Wireless Modulation Classification Using Deep Learning on FPGAs

Cory Nezin

October 15, 2017

1 Abstract

Wireless modulation classification has been, and continues to be an important engineering problem. Sensing and classifying wireless signals is relevant to applications including government spectrum regulation, cognitive radio, and situational awareness in military/adversarial environments. [1] Deep neural networks have recently achieved impressive performance in classifying audio, images, and video. The application of neural networks to wireless communication has recently grown in the machine learning community. Applications include nonlinear channel modeling, learned data encoding, and modulation classification. [2] While promising results have been achieved, they have only been implemented on graphics processing units (GPUs) which have relatively large size, weight, power, and latency compared to FPGAs. [3] We propose a general framework for converting computational graphs (a more general term than neural network) bulit in TensorFlow [4] into synthesizable VHDL code for implementation on field programmable gate arrays (FPGAs).

In addition to the size, weight, power, and latency advantages offered by FPGA's, they have also drawn attention in deep learning applications for their reconfigurability from large companies like Microsoft. [5] Google has also recently developed specialized hardware for deep learning performance enhancement in the form of the "Tensor Processing Unit" (TPU). The TPU was originally planned to be an FPGA when "[Google] saw that the FPGAs of that time were not competitive in performance compared to the GPUs of that time." [6]

2 Literature Search

2.1 Multiply Accumulate

The multiply accumulate (MAC) operation is perhaps the most important and fundamental operation in signal processing and deep learning. It has the simple form:

$$a \leftarrow a + (b \times c)$$

This operation is fundamental in signal processing because it easily describes feedback. However it is even more important in general because it is the building block of matrix multiplication, which can be used to describe any finite linear transformation. Two of the most computationally intense operations in deep learning for signal analysis are filtering (convolutional layer) and matrix multiplication (fully connected layer). Since both operations are linear, one may think of a convolutional layer as a special case of a fully connected layer but having a special structure which makes its computation faster.

There are two primary architectures for high performance MAC computation: temproal and spatial. Temporal architectures use a central controller to aggregate the data of many arithmetic logic units (ALUs). Spatial architectures allow direct communication between many ALUs in what is called "dataflow processing." [7]

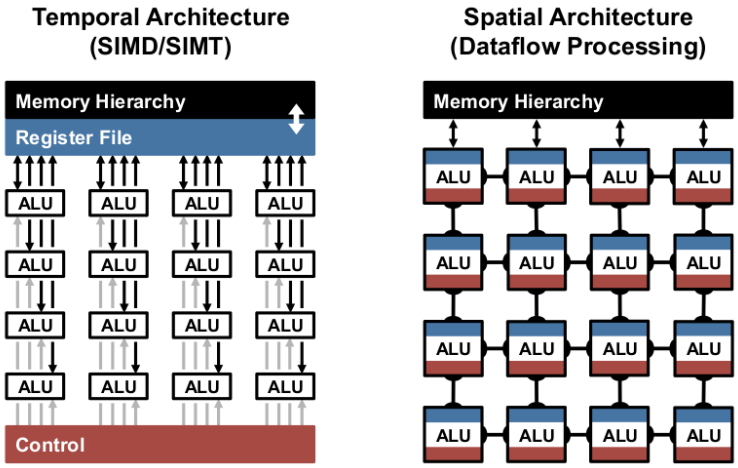

Figure 1: Temporal and Spatial architectures [7]

2.2 Optimization

2.2.1 Convolution

A standard (finite) circular convolution can be computed using the "flip and slide" method [FF] with exactly $N_s N_f$ MACs where N_s is the number of samples in the signal and N_f is the number of filter coefficients. However, it may be possible to do better. The well known convolution theorem states that "Convolution in the time (or spatial) domain is Hadamard (elementwise) multiplication in the frequency (temporal or spatial) domain." That is,

$$x[n] \circledast y[n] = z[n] \implies X[k] \odot Y[k] = Z[k]$$

So we may calculate

$$z[n] = \mathcal{F}^{-1}(\mathcal{F}(x[n]) \odot \mathcal{F}(y[n]))$$

It has been show that the lower bound on the number of multiplications for the fast fourier transform (for inputs of length 2^m) is $4N - 2\log_2^2(N) - 2\log_2 N - 4$, which has $O(N)$ complexity. And in fact, there are known algorithms [8] which achieve this lower bound, but they use too many additions to be practical on modern processors. [9] Practical algorithms such as the split-radix FFT achieve $4N\log_2 N - 6N + 8$ real additions and multiplications, which is $O(N\log(N))$ complexity. [10] Additionally, when dealing with purely real data, there are algorithms which are capable of roughly halving the number of operations. [11] Since our goal is to classify modulation using complex I-Q samples, it is unclear whether this optimization will be helpful, though previous work has suggested that complex neural networks offer only marginal imporvement. [12]

It is clear that direct convolution computation has a complexity of $O(N_s N_f)$ while the Fourier transform method has a complexity of $O(N_s \log N_s)$, assuming that $N_f \leq N_s$ and we don't take advantage of the relative sparsity of the filter. It is not immediately clear which of these is more practical. If $N_s \to \infty$ as N_f remains constant, the direct method is better. If $N_s \to \infty$ and $N_f \to \infty$, the Fourier transform method is better. We are also not accounting for the differing complexities and the fact that specialized hardware may exist for either. Therefore experiment and deeper analysis will be necessary to determine which method is better for our application.

2.2.2 Matrix Multiplication

References

[1] S. Rajendran, R. Calvo-Palomino, M. Fuchs, B. V. den Bergh, H. Cordobés, D. Giustiniano, S. Pollin, and V. Lenders, "Electrosense: Open and big spectrum data," *CoRR*, vol. abs/1703.09989, 2017. [Online]. Available: http://arxiv.org/abs/1703.09989

[2] T. J. O'Shea and J. Hoydis, "An introduction to machine learning communications systems," *CoRR*, vol. abs/1702.00832, 2017. [Online]. Available: http://arxiv.org/abs/1702.00832

[3] N. et al., "Can fpgas beat gpus in accelerating next-generation deep neural networks?" in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: ACM, 2017, pp. 5–14. [Online]. Available: http://doi.acm.org/10.1145/3020078.3021740

[4] M. A. et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467

[5] P. et al., "A reconfigurable fabric for accelerating large-scale datacenter services," in *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ser. ISCA '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 13–24. [Online]. Available: http://dl.acm.org/citation.cfm?id=2665671.2665678

[6] N. P. J. et al., "In-datacenter performance analysis of a tensor processing unit," *CoRR*, vol. abs/1704.04760, 2017. [Online]. Available: http://arxiv.org/abs/1704.04760

[7] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *CoRR*, vol. abs/1703.09039, 2017. [Online]. Available: http://arxiv.org/abs/1703.09039

[8] S. Winograd, "On computing the discrete fourier transform," *Math. Computation*, vol. 32, no. 1, pp. 175–199, Jan. 1978.

[9] P. Duhamel and M. Vetterli, "Fast fourier transforms: A tutorial review and a state of the art," *Signal Process.*, vol. 19, no. 4, pp. 259–299, Apr. 1990. [Online]. Available: http://dx.doi.org/10.1016/0165-1684(90)90158-U

[10] R. Yavne, "An economical method for calculating the discrete fourier transform," in *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, ser. AFIPS '68 (Fall, part I). New York, NY, USA: ACM, 1968, pp. 115–125. [Online]. Available: http://doi.acm.org/10.1145/1476589.1476610

[11] G. D. Bergland, "Numerical analysis: A fast fourier transform algorithm for real-valued series," *Commun. ACM*, vol. 11, no. 10, pp. 703–710, Oct. 1968. [Online]. Available: http://doi.acm.org/10.1145/364096.364118

[12] C. Trabelsi, O. Bilaniuk, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," *CoRR*, vol. abs/1705.09792, 2017. [Online]. Available: http://arxiv.org/abs/1705.09792