CDA5125 Programming Assignment No. 1: Deep Neural Network for Hand-Written Digit Recognition

(Due: 02/07/2022 for Part 1 and 02/14/2022 for Part 2)

Purpose:

- Experience with a common workload on today's high performance computing (HPC) systems - training deep neural networks.
- Practice loop optimization techniques and apply the techniques to optimize the implementation of deep neural networks.

Statement of work:

This is a group assignment. Each group can have 2 people. This assignment has two parts. In the first part, you will develop a **correct** vanilla implementation of a deep neural network **from scratch** in C or C++ that can be trained to recognize hand-written digit images. You are not allowed to use any AI/machine learning related library or use any language other than C/C++. Such programs will get 0 point for the assignment. Specifically, the following paper describes a number of multi-level perceptrons (MLP) that yield very high accuracies on the MNIST handwritten digits benchmark:

Dan Claudiu Ciresan, et al, "Deep Big Simple Neural Nets Excel on Hand-written Digit Recognition," arXiv:1003.0358, Mar, 2010.

You should read the paper, and select one of the neural networks described in Table 1 in the paper to implement in this project. You should test your program on the MNIST dataset, which can be obtained from https://deepai.org/dataset/mnist.

The neurons in the neural networks in the paper use a scaled hyperbolic tangent function as the activation function, so you will need to adapt the sigmoid activation function that we discussed in class and implement the scaled hyperbolic tangent function.

To demonstrate that your program is correct, you should train your network to recognize at least 4 digits in the MNIST dataset with a very high accuracy (e.g. more than 90% training accuracy, you should be able to get almost 100% as indicated in the paper). Like any machine learning project, this can only be accomplished with intensive debugging/configuration tuning and a long time of training (in my test, the network is reasonable good after around 20 hours training over 400 epochs to recognize 5 digits). Thus, to demonstrate that your program is correct is non-trivial. But you are required to do so -- all of these are parts of the assignment. Note that you want to closely match the configuration specified in the paper. Otherwise, your program may not be trained to reach high accuracy and will be considered as an incorrect program.

In the second part of the assignment, you will apply loop optimization techniques to improve the execution time of the program. *You are required to manipulate array access pattern, apply loop blocking/loop interchange and loop unrolling effectively at least once.* Your optimizations should result in a noticeable improvement in execution time for this program. Note that when

trying to improve the performance of any program, you must first identify which parts of the program dominate the execution time and then focus on optimizing those parts.

Due dates:

Part 1 is due on February 7, 11:59pm. Put all related source code, the makefile, and a README file in a tar file and submit the tar file. In the README file, you must describe how to run the program (e.g. where should the MNIST data be placed?) and whether and how the program achieves high accuracy in recognizing handwriting digits. What is described in the README file must be repeatable with your submitted files.

Part 2 is due on February 14, 11:59pm. Put all related source code, the makefile, and a README file in a tar file and submit the tar file. In the README file, you must describe (1) how to run the program and whether and how the optimized program achieves high accuracy (optimized code should be correct), and (2) what optimization techniques are applied to which parts of the code, and (3) the performance improvement with the optimizations (potentially with each optimization). Similarly, what is described in the README file must be repeatable with your submitted files.

Grading:

Part 1:

1. Submission has all components (all related source code, makefile, README file); the executable can be successfully produced with a ‘make’ command in the directory; a deep neural network for handwriting digit recognition with the MNIST dataset is built (30 points).
2. The README file describes the information as required (5 points).
3. Points in 3) can be obtained only after all points in 1) and 2) are obtained. One can follow the description in the README file to repeat the claims (15 points).

Part 2:

1. Submission has all components (all related source code, makefile, README file); the executable can be successfully produced with a ‘make’ command in the directory; an optimized deep neural network for handwriting digit recognition with the MNIST dataset is built (30 points).
2. The README file describes the information as required (5 points).
3. Points in 3) and 4) can be obtained only after all points in 1) and 2) are obtained. One can follow the description in the README file to repeat the claims (5 points).
4. All the required optimizations have been applied; and the improvement for the overall program execution is substantial (10 points).