Real-time Automatic Modulation Classification

Stephen Tridgell, David Boland, Philip H.W. Leong and Siddhartha
The University of Sydney
Email: {stephen.tridgell, david.boland, philip.leong, siddhartha.siddhartha}@sydney.edu.au

***Abstract*—Deep learning based techniques have shown promising results over traditional hand-crafted methods for automatic modulation classification for radio signals. However, implementation of these deep learning models on specialized hardware can be challenging, as both latency and throughput performance are critical to achieving real-time response to over-the-air radio signals. In this work, we meet our targets by designing an optimized ternarized convolutional neural network that leverages the RF capabilities offered by the Xilinx ZCU111 RFSoC platform. The implemented networks achieve high-speed real-time performance with a classification latency of $\approx$8μs, and an operational throughput of 488k classifications per second. On the challenging open-source RadioML dataset, we achieve up to 81.1% accuracy, which is competitive to existing state-of-the-art *software-only* implementations.**

I. Introduction

Deep neural networks in recent years have surpassed state-of-the-art performance in a variety of fields such as computer vision, speech recognition, and machine translation [10]. Convolutional neural networks (CNNs), in particular, have done very well for pattern recognition problems. There is also growing interest in CNNs for radio frequency (RF) applications, particularly in automatic modulation classification (AMC) [5], [9], [14]. In AMC, the goal is to accurately identify the modulation type used by a transmitting source based on RF samples taken from the environment. Deep neural networks have been shown to outperform the state-of-the-art in this domain, especially when there is a low signal-to-noise (SNR) ratio in the transmission channel [5], [14].

While research into deep learning methods for AMC has gained significant traction, there are far fewer efforts undertaken on realizing these machine-learning based systems as real-time hardware implementations. CNNs are typically compute-bound, and while domain-specific accelerators can offer markedly better performance over commodity general-purpose hardware, they are usually targeted towards computer vision applications where a classification throughput of tens to hundreds of classifications per second is sufficient. Figure 1 shows this effect on a modern NVIDIA RTX 2080 Ti GPU, which tops out at a peak throughput of $\approx$ 30k classifications per second on the smaller models. In addition, a large batch-size plays a vital role in sustaining this throughput, which may not be feasible for a real-time implementation due to latency constraints. For RF applications, the data-rate can be very high on the order of 100s of millions of samples per second, and hence, both latency and throughput of the AMC design are critical for sensing and responding to changes in the RF channel.

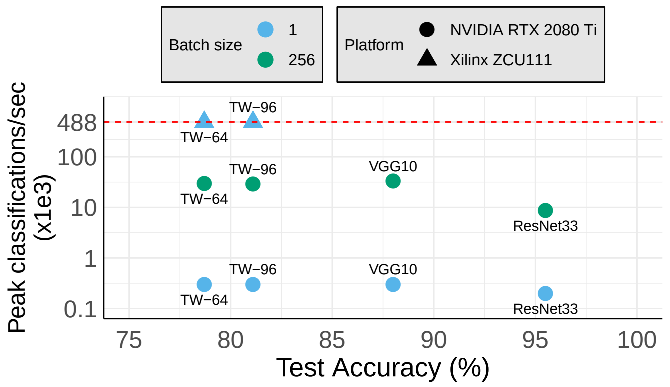

Figure 1: Peak classification throughput vs model accuracy on 24 modulation classes, measured on two modern hardware platforms. Note that increasing batch size also increases the response latency of the model.

The Xilinx ZCU111 RFSoC (Radio-Frequency System-on-Chip) is a new FPGA-based radio platform targeted for research and development on software-defined radio (SDR) applications. The platform is built on the Zynq Ultrascale+ SoC family, and offers runtime-programmable multi-gigasample RF transceivers for building high-speed radio systems. As seen from Figure 1, we are able to improve the classification throughput by several orders of magnitude on the RFSoC platform (batch size 1), while achieving competitive classification accuracy on a large number of modulation classes. The contributions in this work are as follows:

- A fully-pipelined ternary-weights convolution neural network implementation that is capable of producing high-speed real-time classifications on the signal modulation classification problem.
- An exploration on the effect of hardware design parameters on model accuracy, alongside an emphasis on performance/resource efficiency.
- The first real-time implementation of a high-speed machine learning model on a radio frequency application that takes advantage of the Xilinx ZCU111 SoC.

II. Background

A. Automatic Modulation Classification

Automatic Modulation Classification is a common requirement in cognitive radio networks (CRN) for both military and civilian applications. For example, AMC plays a pivotal role in dynamic spectrum management in CRNs, where cognitive radios are able to detect idle frequency bands and transmit data in these bands such that the primary users of these channels are not affected [8]. AMC, however, can be a challenging task for two reasons: (1) often, there is no apriori information known about the transmission signal/channel (*e.g.* signal-to-noise ratio, carrier frequency, etc), especially in military contexts, and (2) latency/throughput requirements to achieve real-time processing often limit the complexity of the implementation and its runtime effectiveness.

The classical approach to AMC is built on statistical/stochastic methods that require high domain expertise and frequent manual tuning to achieve reliable performance in each specific environment [1], [6], [8]. Recently, notable efforts have been made into using deep learning techniques to push this field further [5], [9], [14]. While these studies have demonstrated an improvement to classification accuracy and reliability, the practicality of the realizing a real-time machine-learning based AMC implementation is left unexplored. Our work adopts a more holistic approach, where expertise in both machine-learning and hardware design is used in tandem to set a benchmark for real-time automatic modulation classification.

B. Deep Neural Networks on FPGAs

Convolutional neural networks (CNNs) are deep neural networks that comprise of convolution, pooling, and dense layers. CNNs come in a variety of flavors, and can be highly customized to suit each application domain. Readers are recommended to refer to [10] for a comprehensive overview of modern CNN design.

While CNNs are typically trained with single/double floating-point precision, inference can be done with reduced precision and/or pruning without a significant loss in accuracy, which opens up opportunities for building accelerators for deployment. FPGAs have become a popular platform to implement these quantized CNNs in recent years as demonstrated in [12]. In addition, ternarization of weights [2], [3] is a popular quantization choice, as it delivers a significant reduction in the memory footprint of the CNN model, while also preserving the model accuracy to a large degree. In ternary weights networks (TWNs), parameters are quantized during training as follows:

$$W_i^l = \begin{cases} +1, & \text{if } W_i^l > \Delta \\ 0, & \text{if } \left|W_i^l\right| \leq \Delta \\ -1, & \text{if } W_i^l < -\Delta \end{cases} \tag{1}$$

where W_i^l are the parameters in each layer of the network, and Δ is a positive threshold parameter used for quantization.

C. RadioML Dataset

We train and evaluate all our network designs on the open-source RadioML 2018.01A dataset[1], which is a collection of raw I/Q samples that have been captured over-the-air using USRP devices as described in [5]. Each training sample is a time-series of 1024 I/Q sample pairs, accompanied by a label that identifies its modulation class. There are a total of 24 modulation classes recorded at 26 signal-to-noise ratio (SNR) levels, ranging from -20dB to +30dB in increments of 2dB. Each {*modulation class,SNR*} pair has 4096 training examples, and hence, there are a total of 2.56M labeled I/Q time-series examples in the entire dataset. The 24 modulation classes include a broad range modulation types, details of which can be found in [5].

[1]https://www.deepsig.io/datasets, *Accessed: July 2019*

D. Related Work

Deep learning for RF applications is a relatively new field, and in particular, existing work on AMC has been restricted to model design and evaluation purely in software. Mendis et. al [4] compute the spectral correlation function (*i.e.* a cyclo-stationary feature extraction step) and implement a deep-belief network for classifying five modulation classes. In [7], the authors report the effectiveness of various deep neural networks on ten modulation classes, and demonstrate several strategies that help reduce training time. Zhang et. al [13] propose a heterogeneous CNN / LSTM (long short-term memory) model that delivers high classification accuracy on eleven modulation classes. While inference runtime is reported for CNN networks (on the order of a few seconds), the authors do not report runtime of the proposed heterogeneous models, which is likely to be larger due to the increased complexity in the models. O'Shea et al. [5] design and evaluate two CNN models (VGG10 and ResNet33) and demonstrate competitive accuracy performance on 24 modulation classes. In all of these studies, runtime boundaries for real-time implementation are not reported, or are incomplete, which is a research gap that we aim to fill with this work.

III. CNN-based Automatic Modulation Classification

We use the models proposed in [5] – VGG10 and ResNet33 – as our baseline, and explore design strategies to achieve a high-speed and resource-efficient FPGA implementation. Unfortunately, implementing a high-throughput ResNet33 model on an FPGA is infeasible due to two reasons: (1) the model size is too large to be spatially mapped to the FPGA fabric, which limits the achievable classification throughput significantly, and (2) the residual connections create an unbalanced design which can result in bottlenecks if large amounts of on-chip memory are not available to store intermediate activations. Hence, we elected to use the smaller VGG10 model instead for our real-time FPGA-based AMC implementation. We experiment with low precision variations of the VGG10 network and explore the tradeoff of computation with accuracy.

The VGG10 model has seven 1D convolutional layers followed by three dense layers. All of the convolutional layers have a kernel size of three and a stride of one. The convolutions are followed by maxpool, batch normalization and the ReLU activation layers. In [5], the first two dense layers use alpha dropout followed by the SELU activation function. We use this training method for networks trained

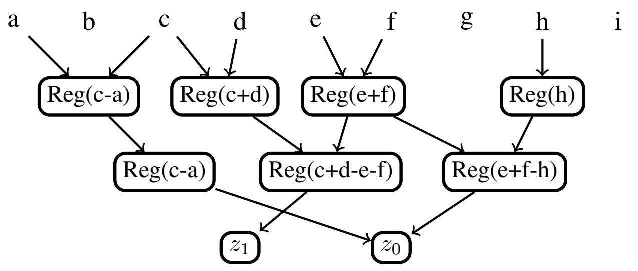

Figure 2: Computing $z_0 = c + e + f - (a + h)$ and $z_1 = c + d - e - f$

with floating-point precision, but swap alpha dropout layers with batch normalization layers when training low-precision networks for improved training stability.

A. Training Method

All models are trained on the RadioML 2018.01A dataset. We set the batch size to 128, the initial learning rate to 10^{-3}, and train for 250k steps. The learning rate was set to smoothly decay exponentially at a rate of 0.5 every 100k steps. The dataset was partitioned into a 90% – 10% split to create the train and test sets respectively, such that each {SNR, modulation class} pair had 3686 train and 410 test examples. For training, we use samples captured at $\geq$ +6dB SNR, which is typically the minimum signal strength observed in most wireless communication systems. This improves the training time, and ensures that the CNN is able to achieve the best classification accuracy for typical real-time use cases. In addition, we also use the teacher-student [2] training methodology to further improve accuracy. In this case, for all VGG10 networks, including floating-point implementations, a trained ResNet33 model was used as the teacher.

Ternary weight networks were quantized using the method described in [3], which results in a network with ternary weights and floating-point activations. The authors in [3] choose a threshold using a value of $\Delta = \nu \cdot E(|W|)$, where they recommend $\nu = 0.7$. As discussed in [11], ν can be used to control the sparsity of the weights, and hence, it has a direct impact on the implementation of the network. For networks with quantized activations we first clip the floating-point activation values between 0 and 1, and then compute

$$x' = round(x * (2^k - 1))/(2^k - 1)$$

where x is an activation, and k is the number of bits to use for the quantization. This activation quantization is done after the ReLU of each layer.

B. Model Design

We leverage model design techniques described in our previous work [11] to implement and evaluate low precision CNNs. Figure 2 shows an example of how a convolution can be visualized as a dataflow graph for evaluating two output map pixels, z_0 and z_1, in a convolution window. This window can then be fed inputs in a pipelined fashion by doing an *im2col* transformation on the input image. The dataflow graph is implemented spatially on the FPGA as hardware blocks in order to maximize available parallelism on the FPGA fabric. We also further optimize the hardware implementation for each specific network by merging common subexpressions to reduce the area required. This method of implementing a ternary neural network in hardware allows for a very compact and high throughput design. See our previous work in [11] for more details.

We explore different model sizes, and our goal is to maximize classification accuracy as much as possible while fully utilizing the available FPGA resources. Table I details all the models and their properties explored in this paper. The first four networks in Table I are trained and tested with floating-point weights and activations. All the networks with the prefix `TW-` are trained with ternary weights (*i.e.* {-1, 0, 1}). For all `TW-` networks, we use $\nu = 0.7$ for the first layer, $\nu = 1.2$ for the remaining convolutional layers, and $\nu = 0.7$ for the two dense layers with ternary weights. All networks have floating-point batch normalization variables. Neither the weights and activations of the final dense layer, nor the input data in all networks is ever trained with quantization.

C. Quantization error

The networks trained in Table I use 32b floating-point for the batch normalization variables and the final dense layer. During inference, the batch normalization variables can be multiplied into the scaling factors of the convolution operation to give an equation $bn(x) = ax + b$, where a and b are known constants that are pre-loaded into the hardware design. In order to avoid implementing floating-point blocks for portions of the computation, calculations are done in fixed-point instead. We empirically determine that these models deliver the best accuracy when 6 fractional bits in the activation layers, and 8 fractional bits for the batch normalization variables are used. We observe minimal numerical difference at the output, typically around 2% for the classes with the largest output activation values. Table I shows that this quantization strategy provides enough bits for stable implementations with minimal differences in accuracy.

IV. Methodology

We use Vivado 2018.3 to synthesize all designs. For training, we use the Tensorflow framework to train and test various models, and we quantize weights in the networks to 2b ternary representations. In order to support our claim for real-time modulation classification, we choose an I/Q sample rate of 500MHz and design the CNN to accept 2$\times$I/Q samples each cycle with a 250MHz clock.

V. Results

A. Resource Utilization

Table II shows the resource utilization for the various models compared in this paper. The DSP usage is relatively constant as the convolutions do not use any DSPs. For both dense layers with 512 outputs, this is 1024 multiplications,

Table I: Properties of various models designed and explored in this paper.

Model Name	Architecture	Precision[1] (Weights/Activations)	# parameters	# MACs	Accuracy[2] (FxP[3])
ResNet33 [5]	{ResBlock}×6, (FC, 128)×2, (FC/Softmax, 24)	32b/32b (FP)	507k (2.03Mb)	111m	95.5
VGG10 [5]	{(Conv, K3, 64), (MaxPool, S2)}×7, (FC, 128)×2, (FC/Softmax, 24)	32b/32b (FP)	102k (407Kb)	12.8m	88.0
VGG10-64	{(Conv, K3, 64), (MaxPool, S2)}×7, (FC, 512)×2, (FC/Softmax, 24)	32b/32b (FP)	381k (1.5Mb)	13.3m	89.6
VGG10-128	{(Conv, K3, 128), (MaxPool, S2)}×7, (FC, 512)×2, (FC/Softmax, 24)	32b/32b (FP)	636k (2.5Mb)	51.1m	90.9
TW-64	{(Conv, K3, 64), (MaxPool, S2)}×7, (FC, 512)×2, (FC/Softmax, 24)	2b/32b (FP)	381k (95kb)	13.3m	78.8 (78.7)
TW-96	{(Conv, K3, 96), (MaxPool, S2)}×7, (FC, 512)×2, (FC/Softmax, 24)	2b/32b (FP)	490k (123kb)	29.1m	82.4 (81.1)
TW-128	{(Conv, K3, 128), (MaxPool, S2)}×7, (FC, 512)×2, (FC/Softmax, 24)	2b/32b (FP)	636k (159kb)	51.1m	82.1 (81.7)

[1]FP = 32b floating-point; [2]Best accuracy at +30dB SNR; [3] with fixed point activations

Table II: Out-of-context resource utilization. Target device: `xczu28dr-ffvg1517-2-e` at 250MHz. [1] Failed to Route

Model	CLBs	LUTs	FFs	BRAMs	DSPs
TW-64	28k (53.5%)	124k (29.1%)	217k (25.5%)	524 (48.5%)	1496 (35%)
TW-96	47k (89.3%)	232k (54.7%)	369k (43.4%)	524 (48.5%)	1207 (28.3%)
TW-128[1]	51k (96.7%)	320k (75.3%)	506k (59.5%)	524 (48.5%)	1431 (33.5%)

whereas for 128 outputs only 256 multiplications need to be performed. It should be noted that this is typically multiplying a 16-bit number with a 2-bit weight, hence mapping it to a DSP is optional for Vivado as this could be computed with CLBs. The largest network, `TW-128`, fails to complete routing. All other designs meet timing constraints with a 250 MHz clock.

VI. Conclusion

In this paper, we demonstrate real-time automatic modulation classification with CNNs on an FPGA. We explore the effect of hardware design parameters on model accuracy and on the performance/resource efficiency. This paper also presents the first real-time implementation, to the best of our knowledge, of a high-speed machine learning model on a radio frequency application that takes advantage of the Xilinx ZCU111 SoC. Our design achieves a very high throughput of 488K classifications/s and a classification latency of just 8μs, while delivering a competitive 81.1% accuracy on the 24-class RadioML dataset.

References

[1] Ameen Abdelmutalab, Khaled Assaleh, and Mohamed El-Tarhuni. Automatic Modulation Classification Based on High Order Cumulants and Hierarchical Polynomial Classifiers. *Phys. Commun.*, 21(C):10–18, December 2016.

[2] Hande Alemdar, Vincent Leroy, Adrien Prost-Boucle, and Frédéric Pétrot. Ternary Neural Networks for Resource-Efficient AI Applications. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2547–2554. IEEE, 2017.

[3] Fengfu Li, Bo Zhang, and Bin Liu. Ternary Weight Networks. *arXiv preprint arXiv:1605.04711*, 2016.

[4] Gihan J Mendis, Jin Wei, and Arjuna Madanayake. Deep Learning-based Automated Modulation Classification for Cognitive Radio. In *2016 IEEE International Conf. on Communication Systems*, pages 1–6. IEEE, 2016.

[5] T. J. 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*, 12(1):168–179, Feb 2018.

[6] Prokopios Panagiotou, Achilleas Anastasopoulos, and A Polydoros. Likelihood Ratio Tests for Modulation Classification. In *MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No. 00CH37155)*, volume 2, pages 670–674. IEEE, 2000.

[7] Sharan Ramjee, Shengtai Ju, Diyu Yang, Xiaoyu Liu, Aly El Gamal, and Yonina C Eldar. Fast Deep Learning for Automatic Modulation Classification. *arXiv preprint arXiv:1901.05850*, 2019.

[8] B. Ramkumar. Automatic Modulation Classification for Cognitive Radios using Cyclic Feature Detection. *IEEE Circuits and Systems Magazine*, 9(2):27–45, Second 2009.

[9] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury. Deep Learning Convolutional Neural Networks for Radio Identification. *IEEE Communications Magazine*, 56(9):146–152, Sep. 2018.

[10] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S Emer. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[11] Stephen Tridgell, Martin Kumm, Martin Hardieck, David Boland, Duncan Moss, Peter Zipf, and Philip H.W. Leong. Unrolling Ternary Neural Networks. *ACM Transactions on Reconfigurable Technology and Systems*, 12(4), 2019.

[12] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. FINN: A Framework for Fast, Scalable Binarized Neural Network Inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 65–74. ACM, 2017.

[13] Duona Zhang, Wenrui Ding, Baochang Zhang, Chunyu Xie, Hongguang Li, Chunhui Liu, and Jungong Han. Automatic Modulation Classification Based on Deep Learning for Unmanned Aerial Vehicles. *Sensors*, 18(3):924, 2018.

[14] Siyang Zhou, Zhendong Yin, Zhilu Wu, Yunfei Chen, Nan Zhao, and Zhutian Yang. A Robust Modulation Classification Method using Convolutional Neural Networks. *EURASIP Journal on Advances in Signal Processing*, 2019(1):21, Mar 2019.