ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/339229389

Novel Implementation and Benchmarking of AlexNet for Edge AI: From CPU and GPU to FPGA

Preprint · April 2020

CITATIONS
0

READS
858

5 authors, including:

Firas Al-Ali
Manukau Institute of Technology
8 PUBLICATIONS **20** CITATIONS
SEE PROFILE

Thilina Prasanga Doremure Gamage
Manukau Institute of Technology
4 PUBLICATIONS **9** CITATIONS
SEE PROFILE

Sayan Kumar Ray
Taylor's University
74 PUBLICATIONS **1,192** CITATIONS
SEE PROFILE

Hewa WTS Nanayakkara
Manukau Institute of Technology
4 PUBLICATIONS **9** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Fast Handover in Mobile WiMAX View project

Spectrum Auctions for Spectrum Sharing View project

All content following this page was uploaded by Firas Al-Ali on 16 July 2020.

The user has requested enhancement of the downloaded file.

Novel Casestudy and Benchmarking of AlexNet for Edge AI: From CPU and GPU to FPGA

Firas Al-Ali
Manukau Institute of Technology
Auckland, New Zealand
Firas.Al-Ali@manukau.ac.nz

Thilina Doremure Gamage
Manukau Institute of Technology
Auckland, New Zealand
Dore16@Manukaumail.com

Hewa WTS Nanayakkara
Manukau Institute of Technology
Auckland, New Zealand
Nana87@Manukaumail.com

Farhad Mehdipour
Otago Polytechnic, AIC
Auckland, New Zealand
Farhad.Mehdipour@OP.ac.nz

Sayan Kumar Ray
Manukau Institute of Technology
Auckland, New Zealand
Sayan.Ray@manukau.ac.nz

***Abstract*— Convolutional Neural Networks (CNNs) require massive parallelism due to the high-precision floating-point arithmetic operations they perform. So, demand of processing power in them is significantly higher than what a standard CPU can offer. This has traditionally made CNNs more suited for running on a Graphics Processing Unit (GPU). However, GPUs consume much more power than CPUs, rendering the former impractical for implementing CNNs in Edge AI (Artificial Intelligence), where restraining power consumption is paramount. On the other hand, FPGAs (Field Programmable Gate Arrays) are more suited for AI computing at the edge as they consume much lesser power than GPUs and even CPUs. Additionally, GPUs and CPUs are not suited for real-time AI applications, which require both high throughput and low latency at the same time and FPGAs excel at all these requirements. The purpose of this paper is to provide a study of the performance of FPGA as the most suitable platform for AI-based computing at the edge. To achieve this, we chose AlexNet, a popular CNN image classifier, for which we present a case study on four different platforms: CPU, GPU, embedded RISC core, and FPGA fabric. Then, we quantitatively measure and compare the performance in terms of inference time (time needed to classify an image) on all these platforms. Inference time using FPGA is reduced by almost 64, 1.6, and 1.1 times compared to a dual-core ARM, an i5-6400 CPU, and an Nvidia GPU, respectively.**

Keywords*— *Convolutional Neural Networks (CNN), Deep Learning (DL), Edge AI, Field Programmable Gate Array (FPGA), Graphics Processing Unit (GPU), Python Productivity for ZYNQ (PYNQ-Z1)

I. INTRODUCTION

AI is currently one of the key technologies in many application domains. Moreover, DL (Deep Learning), as a popular AI approach has drawn great attention in recent years for its efficiency in solving complex problems with reduced error margins. This is a result of the current wide availability of powerful CPUs and GPUs which make parallel processing (a requirement for DL) ever faster, cheaper, and more powerful. However, an FPGA chip is an alternative hardware platform (utilizing reconfigurable logic fabrics) used to accelerate complex AI/DL architectures such as DNNs (Deep Neural Networks) [1].

Following the emergence of Edge AI and the need for bringing computation to where the data is captured, this has become one of today's heavily-researched areas. Hence, bringing AI/DL to the edge is becoming inevitable, as the number of AI/DL-dependent applications keeps increasing [2]. As a result, this poses several challenges for real-time edge AI/DL applications. Maintaining a low power consumption while running the applications at low latency and high throughput at the same time is one of major challenges. Also, the hardware and computing platform hosting the learning models should provide sufficient flexibility in order to accommodate frequent changes to these models over time. These challenges are the main drive behind modern (and future) evolving AI/DL architectures requiring complex hardware platforms which must be highly customizable, re-programmable, and adaptive such as FPGAs [3].

The purpose of this paper is to provide a study of the performance of FPGA as the most suitable platform for edge AI. Hence, our aim for this paper is: (1) successful implementation of AlexNet (a popular large convolutional neural network) on CPU, GPU, embedded RISC core, and FPGA fabric, and (2) performance comparison and analysis of inference on these four platforms. In Section II, CNN and other types of neural networks which fall under the generic banner of ANN (Artificial Neural Network) are introduced. Section III discusses the advantages of using FPGA against GPU and CPU for implementing convolutional neural networks. Section IV describes the results of implementing AlexNet on CPU, GPU, embedded ARM and FPGA fabric. Section V concludes and outlines future work.

II. REVIEW AND BACKGROUND

A. Artificial Neural Networks (ANNs)

An Artificial Neural Network (ANN) is a computing model that mimics the structure and functionality of the human brain, which consists of billions of neurons and synapses (Fig. 1). Synapses are the connections defining the interrelations between neurons and can be mapped onto respective weights when implemented in an ANN. Similar to a human brain, a neural network is trained first, and then the trained model is used for predicting new patterns (inference). [4].

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

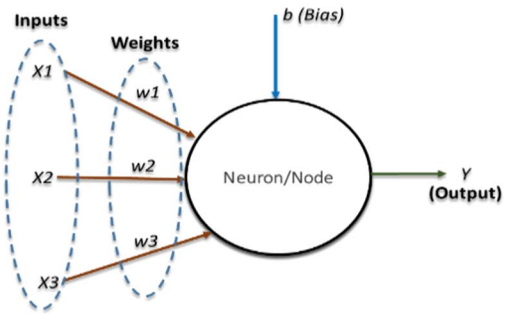

Fig. 1. Structure and Function of an Artificial Neuron (Node).

Referring to Fig. 1, *y* is the output of the neuron, and is activated by an Activation Function (AF) which determines whether the neuron is to be fired or not. For this given neuron, *x* is a vector of input variables, *w* is a weight vector for the edges and *b* is constant bias [5]. Finally, as already explained, ANN is the over-arching umbrella encompassing all types of neural networks, some of which (such as DNN, CNN) are explained next.

B. Deep Neural Networks (DNNs)

Deep Learning is a complex neural network with many hidden layers, allowing it to perform very well in order to produce highly-accurate predictions for a given set of unlabelled or unclassified data. A DNN (Fig. 2) is a popular deep learning architecture consisting of a number of hidden layers between the input and output layers, all of which are made up of neurons/nodes [6]. DNNs come in many flavours, one of which is Convolutional Neural Networks (CNNs).

C. AlexNet and other CNNs

AlexNet is the first CNN introduced in 2012, consequently paving the path for the more complex GoogleNet and DenseNet later on. CNNs are among the most popular DNNs used in machine vision (image recognition, classification and object detection) and natural language processing. AlexNet consists mainly of three types of layers; convolutional layers, pooling layers (max, average pooling layers), and fully connected layers, as shown in Fig. 3.

In a CNN, over 90% of the computations are performed in the convolution layers [7], while the Fully Connected (FC) layers require more memory in order to perform classification based on the features extracted by the previous layers.

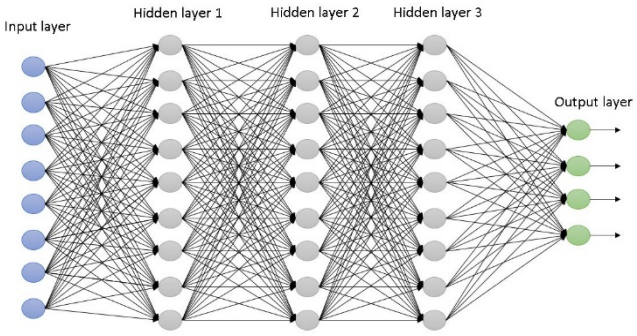

Fig. 2. Architecture of DNN. [7]

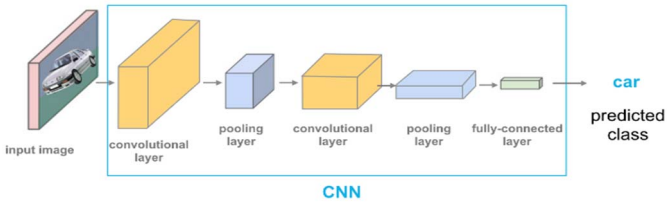

Fig. 3. Architecture of CNN. [9]

Therefore, CNNs are highly compute-intensive and require large memory resources and power. This makes GPUs the major candidate for training CNNs [5]. Thus, the main drawbacks of a CNN are high memory and power consumption as a direct result of the billions of high-precision floating point operations it has to perform. For example, AlexNet has 65 million parameters and 1.5 billion full-precision floating point operations, rendering it unsuitable (until now) for a broad spectrum of applications and embedded devices [8]. However, this is not the case anymore, as we demonstrate in this paper.

D. FPGAs and SoC FPGAs

An FPGA is a prefabricated yet reconfigurable silicon chip containing an array of programmable logic units connected by programmable interconnections to produce a highly flexible semiconductor device. FPGAs can be reconfigured (reprogrammed) post-manufacture, in order to achieve any desired digital output given the correct inputs, adequate resources, and circuitries [10]. The architecture of a basic FPGA is shown in Fig. 4.

However, recent years have witnessed the advent of the SoPC (System on Programmable Chip) a.k.a. the SoC FPGA. A SoC FPGA integrates hardcore processors with highly rich peripherals into the FPGA's programmable fabric to form highly-integrated, low power chips compared to traditional FPGAs. A complete in-depth review of FPGA architectures can be found in our previous paper [11].

III. FPGA VERSUS GPU AND CPU

A. Limitations of GPUs and CPUs for Edge AI

GPUs are time-efficient at AI training which happens offline. Although training time is not as critical as actual inferencing, GPUs remain the best platform for achieving a minimum training time. However, GPUs are power-hungry, rendering them unsuitable for AI inference at the edge where power consumption is a major limitation [12].

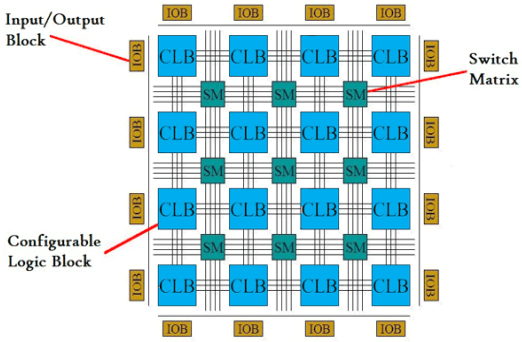

Fig. 4. Architecture of a Basic FPGA. [10]

In GPUs and CPUs, the 'data/weights' model must be loaded as sets called 'batches', then computations are performed on those batches; thereby generating multiple results together. This is due to the fact that, in CPUs and GPUs, memory is a bottleneck which limits multiple loading of data to the neural network. The intrinsic limitation of batching is that it supports either high throughput or low latency, but not both at any given time. In contrast, many edge AI/DL applications require both (high throughput and low latency) simultaneously while also maintaining a high performance per watt value [12] [3].

B. *The Solution: FPGAs for Edge AI*

In FPGAs, different inputs are pipelined through massively-parallel reconfigurable logic fabric and produce outputs concurrently leading to high bit-level parallelism. Moreover, no waiting is required for all the weights to be loaded (in contrast to GPU batches), resulting in low latency. Pipelined designs guarantee high throughput alongside the FPGA's capabilities of customizable data precisions as per the design requirements. Furthermore, with customizable memory hierarchies on the FPGA, the design can utilize faster on-chip memories. As a result, FPGAs can achieve high throughput with low latency at the same time. This makes FPGAs the most suitable hardware platform for real-time inferencing of most practical AI/DL solutions [3].

IV. Case Study: Implementing and Benchmarking AlexNet

In this section, we present implementation and benchmarking of AlexNet CNN on four different hardware platforms; CPU, GPU, embedded ARM core and FPGA fabric. In order to achieve this, we ran Nvidia-Docker on a Linux system. Nvidia-Docker enables CUDA (Compute Unified Device Architecture) containers to run on a Nvidia GPU. We were able to produce identical environments using Theano and Lasagne libraries for both our CPU and GPU (explained next) to run AlexNet inference. Additionally, these conditions were almost identical to the AlexNet inference environment which was later performed on SoC FPGA. Hence, the performance benchmarking and comparison between each of these devices can be justified.

A. *AlexNet Inference on CPU*

The CPU used in our experiments is an Intel core i5-6400 rated at 65W TDP (Thermal Design Power) and 2.7 GHz operating frequency. The inference results in Fig. 5 show an inference time of 0.539 seconds with 82% accuracy.

Fig. 5. AlexNet Inference Results on intel core i5-6400 CPU.

B. *AlexNet Inference on GPU*

The GPU used is an Nvidia GeForce GTX 960 which has a rating of 120W TDP and 1.127 GHz operating frequency. The results in Fig. 6 show an inference time of 0.358 seconds with 83% accuracy.

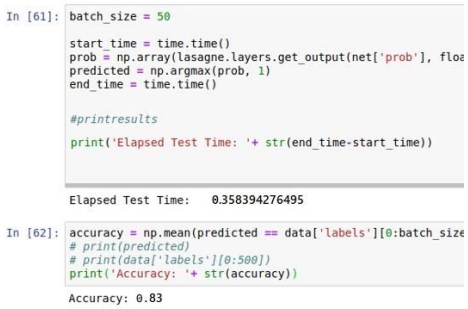

Fig. 6. AlexNet Inference Results on Nvidia GeForce GTX 960.

C. *AlexNet Inference on SoC FPGA*

1) *The PYNQ-Z1 Development Board*

The multi-processing system-on-chip FPGA that we have used is PYNQ-Z1 (Python Productivity for ZYNQ) board shown in Fig. 7. It hosts a Xilinx ZYNQ MPSoC which combines Programmable Logic (PL), referring to the FPGA fabric itself, with a Programmable System (PS), an embedded 650MHz hard dual-core ARM Cortex-A9 processor. We chose this PYNQ-Z1 board because it integrates with the Python language and other necessary programming libraries and facilitates development of ML/NN (Machine Learning/Neural Network) embedded systems based on FPGA[13] . The board generally consumes between 1 to 4 Watts depending on the application [14].

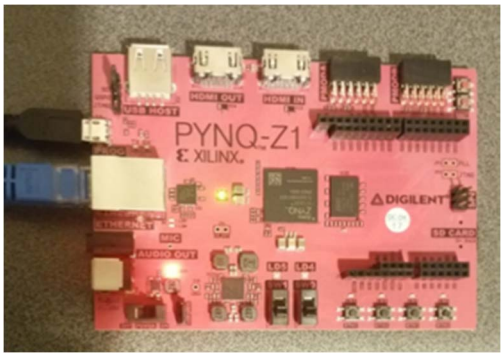

Fig. 7. Our PYNQ-Z1 Development Board.

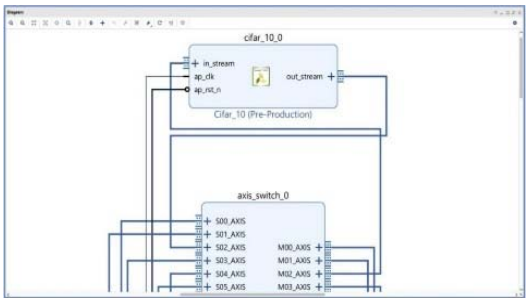

Fig. 8. PYNQ-Z1 customized overlay with AlexNet.

Xilinx's Vivado Design Suite was used to design and configure the PS (embedded ARM) and PL (FPGA fabric) portions of the ZYNQ MPSoC and its overlays. Overlays are used to exchange data between the hardware logic and software through a Python API (Application Programming Interface). This is what makes the PYNQ board more suitable for ML and for natural language processing where Python is often the programming language of choice [13].

2) Redesigning the PYNQ-Z1 Overlay

We used Xilinx HLx and Vivado tools to redesign and add AlexNet to PYNQ base overlay provided by XILINX. As AlexNet is written in C language, it had to undergo a C-to-RTL (Register Transfer Level) co-simulation by using Xilinx HLx tool. This workaround saved a significant amount of development time by bypassing time-consuming hardware programming using HDL (Hardware Description Language). Then, these synthesized and connected AlexNet layers - through AXI (Advanced eXtensible Interface) stream interfaces - were exported into Xilinx Vivado and wrapped into a single IP (Intellectual Property) core, and then connected to the PYNQ overlay in block designer as shown in Fig. 8. Finally, we generated a bitstream to make a link the PS and PL through the Python API [15].

3) Running AlexNet on PYNQ-Z1 Embedded ARM (PS)

As our focus is not on training AlexNet, we imported a pre-trained AlexNet model using CIFAR-10 dataset into Caffe framework. We used Python and Python libraries NumPy, Theano and Lasagne to configure our Jupyter notebook in order to run the inference on the embedded dual ARM core. The inference results showed an inference time of 20.67 seconds with 82% accuracy.

4) Running AlexNet on PYNQ-Z1 FPGA Fabric (PL)

Next, we ran the inference on the FPGA fabric itself. The remarkable inference results show the shortest inference time of 0.324 seconds and highest accuracy of 88% when compared to the other three platforms.

D. Benchmarking and Results Analysis

Analysis of these results indicate that running AlexNet directly on the FPGA fabric reduces the inference time by almost (a) 64 times compared to the dual-core ARM (embedded in the ZYNQ SoPC), (b) 1.6 times compared to the i5-6400 CPU, and (c) 1.1 times compared to the Nvidia GPU. Furthermore, with lowest inference time recorded and with a small power consumption (note that PYNQ-Z1 can only draw power up to a maximum of 4 to 5 Watts from the USB 3 port used.) we can generally visualize FPGA's massive energy efficiency according to the formula (1);

$$Energy\ Efficiency = Workload / (Power * Inference\ Time) \quad (1)$$

Workload, in this scenario is equal to a constant batch size across all devices and can be normalized to "1".

$$Energy\ Efficiency = 1 / (Power * Inference\ Time) \quad (2)$$

V. Conclusion and Future work

Current AI inference engines require highly efficient and massively parallel hardware platforms. We have shown that MPSoC FPGA can be used for implementing AlexNet with higher speed and power efficiency compared to its GPU and CPU counterparts. Measuring power consumption more accurately for the GPU, CPU and FPGA remains as our future work.

Acknowledgment

The authors would like to thank Manukau Institute of Technology Strategic Research Fund for funding this research.

References

[1] F. Al-Ali, T. Gamage, H. Nanayakkara, F. Mehdipour and S. Ray, "Benchmarking a Machine Vision Image Classifier Implementation on FPGA Using Binarized Neural Networks," in *The 10th Annual Conference 2019*, Wellington, 2019.

[2] M. Wielgosz and M. Karwatowski, "Mapping Neural Networks to FPGA-Based IoT Devices for Ultra-Low Latency Processing," *Sensors,* vol. 19, 2019.

[3] C. Abramson, "Linaro Connect Bangkok," 3 April 2019. [Online]. Available: bkk19.sched.com/event/LNnG/bkk19-321-fpgas-for-highest-performance-inference. [Accessed 01 January 2020].

[4] L. Baruah, "Performance Comparison of Binarized Neural Network with Convolutional Neural Network," Michigan Tech, 2017.

[5] N. Nazari, M. Loni, M. Salehi, M. Daneshtalab and M. Sjodin, "TOT-Net: An Endeavor Toward Optimizing Ternary Neural Networks," in *22nd Euromicro Conference on Digital System Design*, Sweden, 2019.

[6] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. E. Alsaadid, "A survey of deep neural network architectures and their applications," *Neurocomputing,* 2016.

[7] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, 2017.

[8] M. Nielsen , "Deep learning," June 2019. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap6.html.

[9] M. Rastegari, V. Ordonez, J. Redmon and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *Computer Vision – ECCV 2016*, Munich, 2016.

[10] T. Gamage, H. Nanayakkara, R. Bhaskar and F. Al-Ali, "Comprehensive Survey Of FPGA Architectures: Past, Present And Future," in *Academics World* , Auckland, 2019.

[11] A. Pandit, "Introduction to FPGA and It's Programming Tools," circuitdigest, 24 April 2019. [Online]. Available: https://circuitdigest.com/tutorial/what-is-fpga-introduction-and-programming-tools. [Accessed 1 January 2020].

[12] S. Biookaghazadeh, F. Ren and M. Zhao, "Are FPGAs Suitable for Edge Computing?," in *hotedge18*, Bostan, 2018.

[13] Y. Hao and S. Quigley, "The implementation of a Deep Recurrent Neural Network Language Model on a Xilinx FPGA," *ArXiv,* 2017.

[14] X. Xu, X. Zhang, B. Yu, S. X. Hu, C. Rowen, J. Hu and Y. Shi, "DAC-SDC Low Power Object Detection Challenge for UAV Applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2019.

[15] E. Wang, J. J. Davis and P. Y. K. Cheung, "A PYNQ-Based Framework for Rapid CNN Prototyping," *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM),* pp. 223-223, 2018.

View publication stats