Accelerating Artificial Neural Networks on Embedded FPGA with Hybrid Custom Floating-Point and Logarithmic Dot-Product Approximation

Yarib Nevarez, Alberto Garcia-Ortiz
Universität Bremen, nevarez@item.uni-bremen.de, agarcia@item.uni-bremen.de

Abstract

We accelerate artificial neural networks (ANN) optimizing the floating-point computation with vector dot-product approximation. The proposed method exploits the intrinsic error resilience of machine learning (ML) algorithms to reduce computational latency, memory footprint, and power dissipation while preserving inference accuracy. This method is integrated in a hardware/software co-design exploration framework. To demonstrate our approach, we address a hardware design exploration with convolutional neural networks (CNNs) and spiking neural networks (SNNs).

Introduction

We accelerate ANN with a dot-product hardware design based on approximate computing with hybrid custom floating-point and logarithmic number representation[1]. This hardware unit has a quality configurable scheme based on the bit truncation of the synaptic-weight vector. **Fig.** 1 illustrates the dot-product hardware module with standard floating-point (IEEE 754) arithmetic, and our approach with hybrid custom floating-point as well as logarithmic approximation. As a design parameter, the mantissa bit-width of the weight vector provides a tunable knob to trade-off between efficiency and quality of result (QoR)[2]. Since the lower-order bits have smaller significance than the higher-order bits, truncating them may have only a minor impact on QoR [3]. Further on, we can remove completely the mantissa bits in order to use only the exponent of a floating-point representation. Therefore, the most efficient configuration becomes a logarithmic representation, which consequently leads to significant hardware-level optimizations using only adders and shifters for dot-product approximation. Moreover, since approximations and noise have qualitatively the same effect[4], we propose noise tolerance plots as an intuitive visual measure to provide insights into the quality degradation of ANN under approximate processing effects.

Dot-Product Block-Level Design

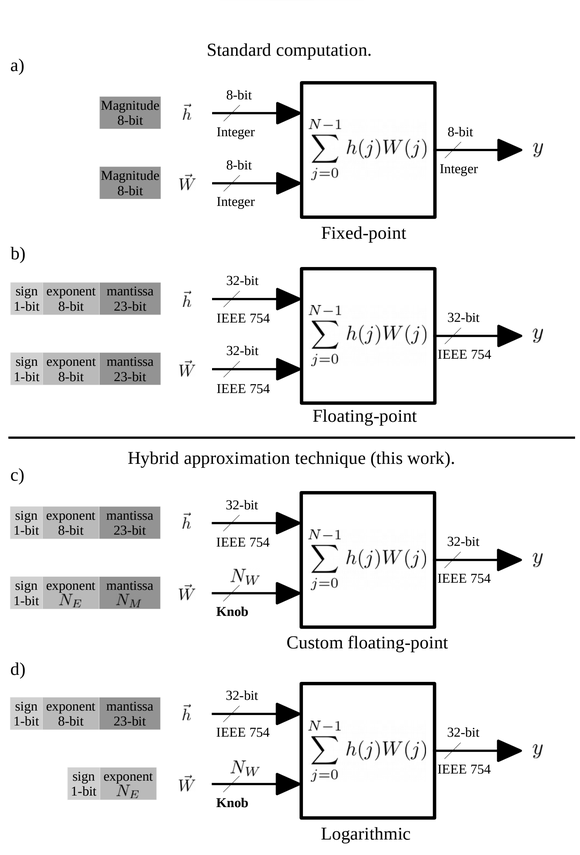

Figure 1: Hardware alternatives for vector dot-product.

Tensor Processor

This section presents a tensor processor (TP) compatible with TensorFlow Lite to accelerate *Conv2D* (**Eq.** (1)) and *DepthwiseConv2D* (**Eq.** (2)) operations on embedded FPGA. This TP instantiates the compute engines shown in **Fig.** 1.

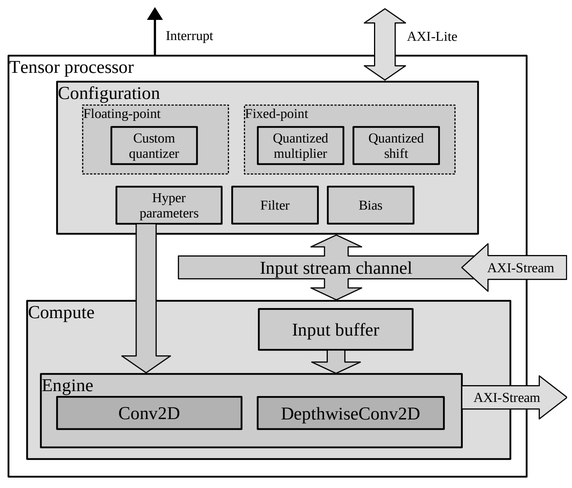

Figure 2: Hardware architecture of the proposed tensor processor.

Embedded System Architecture

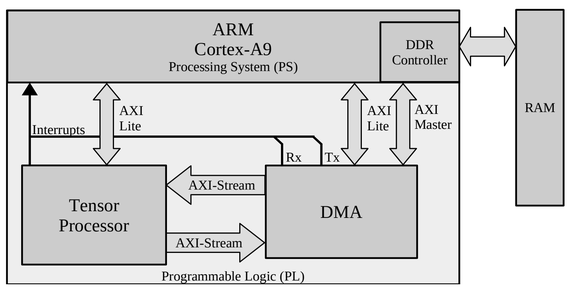

Figure 3: Base embedded hardware architecture.

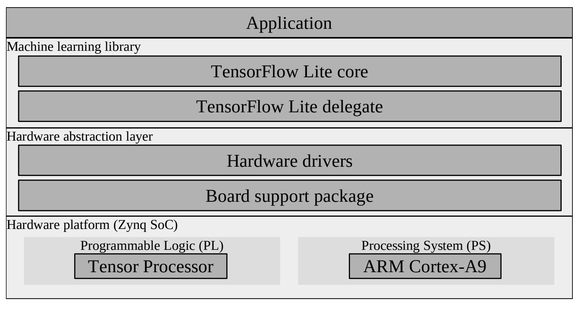

Figure 4: Base embedded software architecture.

Experimental Results

A single TP achieves a peak acceleration of 45× on *Conv2D* tensor operation. The energy-delay product (EDP) of the different compute engines are compared in **Tab.** 1. The implemented custom floating-point formats and their classification accuracy are shown in **Fig.** 5.

Table 1: Energy consumption of a single TP at peak acceleration of 45X.

Engine	Power (W)	EDP (J)	Reduction
CPU (ARM Cortex-A9)	1.404	7,812.97	1.00
TP (Fixed-point)	0.136	16.67	468.66
TP (Floating-point LogiCORE)	0.070	39.85	196.05
TP (Hybrid custom floating-point)	0.066	8.19	**954.43**
TP (Hybrid logarithmic)	0.060	7.40	**1,055.92**

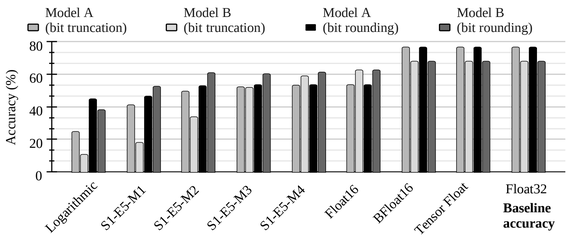

Figure 5: Classification accuracy using hybrid custom floating-point approximation with various formats.

Tolerance Plot

For quality monitoring, we propose tolerance plots as an intuitive visual measure to provide insights into the accuracy degradation of ANN under different approximate processing effects. This plot revels inherent error resilience, hence, the possibilities for approximate processing. **Fig.** 6 shows the tolerance plot of a SbS network with hybrid logarithmic approximation [1], this plot reveals elevated approximation resilience. The SbS dynamics is described in **Eq.** (3).

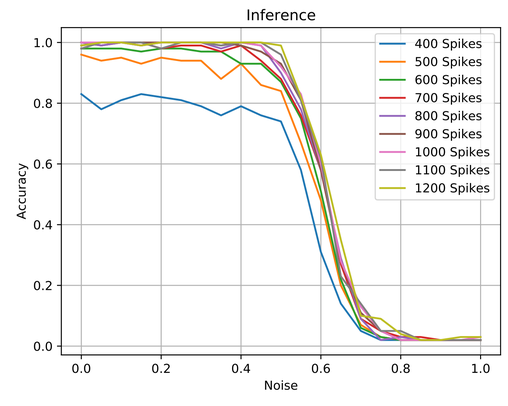

Figure 6: Tolerance plot.

Conclusions

We present a hardware/software co-design framework to accelerate floating-point tensor computation in embedded FPGA with approximation techniques to reduce energy consumption and resource utilization.

Implemented operations

$$Conv2D\,(W,X)_{i,j,o} = \sum_{k,l,m}^{K,L,M} W_{(o,k,l,m)} \cdot X_{(i+k,j+l,m)} + b_o \tag{1}$$

$$DepthwiseConv2D\,(W,X)_{i,j,n} = \sum_{k,l}^{K,L} W_{(k,l,n)} \cdot X_{(i+k,j+l,n)} + b_n \tag{2}$$

$$h_\mu^{new}(i) = \frac{1}{1+\epsilon}\left(h_\mu(i) + \epsilon \frac{h_\mu(i)W(s_t|i)}{\sum_j h_\mu(j)W(s_t|j)}\right) \tag{3}$$

References

[1] Yarib Nevarez, David Rotermund, Klaus R Pawelzik, and Alberto Garcia-Ortiz. Accelerating spike-by-spike neural networks on fpga with hybrid custom floating-point and logarithmic dot-product approximation.

[2] Jie Han and Michael Orshansky. Approximate computing: An emerging paradigm for energy-efficient design.

[3] Sparsh Mittal. A survey of techniques for approximate computing.

[4] Swagath Venkataramani, Srimat T Chakradhar, Kaushik Roy, and Anand Raghunathan. Approximate computing and the quest for computing efficiency.

Acknowledgements

This work is funded by *CONACyT*.

Universität Bremen

ITEM.IDS