Neural Net Implementation on FPGA's: Proposal

Cory Nezin, Brenda So

September 6, 2017

Introduction

Neural networks has recently gained momentum in the machine learning and artificial intelligence community. While neural networks provides state-of-the-art accuracy in many tasks, it comes with the cost of high computational and space complexity which modern CPU's can barely handle. The two main contenders that can achieve the computing ability required by neural nets includes FPGA's and GPU's. The comparison between FPGA's and GPU's are still in open research. Our project focuses on exploring ways to optimize neural network performance on FPGA, and ultimately provides an analog front end to demonstrate the ability of neural networks in real-life applications.

Goals

1. Design and implement hardware optimized inference on an FPGA.
   (a) Create a framework to convert symbolic neural nets to FPGA code. Some common neural net components include
      i. ReLU, Sigmoid, Threshold.
      ii. Fully Connected (matrix multiply - with Strassen algorithm?).
      iii. Sparsely Connected? (Sparse martrix multiply).
      iv. 1D (2D and 3D also?) Convolution - just FIR filters.
      v. Max pooling, average pooling, downsampling.

vi. Recurrent elements?

(b) Optimize neural net implementation by improving energy efficiency and speed without sacrificing accuracy.

i. Design high level architecture to increase data reuse and reduce energy consumption. Some examples of high level architectures include:

A. Weight stationary (WS) [3]

B. Output stationary (OS) [3]

C. Row stationary (RS) [3]

ii. Implement weight compression to decrease neural net size. Ways to implement weight compression include:

A. Use multiplierless multiplication?

B. Use SVD for fully connected compression?

C. Implement Optimal Brain Damage.

D. Implement Optimal Brain Surgeon.

E. Implement Deep Compression for specialized hardware. [2]

iii. Test neural net on a toy data set (spiral, checkerboard, wdbc).

A. Implement a simple live input method - ADC?

B. Perform live classification of hand written digits.

2. Benchmark performance of FPGA's against GPU's and CPU's.

(a) Quantify operation in floating point and fixed point.

(b) Measure performance in terms of

i. power and energy consumption, of both processor and off-chip access (e.g. DRAM accesses)

ii. latency and throughput

iii. memory consumption

iv. FPGA-specific measurements, e.g. utilization of resources such as DSP, BRAM, LUT, FF

v. accuracy

3. Build a general front end, pick one of the following:

(a) Radar - Gender identification (toy problem), people in room.

(b) Style Transfer - live feed.

(c) Audio - Frank's speech denoising.

(d) Video - Live lip reading [1], live blackboard transcription.

(e) Image - Handwriting transcription.

(f) Lidar - Car or pedestrian detection.

(g) Wireless - Signal classification, smart jamming?

References

[1] Y. M. Assael, B. Shillingford, S. Whiteson, and N. de Freitas. Lipnet: Sentence-level lipreading. *CoRR*, abs/1611.01599, 2016. URL `http://arxiv.org/abs/1611.01599`.

[2] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. URL `http://arxiv.org/abs/1510.00149`.

[3] V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *CoRR*, abs/1703.09039, 2017. URL `http://arxiv.org/abs/1703.09039`.