Final Project One Page Report

Current Strategy

In the current code, the focus is on the implementation of General Matrix-Matrix Multiplication (GEMM) on GPU and parallelization of the code by MPI so the code can run on several nodes.

In the first version of GEMM, a naïve approach using 1D blocks was implemented. If the GEMM operation is expressed as $D = \alpha * A * B + \beta * C$, then each thread of the 1D blocks is responsible for the computation of one element in matrix D. It is found that the speed of this 1D block approach for computing GEMM is much slower than the GEMM by using CPU with the Armadillo library. This is because in multiplying A by B, we have to access elements along rows of A and columns of B. However, since matrices are in column-major order, the reads in A is not coalesced and each row/column of B is large so there is no good reuse of cache. Therefore, to better reuse data loaded into cache, an improved approach of using 2D blocks is currently implemented. In the 2D block approach, each thread is still responsible for one thread but the speed of GEMM increases a lot because of the reuse of data loaded into cache in each block.

For the MPI implementation, the input X is first scattered to each node evenly before the iteration over the sub-batches. At the end of each iteration of each process, ΔW and Δb computed from different nodes are summed up by using MPI_Allreduce(). Then, each node updates the network parameters by using the reduced ΔW and Δb.

Intended Optimizations

A better GEMM should be implemented instead of using the naïve 2D block approach because that approach uses global memory that resides in device memory which is slower than shared memory in accessing data. In the shared memory approach, 2D blocks are used and each block compute a sub-matrix of the output matrix D. A loop is used to loop over sub-matrices of matrices A and B to compute the sub-matrix of D. In each loop, sub-blocks of matrices A and B are loaded into shared memory and then each thread in the block loops along the sub-blocks of A and B to compute one element of D.

To further optimize the code, the data sent over the PCI express bus should be minimized. So far, only GEMM part of feedforward() and backprop() is changed to the GPU version and data such as matrices X and W is sent to GPU repeatedly between successive GEMM processes. This kind of communication should be minimized by implementing the feedforward() and backprop() on GPU entirely so data can be reused for GEMM and other operations like sigmoid and softmax on GPU.