Integrated system architecture

Implementation of a quantized FPGA accelerator for convolutional neural networks

Introduction

Computer vision is the field that studies how computers can extract high level understanding from images, to automatize complex tasks such as facial recognition, autonomous driving, and cancer detection. In recent years, methods based on deep neural networks (DNNs) have become the standard approach for such problems. DNNs are loosely inspired by biological neurons and process raw sensory data through multiple layers. Convolutional neural networks (CNNs) are a class of DNNs specialized on image processing. CNNs learn how to extract high level features through statistical learning from a large pool of labelled data, in a process called training. By comparison, traditional algorithms are based on hard-coded feature extractors written by experts, which cannot be easily converted or extended to new data sets, for the same task. However, the superior accuracy of CNNs comes at the cost of high computational complexity, typically in the order of billions of multiply and accumulate (MAC) operations for recent state of the art networks. In recent years, both software and hardware researchers focused on minimizing the computation by compressing the networks and searching for optimized accelerators.

Project description

CNNs are usually trained on GPUs using floating points precision for the data (weights and activations), whereas are quantized to 8-16 bits when executed on low power devices (such as smartphones or drones). GPUs are very efficient when processing high amount of data in a massively parallelized way, such as during the training, but are inherently energy hungry and infeasible for deployment on battery powered devices. By contrast, FPGAs and ASIC are extremely efficient and can be designed around the workload that will be processed at the edge. Developing an efficient CNN hardware accelerator is not an easy task, therefore there had been many efforts in finding flexible architectures that can be easily ported to different workloads.

A recent solution is to co-design both the CNN and its accelerator, carefully tailoring the computation requirements and resources availability with the target task accuracy. Xilinx's researchers developed a framework called Brevitas, built upon Pytorch, to train low bit-width quantized CNNs with the target accelerator in mind. The trained CNN model is then passed to another framework called FINN (Xilinx again) which generates an accelerator based on a systolic array that can execute the network efficiently, respecting the performance targets determined by the designer. The result is a bitstream that can be mounted on a FPGA accelerator, providing an easy way to design and deploy efficient CNNs.

In this special project, you will learn the workflow of implementing a quantized convolutional neural network accelerator on a Xilinx FPGA using a python-HLS toolchain. Then you will optimize the performance by trading-off resource usage, energy consumption, and latency with accuracy by reducing the quantization. Two CNNs trained on CIFAR-10 and MNIST and quantized with Brevitas are provided, they can be used as baseline for the implementation, but the candidates can also provide their own trained CNNs with different layers. The hardware accelerator for the two CNN models is generated with FINN, a systolic array generator, and deployed on a Xilinx FPGA. The toolchain is written in Python, C++ and HLS.

Project summary

1- Read the introduction to convolutional neural networks [a-b-c], in this step you will learn the basic concepts such as the mathematical model.
2- Understand the basic concepts of efficient execution of CNNs [d-e-f], such as architectural choices during the design phase of the CNN and quantized execution during the inference.
3- Learn how to implement a simple CNN for MNIST using Python and Pytorch [g], some reference scripts will be provided.
4- Learn how to use the PYNQ-FINN-BREVITAS toolchain using the tutorial and deployment-ready examples provided on GitHub [h-i-j].
5- Generate two accelerators for each CNN using the toolchain, set the FINN framework to use DSP48 blocks to implement the MAC units, measure baseline performance metrics for the inference, report resource usage, power and latency.
6- Repeat step ***5*** but set the FINN framework to implement the MAC units in LUT instead of DSP48 blocks**.**

Notes

The project can be delivered before the deadline.

The concepts learned during this project will prove useful in case you want to carry out your master thesis on a ML topic. Additionally, this project can evolve into a master thesis.

For any questions, please contact us via email at:
maurizio.martina@polito.it
emanuele.valpreda@polito.it

References:

Before reading the references, have a look at this YouTube video, for a quick introduction on CNNs:
Convolutional Neural Networks – Explained (a simple and informative video for beginners)

[a] A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way
[b] Neural networks tutorial part 1, part 2, part 3
[c] Convolutional neural networks tutorial part1, part 2
[d] Efficient Processing of Deep Neural Networks: A Tutorial and Survey
[e] A Survey of Quantization Methods for Efficient Neural Network Inference
[f] FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks
[g] Pytorch tutorials
[h] FINN: https://xilinx.github.io/finn/
[i] Brevitas: https://github.com/Xilinx/brevitas
[j] PYNQ: https://pynq.readthedocs.io/en/latest/index.html

Extra references:

A Survey of Quantization Methods for Efficient Neural Network Inference
A Survey of Accelerator Architectures for Deep Neural Networks