MagmaDNN Core Development and Applications

Daniel Nichols
University of Tennessee

Sedrick Keh
Hong Kong University of Science and Technology

Kam Fai Chan
Chinese University of Hong Kong

Mentors: Kwai Wong, Stan Tomov, Junqi Yin, Ying Wai Li, Markus Eisenbach, Max Lupo Pasini
What led to the recent emergence of deep learning?

- More available data
- Improved computing capabilities
Existing Deep Learning Frameworks
What else can we add to this space?

Scalability
What happens as we increase the number of data / layers / parameters?

Flexibility
What if I want to add my own feature / model / optimizer / loss function?

Speed and Efficiency
How do we ensure faster training times?
MagmaDNN
What is MagmaDNN?

MagmaDNN is a modularized deep learning framework that is optimized for parallel computation and distributed training on GPUs. It is built around the MAGMA linear algebra library and CuDNN to accelerate some deep learning-related computations.
Current Features

- Basic neural network features
  - Forward and backward propagation
- **CNN** support
  - Convolution, Pooling
  - Dropout, Batch Normalization
- Basic **Graph** convolution
- Various **Optimizers**
  - SGD, Adam, AdaGrad, RMSProp
MagmaDNN Parallelism
MNIST MLP Time Comparison
Profiled on Nvidia 1050 Ti

Tensor Reductions in MagmaDNN
Data collected on P100 GPU
In Progress

- Distributed Training
- Further optimization
  - Better compute graph optimization
  - Better memory management
- RNN/LSTM support
- Transfer learning
- Large model (e.g. ResNet)
- Hyperparameter optimization
- User-friendly interface
MagmaDNN Applications:
Computational Materials Science
Ising Model on Lattice Structure

- **Particles** (e.g. dipoles) stack in certain structure
- Each particle has a spin, upward/downward
  - Denoted as +1 or -1
- Each particle interacts only with neighbours
  - Interaction strength depends on location and spins
- Physical properties determined by their interactions
  - e.g. heat capacity, magnetic susceptibility
- Similar model also exists in neuroscience

Simple illustration of Ising model on 2D plane
Problem: Computing the Hamiltonian

- An important quantity, Hamiltonian, is related to local configurations

\[ H = - \sum_{(i,j)} J_{ij} \sigma_i \sigma_j \]

- Need to compute for large number of configurations
- Structure can be highly irregular (e.g. different neighbourhoods)
- Good and basic example for problems in material science
Idea

- Use **Graph Convolutional Network (GCN)** on lattice
  - Can capture local features on irregular graph
- For benchmark, **Compare** with usual **CNN** on 2D grid
- Implemented in **MagmaDNN**
  - Customizable, Efficient, Open source
Comparing CNN and GCN

1. **Generate samples**: 8x8 2D planar grid, periodic boundary, uniform interaction strength
2. **Compute Hamiltonian** of samples directly
3. **Use CNN** to learn the Hamiltonian
4. Use same model but **replace CNN with GCN**
## Result

Trained on 1.76M training samples, 20 epochs, 315k testing samples

<table>
<thead>
<tr>
<th></th>
<th>Training</th>
<th></th>
<th>Testing</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>RMSE</td>
<td>MAE</td>
<td>RMSE</td>
</tr>
<tr>
<td>CNN</td>
<td>2.98</td>
<td>3.93</td>
<td>2.98</td>
<td>3.93</td>
</tr>
<tr>
<td>GCN</td>
<td>5.77</td>
<td>7.75</td>
<td>5.77</td>
<td>7.77</td>
</tr>
</tbody>
</table>

MAE: Mean Absolute Error, RMSE: Root-Mean-Square Error

Converge slower than CNN, capable to handle general graphs
→ Not a bad substitute for regular convolution on irregular structures

All these results are obtained with MagmaDNN
MagmaDNN Applications: Computational Microscopy
Computational Microscopy

Use of numerical approaches to measure and analyze images on a very small scale.
Crystallographic Space Groups

● Every material has a corresponding crystal structure.

● There are 230 possible symmetric space groups.
Crystallographic Space Groups

Crystal Lattice Structure

Convergent Beam Electron Diffraction

CBED Images
We use deep learning!

Input: (3, 512, 512)
Conv1: (2,2) filter, (1,1) stride, 64 channels
MaxPool1: (2,2) stride
Conv2: (2,2) filter, (1,1) stride, 16 channels
MaxPool2: (2,2) stride
Flatten
Dense (256)
Dense (231)
Output

Accuracy: 11%
We use deep learning!

Accuracy: 11%
Consider:

Accuracy: 10%

There is a **degradation** problem.
Consider:

\[ y = F(x) \quad \text{or} \quad y = F(x) + x \]
ResNet\textsuperscript{[1]}

Shortcut connections

How can MagmaDNN be used in this task?

- Very flexible, easy to build custom models
- 2D Convolution, Batch Normalization, Pooling, Dropout
- Shortcut connections can be implemented using the addition operation.
How can **MagmaDNN** be used in this task?

**Accelerated GPU Computations**
Use MAGMA for linear algebra routines, CuDNN for operations like convolutions

**Dynamic Memory Manager**
Define its own custom memory manager similar to CUDA’s

**Data and Model Parallelism**
Support MPI capabilities
ResNet 18

Accuracy: 16%
Challenges:

- Many output classes (230)
- Data imbalance
MagmaDNN scales well

On ResNet 18 benchmark (on 1050 GPU card):

- **TensorFlow**: 726 seconds per epoch
- **MagmaDNN**: 195 seconds per epoch
Thank you!

MagmaDNN v1.0 is available at
https://bitbucket.org/icl/magmadnn/