Outline

- CUDA
- MAGMA
- MagmaDNN
- Tensors
- Operations
  - Compute trees
- Layers
  - LSTM
Vector Addition

```c
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
    ...
    // Kernel invocation with N threads
    VecAdd<<<1, N>>>(A, B, C);
    ...
}
Layer-based model building
Why LSTM?

\[0.9 \times 0.9 \times 0.9 \times 0.9 \ldots = 0; \quad 1.1 \times 1.1 \times 1.1 \ldots = \infty\]

The Vanishing Gradient Problem
What is LSTM?

- Long Short-Term Memory
- Recurrent Neural Network
LSTM Uses
\[
\begin{align*}
  i_t &= \sigma(x_t U^i + h_{t-1} W^i) \\
  f_t &= \sigma(x_t U^f + h_{t-1} W^f) \\
  o_t &= \sigma(x_t U^o + h_{t-1} W^o) \\
  \tilde{C}_t &= \tanh(x_t U^g + h_{t-1} W^g) \\
  C_t &= \sigma(f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t) \\
  h_t &= \tanh(C_t) \cdot o_t
\end{align*}
\]
LSTM Implementation Requirements

- Forward Pass
- Backward Pass
First Implementation

- Combine pre-existing operations with new ones
- CPU and GPU
- Forward and Backward method
  - Each individual operations’ eval and gradient
Needed Operations

● Currently implemented
  ○ Sigmoid
  ○ Tanh
  ○ Matrix Multiplication
  ○ Matrix Addition
  ○ Element-wise Product

● Not Implemented
  ○ Slice
  ○ Concatenation
Slice and Concat

- Slice
- Concat
Slice and Concat Testing

- Both implemented on CPU and GPU
- Forward and Backward tested with examples on both
  - Slice/Concatenate passed in tensor and compare output
  - Pass in upstream gradient and compare downstream

```
Tensor size of [1, 2, 3, 4]
[
  [0.00000, 1.31530, 7.55605, 4.58650, 5.32767, 2.18959, 0.47945, 6.78865, 6.79296, 9.34693, 3.83502, 5.19416, 6.38965, 0.34572, 0.53462, 5.29780, 6.71149, 0.07698, 3.83416, 0.66842, 4.17486, 6.80773, 5.88977, 9.30437, 0.00000, 1.31530, 7.55605, 4.58650, 5.32767, 2.18959, 0.47945, 6.78865, 6.79296, 9.34693, 3.83502, 5.19416, 6.38965, 0.34572, 0.53462, 5.29780, 6.71149, 0.07698, 3.83416, 0.66842, 4.17486, 6.80773, 5.88977, 9.30437, 0.00000, 1.31530, 7.55605, 4.58650, 5.32767, 2.18959, 0.47945, 6.78865, 6.79296, 9.34693, 3.83502, 5.19416, 6.38965, 0.34572, 0.53462, 5.29780, 6.71149, 0.07698, 3.83416, 0.66842, 4.17486, 6.80773, 5.88977, 9.30437, 0.00000, 1.31530, 7.55605, 4.58650, 5.32767, 2.18959, 0.47945, 6.78865, 6.79296, 9.34693, 3.83502, 5.19416, 6.38965, 0.34572, 0.53462, 5.29780, 6.71149, 0.07698, 3.83416, 0.66842, 4.17486, 6.80773, 5.88977, 9.30437, 0.00000, 1.31530, 7.55605, 4.58650, 5.32767, 2.18959, 0.47945, 6.78865, 6.79296, 9.34693, 3.83502, 5.19416, 6.38965, 0.34572, 0.53462, 5.29780, 6.71149, 0.07698, 3.83416, 0.66842, 4.17486, 6.80773, 5.88977, 9.30437, 0.00000, 1.31530, 7.55605, 4.58650, 5.32767, 2.18959, 0.47945, 6.78865, 6.79296, 9.34693, 3.83502, 5.19416, 6.38965, 0.34572, 0.53462, 5.29780, 6.71149, 0.07698, 3.83416, 0.66842, 4.17486, 6.80773, 5.88977, 9.30437, 0.00000, 1.31530, 7.55605, 4.58650, 5.32767, 2.18959, 0.47945, 6.78865, 6.79296, 9.34693, 3.83502, 5.19416, 6.38965, 0.34572, 0.53462, 5.29780, 6.71149, 0.07698, 3.83416, 0.66842, 4.17486, 6.80773, 5.88977, 9.30437]
```

Below is the given tensor sliced about axis 1 and by index 2 on GPU

```
Tensor size of [1, 1, 2, 4]
[
```
Problems in Development cont.

Add these two lines:

```cpp
for (unsigned int r = x_idx; r < n_rows; r += x_stride) {
    for (unsigned int c = y_idx; c < n_cols; c += y_stride) {
        out[r * n_cols + c] = (axes == 1) ? grad[r] : grad[c];
    }
}
```

We had to swap these two.

(the order shown in the image is correct)
First Implementation Analysis

Successes
- Calculations
- Training on small data

Shortcomings
- Adding operation overhead takes too much time for time sequences >10
  - Perhaps exponential growth
- Compute tree eval issues
- Can not train on large data
Operation Overhead

```c++
user1@REU1902-HP-Z800-Workstation: ~/magma/magmadnn/src/compute/reducesum

56 // TODO: With this uncommented, the simplelstm constructs exponentially slower at larger
57 // sequence lengths (~lengths > 9)
58 // But with it commented, the GPU cannot work with any layer
59
60 #if defined(MAGMADNN_HAVE_CUDA)
61   // Use default stream for CUDA kernels
62   // this->cudnn_ = nullptr;
63   this->set_cudnn_handle(0);
64
65   this->cudnn_handle(magmadnn::internal::MAGMADNN_SETTINGS->cudnn_handle);
66
67   this->set_cublas_handle(magmadnn::internal::MAGMADNN_SETTINGS->cublas_handle);
68
69   this->set_async(false);
70 #endif
71 ```
Compute Tree Reevaluation

$O(5^n)!!$

No. times matmul is evaluated

Time step

No. times matmul is evaluated

1 2 3 4 5 6
Second Implementation

- GPU only
- Use a single LSTM Operation
  - Cuda code from previous operations
  - New Cuda code for intermediate calculations
- Forward pass methodology
  - Kernels used
  - MAGMA sgemm and dgemm
  - Value caching
- Backward pass methodology
  - Kernels used
  - MAGMA sgemm and dgemm
  - Use of store values
Second Implementation Analysis

Resolved Issues
○ Operation Overhead
○ Eval issues

Ongoing Problems
○ Large data
○ Runs out of memory
Testing

- Compared calculations against python script
  - Script verified against Dr. Wong test case
  - With and without return sequences
- Taught to predict zeros
input -> lstm(5, return_sequences = true) -> lstm(1, false) -> output
random initialization, all target values set to zero, 200 epochs, 300 input/output pairs
Future LSTM Work

- Understand the problem with the first implementation
  - i.e. why is it get so slow? what makes it evaluate exponentially more operations?
- Fix memory issue with second implementation
- Add support for dynamically changing the input/output sequence length
  - This would likely require reworking the NeuralNetwork class
Future MagmaDNN Work

ADD ERROR MESSAGES

segmentation fault.
(Core dumped)

Cannot Rename
A file with the same name already exists. Specify another name.

Cannot Rename
Specify another name.

The message contains the problem, cause of error and solution.

The cause of the error is missing in this error message.
Future MagmaDNN Work cont.

- Abstract methods do not have clear descriptions of their responsibilities.