CNN Acceleration and Simulation on an FPGA

The Iby and Aladar Fleischman Faculty of Engineering
Tel Aviv University

Project Number: 20-1-1-2187

Names: Chaim Gruda, Shay Tsabar

Advisor: Yoni Seifert

Introduction

In recent years image recognition has become a vital feature in ever more applications, ranging from autonomous vehicles, security, healthcare and more. Convolution neural networks (CNN) algorithms are widely used in many such applications since their high accuracy for image recognition. However, with the high accuracy come long computation times and high-power needs. The need to include such image recognition capabilities in embedded systems with tight real-time and power constraints, lead the design for hardware accelerators for CNNs.

Objectives

The objective of the project is to explore the advantages of hardware implementation of a CNN compared to the same CNN implemented in software, regarding the following:

- **Time acceleration** –hardware can be utilized to parallelize computations, which in software are performed in series.
- **Prediction accuracy** – in order of simplifying hardware implementation, fixed point arithmetic is used in calculations, and may cause accuracy loss of the CNN
- **Power consumption** – dedicated CNN hardware processing elements are expected to utilize less power than a CPU.

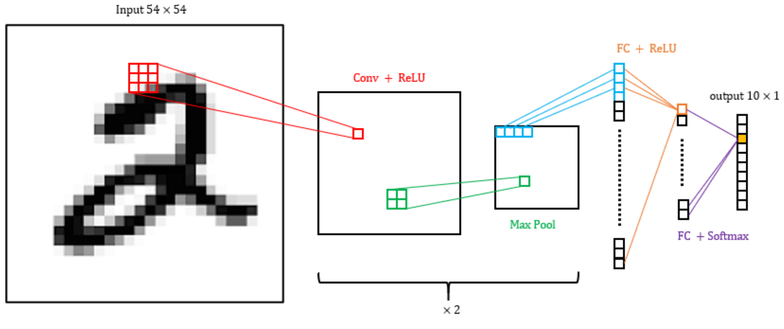


Figure 1: Accelerated CNN model on MNIST dataset

The CNN model which will be tested for acceleration, consists of convolution, pooling and fully connected layers as described in the **Figure 1**. The network is trained using Python Keras library and the MNIST dataset, which consists of 60,000 images of labeled handwritten numbers. The Hardware platform is the Zynq-7000 SoC based Zedboard.

Implementation

The Zedboard Zynq-7000 SoC includes two subsystems that are utilized for the implementation of the CNN accelerator:

- **Processing System (PS)** includes an ARM cortex-A9 667MHz processor, which runs C code on bare-metal, that configures and controls the CNN, and is the interface with the outside world.
- **Programmable Logic (PL)** which is an FPGA, programmed to contain Processing Elements (PE) each implementing a layer of the CNN, as well as DMA engine and Block RAM.

A block diagram of the CNN accelerator system is seen in **Figure 2**. (**red** = data; **black** = control).

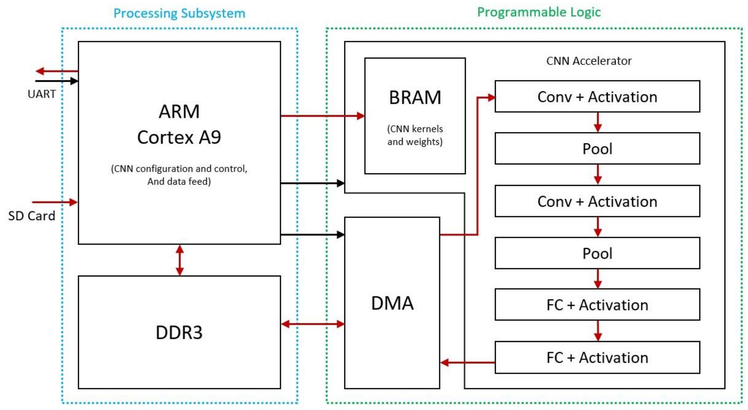


Figure 2: Block diagram of CNN accelerator on Zedboard

Each PE has access to BRAM that stores static kernels and weights of the CNN. The image data is streamed into and out of each PE using AXI stream interface. The first and last PEs complete the data-path with the PS using a DMA engine.

Since each pixel is included in multiple calculations in the convolution and pool layers, each corresponding PE has a 'Sliding Window' mechanism. that efficiently utilizes the stream of data and BRAM to allow Data reuse of image pixels, as illustrated in **Figure 3**. This also allows each PE to start calculations as soon as possible, Increasing throughput.

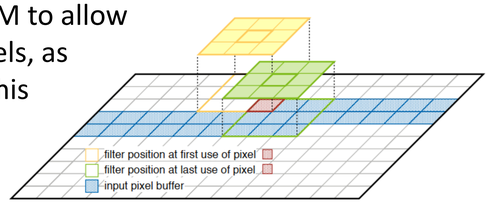
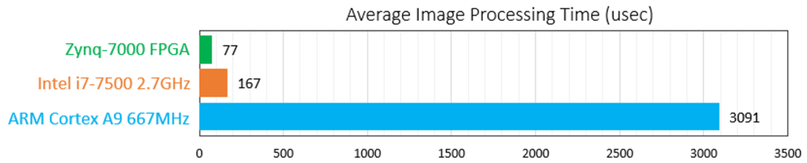


Figure 3: Sliding window mechanism illustration

Results

Following graph shows average image processing time for the hardware CNN accelerator, compared to the software implementation, executed on an Intel i7 high-end processor, and on the Zedboard embedded ARM Cortex-A9 processor:



Graph 1: Time comparison

The following table shows that the accuracy and certainty of the hardware implemented CNN were only slightly affected by the fixed-point arithmetic:

	Floating point (FP64)	Fixed point (Q32.10)
Accuracy	96.03%	96.02%
Certainty	98.83%	98.82%

Table 1: Accuracy comparison

The following table shows estimated power consumption percentage of Zynq-7000 subsystems:

	Processing System	Programmable Logic
Dynamic Power	77%	23%
Total Power	71%	29%

Table 2: Power consumption comparison of PS vs PL

Conclusions

Hardware implementations of CNNs give great results compared to software implementations in the following:

- **Calculation time** can be accelerated, especially in embedded systems which include low-end CPUs, and even in high-end CPUs.
- **Prediction accuracy** of the CNN can be kept in similar ranges when using fixed-point arithmetic which is simpler to implement in hardware.
- **Power consumption** of dedicated processing elements is significantly less than that of a CPU, thus making the hardware implementation much more energy conserving.