Pareto-Optimal Approximate Multiplication via Cartesian Genetic Programming for Neural Networks

Contact Person: Mr. William Simon (william.simon@epfl.ch),
Dr. Marina Zapater (marina.zapater@epfl.ch),
Prof. David Atienza (david.atienza@epfl.ch)

Project Description

Approximate computing is useful for improving the efficiency of error tolerant applications such as neural networks by trading accuracy for reduced delay, area, and power consumption. However, approximate multipliers still reduce network accuracy to unacceptable levels, requiring network retraining. This retraining is very time consuming, as the approximate multiplier must be simulated, and, therefore, hardware accelerators such as GPUs cannot be used to accelerate training. To overcome this, we have previously demonstrated an approximate multiplier that guarantees exact results for any input value when the weight values are fixed to specific values. By quantizing weights to these values, we can again perform retraining with acceleration.

The current approximate multiplier is hand-designed. However, Cartesian Genetic Programming (CGP) has been previously demonstrated as an effective way to find Pareto-optimal approximate multipliers with respect to power consumption and various error metrics such as mean relative error distance[1]. CGP is performed by evolving a circuit across generations. In each generation, the circuit is mutated to produce multiple offspring, which are evaluated via a fitness function, before selecting the best offspring for new mutation.

Previous works have applied CGP to evolve approximate multipliers that minimize various error metrics against power. In this project, we plan to maximize the number of weight values that produce exact results for any input value. This will be accomplished by defining a mutation function that apply *n* random mutations to the multiplier graph, producing *n* offspring, which will be evaluated for the number of useful weight values, before passing the best offspring for further mutations.

Simultaneously, we would also like to accelerate the mutating and testing process by treating the Cartesian graph representing the multiplier as a tensor-based graph that can be accelerated by tensor libraries such as Tensorflow or Pytorch. To this end, we have already designed a framework in Pytorch that converts a graph description of a multiplier into a Pytorch model, greatly reducing computation time and, hence, enabling more offspring to be tested. This will need to be expanded to support the mutation function as described above.

Tasks of the Student:

1. Become familiar with Tensorflow or equivalent framework if not already.
2. Become familiar with current work applying CGP to approximate multipliers.
3. Develop mutation and fitness function and integrate them into CGP generational flow.
4. Expand current tensor-based acceleration in Pytorch to support full CGP flow.
5. If previous objectives are fulfilled, this work will be a part of a submission in a peer-reviewed conference or journal.

Requirements:

- Previous work with Python, with an understanding of efficient, high-performance coding techniques such as list comprehension.

Appreciated Skills:

- Previous work with Tensorflow or equivalent neural network framework.
- Motivation to learn new skills.
- Autonomous work ability.

References:

[1] EvoApprox8b: Library of Approximate Adders and Multipliers for Circuit Design and Benchmarking of Approximation Methods