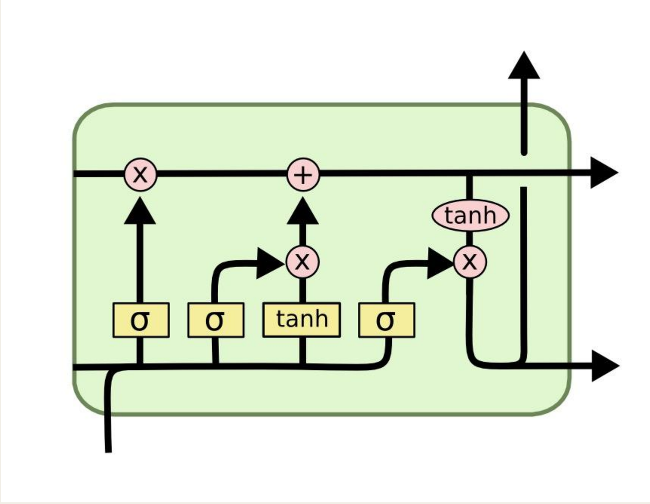
MagmaDNN LSTM Implementation

Students: Pierluigi Cambie-Fabris (UTK), Joshua Zingale (SDSU)
Mentors: Kwai Wong (UTK)

Background

MagmaDNN is a deep learning engine built atop the computational framework Magma. Until now, MagmaDNN has lacked features which are necessary for learning from and predicting sequences of data. A recurrent neural network (RNN) is a neural network that takes sequences of data as input and uses each step of the sequence to determine the next output. The long short-term memory network (LSTM) is an RNN that addresses the vanishing gradient problem prevalent in RNN architectures. The vanishing gradient problem, in the context of RNNs, occurs when time-consecutive gradient calculations result in the gradient value approaching zero. The LSTM addresses this by having a forget gate that links early time steps to later time steps. The equations defining an LSTM cell are given by the following, where for sequence step t, x_t is the input, c_t and h_t are internal states, i_t and f_t are hidden weights, and o_t is the output.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)$$
$$h_t = o_t \circ \sigma_h(c_t)$$



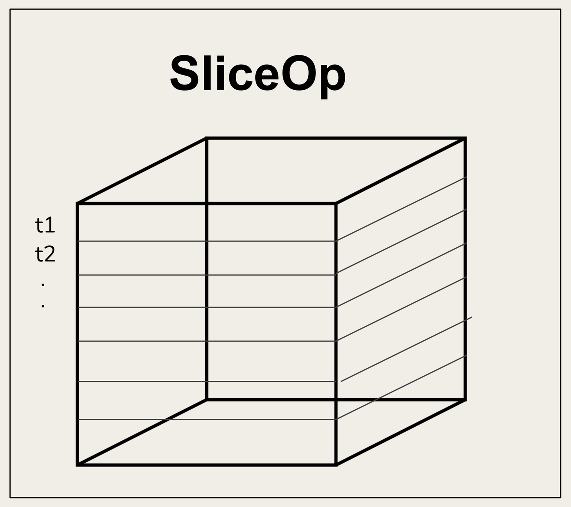
Architecture

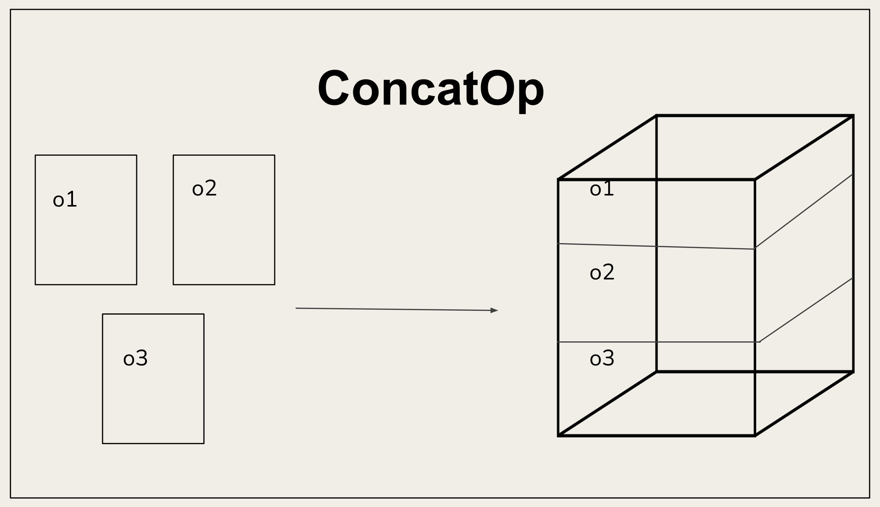
MagmaDNN is a deep learning framework written in C++ with the intent easily integrating with Magma, a linear algebra package. MagmaDNN has four main levels of abstraction: Tensors, Operations, Layers, Models. Tensors are multidimensional arrays which abstract the process of memory allocation and management on both main memory and GPU memory. Operations store input data as Tensors and compute outputs, which themselves are Tensors; Operations can be superimposed to create complex compute trees, which allow for gradients to be automatically computed. Comprised of Operations, a Layer receives input data in the form of Operations either directly or from another Layer; using its comprising Operations, a layer compute an output which can be either accessed directly or passed to another layer for further computation. Finally, Models are built using layers: these can be trained on data to learn accurate mathematical models for natural and synthetic phenomena.

As discussed in implementation, the LSTM is implemented as a Layer in MagmaDNN, requiring use of the above abstractions.

Implementation

The LSTM layer implementation needed two operations: Slice and Concat (concatenation). Slice is used to cut three-dimensional input data, Tensors, along the time axis, returning two-dimensional data. Concat is used to concatenate the many two-dimensional outputs of the LSTM into one three-dimensional output; this is used to feed the outputs of an LSTM layer into the inputs of another Layer that expects three-dimensional inputs. Each of these operations is implemented on CPU and GPU. The CPU implementations perform calculations in C++. The GPU implementations instead use CUDA kernels to parallelize computation of outputs. These operations are paired with pre-existing operations for sigmoid, tanh, matrix multiplication, element-wise multiplication, and addition to create the LSTM layer.





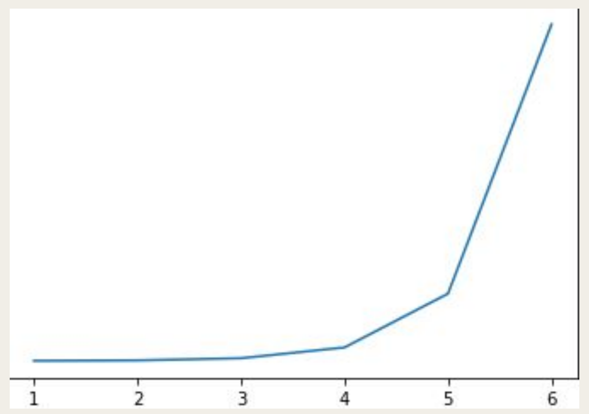
Testing

The current version of the LSTM layer performs the correct calculations expected from the LSTM equations.
Below we have an example of such testing.

```
 Test with return_sequences = false

vvvvvvv MODEL OUTPUT vvvvvvv
The output should be about 0.86922
Tensor size of {1, 1, 1, 1}
{
  {
    | 0.86923, |
  }
}

 Test with return_sequences = true

vvvvvvv MODEL OUTPUTS vvvvvvv
The outputs should be about 0.719607, 0.86922
Tensor size of {1, 1, 2, 1}
{
  {
    | 0.71961, |
    | 0.86923, |
  }
}
```

Performance

The current LSTM performs rather poorly with a time complexity of about O(5^n) with n being the number of time sequences. As a result of this, the LSTM struggles to construct any test with more than 10 time steps. This issue is in part due to the overhead that is necessary for the many Operations the LSTM layer uses.



Analysis

The LSTM implemented into MagmaDNN currently works on CPU and GPU for both forward and backward propagation but only for small time sequences since, as the sequence length increases, the layer becomes exponentially slower. Also, the input and output sequence lengths must be constant. This requirement of constant length prevents this LSTM from acting in many of the use cases typical of LSTMs, such as being trained on one input length and then later accept input sequences of any length.

Future Work

As mentioned in the analysis, this LSTM does not support arbitrarily sized input sequences. For this to be implemented, the current compute-tree architecture within MagmaDNN must be modified to be dynamic. Also, to circumvent the computational inefficiency of Operation overhead mentioned in performance, we have begun work on an LSTM implementation which does not utilize external Operations; that is, all computations are to be implemented from scratch, both forward and backward.

Acknowledgments

Ferlay, J., Héry, C., Autier, P. & Sankaranarayanan, R. (2010). Global burden of breast cancer. Breast cancer epidemiology, 1–19, Springer.
Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. European conference on computer vision (pp. 21-37). Springer, Cham.
Ting, F. F., Tan, Y. J., & Sim, K. S. (2019). Convolutional neural network improvement for breast cancer classification. Expert Systems with Applications, 120, 103-115.

THE UNIVERSITY OF TENNESSEE KNOXVILLE