Deep Learning on FPGAs: Past, Present, and Future [1]

G. Lacey, G. Taylor, S. Areibi (2016)

Cristina Venturini, Samuele Piccinelli

1st March 2021

UNIVERSITÀ DEGLI STUDI DI PADOVA

Introduction

Recently, **machine learning** has been seeing a shift in philosophy for **algorithm design**.

First attempts at learning involved much feature engineering by hand while now composable feature structure systems are learnt automatically from massive amounts of data, with great improvement in performance.

Still, scientists are limited by the need for better **hardware acceleration** to accommodate scaling beyond current data and algorithm sizes.

Hardware acceleration

At the moment, hardware acceleration for deep learning is dominated by using clusters of GPUs as **general purpose processors, GPGPU**.

To program GPGPUs the most dominant platform is **NVIDIA CUDA**, but also **OpenCL** is gaining momentum.

GPGPUs are easier to use in the sense that they don't need hardware level programming capabilites, which most of researchers and application scientist don't have, so they are in this sense more user friendly.

OpenCL

OpenCL is an open source, standardised framework for algorithm acceleration on heterogenous architecture.

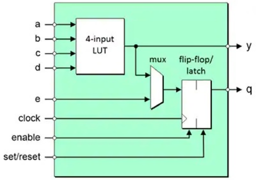
Provides low-level access to hardware and can support programming a wide variety of hardware platforms - including FPGAs.

However, this means that all supported platforms are not guaranteed to support all OpenCL function: especially in the case of FPGAs only a subset of OpenCL functions are currently supported.

FPGAs as an alternative?

FPGAs as an alternative?

FPGAs are the main competitors to GPUs for algorithm acceleration: they offer **flexible configuration** and provide **better performance per watt** then GPUs, thus reducing costs.



They can implement sequential logic through the use of *flip-flops* and combinational logic through the use of look-up tables (LUT).

Modern FPGAs also contain hardened components such as full processors cores, communication cores, arithmetic cores and block RAM.

FPGAs as an alternative?

For deep learning, FPGAs provide an obvious potential for acceleration:

- they are capable of exploiting **distributed on-chip memory**;
- they are capable of exploiting large degrees of **pipeline parallelism**, which fits naturally with the *feed-forward nature* of deep learning methods;
- modern FPGAs support **partial dynamic reconfiguration**, where part of the FPGAs can be reprogrammed while another part is used. This is particularly useful for large deep learning models;
- FPGAs architecture is tailored directly for executing task in parallel: when developing DL techniques there is less emphasis on adapting algorithms for a fixed computational structure, allowing more freedom to explore optimisation.

FPGAs as an alternative?

Main Issues with FPGAs

1. To be programmed they require specific hardware knowledge that many scientists do not possess;
2. The flexibility that FPGAs offer comes at the cost of large compile times.

1. Because of this, FPGAs have now adopted software level programming models (including OpenCL).

   Additionally, FPGAs trends are tending towards a system on-chip design approach and are rapidly replacing ASICS for fixed function logic.

Practical implementation

2. The main challenge to successfully integrate FPGAs into deep learning design flows is related to design compile time.

What is a kernel?

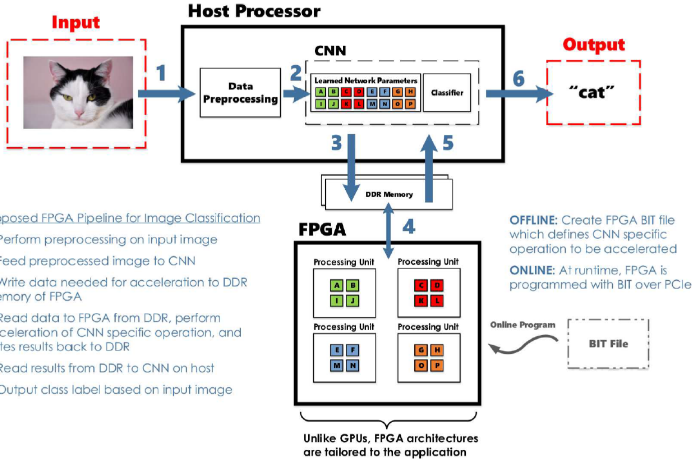
The kernel is a computer program at the core of a computer's operating system that has complete control over everything in the system and facilitates interactions between hardware and software components.

OpenCL kernel compilation time for FPGAs is on the order of ten of minutes to hours, whereas compiling generic OpenCL kernels for GPP or GPU is on the order of milliseconds to seconds.

However, this is not that much of a problem for deep learning, as deep learning tools often reuse the same pre-compiled kernels during the design phase.

Doing one-time offline compilation of commonly used deep learning kernels is a reasonable compromise.

Proposed deployment flow



Proposed FPGA Pipeline for Image Classification

1. Perform preprocessing on input image
2. Feed preprocessed image to CNN
3. Write data needed for acceleration to DDR memory of FPGA
4. Read data to FPGA from DDR, perform acceleration of CNN specific operation, and writes results back to DDR
5. Read results from DDR to CNN on host
6. Output class label based on input image

OFFLINE: Create FPGA BIT file which defines CNN specific operation to be accelerated

ONLINE: At runtime, FPGA is programmed with BIT over PCIe

Unlike GPUs, FPGA architectures are tailored to the application

Overview of main implementations

- *Cloutier et al.* where among the firsts but were strongly limited by FPGAs sizes constraints. Also, FPGAs at this time did not contain MAC units and so arithmetic was very slow and resource expensive.
- State of the art performance for forward propagation CNNs was achieved by *Ovtcharov et al.*with a throughput of 134 images/second while operating at 25 watt. Performance is projected to increase to an estimated throughput of 233 images /second, consuming the same power.

Since there aren't many implementations yet and FPGAs specific differences can be quite vast, it is hard to determine specific optimal architecture decision and so more research is needed.

Most FPGAs efforts have involved accelerating the forward propagation, but accelerating backward propagation is also an area of interest.

Looking forward

For this techniques to succeed in the problems of tomorrow, they must scale to accomodate the data sizes and architectures that continue to grow at an incredible rate.

The hardware is headed towards larger memory, smaller feature size, interconnected improvements as to reach multi-FPGA configuration.

Design tools will likely tend to higher level abstraction and software-like experiences.

Increasing degrees of freedom for training

While training ML algorithms tunable hyper-parameters need to be adjusted. Recently, researchers have been using adaptive methods, which exploit the results of hyper-parameters tuning attempts.

Training procedures using fixed architectures are limited in their ability to grow sets of possible models, since they make it difficult to explore settings between models.

The flexible architecture of FPGAs may be better suited for these types of optimisation as a completely different hardware structure can be programmed and accelerated at runtime.

Low power compute clusters

Deep learning techniques are often scaled-up across multi-node computing infrastructures. **Current solutions** to this problem involve using clusters of GPUs with Infiniband interconnects and MPI to allow high level of parallel computing power and fast data transfer between nodes.

However, as the workloads of this large-scale applications become increasingly heterogeneous, the use of FPGAs may prove to be a superior alternative.

Programmability of FPGAs would allow re-configurability based on the application and workload and also FPGAs attractive performance/watt ratio would lower costs for data centres.

2021 Developments

Current status

2021 Developments [2]

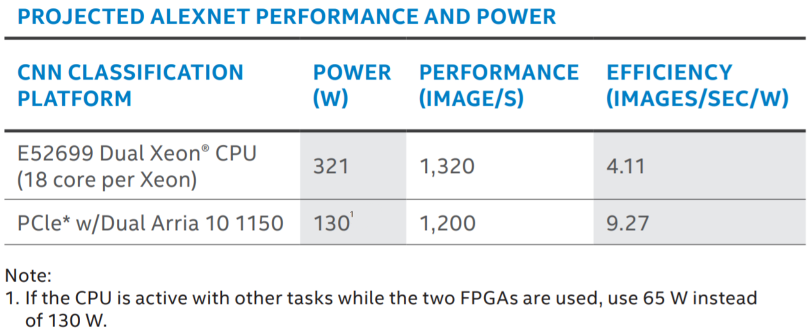
At the moment, the AlexNet CNN topology on Intel FPGAs classifies 50.000 validation set images at > 500 images/second at ~ 35 W (developed with OpenCL device).

The design uses OpenCL kernels to implement each CNN layer. Data is passed from one layer to the next through a mechanism which allows data to move from one kernel to the next without sending data to the external memory.

This FPGA implementation is a data buffer that uses the internal memory structures next to the kernel itself.

Arria 10 and Stratix 10 FPGAs have innovative DSP blocks that also deliver IEEE-754 floating-point addition, which results in less FPGA logic and higher clock speeds for better performance and lower power.

2021 Developments



PROJECTED ALEXNET PERFORMANCE AND POWER

CNN CLASSIFICATION PLATFORM	POWER (W)	PERFORMANCE (IMAGE/S)	EFFICIENCY (IMAGES/SEC/W)
E52699 Dual Xeon® CPU (18 core per Xeon)	321	1,320	4.11
PCIe* w/Dual Arria 10 1150	130[1]	1,200	9.27

Note:
1. If the CPU is active with other tasks while the two FPGAs are used, use 65 W instead of 130 W.

2021 developments [3]

Intel is also further reducing the amount of expertise needed with a software-based programming model. This higher-level FPGA programming model allows a data scientist to create a NN using a common AI framework—such as TensorFlow or Caffe - and deploy it on an FPGA without knowing the details of the FPGA architecture.

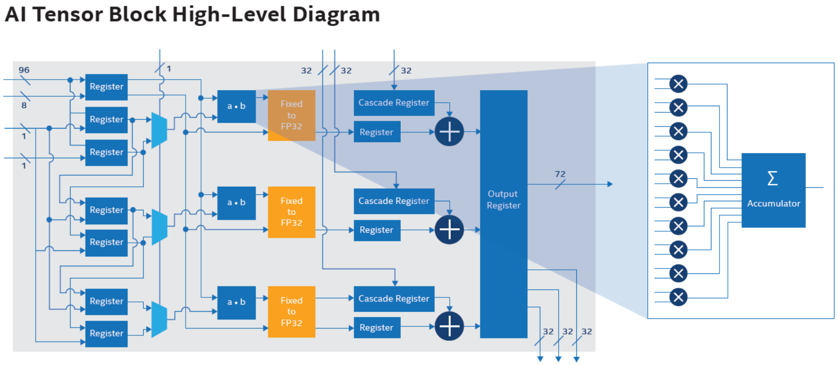
Examples are:

- Intel Distribution of OpenVINO™ toolkit
- Intel FPGA Deep Learning Acceleration Suite
- Intel FPGA SDK for OpenCL™ software technology acceleration

Example

Intel Stratix 10 NX FPGA

Furthermore, Intel Stratix 10 NX FPGA is Intel's first AI-optimized FPGA. It embeds a new type of AI-optimized block, the AI Tensor Block, tuned for common matrix-matrix or vector-matrix multiplications.



Bibliography

- G. Lacey, G. W. Taylor, and S. Areibi, “Deep learning on FPGAs: Past, present, and future,”
- “MS Windows NT kernel description.” https://www.intel.it/content/www/it/it/products/docs/storage/programmable/applications/machine-learning.html. Accessed: 25/02/2021.
- “MS Windows NT kernel description.” https://www.intel.it/content/www/it/it/artificial-intelligence/programmable/fpga-gpu.html. Accessed: 25/02/2021.