Real-time FPGA-based Anomaly Detection for Radio Frequency Signals

Duncan J.M. Moss*, David Boland*, Peyam Pourbeik†, Philip H.W. Leong*
*School of Electrical and Information Engineering
The University of Sydney, Australia 2006
Email: {duncan.moss,david.boland,philip.leong}@sydney.edu.au
†CEWD, Assured Communications, Defence Science & Technology Group, Australia 5111
Email: peyam.pourbeik@dsto.defence.gov.au

***Abstract*—We describe an open source, FPGA accelerated neural network-based anomaly detector. The detector derives its training set from observed exemplar data and performs continuous learning in software via stochastic gradient descent, thus proceeding in an unsupervised manner. Trained network weights are then passed to the FPGA, which performs continuous high-speed anomaly detection, combining parallelism reduced precision, and a single-chip design to maximise performance and energy efficiency. Our design can process continuous 200 MS/s complex inputs, producing anomaly classifications at the same rate, with a latency of 105 ns, an improvement of at least 4 orders of magnitude over a software radio such as GNU Radio.**

I. Introduction

The processing of physical-layer radio-frequency (RF) signals remains challenging due to the very high data rates involved. Over recent years, neural networks (NNs) have achieved results surpassing all other approaches on difficult pattern recognition problems such as image analysis, speech recognition and machine translation. While previous work has demonstrated the utility of applying NNs to RF applications, little has been published on their real-time implementation.

NN-based algorithms are massively parallel in nature, and amenable to computation using low-precision. Together, these two properties make them very suitable for efficient digital implementations. In this paper we describe a field-programmable gate array (FPGA) implementation of an integrated anomaly detection system which is energy efficient, parallel, integrated on the same chip as the other processing hardware, and customised to achieve high performance.

- The first reported single-chip RF physical layer NN-based anomaly detector which is fully pipelined and can produce an output every cycle.
- To the best of our knowledge, we achieve the highest reported performance to date, supporting continuous 200 MS/s complex inputs with latencies of 105 ns (time-domain) and 185 ns (frequency domain), at least 4 orders of magnitude lower than the processing time in GNU radio [1].
- A heterogeneous architecture which supports updating of weights while inference proceeds, enabling simultaneous learning and inference with the former conducted on a processor or graphics processing unit (GPU).

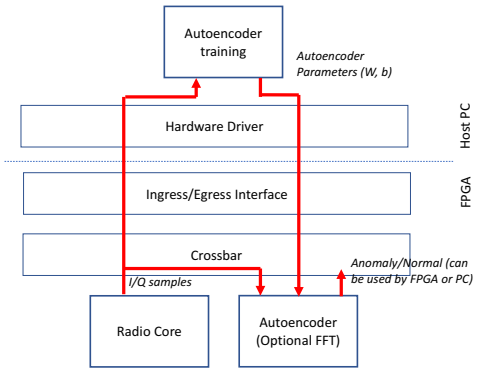


Fig. 1: Block diagram of system implementation. The Autoencoder module accepts the raw IQ data from the radio core.

- An open source design supporting reproducible research. The implementation and scripts to generate all results are available from https://github.com/djmmoss/autoencoder

While this paper focuses on anomaly detection on RF signals, it is relevant to the general problem of real-time neural network processing and can be generalized to other applications domains and NN architectures. Additionally, this anomaly detector could perform initial analysis of the signal, passing its prediction onto a much larger and more computationally expensive algorithm.

II. Background

There has been considerable recent interest in utilising advances in NN technology for RF applications. The most relevant to the present paper is by O'Shea et. al [2], who applied a number of novelty detectors to RF signals. Starting with Frequency Modulation (FM), Global System for Mobile Communications (GSM), industrial, scientific, and medical radio (ISM), and long-term evolution (LTE) band data; Short-time Broadband Bursts, Brief Periods of Signal Non-Linear Compression, Pulsed QPSK Signals and Pulsed Chirp Events were introduced as anomalies. A comparison of a number of algorithms: a 3rd order Unscented Kalman Filter/Predictor, dense neural network, long short-term memory and Dilated Convolutional Neural Network was conducted and it was observed that in most cases, the NN approaches outperform the

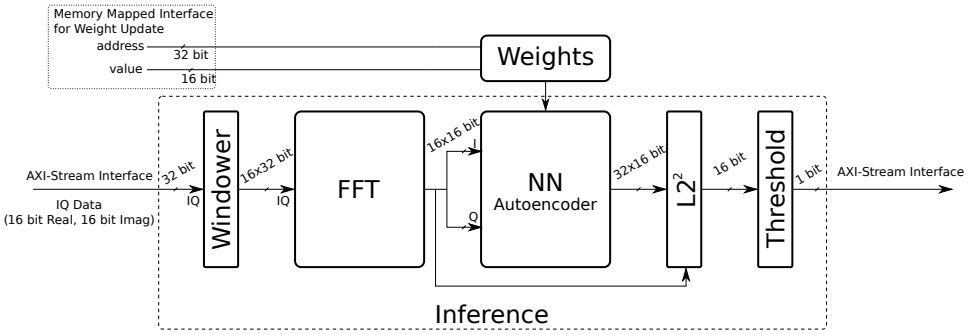


Fig. 2: Block diagram of the anomaly detector FPGA implementation.

Kalman-based approach. This work did not address the issue of real-time implementation. Moss et. al [3] presented an O(1) time-complexity spectral anomaly detector that can handle uniform or irregularly sampled data that achieved throughput of 40 ns and latency of 68 ns. This employed an incremental algorithm to compute the power spectral density and a much simpler anomaly detector operating on the distribution of quantised signal levels. Latency in GNU radio/Universal Software Radio Peripheral platform (USRP - they measured USRP1 and USRP2 whereas our device is a newer USRP3) was thoroughly analyzed by Trong et. al [1]. They concluded that the time for processing at the host computer dominates the communication bus latency and measured values well in excess of 1 ms.

In the past, FPGA devices did not have sufficient capacity to implement entire neural networks on-chip, and single-chip radio and machine learning applications were intractable. However, in recent years, high-performance FPGA implementations of neural networks for inference have been reported. An illustrative example by Zhang et. al in 2015, achieved 62 GFLOPS in single precision floating-point [4], using a roofline model to balance computational resources and memory bandwidth. Very low precision implementations have also been reported. For example, binarized (1-bit) implementations of neural networks can achieve 12.3 million image classifications per second with $0.31\mu s$ latency on the MNIST dataset with 95.8% accuracy [5]. By reducing precision, it is possible to keep all weights on-chip, allowing higher performance with lower energy consumption. In a manner similar to the present design, both implementations used high level synthesis from a C description.

Our paper extends the previous work reviewed by demonstrating, for the first time, an ultra-low latency, single-chip anomaly detector for processing of physical layer RF data.

III. Implementation

A. System Architecture

Fig. 1 illustrates the system level implementation of our design. The host PC is responsible for training, i.e. computing new parameters from the raw inphase/quadrature (I/Q) samples. Initial and/or updated biases and weights thus computed are downloaded to the FPGA which implements the

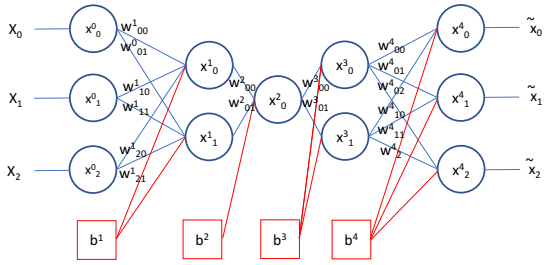


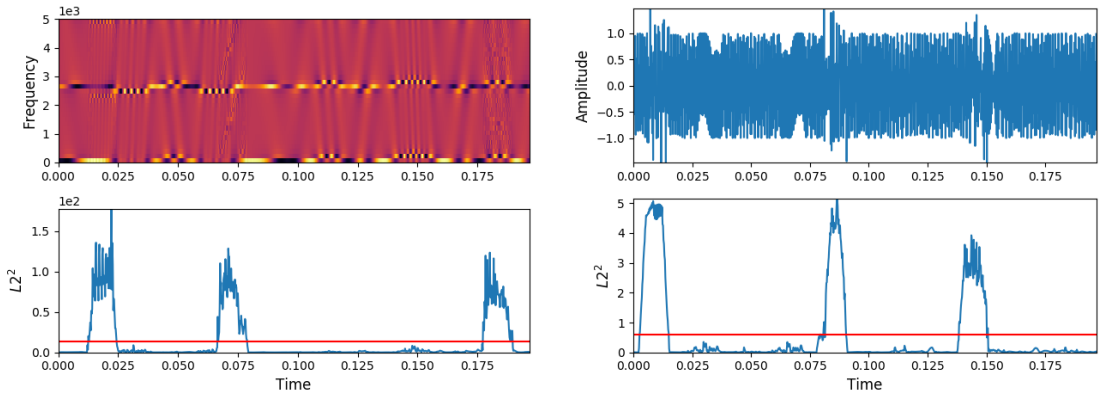
Fig. 3: Simplified 4-layer (3,2,1,2,3) autoencoder signal flow diagram.

autoencoder, operating independently of the host PC. The hardware autoencoder operates at the baseband sampling rate, maximising the probability of detecting transient anomalies, and ensuring the system is able to react as soon as possible. Training on the PC allows the anomaly detector to adapt to changing environmental conditions, while avoiding the problem of training on the FPGA, which impacts performance.

Fig. 2 shows a block diagram of our fully pipelined spectral anomaly detection architecture that supports 200 MS/s input and output. This first stage of this design is the Windower, which accepts complex I/Q samples from the source and uses a shift register to produce a sliding window of past inputs to the next stage. This is followed by an optional FFT stage, enabling anomaly detection in either the time or frequency domain. The FFT is implemented using a 5-stage radix-2 based algorithm derived from reference [6]. It was not possible to use the FPGA vendor's cores for this as they are not capable of accepting an entire input vector per cycle. The window is then passed to an autoencoder [7], which is trained to reconstruct an input window after performing dimensionality reduction. If anomalies occur, the autoencoder will be unable to reconstruct the input. Therefore, comparing $L2^2$ to a threshold yields a binary anomaly/normal result.

B. Autoencoder

An example autoencoder is illustrated in Fig. 3 for a (3,2,1,2,3) 4-layer network. The inputs are given by the vector $x \in \mathbb{R}^n$, and the outputs are $\hat{x} \in \mathbb{R}^n$, where n is the length of the input vector. It is trained on the identity mapping, enabling unsupervised training. Each layer implements the



(a) Frequency domain anomaly detection example.

(b) Time domain anomaly detection example.

Fig. 4: Anomaly detection example. The top panel displays the signal in the (a) frequency or (b) time domain. The 3 different types of anomaly can clearly be seen from left to right: pulsed sinusoid, chirp and Gaussian. The bottom panel shows $L2^2$ and the anomaly threshold.

vector function $F^l : \mathbb{R}^{m_l} \rightarrow \mathbb{R}^{m_{l+1}}$ (m_l is the number of inputs of the layer):

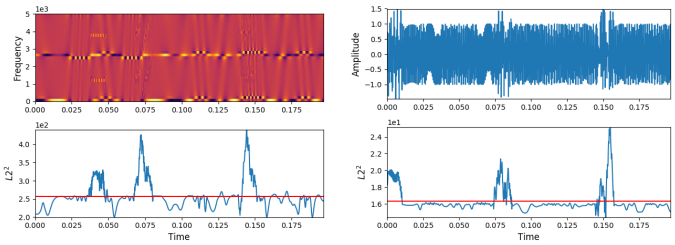
$$x^{l+1} = \sigma^l(W^l x^l + b^l) \tag{1}$$

where L is the number of layers, $0 \leq l \leq L$ the layer number, $W^l \in \mathbb{R}^{m_{l+1} \times m_l}$ the weight matrix, $b^l \in \mathbb{R}^m_{l+1}$ the bias and σ is the activation function (the identity function for the 0'th layer and the element-wise rectified linear unit ($relu(x) = \max(0, x)$) otherwise).

Dimensionality reduction is performed since the number of neurons in the middle layers are fewer than the input/output, e.g. in Fig. 3 the first 2 layers act as an encoder which compresses the input to 1-dimension. Similarly, the final 2 layers act as a decoder by taking the compressed representation and reproducing the output. A squared L_2 error norm similarity measure

$$L2^2(x, \hat{x}) = \sum_{i=0}^{n} (x_i - \hat{x_i})^2$$

(subscripts denote a vector element) is used to evaluate reconstruction error. During training, the mean reconstruction error over all patterns is minimised using standard backpropagation [7].



(a) Frequency Domain Detection

(b) Time Domain Detection

Fig. 5: FM Anomaly Detection: Sample using the Ettus X310

C. Precision and Interface

All processing is done using 16-bit fixed point numbers with truncated rounding and saturating arithmetic [8]. This maps efficiently to the FPGA device's digital signal processing (DSP) blocks, which include dedicated 18-bit multiply-accumulate blocks. The FFT block operates with complex inputs and outputs, while the autoencoder uses real values. This relatively high precision for inference ensures that our fixed-point results will achieve similar accuracy to floating-point [9]. The real and imaginary outputs of the FFT are concatenated to form the inputs of the autoencoder.

Our implementation allows the NN weights and biases, as well as the threshold value, to be updated during operation via a memory mapped register interface. This allows the FPGA to continuously perform inference, with continuous training at a lower speed on a microprocessor. Periodic updates of the weights and biases are made to the FPGA.

The entire implementation is generated from a Python description using standard module generation techniques to produce a synthesisable C output. Google's TensorFlow package is used on the host machine for training.

IV. Results

Our implementation is configurable so that arbitrary sized FFT and NNs can be implemented. In this section, we study a 16-point complex FFT and a 32-input, 4-layer network (32,16,8,16,32). The radio platform was an Ettus X310 software defined radio, which supports DC-6 GHz operation with up to 160 MHz of baseband bandwidth; PCIe, dual 10 GigE, and dual 1 GigE interfaces; and utilises a Xilinx Kintex-7 xc7k410tffg900-2 FPGA. The design was synthesised from C to register transfer language (RTL) using the Xilinx Vivado HLS tool [8] and a bitstream generated using the Xilinx Vivado 2015.4 Design Suite. Verification of the hardware implementation was completed at three levels: C simulation, register transfer level simulation and testing on the Ettus X310.

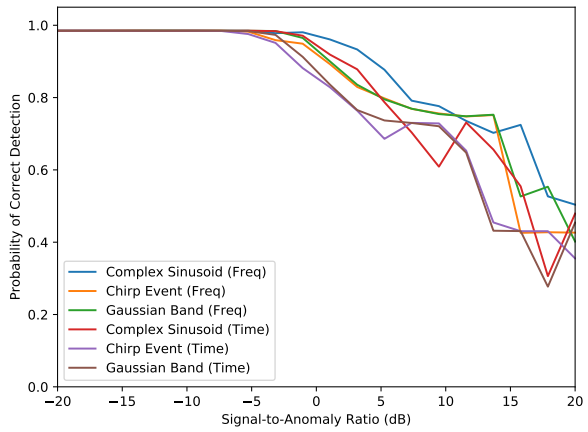


Fig. 6: Probability of detection. Multiple signals with anomalies of varying amplitude were generated for the frequency and time domains. The figure displays the average probability of making a correct detection.

TABLE I: Raw anomaly detection performance.

Operation	Throughput	Latency
Inference(FFT+NN)	5ns	185ns
Inference(NN)	5ns	105ns
Weight Update	1290ns	1285ns

TABLE II: Breakdown of autoencoder performance and resource utilisation

Module	II	Latency (cycles)	BRAM	DSP	FF	LUT
Windower	1	0	0	0	1511	996
FFT	1	8	0	48	4698	2796
NN	1	17	4	1280	213436	13044
$L2^2$	1	4	0	32	1482	873
Thres	1	0	0	0	3	21
Weight Update	258	257	0	0	21955	4528
Inference (FFT+NN)	1	37	1068	1360	241522	45448
Inference (NN)	1	29	1068	1312	236824	42652
Total	N/A	N/A	1068	1360	263477	49976
Total Util.	N/A	N/A	67%	88%	51%	19%

Most BRAMs are contained within the top-level, 'Inference', module

A. Detector Performance

A synthetic signal was generated by modulating a sine wave with a randomly modulated tone and an FM signal was recorded using the Ettus X310 SRD. Three types of anomalies, similar to those in [2], were introduced across each signal. The anomalies have a given amplitude, A, and exist over time window $t \in [t_s, t_e)$, where t_s and t_e are the start and end time:

- Period of Gaussian Noise across the entire bandwidth: Modeled as $n(t) = Gaussian(-A, A)$.
- Pulsed Complex Sinusoid: $n(t) = A * exp(2\pi t F_n)$ where $F_n = Uniform(-F_s/2, F_s/2)/F_s$.
- Pulsed Chirp Event: $n(t) = A * exp(2\pi t F_n)$ where F_n ranges linearly from F_{c1} to F_{c2}. Both F_{c1} and F_{c2} are sampled from $Uniform(-F_s/2, F_s/2)/F_s$.

Fig. 4a and Fig. 4b respectively demonstrate successful detection of different anomalies in the frequency and time domains respectively, the top panel being the signal and the bottom the similarity value from the anomaly detector. In both cases we used $1.1\times$ the running average of $L2^2$ as the threshold value, as illustrated in the figures. This performs very well and allows all anomalous events to be detected. Fig. 5a and Fig. 5b illustrate the anomaly detector operation on real FM signals. The performance is similar to the synthetic case, resulting in correct identification of the anomalies.

In a practical application, the choice of threshold should be guided by the value of $L2^2$, and an analysis over different signal-to-anomaly ratios (SAR), similar to Fig. 6, should be performed.

Fig. 6 shows probability of correctly detecting an anomaly at different SARs. In this particular case, windows were identified as anomalous if $L2^2$ is 10% over its running average. The SAR is calculated using the power, $P = \frac{1}{N}\sum_{i=0}^{N}(x_i^2)$, and (2).

$$SAR = 10\log_{10}(\frac{P_{Signal}}{P_{Anomaly}}) \quad (2)$$

As the power of the anomaly becomes proportionally smaller to the signal power, the probability of detecting it also decreases. As illustrated in Fig. 6, Detection in the frequency domain is slightly more resilient to decreasing anomaly power, however the impact is only marginal and may be overcome with different thresholding schemes.

B. Hardware Performance

The results presented in Tab. II show the latency, initiation interval (II) and resource utilisation of our design. Importantly, all modules to perform inference (Windower, FFT, NN, $L2^2$ and Thres) have an II of 1, meaning it is fully pipelined, and a total latency of only 37 cycles. Since the implementation operates at 200 MHz, the throughput and latency are 200 (complex) MS/s and 185 ns respectively. As both the FFT and NN require multiply-accumulate operations, the DSP resources constrain the parallelism of our design. The implementation is configurable so that arbitrary sized FFT and NNs can be implemented. On-chip block rams (BRAM) are the next resource constraint, this being because all weights and biases are stored on-chip. This could be addressed by using off-chip memory, at the cost of greatly increased latency as it would become the bottleneck in our design.

V. Conclusion

In this paper we demonstrated the feasibility of single-chip, 200 MHz sample-at-a-time anomaly detection, resulting in high throughput and ultra-low latency. This paves the way for the inclusion of real-time neural networks in sophisticated software defined radio systems, with potential applications in fault diagnosis, spectrum enforcement and collaborative spectrum sharing. Future work will involve improving capacity, careful comparison of accuracy with other techniques, and exploring the utility of simultaneous learning and prediction.

REFERENCES

[1] N. B. Truong, Y. J. Suh, and C. Yu, "Latency analysis in gnu radio/usrp-based software radio platforms," in *MILCOM 2013 - 2013 IEEE Military Communications Conference*, Nov 2013, pp. 305–310.

[2] T. J. O'Shea, T. C. Clancy, and R. W. McGwier, "Recurrent neural radio anomaly detection," *CoRR*, vol. abs/1611.00301, 2016. [Online]. Available: http://arxiv.org/abs/1611.00301

[3] D. J. Moss, Z. Zhang, N. J. Fraser, and P. H. Leong, "An FPGA-based spectral anomaly detection system (with errata)," in *FPT '14*, 2014, pp. 175–182. [Online]. Available: http://ieeexplore.ieee.org/document/7082772/

[4] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '15. New York, NY, USA: ACM, 2015, pp. 161–170. [Online]. Available: http://doi.acm.org/10.1145/2684746.2689060

[5] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. H. W. Leong, M. Jahre, and K. A. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," *CoRR*, vol. abs/1612.07119, 2016. [Online]. Available: http://arxiv.org/abs/1612.07119

[6] Freescale, "Complex fixed-point fast fourier transform optimization for altivec," vol. Applications Note AN2114, 2015. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN2114.pdf

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[8] X. Inc., "Vivado design suit user guide, high-level synthesis," vol. UG902 (v2015.4), 2016. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2016_1/ug902-vivado-high-level-synthesis.pdf

[9] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011.