An Experimental Analysis of the Opportunities to Use Field Programmable Gate Array Multiprocessors for On-board Satellite Deep Learning Classification of Spectroscopic Observations from Future ESA Space Missions

Ioannis Kalomoiris [‡,*], George Pitsis [*], Grigorios Tsagkatakis, Aggelos Ioannou, Christos Kozanitis, Apostolos Dollas [*], Panagiotis Tsakalides [‡], Manolis GH Katevenis [†]

Institute of Comp. Science (ICS), Foundation for Research and Technology (FORTH), Irakleio, Greece

Abstract

Satellite-to-earth data transmissions are increasingly becoming a bottleneck, as transmission speed improvements do not keep up with the pace of on-board data generation. Hence, on-board satellite payload data processing becomes essential, provided such processing can be performed with a sufficiently small energy footprint. In this work we demonstrate that with appropriate pruning of weights, suitable data structures to reduce off-chip memory requirements, and a highly parallel application-specific architecture, Field Programmable Gate Array (FPGA) technology can be used for on-board satellite processing of observation by Convolutional Neural Network (CNN) architectures, and at an order-of-magnitude smaller energy requirements compared to Graphics Processing Units (GPUs) running the same algorithms. We demonstrate a 0.4% error vs. results from Tensorflow running on GPUs towards estimation of the galaxy redshift from spectroscopic observations. The results are from actual executions on FPGAs which have space-qualified equivalent parts. The main contribution of this work is the demonstration that accurate observation analysis task can be performed in space, so that only critical information is transmitted to ground stations instead of raw data.

1 Introduction

The exponential growth of data during the last years is an undeniable fact, leading to the need for proper management. As anticipated, in the 21st century global data are cracking the zettabyte barrier. Extracting and analyzing these amounts of information is difficult or even impossible using conventional software tools and technologies. The rapid explosion of digital data brings big opportunities for innovative methods and creates the field to explore ways to extract a high-level understanding of the low-level information given by raw data such as images, video and speech sequences. Among the proposed methods, Convolutional Neural Networks (CNNs) [4] have become the driving force by achieving accuracy even better than humans in many applications related to machine vision (e.g detection [1], classification [5], segmentation [10]) and speech recognition [6]). Satellite based spectroscopic imaging sensors produce massive volumes of data at very high rates. The automated analysis of such observations via machine learning algorithms, e.g. for the precise estimation of the redshift associated with individual galaxies, requires massive floating point operations, generally performed on Graphics Processor Units (GPU). This form of processing is fast but

[*]also at the School of ECE, Technical University of Crete, Chania, Greece
[†]also at the Dept. of Computer Science, University of Crete, Irakleio, Greece

requires substantial energy for the computation, and thus necessitates the transmission of the acquired measurements to ground stations for processing in large data centers. An alternative technology to GPUs is Field Programmable Gate Arrays (FPGAs), which often require substantially less energy per computation compared to GPUs, but are considered too slow for state-of-the-art machine learning methods like CNN based inference. In this work, through the collaboration of two EU Horizon 2020 projects, namely EuroEXA and DEDALE, we demonstrate experimentally that using (i) appropriate data structures to reduce memory bandwidth, (ii) compressed fixed point indices to clustered floating point weights and (iii) massive pipelining, FPGA-based computing can yield extremely high (in the order of 99%) classification accuracy vs. GPUs in the context of top-one classification, at an order-of-magnitude less energy. For this work we considered optimized Tensorflow codes running on various GPUs vs. our proposed FPGA-based architecture for galaxy redshift estimation from extended wavelength range spectroscopic measurements using simulated measurements which are in-line with the publicly available specification of the upcoming ESA Euclid space telescope mission [3]. We demonstrate our results on actual runs in hardware developed within the ExaNeSt project and deployed by EuroEXA, namely the Quad FPGA Daughter Board (QFDB). It offers substantially lower latency vs. a similar-technology Nvidia P1000 GPU for batches of any size, better throughput for batches up to 30 spectral profiles (which can scale out to any batch size), and also roughly an order of magnitude better energy consumption for the same computations. So it is becoming an interesting technology for on-board satellite deep learning classification tasks. An important aspect of this work is that the FPGA model used in this work has an equivalent rad-hard part qualified for space applications.

1.1 Scientific Contributions

The scientific contribution of this work is focused on two aspects: (i) the demonstration that substantial weight pruning and clustering can result in significantly smaller memory transfers, and (ii) the presentation of an appropriate FPGA-based architecture which is highly competitive to GPUs in latency, throughput, and energy requirements. We consider the problem of spectroscopic redshift estimation and employ a 1D multilayer CNN architecture for the estimation of spectroscopic redshift by dividing the redshift range [1,1,8) into 800 equally spaced classes, as described in [9]. The proposed CNN network is parameterized by 22,776,272 (64-bit, double precision floating point) weights, meaning 173.77 MB, thus it is essential to reduce its size efficiently to accelerate the network.

A pipeline was originally created at each layer separately and then it was expanded between the layers. In order to achieve this, we have to transform the order in which the layers export their results so that the next layers are able to start their process before the previous layers complete their own. Another challenge was to limit I/O transactions which are the main bottleneck in every FPGA implementation. We investigated two architectures for a single FPGA (ZCU-102) and the quad FPGA (QFDB). The implemented accelerator was able to achieve 2.5x speedup and 10x energy efficiency over GPU NVIDIA-Quadro-K2200. Both speedup and energy efficiency play an important role when targeting on-board satellite applications. An investigation of different network parameters in system performance is reported in [8].

2 Reduction of Memory Footprint and Requirements

After a Robustness analysis implemented on the structure of the CNN we present the three main techniques and their resulting effects in the CNN processing. A more detailed description of the methodology is available in [7]. These are:

- Clustering of weights
- Hierarchical structure of clusters
- Inverse density normalization

All of our models have been evaluated with MATLAB and the results were compared with the TensorFlow "golden standard". The CNN was designed and trained using the TensorFlow toolbox, while the FPGAs

were tasked with the inference procedure, where spectra were fed into the network in order to predict the associated target classes.

In FPGA designs memory-related constraints such as memory-bandwidth, on-chip Block RAM (B-RAM) size, memory requirements, etc., are the performance bottleneck of most computational applications. Especially for implementing CNN in FPGA-based systems, memory is a performance inhibitor, since most of the time, the challenge is to fit the model into internal B-RAMs. Since these types of networks require typically from tens to hundreds of MB (173.7 MB in our case) it is obvious that low-MB B-RAMs are not a solution (4MB in our case). Of course modern FPGAs support D-RAMs but its limited memory-bandwidth is their main disadvantage. Thus, it is important to understand how weights are distributed in the network stages and to find ways to reduce the overall memory footprint. Table 1 presents the memory footprint of the weights and the stages of the signal using double floating point.

Table 1: Weights Memory Footprint

Layer	#Weights	Footprint
conv1	144	1.1KB
conv2	2,064	16.1KB
conv3	2,064	16.1KB
dense	22,771,200	173.7MB

Using Fixed Point (static or dynamic), we observe that there is indeed a significant improvement in memory footprint but as we reduce bit-width, the accuracy of the network decreases. So, we opted for a hybrid solution, which provides floating point format for the kernels (using double or single precision) with much less memory footprint. The idea is to group weights according to their values in k centroids and store their values in a codebook as shown in Figure 1. Thus, instead of storing in off-chip memory the value of each weight, only the index of the corresponding centroid in a shared code-book is stored. Using this quantization, given k centroids, we only need $log_2 k$ bits to encode the index. There are several algorithms to cluster these centroids such as Lloyds or K-means. These optimizations in our network concern the dense layer only because this is the main memory bottleneck.

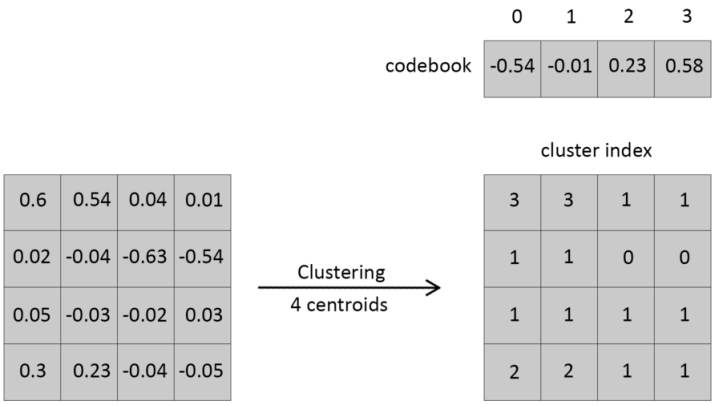

Figure 1: Clustering : A sample of Clustering 4.

An ideal case would be to use Lloyds quantization with 16 centroids (i.e. 4-bit). We developed some techniques so that we can drop the error to lower levels. A problem that arises from a straightforward application of the Lloyds clustering, is that Lloyds is trying to group weights without understanding their differential importance. For example, weight with value 0.69 is much more important for the network than

weight with value 0.0023. Larger weights play a more important role vs. smaller weights [2], but their density is inversely proportional to that of smaller weights.

Inverse Density: Our next step is to add normalization to the clustering algorithm by providing an initial codebook. By knowing that density is inversely proportional to the importance of weights, we propose to initialize the codebook, starting from the minimum value and ending up to the maximum, trying to have a high resolution at the values with large absolute magnitude and as we approach small values to reduce resolution linearly.

Hierarchical Clustering: In addition, we address the same problem from another point of view. As long as we use a larger number of centroids, we increase the resolution across all values (large and small). Thus, we force the algorithm to pay more attention to high values. Then if we apply the clustering algorithm hierarchically we will come up with a better resolution at weights that are of greater importance to us. Table 2 and Figure 2 present the impact of compression on error rates which we observe after the use of our techniques (Hierarchical Clustering and Inverse Density Normalization). Thus we end up with a minor 0.6 % error.

Table 2: Final Compression Methods

Method (#Centroids)	Bit_{width}	Error rate (%)	Compression
Clust. (256)	8	0.03	8x
Clust. (128)	7	0.09	9.1x
Clust. (64)	6	0.16	10.7x
Clust. (32)	5	0.26	12.8x
Clust. (16)	4	1.37	16x
H.Clust.& Norm. (16)	**4**	**0.6**	**16x**
Clust. (8)	3	4.6	21.3x

Clust. = Clustering,
H. Clust. = Hierarchical Clustering,
Norm. = Inverse Density Normalization

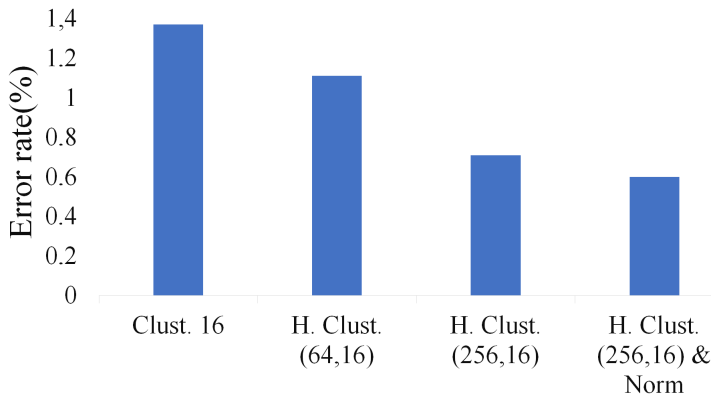

Figure 2: Clustering 16: Error rate for Clustering methods.
Clust. = Clustering , H. Clust. = Hierarchical Clustering

3 FPGA Architecture

The CNN-based inference Hardware Accelerator was implemented using the Xilinx Vivado Design Suite - HL System Edition 2017.1. The tools used are the Vivado HLS, Vivado IDE, and Xilinx SDK. Vivado HLS also provides (optional) directives that can be used to optimize the design: reduce latency, improve throughput, and reduce area and device resource utilization of the resulting RTL code such as Pipeline, Array Partition and Dataflow.

Our architectures targeted two FPGA platforms, the **Xilinx ZCU102** and the **QFDB**. The Quad-FPGA Daughter-Board (QFDB) designed as part of the EU-ExaNeSt project in order to offer both high compute density and high flexibility. It contains four Xilinx Zynq Ultrascale+ FPGAs (model: ZCU9EG).

3.1 System Architecture

In the system architecture we integrated the reduced memory footprint of the weights via sophisticated clustering schemes as shown in the previous sections, with a huge pipeline comprising of all the signals entering the system as well as the communication between the convolutional layers and then with the fully connected layer. Subsequent resource optimizations were performed in order to fit the network into the FPGA.

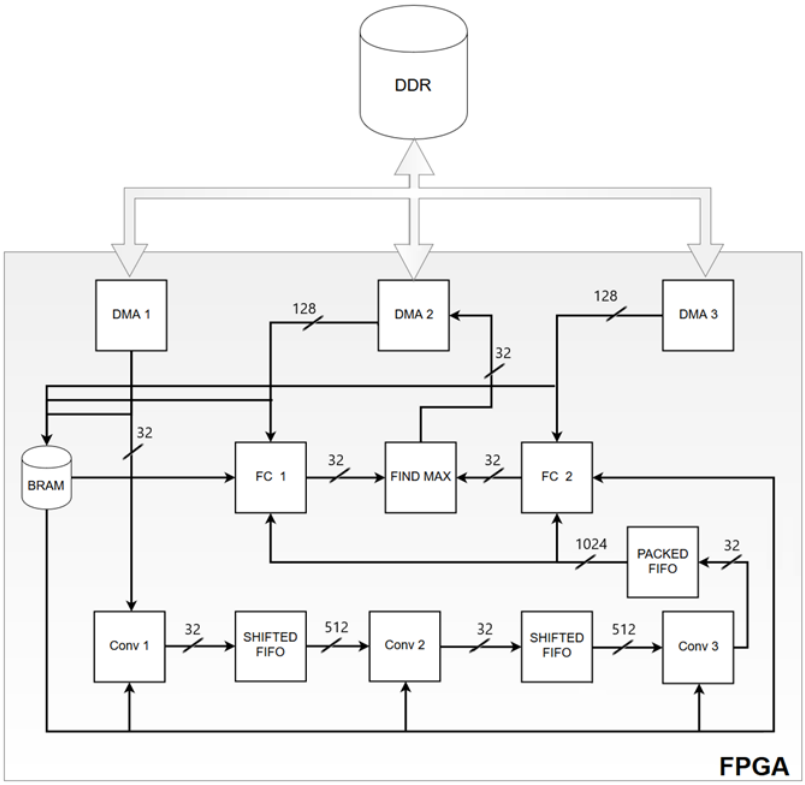

Figure 3: Datapath of the final Architecture

3.1.1 Embedding Compressed Weights

Originally, the analysis was done to reduce the memory footprint of the weights of the fully connected layer, because this is the main memory bottleneck of the algorithm. Furthermore, compressed weights are used in the I/O streaming interface during the processing. We used 256-bit channel from the memory (2-DMA of

128 bits) based on the previous analysis on memory buses. Each compressed weight has a 4-bit precision. Therefore we can stream 64 weights in one cycle (stream read). This gives us a possibility for a huge parallelism at the operation level.

3.1.2 Pipelining Convolutional Layers

The next step is to try to get the most out of all available resources. To accomplish this, a pipeline must be created between the layers of the network (Convolutional and Fully Connected) as shown in Figure 4, in such a way that different parts of the input signal are processed at the same time by the 4 entities.

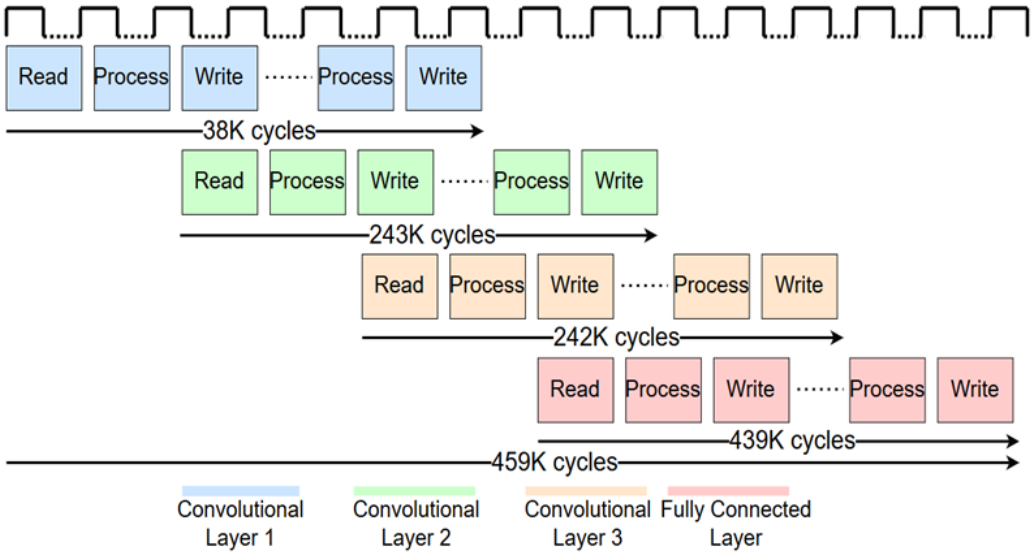

Figure 4: Convolutional Layers Time-Chart of the final Architecture

3.1.3 Resource Optimizations

It is important to ensure that we use efficiently the resources at our disposal. After the successful completion of the first architecture, we introduced another level of parallelism, i.e., the use of batching. Instead of computing the results for a single input signal, we process data for two input signals in parallel. A brute-force approach would be to insert another instance of the already existing accelerator to implement Batch 2. This would, however, lead to a doubling of resources. A better approach is to integrate batch 2 into a single architecture in HLS, which avoids duplication of resources and is easier for the HLS tool to route.

3.1.4 Batching

Given that the vast amount of I/O comprises of the weights, the overhead to process multiple datasets (with the same weights) is minimal, as long as the FPGAs have resources for the processing. It turns out that the FPGAs do have the resources for two such datasets, leading to the "Batch 2" architecture, meaning to run the same algorithm (same weighs, same connectivity) on two datasets. The I/O is not increased substantially - it grows only by 0.0001%. The reason why we can not proceed to larger batches is due to resource restrictions. Table 3 shows the relative performance of the final architecture, for batches of size 1 and 2. We note that the complete input-to-output performance for both batch sizes is lower than the sum of the parts, which is expected given that the pipeline is very deep and it takes some time to fill up, however, the performance for batch 2 is twice that of batch 1 because the I/O overhead is minimal, and the two compute engines run in parallel.

Table 3: Final Architecture Performance

Modules	Latency (cycles)	Comp. Performance (GFLOPS/s)	Bandwidth (GB/s)
conv (b=1)	263K	14.3	1
dense (b=1)	439K	25.8	8.23
conv+dense (b=1)	459K	33.1	9.23
conv (b=2)	264K	28.7	1
dense (b=2)	441K	51.6	8.23
conv+dense (b=2)	459K	66.1	9.23

GB=Gbytes, b = batch

4 Results

In this section, we present the results of the proposed framework, as summarized in Tables 4 and 5. These results were obtained from the final architecture, ported to both the ZCU-102 and the QFDB platforms, designed to make the most out of our resources. Comparisons were made with the GPU platform NVIDIA Quadro K2200, on latency, throughput, power, and energy for 10K input spectral signals, as shown in Table 5. The energy efficiency, i.e. spectrum/joule is quite noteworthy for a technology with the prospect of getting spaceborne.

Given that GPU technologies as well as FPGA technologies change, it is well worth mentioning why the comparison was made against the specific model. The reasons are two: (a) that it is of a corresponding technology as the FPGAs we used, and (b) that it has a space-qualified version - the specific model is designed to be low-power and suitable for aerospace applications. Thus we compare a recent generation with a space qualified FPGA part vs. a recent generation with a space qualified GPU. More recent GPUs may have a 4X energy efficiency improvement vs. the K2200, however, there are newer FPGA platforms as well, as we expect that the trend will continue to favor the FPGA platform in terms of energy efficiency.

Table 4: Architecture comparison with GPU

	ZCU-102	QFDB	K2200
Clock Frequency(MHz)	250	250	1124
Throughput(Signals/s)	1084	4334	2000
Latency(s)	0.003	0.003	0.06
GFLOPS	66.1	265	122.5
Total On-chip Power(Watt)	11.8	47.3	300
Energy Consumption(Joule)	108.8	109.1	1.5K
Signals/Joule	91.6	91.6	6.66

Table 5: Speedup and Efficiency over GPUs

	ZCU vs K2200	QFDB vs K2200
Latency speedup	20x	20x
Throughput speedup	0.55x	2.17x
Power Efficiency	22x	1.83x
Energy Efficiency	11.9x	11.9x

5 Conclusions

In recent years Convolutional Neural Networks (CNNs) have enjoyed extreme growth due to their effectiveness in complex signal analysis problems. The purpose of this study is to accelerate a specific-CNN for space-borne observation analysis using Reconfigurable Logic (FPGAs). After carrying out an extensive Robustness Analysis, computational workloads and memory accesses were analyzed, compression methods and algorithmic optimizations were investigated towards exploiting FPGA parallelism. At the CNN network level, optimizations of the convolutional and fully connected layers were presented and compared, while approximate computing optimization methods were examined in order to minimize the a potential decrease of the Network's accuracy. Two platforms, the ZCU102 and QFDB (a custom 4-FPGA platform developed at FORTH) were considered. The implemented accelerator was able to achieve 20x latency speedup, 2.17x throughput speedup and 11.9x energy efficient vs. the GPU NVIDIA-Quadro-K2200 which has low-power consumption and is suitable for aerospace applications.

6 Future Work

Our next effort is to scale out this study to more FPGAs systems. Our top priority is to scale this work to the Mezanine that hosts four QFDBs (16-FPGAs in total) and which has been already developed at FORTH, expecting a linear speedup due to the parallelism of the application. In addition, we will be targeting a larger FPGA in order to increase the internal parallelism by adding larger batches which will also lead to an almost linear speedup. Finally, since the use-case considers astrophysics applications and is focused on the on-board signal processing, it would be important to study newer generation FPGAs that have resistance to space radiation. The FPGA's suitability for on-board processing is justified by the increase in energy efficiency and throughput compared to GPUs, and the present work has already been performed on FPGAs which have space-qualified equivalents.

7 Acknowledgements

This work was carried out with support from the EuroExa (Grant Agreement 754337) and the DEDALE (Grant Agreement 665044) projects, funded by the European Union Horizon 2020 Research and Innovation Programme. The authors would like to thank Radamanthys Stivaktakis, Theodoros Zois, and Antonis Nikitakis for their contribution.

References

[1] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[2] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[3] R. J. Laureijs, L. Duvet, I. E. Sanz, P. Gondoin, D. H. Lumb, T. Oosterbroek, and G. S. Criado. The euclid mission. In *Space Telescopes and Instrumentation 2010: Optical, Infrared, and Millimeter Wave*, volume 7731, page 77311H. International Society for Optics and Photonics, 2010.

[4] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[5] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[6] H. S. J. K. Olga Russakovsky, Jia Deng. Subject independent facial expression recognition with robust face detection using a convolutional neural network, 2015.

[7] G. Pitsis. Design and implementation of an fpga-based, 2018. URL `http://dias.library.tuc.gr/view/manf/79094`.

[8] G. Pitsis, G. Tsagkatakis, C. Kozanitis, I. Kalomoiris, A. Ioannou, A. Dollas, M. Katevenis, and P. Tsakalides. Efficient convolutional neural network weight compression for space data classification on multi-fpga platforms. In *Acoustics, Speech and Signal Processing (ICASSP), 2019 IEEE International Conference on*. IEEE, 2019.

[9] R. Stivaktakis, G. Tsagkatakis, B. Moraes, F. Abdalla, J.-L. Starck, and P. Tsakalides. Convolutional neural networks for spectroscopic redshift estimation on euclid data. *arXiv preprint arXiv:1809.09622*, 2018.

[10] P. B. Ying Zhang, Mohammad Pezeshki. Towards end-to-end speech recognition with deep convolutional neural networks., 2017.