights. As the neurons in a layer do not depend on each other for their operations, we perform the operations of each neuron in parallel. Parallel execution of the neurons provides low latency execution of the FNN on the FPGA. We implemented the ReLU activation on FPGA using conditional operation.

The resulting TensorFlow model is converted to a format readable by FPGA and the bit file is generated. This bit file is uploaded to the FPGA. This way, the RF signal classifier runs on the FPGA without incurring any delay due to another host machine. The output label returned by the FPGA is passed to an Android smartphone for display. A mobile app is running on the smartphone to display the predicted label. The graphical user interface (GUI) of the smartphone is shown in Figure 4.

V. RF Classification Performance

We measured that the RF signal classifier implemented on the FPGA achieves high accuracy ($> 94\%$ when averaged over different SNRs) with low latency and low energy consumption. Vivado Design Suite [35] is used to simulate and

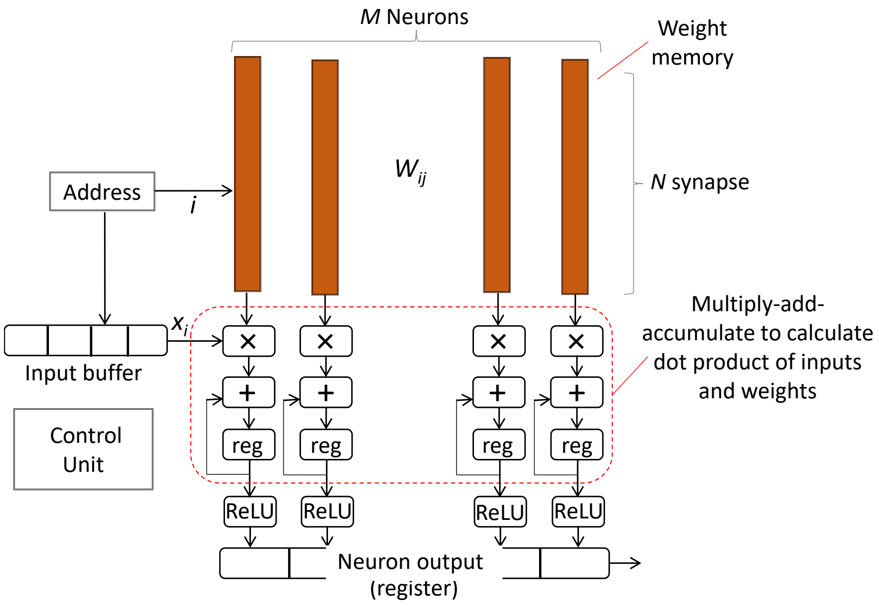
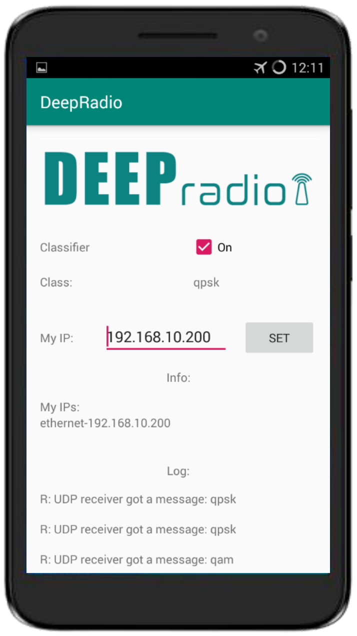


Fig. 3: Block diagram of FNN implementation.


DeepRadio

DEEPradio

Classifier [✓ On]

Class: qpsk

My IP: 192.168.10.200 SET

Info:

My IPs:
ethernet-192.168.10.200

Log:

R: UDP receiver got a message: qpsk

R: UDP receiver got a message: qpsk

R: UDP receiver got a message: qam

Fig. 4: RF classifier GUI.

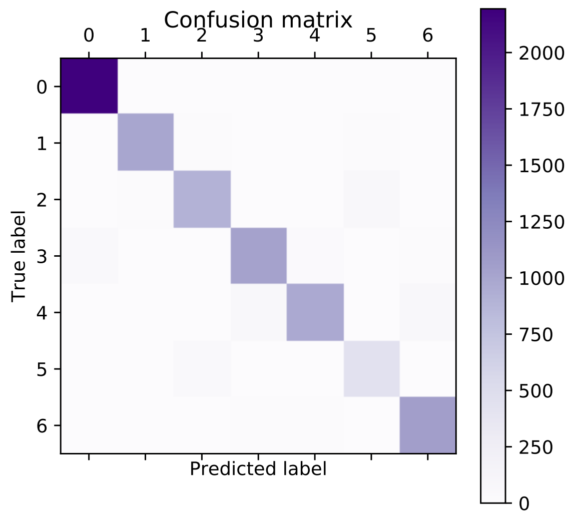


Fig. 5: Confusion matrix of RF signal classifier.

then synthesize the FPGA code. We port this code to FPGA and obtain classification results in real time from FPGA.

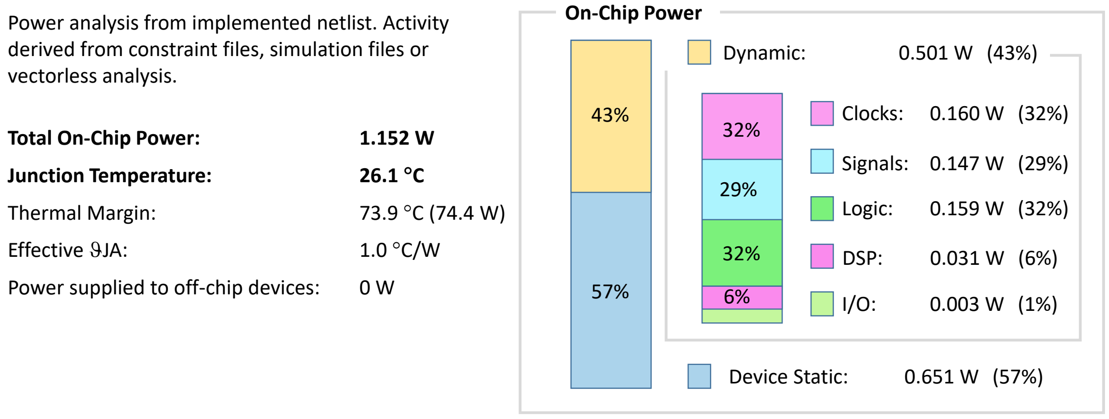
The confusion matrix is shown in Figure 5, where label 0 is noise, label 1 is, BPSK, label 2 is QPSK, label 3 is CPM, label 4 is GFSK, label 5 is QAM16, and label 6 is GMSK. The average probability of error is shown in Table I for each ground truth label.

FPGA resource allocation is measured in Vivado. Resource allocation breakdown for RF signal classifier is shown in Table II, where LUT: lookup table, LUTRAM: lookup table RAM FF: flip flop, DSP: digital signal processing, IO: input output, BUFG: global buffer.

We have the 16-bit implementation of deep neural network on FPGA and the performance gap from the floating implementation in software (on embedded GPU) is within 1%. When the RF classifier is run on the FPGA, the latency is 24 μs per sample and the energy consumption is 28 μJ per sample. Note that 24 μs translates to operating with >37 megasamples per second without any downsampling. It is possible to increase the rate by downsampling, reducing

TABLE I: Average probability of error in classifying signals for each ground truth label.

Ground Truth Label	noise	BPSK	QPSK	CPM	GFSK	QAM16	GMSK
Average Error of Misclassification	3.2%	8.2%	8.6%	13.9%	13.9%	0.10%	3%



Power analysis from implemented netlist. Activity derived from constraint files, simulation files or vectorless analysis.

Total On-Chip Power:	**1.152 W**
Junction Temperature:	**26.1 °C**
Thermal Margin:	73.9 °C (74.4 W)
Effective ϑJA:	1.0 °C/W
Power supplied to off-chip devices:	0 W

Fig. 6: Power consumption on FPGA.

TABLE II: Average probability of error in classifying signals for each ground truth label.

Resource	Utilization	Available	Utilization (%)
LUT	158,435	274,080	57.81
LUTRAM	117,380	144,000	81.51
FF	16,222	548,160	2.96
DSP	210	2,520	8.33
IO	10	328	3.05
BUFG	4	404	0.99

the size of FNN, or using the available FPGA resources to implement another classifier in parallel on the FPGA. Figure 6 shows the detailed breakdown of power consumption obtained from Vivado.

FPGA achieves better latency and power performance compared to other embedded platforms such as embedded GPU. When the RF classifier is run on the Jetson AGX Xavier GPU, the latency is 3.6 ms per sample and the energy consumption is 36 mJ per sample. When the RF classifier is run on the Jetson Nano GPU, the latency is 4.1 ms per sample and the energy consumption is 21mJ per sample. Table III shows the performance comparison of FPGA and embedded GPU.

The predicted label is displayed in the smartphone GUI. Figure 7 shows how the predicted label changes first when there is no transmission, second when a signal with BPSK is transmitted, and when third a signal with QPSK is transmitted.

VI. Conclusion

The paper presents the embedded implementation of deep learning based RF signal classification for low-latency and low-power applications. While deep learning has started finding different applications in wireless security and communications, there is a gap of real radio implementation. This paper fills the gap and opens up new opportunities regarding the use of deep learning in wireless applications. We showed that FPGA implementation of RF signal classifier maintains high accuracy and significantly reduces latency (more than 100 times) and energy consumption (close to 1000 times) compared to an embedded GPU implementation. With this capability, it is possible to support various wireless application such as detecting jammers/emitters, authenticating mobile devices, and identifying spectrum opportunities all in real-time (in microsecond time frame) with low (microJoule) power consumption.

References

[1] S. Haykin, "Cognitive radio: brain-empowered wireless communications,"*IEEE Journal on Selected Areas in Communications*, 2005.

[2] Y. E. Sagduyu, S. Soltani, T. Erpek, Y. Shi and J. Li, "A unified solution to cognitive radio programming, test and evaluation for tactical communications," IEEE Communications Magazine, Oct. 2017.

[3] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *arXiv preprint arXiv:1710.02913*, 2017.

[4] O. Simeone, "A very brief introduction to machine learning with applications to communication systems," *IEEE Transactions on Cognitive Communications and Networking*, 2018.

[5] S. Wang, Y. E. Sagduyu, J. Zhang, and J. H. Li, "Spectrum shaping via network coding in cognitive radio networks", IEEE INFOCOM, 2011.

[6] T. O'Shea, J. Corgan, and C. Clancy, "Convolutional radio modulation recognition networks," *International Conference on Engineering Applications of Neural Networks*, 2016.

[7] T. J. OShea, T. Roy and T. C. Clancy, Over-the-air deep learning based radio signal classification, *IEEE Journal of Selected Topics in Signal Processing*, 2018.

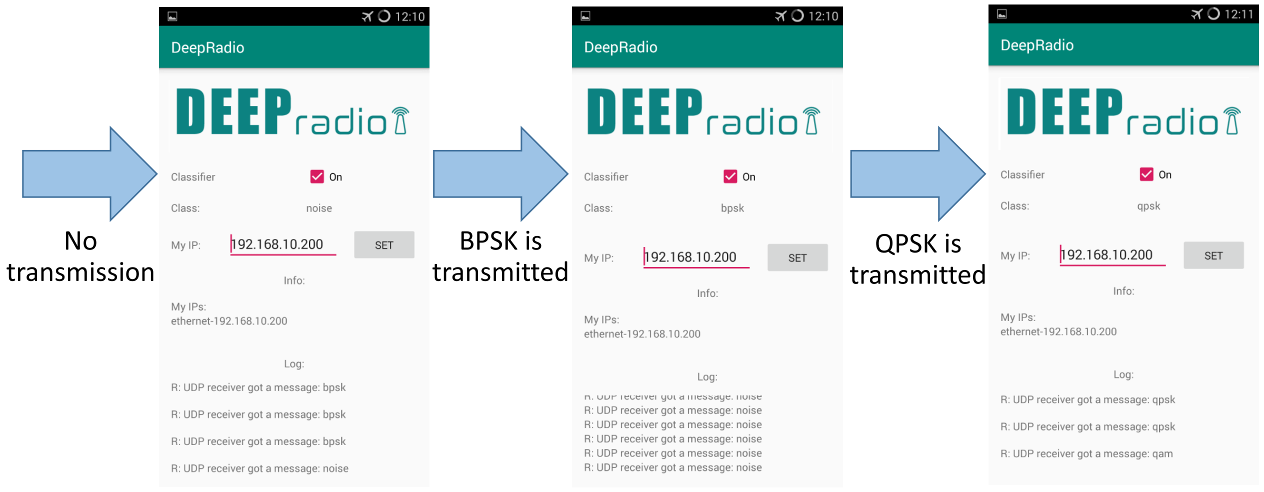
[8] Xilinx AI Inference Acceleration [Online]. Available: https://www.xilinx.com/applications/megatrends/machine-learning.html

[9] W. Lee, M. Kim, D. Cho, and R. Schober, "Deep Sensing: Cooperative spectrum sensing based on convolutional neural networks," *arXiv preprint* arXiv:1705.08164, 2017.

[10] K. Davaslioglu and Y. E. Sagduyu, "Generative adversarial learning for spectrum sensing," *IEEE International Conference on Communications (ICC)*, 2018.

TABLE III: Performance comparison of FPGA and embedded GPU.

Performance Measure \ Device	FPGA	Xavier GPU	Nano GPU
Average Classification Accuracy	94%	95%	95%
Latency (per sample)	24 μs	3.6 ms	4.1 ms
Energy Consumption (per sample)	28 μJ	36 mJ	21 mJ



Classifier [✓ On]
Class: noise
My IP: 192.168.10.200
Info:
My IPs:
ethernet-192.168.10.200
Log:
R: UDP receiver got a message: bpsk
R: UDP receiver got a message: bpsk
R: UDP receiver got a message: bpsk
R: UDP receiver got a message: noise

Classifier [✓ On]
Class: bpsk
My IP: 192.168.10.200
Info:
My IPs:
ethernet-192.168.10.200
Log:
R: UDP receiver got a message: noise
R: UDP receiver got a message: noise
R: UDP receiver got a message: noise
R: UDP receiver got a message: noise
R: UDP receiver got a message: noise
R: UDP receiver got a message: noise

Classifier [✓ On]
Class: qpsk
My IP: 192.168.10.200
Info:
My IPs:
ethernet-192.168.10.200
Log:
R: UDP receiver got a message: qpsk
R: UDP receiver got a message: qpsk
R: UDP receiver got a message: qam

Fig. 7: Recognized modulation is displayed in the smartphone GUI.

[11] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "A model-driven deep learning network for MIMO detection," *arXiv preprint*, arXiv:1809.09336, 2018.

[12] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in OFDM systems," *IEEE Wireless Communications Letters*,2018.

[13] T. J. O'Shea, and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking (TCCN)*, 2017.

[14] Y. Shi, Y. E. Sagduyu, T. Erpek, K. Davaslioglu, Z. Lu, and J. Li, "Adversarial deep learning for cognitive radio security: jamming attack and defense strategies," *IEEE ICC 2018 Workshop - Promises and Challenges of Machine Learning in Comm. Networks*, 2018.

[15] Y. Shi, T. Erpek, Y. E. Sagduyu, and J. Li, "Spectrum data poisoning with adversarial deep learning," *IEEE Military Communications Conference*, 2018.

[16] Y. E. Sagduyu, Y. Shi, and T. Erpek, "IoT network security from the perspective of adversarial deep learning," *IEEE International Conference on Sensing, Communication and Networking (SECON) Workshop on Machine Learning for Communication and Networking in IoT*, 2019.

[17] T. Erpek, Y. E. Sagduyu, and Y. Shi, "Deep learning for launching and mitigating wireless jamming attacks," *IEEE Transactions on Cognitive Communications and Networking*, 2019.

[18] Y. Shi, K. Davaslioglu, and Y. E. Sagduyu, "Generative adversarial network for wireless signal spoofing," *ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec) Workshop on Wireless Security and Machine Learning (WiseML)*, 2019.

[19] N. Abu Zainab, T. Erpek, K. Davaslioglu, Y. E. Sagduyu, Y. Shi, S. Mackey, M. Patel, F. Panettieri, M. Qureshi, V. Isler, A. Yener, "QoS and jamming-aware wireless networking using deep reinforcement learning," *IEEE Military Communications Conference (MILCOM)*, 2019

[20] B. Kim, J. Kim, H. Chae, D. Yoon, and J.W. Choi, "Deep neural network-based automatic modulation classification technique," *IEEE International Conference on Information and Communication Technology Convergence (ICTC)*, 2016.

[21] G. J. Mendis, J. Wei, and A. Madanayake, "Deep learning-based automated modulation classification for cognitive radio," *IEEE International Conference on Communication Systems (ICCS)*, 2016.

[22] S. Peng, H. Jiang, H. Wang, H. Alwageed, and Y.-D. Yao, "Modulation classification using convolutional neural network based deep learning model," *IEEE Wireless and Optical Communication Conference (WOCC)*, 2017.

[23] Y. Tu, Y. Lin, J. Wang, and J.-U. Kim, "Semi-supervised learning with generative adversarial networks on digital signal modulation classification," *Comput. Mater. Continua*, 2018).

[24] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders and S. Pollin, "Deep learning models for wireless signal classification with distributed low-cost spectrum sensors," *IEEE Transactions on Cognitive Communications and Networking*, 2018.

[25] C. de Vrieze, L. Simic, and P. Mahonen, "The importance of being earnest: Performance of modulation classification for real RF signals," *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2018.

[26] A. Ali and Y. Fan, "Unsupervised feature learning and automatic modulation classification using deep learning model," *Physical Communication*, vol. 25 (2017): 75–84.

[27] Y. Shi, K. Davaslioglu, Y. E. Sagduyu, W. C. Headley, M. Fowler, and G. Green, "Deep learning for signal classification in unknown and dynamic spectrum environments," *IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, 2019.

[28] F. Restuccia, S. D'Oro, A. Al-Shawabka, M. Belgiovine, L. Angioloni, S. Ioannidis, K. R. Chowdhury, T. Melodia, "DeepRadioID: Real-time channel-resilient optimization of deep learning-based radio fingerprinting algorithms," *ACM Intl. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2019

[29] S. Soltani, Y. E. Sagduyu, Y. Shi, J. Li, J. Feldman, and J. Matyjas, "Distributed cognitive radio network architecture, SDR implementation and emulation testbed," *IEEE Military Communications Conference (MILCOM)*, 2015.

[30] Y. E. Sagduyu, R. Berry, and A. Ephremides, "Jamming games in wireless networks with incomplete information," *IEEE Communications Magazine*, 2011.

[31] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," *EEE Symposium on Security and Privacy*, 2017.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, 1986.

[33] M. Abadi, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[34] D. P. Kingma, and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[35] Xilinx Vivado Design Suite, [Online]. Available: https://www.xilinx.com/products/design-tools/vivado.html