Comparison of CNNs on Amazon EC2 versus FPGAs

Kyle Daruwalla and Akhil Sundararajan

ECE 901 Fall 2016

October 31, 2016

Overview

Introduction to CNNs and FPGAs

Proposed work

Proposed analysis

Convolutional Neural Networks (CNNs)

$$\min_{f \in F} \sum_{i=1}^{n} \mathcal{L}(f(x_i); y_i) \tag{1}$$

- Uses SGD with backpropagation to arrive at optimum
- Inherently serial
- Potential system overhead to perform per-iteration computation

Field-Programmable Gate Arrays (FPGAs)

- Reconfigurable hardware platform
- Common target for real-time applications
- Written in hardware description language (HDL)
- Project Catapult is targeting FPGAs for NN implementation

Software Implementation

- Google's TensorFlow – describe CNNs at the layer level
- Amazon EC2 for deployment
    - Single CPU implementation
    - GPU implementation
- HOGWILD! implementation for GPUs

Hardware Implementation

- Use FPGAs to build CNN structure
- Modularize design into filters
- Use controller to pass data through filters and update weights
- Target cost-per-iteration speedup by optimizing filter units

Theoretical Analysis

- Use FPGA implementation to define constant bounds on time per filter operation
- Use timing constants + CNN structure to provide theoretical bound on cost-per-iteration
- Analyze computational complexity in terms of this cost-per-iteration

Emperical Results

- Comparing generalization error between the CPU, GPU, Hogwild!, and FPGA implementations.
- Comparing convergance rates between the CPU, GPU, Hogwild!, and FPGA implementations.
- Provide a metric of when FPGAs might provide a larger speedup than Hogwild!

References

F. Niu, et al. (2011, Nov. 11). *Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent* (v2) [Online]. Available: `https://arxiv.org/abs/1106.5730v2`

The End