MagmaDNN: Accelerated Deep Learning Using MAGMA

D. Nichols¹, K. Wong¹, S. Tomov¹, L. Ng², S. Chen², A. Gessinger³

¹ - University of Tennessee, Knoxville
² - The Chinese University of Hong Kong
³ - Slippery Rock University
Organization

- Motivation
- Magma
- MagmaDNN Overview
  - Framework Overview
  - Compute Graph Optimization
  - Tuning
  - Distributed Training
- Results
- Current and Future Work
- Availability
Motivation

- Utilize state of the art MAGMA LA framework
- Provide a modular C++ Deep Learning Interface
- Support state of the art distributed training techniques
MAGMA

- Accelerated Linear Algebra on Heterogeneous Architectures

\[ g_n \left( U_{n-1} W_n + b_n \right) \]

\[ Y = A^T \left[ \left[ GgG^T \right] \odot \left[ B^T dB \right] \right] A \]

P100 1.19GHz
Peak: 4700 Gflops
MAGMA for Data Science

- Better performance than other leading LA packages in SVD
- Scalable FFTs for Convolutions

Strong scalability of 3D FFT on Summit (N = 1024)

Memory bound scalability peak:
- 78 Gflop/s per node
  - Assuming max bandwidth
    - $2 \times 2 \times 12.5 \text{ GB/s} = 50 \text{ GB/s}$
- Achieved performance is
  - $1223/32 \approx 38 \text{ Gflop/s per node}$
  - or ~25 GB/s (this is maximum if there is no duplexing)
MagmaDNN Overview

**Applications**
High-performance data analytics and machine learning for many-core CPUs and GPU accelerators

**MagmaDNN**
Scalable LA on new architectures
Data abstractions and APIs
Heterogeneous systems portability

**MAGMA Templates**
Tile algorithms
LAPACK++
BLAS++

SLATE
ScalAPACK API
MPI

Single Heterogeneous Node

Load Data
Preprocessing
Create/Load Model
Train Model
Export Model
Predict
Framework Overview

Memory Manager
Tensor
Operation & Graph
Optimizer
Model
size_t n_vals = 20;
// HOST, DEVICE, MANAGED, CUDA_MANAGED
memory_t mem_type = MANAGED;
device_t device_id = 0;
MemoryManager<float> mm(n_vals, mem_type, device_id);
Framework Overview

```cpp
Tensor<float> x({32,28,28}, {UNIFORM, {-1.0f,1.0f}}, DEVICE);
std::cout << x.get({1,3,0});
x.set({1,3,0}, 8.0f);
```
Framework Overview

Memory Manager  | Tensor  | Operation & Graph  | Optimizer  | Model

auto X = op::var<float>("X", {5,3},{UNIFORM});
auto W = op::var<float>("W", {6,5},{UNIFORM});
auto b = op::var<float>("b", {6,3},{UNIFORM});

auto transform = op::add( op::matmul(W, X), b );

Tensor<float> *output = transform->eval();
Framework Overview

- Memory Manager
- Tensor
- Operation & Graph
- Optimizer
- Model

```cpp
auto x = op::var<float>("x", NONE);
auto c = op::var<float>("c", {CONSTANT, -2.0f});

optimizer::GradientDescent opt(0.05);

opt.minimize( op::add(op::pow(x, 2), c), {x});
```

minimize $x^2 + c$

with respect to $x$
Tensor<float> data({60000, 28*28}, HOST);

io::read_csv_to_tensor(data, "mnist_data_set.csv");

Tensor<float> labels({60000, 10}, HOST);

io::read_csv_to_tensor(labels, "mnist_labels_set.csv");

/* create a vector of layers ... */

model::NeuralNetwork<float> model(layers_vector, optimizer::CROSS_ENTROPY, optimizer::ADAM,
{batch_size, n_epochs, learning_rate});

model.fit(data, labels, params_out, verbose);
Compute Graph Optimization

- Mixed Precision Training
- Make use of Volta Tensor Cores

\[
\sigma
\]

\[
+ \quad \text{gemm} \quad + 
\]

\[
W(L) \quad H(L) \quad b(L)
\]

\[
\text{FP32 to FP16} \
\text{FP32 to FP16}
\]

\[
W(L) \quad H(L)
\]
Compute Graph Optimization (cont.)

- Fused Operations

![Diagram showing fused operations]
Compute Graph Optimization

- Combined

\[
\sigma(W(L) \cdot H(L) + b(L)) \rightarrow \text{BIAS AND ACTIVATE}
\]

\[
\text{gemm (out FP32)} \rightarrow \text{FP32 to FP16}
\]

\[
\text{gemm (out FP32)} \rightarrow \text{BIAS AND ACTIVATE}
\]

\[
\text{FP32 to FP16} \rightarrow \text{FP32 to FP16}
\]
Tuning

- MagmaDNN tunes tensor and deep learning kernels

- Magma tunes matrix algebra kernels
Distributed Training

Data Parallelism

Model Parallelism
Distributed Training

- ASGD
- AllReduce
- Ring Reduce
Results

- Best performance across a single node
- Best scaling with network size

MLP Time Comparison
Profiled on Nvidia 1050 Ti

![Graph showing time comparison between MagmaDNN, TensorFlow, Theano CPU, and Theano GPU across layers.](image-url)
Summary

- Accelerated single node training times
- Modern C++ interface
- Support for distributed training
Current and Future Work

- Competitive distributed ResNet-50 training time
- Move to new C++ standard
- More “Bells and Whistles”
- Development
- Python Interface
Acknowledgements

- Joint Institute for Computer Science (JICS)
- National Institute for Computational Sciences (NICS)
- University of Tennessee, Knoxville (UTK)
- NSF Award #1659502
- Extreme Science and Engineering Discovery Environment (XSEDE)
- BP High Performance Computing
Availability

Development: https://github.com/MagmaDNN/magmadnn

Releases: https://bitbucket.org/icl/magmadnn

Tutorials: https://github.com/MagmaDNN/magmadnn/tree/master/docs/tutorials

Contact: Daniel Nichols (danielnichols@utk.edu)