**ECE 4822: Engineering Computation IV**

**Homework No. 10: Neural Networks and Dense Graphs**

**Goal:** Appreciate how to implement computations on large-scale directed graphs.

**Description:**

You must do your own coding for this assignment. You cannot use one of the deep learning packages.

In this location:

*https://www.isip.piconepress.com/courses/temple/unvs\_0822/lectures/current/0001/*

there is a brief introduction to neural networks and deep learning. On slide 4 of this lecture:

*https://www.isip.piconepress.com/courses/temple/ece\_8527/lectures/2020\_00\_spring/lecture\_34.pptx*

there is a description of how the computations work for a simple feed-forward network. In this assignment, you will implement a large three-level feed-forward network.

Each layer in your network will be fully connected to the previous layer. The weights for each arc will be assigned randomly with the constraint that the sum of the weights entering a node is one. The inputs will be random values in the range [0,1].

Create a network that has three levels ­­– an input layer, a hidden layer and an output layer. Use a sigmoid distribution for the nonlinearity with a slope of 0.5 (do not hardcode this – make it a #define so it can be easily changed). Allocate 10,000 nodes per level.

Your program should read the network weights from a binary file (number of weights is $10,000 ×10,000×3$ 32-bit floats). It should also read the input data, which are $10,000 x 1$ vectors, from a second binary file (10,000 32-bit floats per epoch for 1,000 epochs). Process all the data in this file through your network. Do not hardcode the number of input vectors – make sure you loop over the input file until all the data has been processed.

Implement your solution in C/C++ first, and then implement the same code using parallel operations on a single GPU. Next, implement this network efficiently using 4 GPUs.

Compare the timing in a table (compute time should be linear with the number of input vectors ignoring startup costs). Discuss the pros and cons of a single GPU vs. multiple GPU implementation for a network of 100 million nodes per layer (which won’t fit inside the memory of a single GPU) based on your findings.

You might find this article useful:

*http://luniak.io/cuda-neural-network-implementation-part-1/*

for understanding how to code this.