Implementing MAGMA Tensor Interfaces

Emilio Villasana, Clifford Li, & Kristina Wilson
Mentors: Dr. Kwai Wong and Dr. Stan Tomov

Background

MAGMA - Matrix algebra on GPU and Multicore Architectures

- performs calculations on GPUs and multicore CPUs for more efficient computations.
- supports matrix-matrix, matrix-vector, vector-vector, scalar-vector, and scalar-matrix operations
  - i.e. **SGEMM- S**ingle Precision **G**eneral **M**atrix **M**atrix Multiplication

Tensors

- Tensors are mathematical data containers that represent multi-dimensional arrays of numerical values
  - Have important applications in machine learning, electro dynamics, and other fields.
  - 0-rank tensor (**scalar**) α
  - 1-rank (**vector**) $\begin{bmatrix}\alpha & \beta & \gamma & \delta\end{bmatrix}$
  - 2-rank (**matrix**) $\begin{bmatrix}\alpha & \beta \\ \gamma & \delta\end{bmatrix}$
  - 3-rank $\begin{bmatrix}\begin{bmatrix}\alpha \\ \beta\end{bmatrix} & \begin{bmatrix}\hat\alpha \\ \hat\beta\end{bmatrix} & \begin{bmatrix}\alpha' \\ \beta'\end{bmatrix}\end{bmatrix}$
  - 4-rank $\begin{bmatrix}\begin{bmatrix}\alpha & \beta \\ \gamma & \delta\end{bmatrix} & \begin{bmatrix}\hat\alpha & \hat\beta \\ \hat\gamma & \hat\delta\end{bmatrix} \\ \begin{bmatrix}\alpha' & \beta' \\ \gamma' & \delta'\end{bmatrix} & \begin{bmatrix}\tilde\alpha & \tilde\beta \\ \tilde\gamma & \tilde\delta\end{bmatrix}\end{bmatrix}$

$$A = \begin{bmatrix}\begin{bmatrix}1.5 & 2.1 & 5.4 & 9.1 \\ 4.2 & 7.2 & 3.2 & 8.0\end{bmatrix} \\ \begin{bmatrix}2.3 & 4.1 & 7.4 & 9.1 \\ 4.6 & 5.2 & 6.7 & 7.0\end{bmatrix} \\ \begin{bmatrix}4.5 & 3.4 & 2.3 & 6.7 \\ 7.9 & 4.2 & 5.1 & 7.9\end{bmatrix}\end{bmatrix} \quad B = \begin{bmatrix}\begin{bmatrix}4.3 \\ 6.7 \\ 1.5 \\ 9.4\end{bmatrix} \\ \begin{bmatrix}4.2 \\ 4.9 \\ 6.8 \\ 3.9\end{bmatrix} \\ \begin{bmatrix}7.4 \\ 1.5 \\ 9.6 \\ 0.5\end{bmatrix}\end{bmatrix} \quad C = \begin{bmatrix}\begin{bmatrix}1.5 & 2.1 & 5.4 & 9.1 \\ 4.2 & 7.2 & 3.2 & 8.0\end{bmatrix}\begin{bmatrix}4.3 \\ 6.7 \\ 1.5 \\ 9.4\end{bmatrix} \\ \begin{bmatrix}2.3 & 4.1 & 7.4 & 9.1 \\ 4.6 & 5.2 & 6.7 & 7.0\end{bmatrix}\begin{bmatrix}4.2 \\ 4.9 \\ 6.8 \\ 3.9\end{bmatrix} \\ \begin{bmatrix}4.5 & 3.4 & 2.3 & 6.7 \\ 7.9 & 4.2 & 5.1 & 7.9\end{bmatrix}\begin{bmatrix}7.4 \\ 1.5 \\ 9.6 \\ 0.5\end{bmatrix}\end{bmatrix} = \begin{bmatrix}\begin{bmatrix}114.16 \\ 146.30\end{bmatrix} \\ \begin{bmatrix}115.56 \\ 117.66\end{bmatrix} \\ \begin{bmatrix}63.83 \\ 117.67\end{bmatrix}\end{bmatrix}$$

$A = 3 \times 2 \times 4$ $\quad B = 3 \times 4 \times 1$ $\quad C = 3 \times 2 \times 1$

Figure 1: Example of 3-D tensor multiplication

Tensor Multiplication

Non-batched *n*-D tensor multiplication

- Loop through the batches of *A* and *B* and perform *(n-1)*-D multiplication on the batches.

Batched 3D tensor multiplication

- Initialize 2D array (double pointer) and point to single arrays. Each of 1D arrays stores 2D matrix. Allocate memory and compute 2D array without loops.

stranspose transposes a 3-D tensor with a given permutation.

- *A: m x n x k*, and perm = [2,0,1], then AT: *k x m x n.*

A: 2 x 3 x 4
A = [[1 2 3 4;
5 6 7 8;
9 10 11 12],
[13 14 15 16;
17 18 19 20;
21 22 23 24]]

perm = [0, 2, 1]
AT: 2 x 4 x 3
AT = [[1 5 9; [13 17 21;
2 6 10; 14 18 22;
3 7 11; 15 19 23;
4 8 12], 16 20 24]]

perm = [2, 1, 0]
AT: 4 x 3 x 2
AT = [[1 13; [2 14; [3 15; [4 16;
5 17; 6 18; 7 19; 8 20;
9 21], 10 22], 11 23], 12 24]]

Figure 3: Example of 3-D transposition

SGETT

Single precision **Ge**neral **T**ensor **T**ensor Multiplication

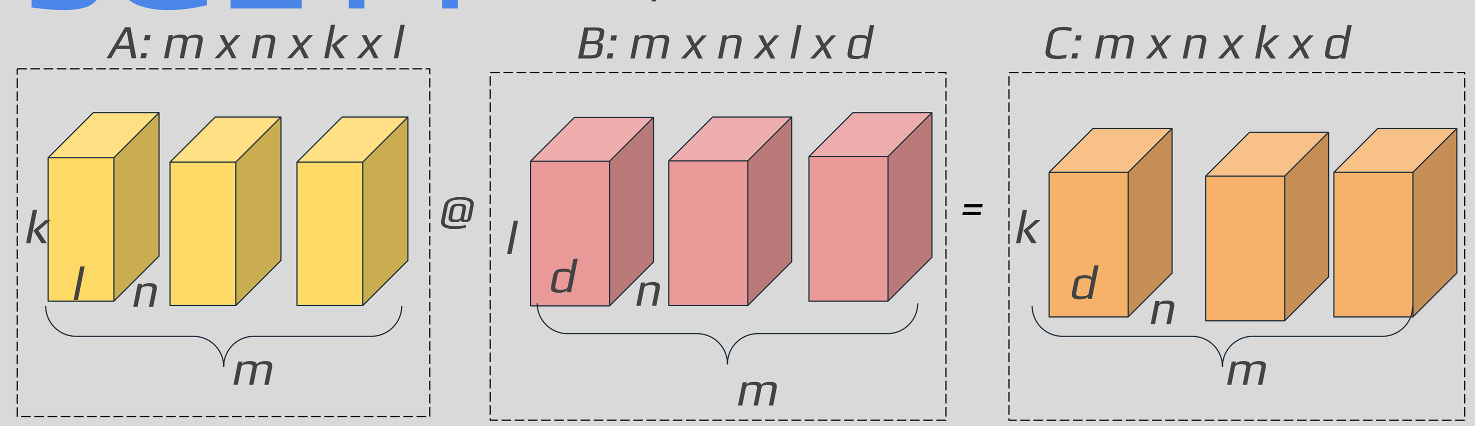
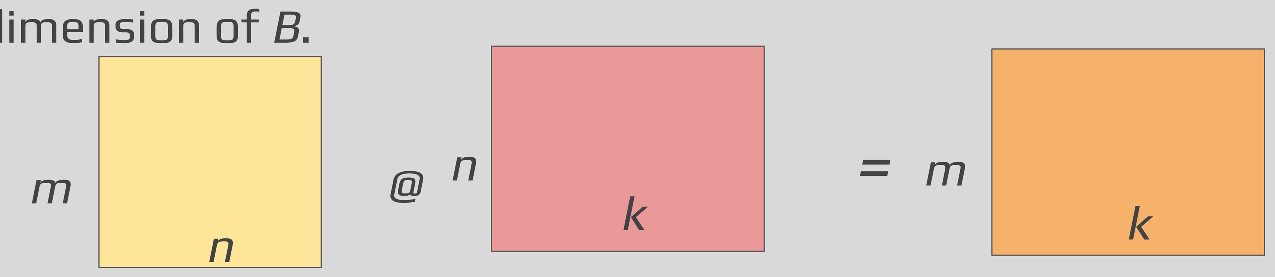


Figure 4: Visualization of 4-D tensor multiplication

In general, we contract the last dimension of *A* with the second to last dimension of *B*.



Results

We evaluated the performance of MAGMA and TensorFlow. Vertical axes in both figures below represent the time elapsed, while horizontal axes represent the tensor size. Regarding the performance test shown in Figure 5, TensorFlow performs the best when tensor size is below 600. It becomes the worst afterwards. While MAGMA SGEMM batched function is more efficient in larger tensor sizes.

Figure 6 indicates the better performance of MAGMA SGETT when tensor dimension becomes higher.

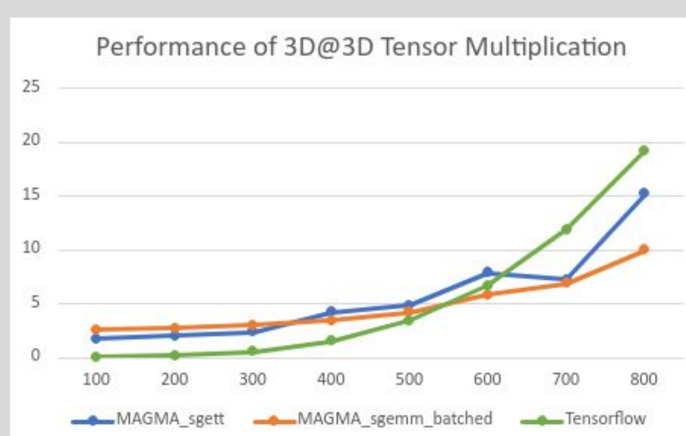


Figure 5: Performance of 3D@3D Tensor Multiplication

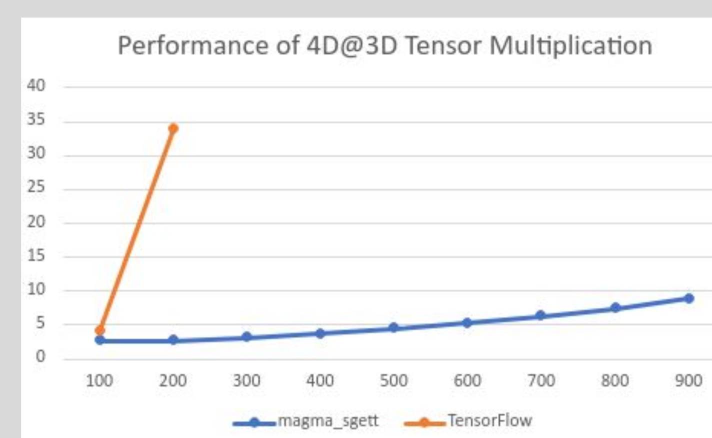


Figure 6: Performance of 4D@3D Tensor Multiplication

MagmaDNN

MAGMADNN

- Neural Network Library
  - Native Tensor Capabilities
  - MAGMA computational backend
  - Modularized
- Efficient Machine Learning Framework
- C++

SWIG

- Wrapper Software
- Translates Coding Languages

PyMagmaDNN

- Python interface for MagmaDNN
- Akin to TensorFlow or PyTorch
- Tensor creation and manipulation implemented

Conclusion and Future Direction

MAGMA

- SGETT provides a potential of using GPU to compute higher dimension of tensors efficiently, particularly in larger tensor size. We plan to evaluate SGETT performance on 5D tensor multiplication, and to find out feasibility of 4D tensor computation in batched approach.
- The transpose routine can be expanded upon (add 4D, 5D, etc), and the transpose option can be added as a SGETT parameter.
- Other contraction methods can be implemented in SGETT.

PyMagmaDNN

- As a way to increase accessibility of MagmaDNN and to provide those in the ML community another resource, continual development and the completion of PyMagmaDNN is a high priority.
- Next steps include implementation of MagmaDNN's layer framework, model evaluation, and Tensor-Tensor operations.

We would like to thank UTK, Davidson College, CityU HK, and NSF for their support of our project, and we would like to thank Dr. Wong and Dr. Tomov for their help and guidance with our work.