MagmaDNN: Towards High Performance and Distributed Deep Learning

Daniel Nichols[1], Sedrick Keh[2], Kam Fai Chan[3], Stan Tomov[1,4], Kwai Wong[1,5]

[1]University of Tennessee, Knoxville, [2]Hong Kong University of Science and Technology,
[3]Chinese University of Hong Kong, [4]Innovative Computing Laboratory, [5]Joint Institute for Computational Science

Introduction

Deep Learning (DL) has become an increasingly attractive tool with the advent of GPUs and distributed computing. Models that would have taken years to train can now be trained in several hours or less. Progress in training techniques has led to the introduction of larger and more complicated models, which only increase the resources and tools necessary to train. Most DL solutions aim to provide a modular and user-friendly interface, typically in Python, often at the cost of performance.
MagmaDNN is a c++ framework powered by the MAGMA [1] linear algebra package aimed at providing a modular interface for accelerated and distributed DL.
At the heart of DL is an optimization problem, namely,

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \left[\mathcal{L}\left(f^*(\boldsymbol{x}), f(\boldsymbol{x};\boldsymbol{w})\right)\right]$$

where is $\mathcal{D}$ the data set, $\boldsymbol{w}$ are the network weights, f is the neural network, and $f*$ is the ground truth.
This optimization problem is highly non-convex and poses immense training difficulties. Thus large amounts of data, GPUs, and distribution strategies are necessary to train interesting models.

Design

MagmaDNN is designed completely in c++ for better performance and easier access to frameworks such as MAGMA, CuDNN, and MPI. To remove the memory management burden of c++ MagmaDNN uses its own dynamic MemoryManager and atomic reference counting.
At the heart of MagmaDNN's design is the Tensor core. Tensors in MagmaDNN are multi-dimensional arrays capable of being stored on CPU and GPU memory. Around the Tensor core is a library of math functions and operations used in MagmaDNN's compute graph system.

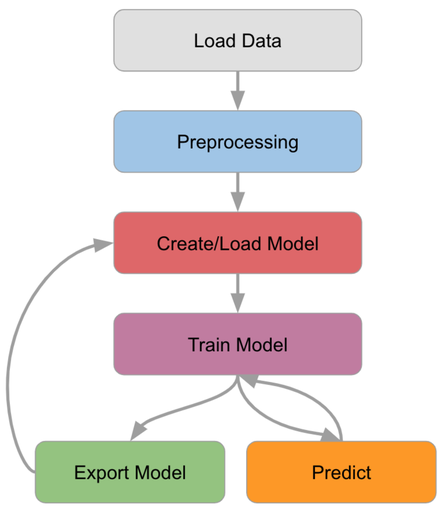


Fig. 1: MagmaDNN Workflow

Distributed Deep Learning

MagmaDNN employs *data parallelism* as its approach to distributed DL. Models are copied across GPUs and the gradients are averaged across networks at mini-batch intervals. A second approach is *model parallelism*, where different parameters of the model are distributed across nodes. Typically, a hybrid approach provides optimal scaling. However, hybrid approaches are not generalizable as they are model dependent.

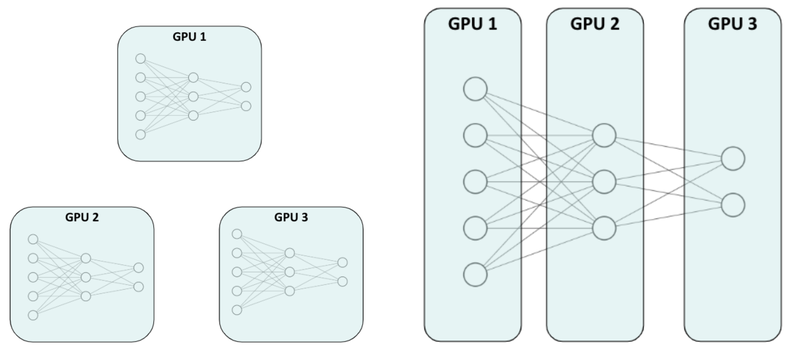


Fig. 2: Data (left) and Model (right) Parallelism

Older implementations of data parallelism use a master-worker model for averaging weights. In this model weights are sent from a master node to N worker nodes. Let w^j be the weights of the j-th worker node. Each node computes the gradient ∇w^j and sends it back to the master node. Once the master node has received the gradients from each worker it calculates

$$\overline{w} \leftarrow \overline{w} - \eta/N \sum_{j=1}^{N} \nabla w^j \,,$$

the average weight, and broadcasts $\overline{w}$ back to each worker. Modern implementations, as well as MagmaDNN's implementation, remove the Master node and average the gradients using `MPI_Allreduce`. Any CUDA-aware MPI implementation can perform this operation, however, Nvidia NCCL's `ncclAllReduce` has shown optimal performance between dense GPU nodes.

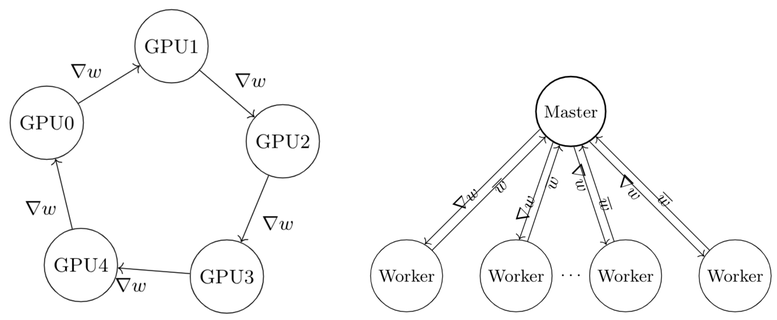
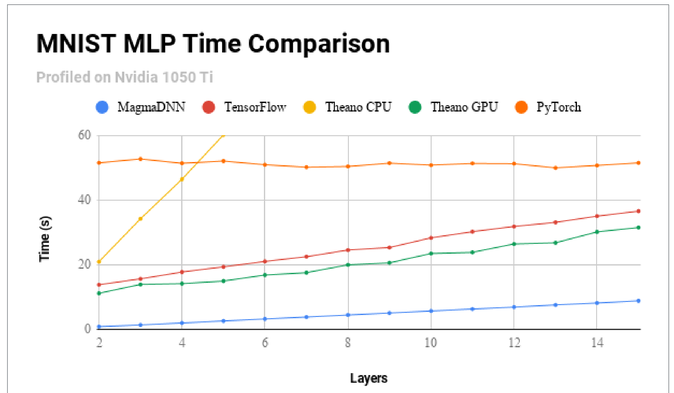


Fig. 3: Distributed Training Techniques

Performance

MagmaDNN is compared against several other popular DL frameworks: Tensorflow, PyTorch, Theano(CPU & GPU). All the testing code is written in Python except for MagmaDNN, which is written in c++.
Each test is ran on a 1050Ti Nvidia GPU with 4GB memory paired with an Intel Xeon 12 core processor. A variable amount of dense layers are used to show scalability with model complexity.



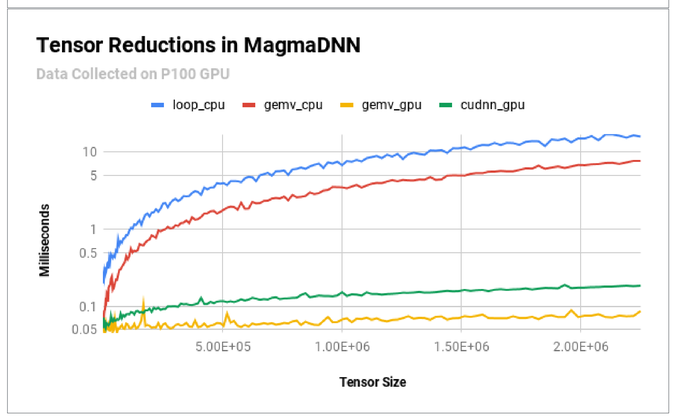


Fig. 4: Training Comparisons

Remarks and Future Work

MagmaDNN has the best performance and scales well on dense networks when compared to other frameworks.
MagmaDNN has 2 main development goals: provide a high-performance framework for distributed DL and maintaining a usable interface. By MagmaDNN v2.0 we aim to push forward in each of these categories with two major tasks. MagmaDNN should have a competitive ResNet50 training on large scale systems, such as Summit. By v2.0 it will also fully utilize a modern c++ interface and c++2*x* STL features.

Acknowledgements

This work is sponsored by the National Science Foundation, Joint Institute for Computational Science, Innovative Computing Laboratory, and University of Tennessee, Knoxville.

References

[1] Stanimire Tomov, Jack Dongarra, and Marc Baboulin. "Towards dense linear algebra for hybrid GPU accelerated manycore systems". In: *Parallel Computing* 36.5-6 (June 2010), pp. 232–240. ISSN: 0167-8191. DOI: 10.1016/j.parco.2009.12.005.