CS554 Project Ideas

uDNN: Micro Deep Neural Networks with Variable Precision Computations

Overview

Deep learning has revolutionized many domains from understanding complex images and videos, to speech recognition, to self-driving cars. NVIDIA saw an opportunity in deep neural network's ability to function correctly even with low precision data types (e.g. quarter precision and half precision data, 8-bit, 16-bit), and has produced optimized GPUs to support significantly faster computations when using these limited precision data types. This project will explore the performance of several classes of neural networks used in the design of deep neural networks, deep belief networks and deep reinforcement learning networks, more precisely convolutional neural networks, recurrent neural networks, belief networks and reinforcement learning algorithms, analyzing the training speed and the learning quality of various neural networks when using different precision levels (8-bit, 16-bit, 32-bit, and 64-bit).

You will need to find a problem that can be solved by writing an application that uses simple neural networks, belief networks or reinforcement learning algorithms and you will have to implement several variants of the neural network or algorithm, each variant being targeted for a specific computational architecture: CPUs, GPUs and FPGAs. Start with a naive implementation, evaluate the performance and then optimize the application by using intrinsic instructions that each platform provides (AVX instructions, GPU low precision intrinsic, FPGA dedicated arithmetical module). You are allowed to use open-source C or Fortran optimized mathematical libraries to speed up the performance of your implementation, but you are not allowed to use artificial intelligence, machine learning and deep learning frameworks.

Relevant Systems and Reading Material

- https://en.wikipedia.org/wiki/Deep_learning
- https://developer.nvidia.com/deep-learning
- http://proceedings.mlr.press/v37/gupta15.pdf
- https://arxiv.org/pdf/1410.0759.pdf
- https://arxiv.org/pdf/1412.7024.pdf
- http://ac.els-cdn.com/S0893608014002135/1-s2.0-S0893608014002135-main.pdf?_tid=aaa9f742-9809-11e7-8f8f-00000aacb362&acdnat=1505255474_1e63be99a40f9d5c311df0340d9debf1
- https://devblogs.nvidia.com/mixed-precision-programming-cuda-8/
- https://software.intel.com/sites/landingpage/IntrinsicsGuide/
- https://artint.info/html/ArtInt_262.html (Reinforcement Learning)
- https://artint.info/html/ArtInt_148.html (Belief Networks)
- https://artint.info/html/ArtInt_256.html (Learning Belief Networks)
- https://artint.info/html/ArtInt_183.html (Neural Networks)
- https://en.wikipedia.org/wiki/MNIST_database (good dataset to begin evaluation)

Preferred/Required Skills

- Principles: operating systems, parallel computing, artificial intelligence, machine learning;
- Programming: C, C++, Python, Bash, multi-threaded/multi-process programming, CUDA, OpenCL;
- Operating System: Linux/UNIX;

Evaluation and Metrics

You will evaluate the performance of your implementation in terms of training speed, convergence speed, convergence factor (does it ever converge?) and in terms of quality specific for what type of neural network or algorithm you picked (i.e. accuracy, precision, recall for neural networks, cumulative reward for reinforcement learning). The evaluation should be done on real datasets (both training and testing) and experiments should be conducted on bare-metal Chameleon Cloud instances.

Project Mentor

Alexandru Iulian Orhean -- aorhean@hawk.iit.edu