Project Report

cuDNN implementation of CNN

Table of Contents

1. Objective

The main objective of this project is a distinct analysis of CNN implementation using CUDA and comparing the performance of a similar kind of code on a CPU.

2. Description

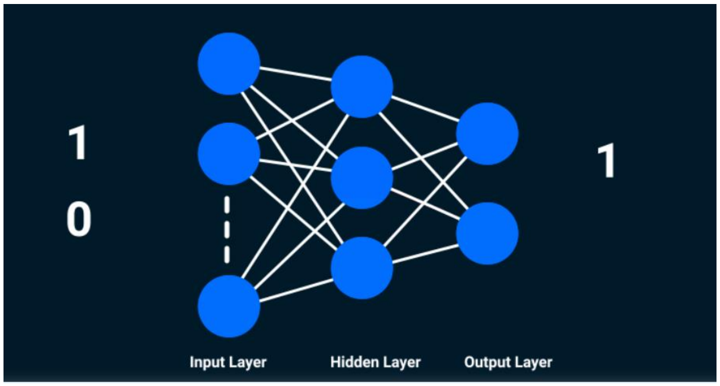
Convolutional Neural Network is a Deep Learning algorithm that takes in input as images and appoints weights and predispositions to significant variables to recognize the components of the image.
CNN utilizes an organization of neurons to foresee the image dependent on the trained data. The bigger the CNN organization the more exact it should accomplish including some different factors excessively, for example, input information size, and the streamlining agent being utilized.

CNN's performance relies on the way the network calculations are done. If they are done in parallel, the output is faster to achieve.
This is where cuDNN implementation arrives at the big picture.

This project is on cuDNN implementation on CNN and a comparison of accuracy rate and time taken on CPU vs GPU

3. Project Implementation

CNN involves three actions – Receiving input, Processing Information and Generating output are represented in the forms of layers as input, hidden and output.



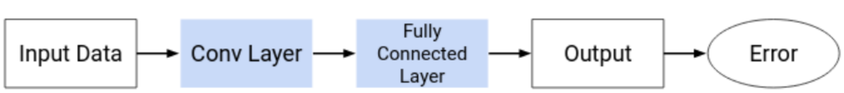
These individual units are called as neurons.
Neural Network training process involves 2 steps.

1. Forward Propagation - Images are taken care of into the input layer as numbers. These mathematical qualities denote the intensity of the pixels in the image. The neurons in the hidden layers apply a couple of numerical procedure on these values. To play out these mathematical tasks, certain parameters are randomly initialized. After these mathematical operations at the hidden layer, the result is sent to the output layer which generates the final prediction values.

2. Backward Propagation- Once the output is generated, in the next step we need to compare the output with the actual value. Based on the final values and the distance from the actual error, the values of the parameter are updated. The forward propagation process is repeated using the updated parameter values and new parameter are generated.

   Convolution is often represented mathematically with an asterisk * sign. If we have an input image represented as X and a filter represented with f, then the expression would be: $Z= X*f$;
   These are the base of Neural Network Algorithm.

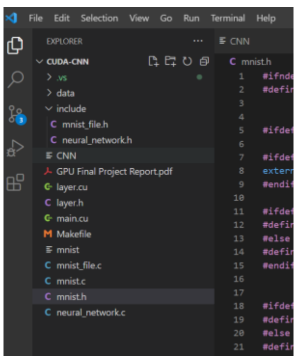
   Convolutional Neural Network (CNN) Architecture
   Complete Network can be broken into 2 parts-
   I. The Convolution layers- Extract features from the input.
   II. Fully connected (Dense) layers- Uses data from the convolution layer to generate output.



3. Project Structure

The architecture of the project is as follows-

- mnist .c (CPU implementation file)
- mnist_file.c (CPU implementation file)
- neural_network.c (CPU implementation file)
- layer.cu (GPU implementation file)
- layer.h (GPU implementation file)
- main.cu (GPU implementation file)
- mnist.h (GPU implementation file)
- /data (MNIST dataset)
- /include(mnist and neural network header files for CPU implementation)



5. Tools and Libraries to be used

cuDNN, Visual Studio Code, C, C++

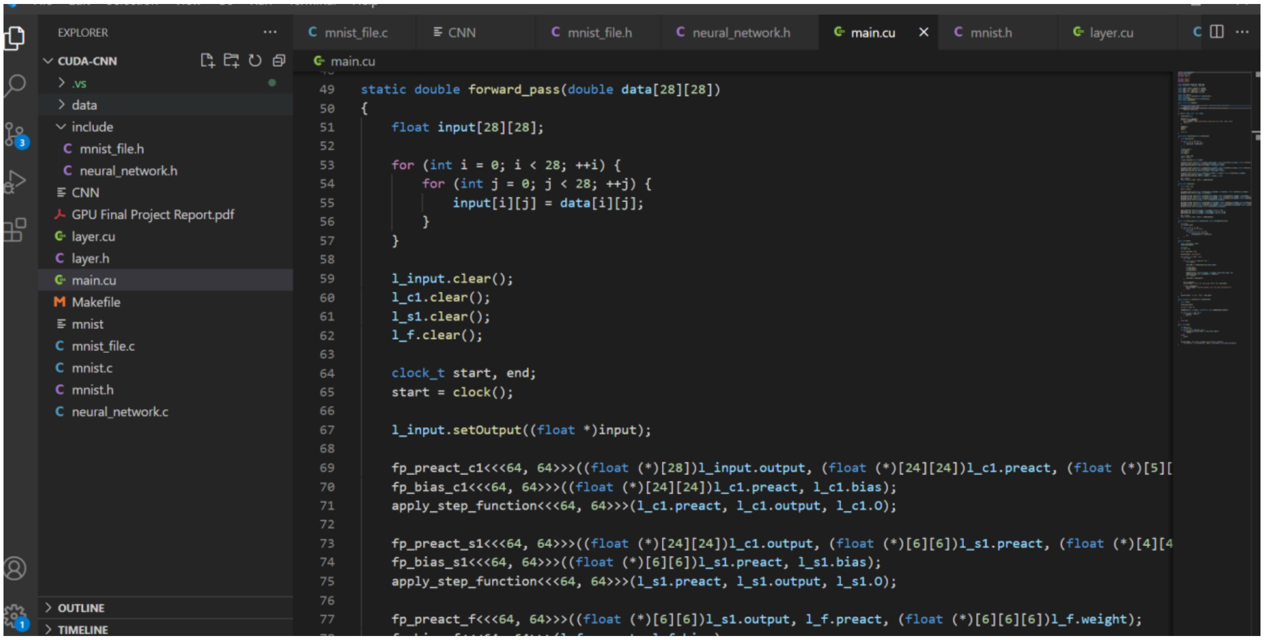
6. Project/Code Walkthrough

GPU gives an advantage of parallelizing the code, thus the parallel method is used by us to implement the forward propagation and backward propagation where the gradient calculations and weight/biases calculations are done in CUDA.

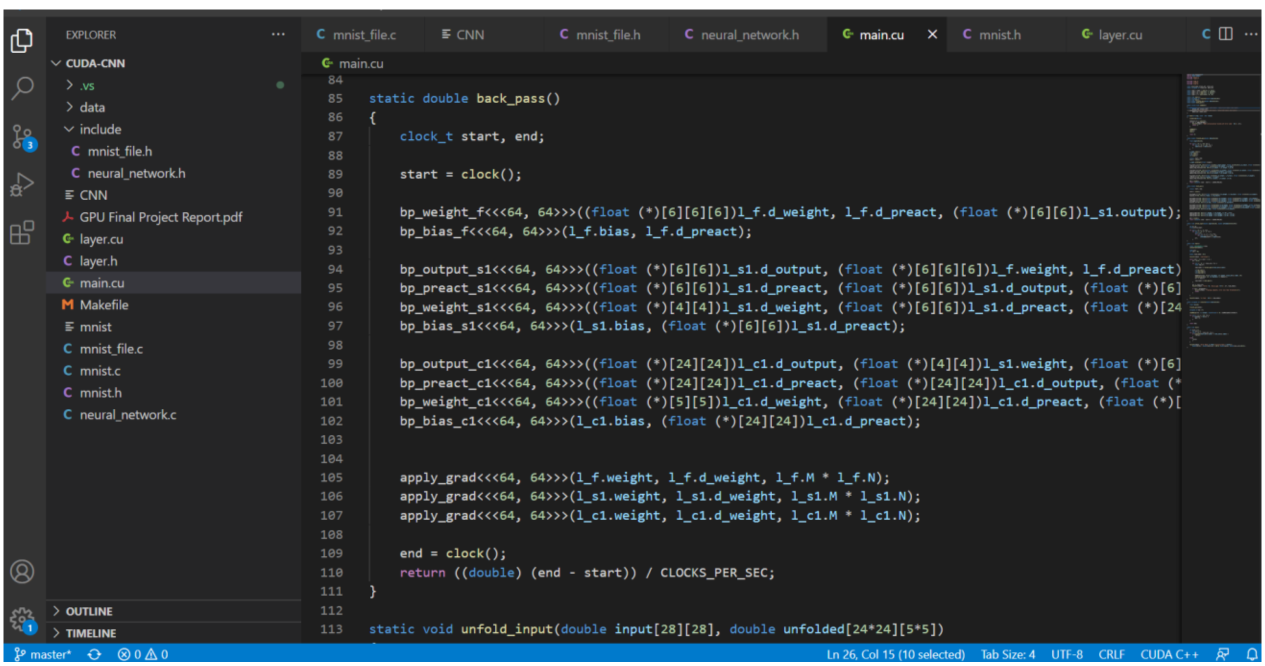
Implementation design for CNN is done in the same manner for CNN and GPU to get a comparison. The network contains 3 layers with one input layer and 2 hidden layers.

CPU implementation calculates the gradient after each layer and serially runs the code. The major difference between the GPU implementation and CPU implementation is the call to write a CUDA code in GPU and feed the weights and biases.

Implementation of Forward Pass



Implementation of Backward Pass



7.Comparison between CPU and GPU

	Accuracy	Time Taken
CPU	81 %	207 seconds
GPU	97 %	172 seconds

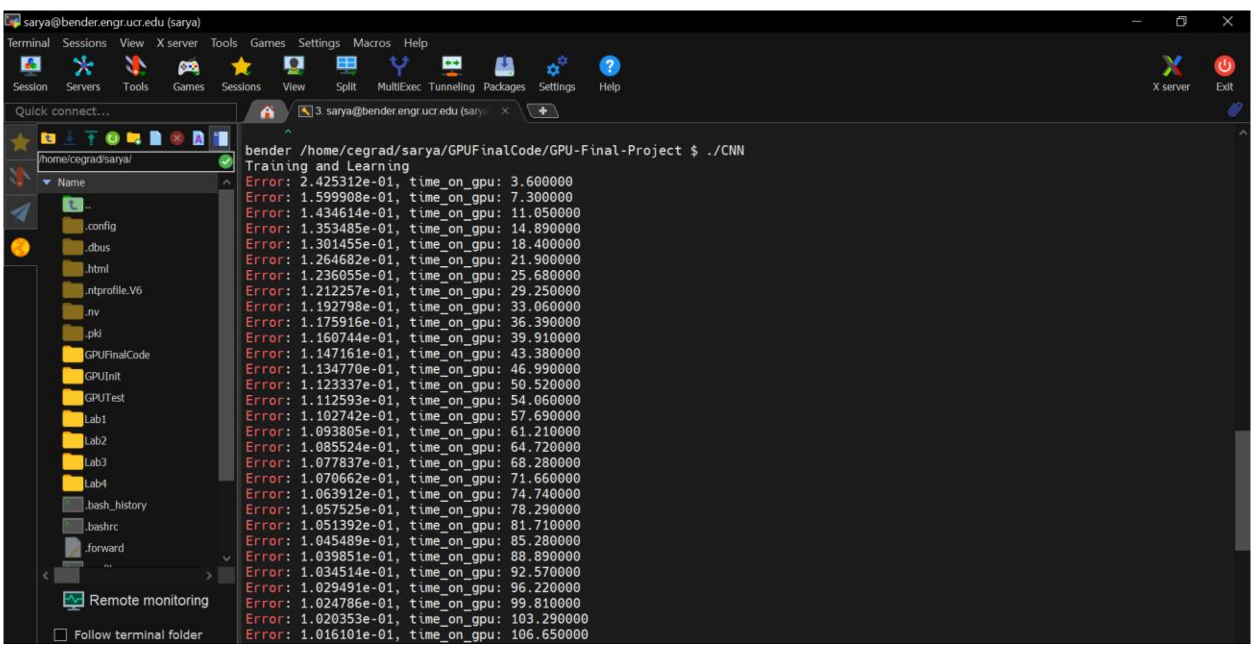
GPU takes less time and provides accuracy to a great extent as compared to CPU implementation. If we increase the number of epochs the result would be significantly different and a larger variation in time taken.

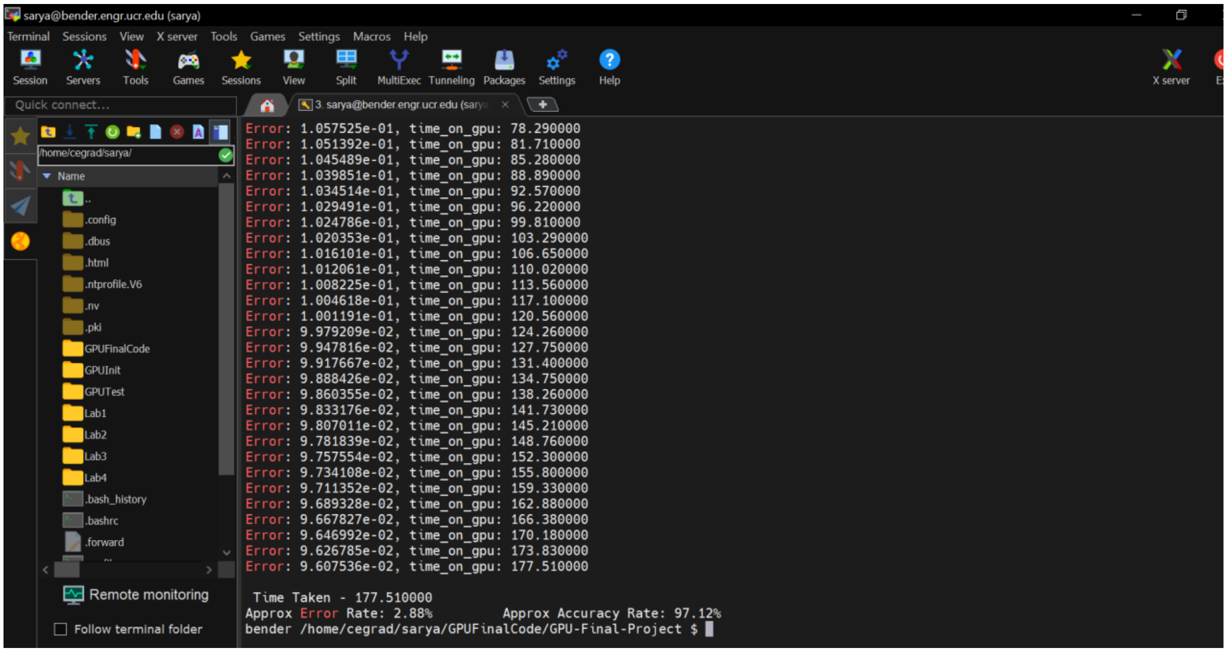
9. Steps to Run the Code

1. Extract all the files from the project folder.
2. Run **make** command inside CuDNN folder.
3. Two files, CNN and mnist will be generated.
4. Run - ./CNN for GPU implementation.
5. Run- ./mnist for CPU implementation.

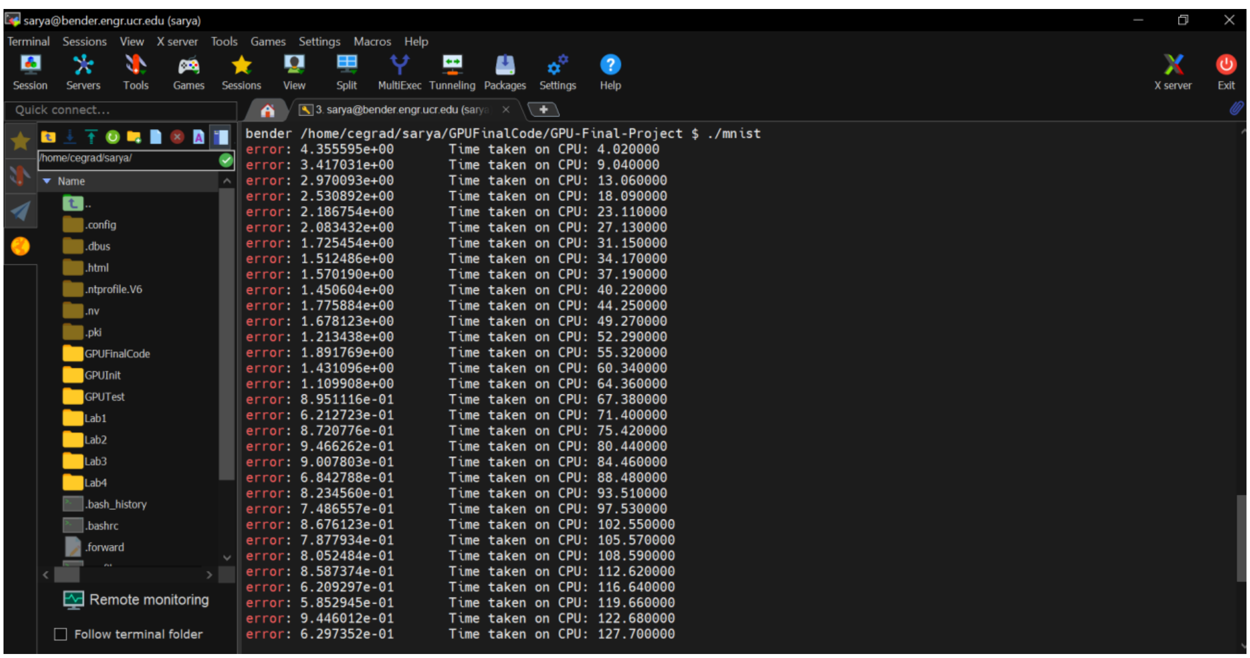
Results produced on the bender

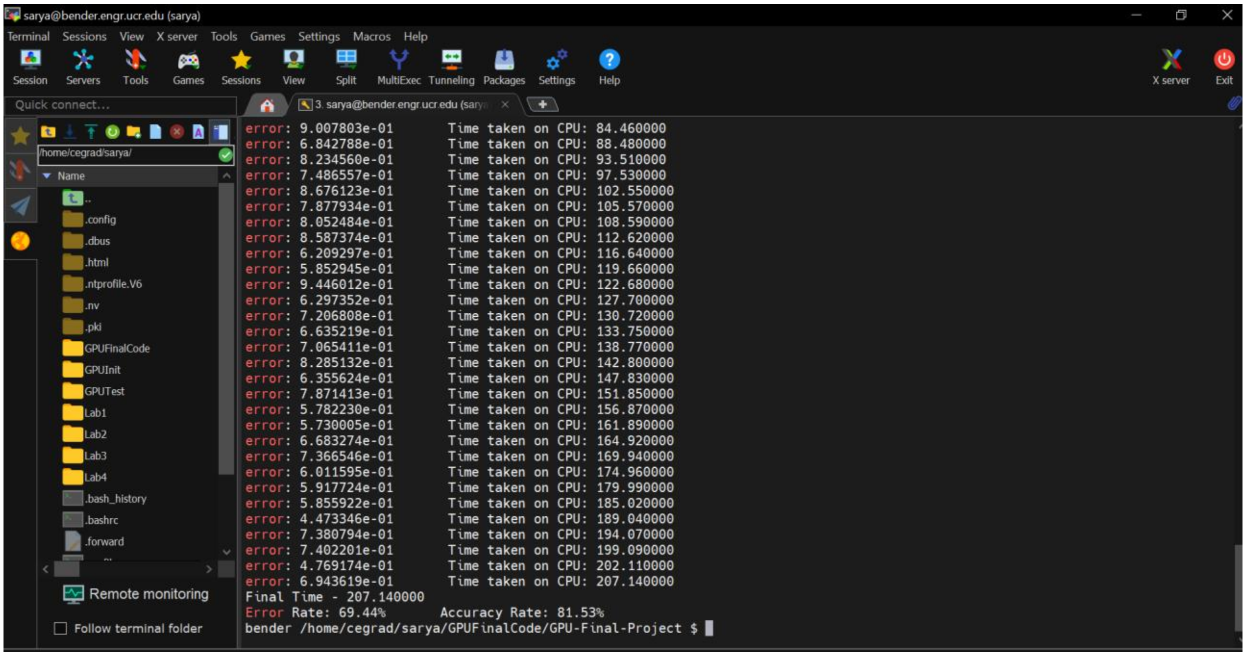
GPU Results




sarya@bender.engr.ucr.edu (sarya)
Error: 1.057525e-01, time_on_gpu: 78.290000
Error: 1.051392e-01, time_on_gpu: 81.710000
Error: 1.045489e-01, time_on_gpu: 85.280000
Error: 1.039851e-01, time_on_gpu: 88.890000
Error: 1.034514e-01, time_on_gpu: 92.570000
Error: 1.029491e-01, time_on_gpu: 96.220000
Error: 1.024786e-01, time_on_gpu: 99.810000
Error: 1.020353e-01, time_on_gpu: 103.290000
Error: 1.016101e-01, time_on_gpu: 106.650000
Error: 1.012061e-01, time_on_gpu: 110.020000
Error: 1.008225e-01, time_on_gpu: 113.560000
Error: 1.004618e-01, time_on_gpu: 117.100000
Error: 1.001191e-01, time_on_gpu: 120.560000
Error: 9.979209e-02, time_on_gpu: 124.260000
Error: 9.947816e-02, time_on_gpu: 127.750000
Error: 9.917667e-02, time_on_gpu: 131.400000
Error: 9.888426e-02, time_on_gpu: 134.750000
Error: 9.860355e-02, time_on_gpu: 138.260000
Error: 9.833176e-02, time_on_gpu: 141.730000
Error: 9.807011e-02, time_on_gpu: 145.210000
Error: 9.781839e-02, time_on_gpu: 148.760000
Error: 9.757554e-02, time_on_gpu: 152.300000
Error: 9.734108e-02, time_on_gpu: 155.800000
Error: 9.711352e-02, time_on_gpu: 159.330000
Error: 9.689328e-02, time_on_gpu: 162.880000
Error: 9.667827e-02, time_on_gpu: 166.380000
Error: 9.646992e-02, time_on_gpu: 170.180000
Error: 9.626785e-02, time_on_gpu: 173.830000
Error: 9.607536e-02, time_on_gpu: 177.510000
Time Taken - 177.510000
Approx Error Rate: 2.88%   Approx Accuracy Rate: 97.12%
bender /home/cegrad/sarya/GPUFinalCode/GPU-Final-Project $

CPU Results


sarya@bender.engr.ucr.edu (sarya)
bender /home/cegrad/sarya/GPUFinalCode/GPU-Final-Project $ ./mnist
error: 4.355595e+00   Time taken on CPU: 4.020000
error: 3.417031e+00   Time taken on CPU: 9.040000
error: 2.970093e+00   Time taken on CPU: 13.060000
error: 2.530892e+00   Time taken on CPU: 18.090000
error: 2.186754e+00   Time taken on CPU: 23.110000
error: 2.083432e+00   Time taken on CPU: 27.130000
error: 1.725454e+00   Time taken on CPU: 31.150000
error: 1.512486e+00   Time taken on CPU: 34.170000
error: 1.570190e+00   Time taken on CPU: 37.190000
error: 1.450604e+00   Time taken on CPU: 40.220000
error: 1.775884e+00   Time taken on CPU: 44.250000
error: 1.678123e+00   Time taken on CPU: 49.270000
error: 1.213438e+00   Time taken on CPU: 52.290000
error: 1.891769e+00   Time taken on CPU: 55.320000
error: 1.431096e+00   Time taken on CPU: 60.340000
error: 1.109908e+00   Time taken on CPU: 64.360000
error: 8.951116e-01   Time taken on CPU: 67.380000
error: 6.212723e-01   Time taken on CPU: 71.400000
error: 8.720776e-01   Time taken on CPU: 75.420000
error: 9.466262e-01   Time taken on CPU: 80.440000
error: 9.007803e-01   Time taken on CPU: 84.460000
error: 6.842788e-01   Time taken on CPU: 88.480000
error: 8.234560e-01   Time taken on CPU: 93.510000
error: 7.486557e-01   Time taken on CPU: 97.530000
error: 8.676123e-01   Time taken on CPU: 102.550000
error: 7.877934e-01   Time taken on CPU: 105.570000
error: 8.052484e-01   Time taken on CPU: 108.590000
error: 8.587374e-01   Time taken on CPU: 112.620000
error: 6.209297e-01   Time taken on CPU: 116.640000
error: 5.852945e-01   Time taken on CPU: 119.660000
error: 9.446012e-01   Time taken on CPU: 122.680000
error: 6.297352e-01   Time taken on CPU: 127.700000


sarya@bender.engr.ucr.edu (sarya)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
/home/cegrad/sarya/
Name
.config
.dbus
.html
.ntprofile.V6
.nv
.pki
GPUFinalCode
GPUInit
GPUTest
Lab1
Lab2
Lab3
Lab4
.bash_history
.bashrc
.forward
Remote monitoring
Follow terminal folder
error: 9.007803e-01 Time taken on CPU: 84.460000
error: 6.842788e-01 Time taken on CPU: 88.480000
error: 8.234560e-01 Time taken on CPU: 93.510000
error: 7.486557e-01 Time taken on CPU: 97.530000
error: 8.676123e-01 Time taken on CPU: 102.550000
error: 7.877934e-01 Time taken on CPU: 105.570000
error: 8.052484e-01 Time taken on CPU: 108.590000
error: 8.587374e-01 Time taken on CPU: 112.620000
error: 6.209297e-01 Time taken on CPU: 116.640000
error: 5.852945e-01 Time taken on CPU: 119.660000
error: 9.446012e-01 Time taken on CPU: 122.680000
error: 6.297352e-01 Time taken on CPU: 127.700000
error: 7.206808e-01 Time taken on CPU: 130.720000
error: 6.635219e-01 Time taken on CPU: 133.750000
error: 7.065411e-01 Time taken on CPU: 138.770000
error: 8.285132e-01 Time taken on CPU: 142.800000
error: 6.355624e-01 Time taken on CPU: 147.830000
error: 7.871413e-01 Time taken on CPU: 151.850000
error: 5.782230e-01 Time taken on CPU: 156.870000
error: 5.730005e-01 Time taken on CPU: 161.890000
error: 6.683274e-01 Time taken on CPU: 164.920000
error: 7.366546e-01 Time taken on CPU: 169.940000
error: 6.011595e-01 Time taken on CPU: 174.960000
error: 5.917724e-01 Time taken on CPU: 179.990000
error: 5.855922e-01 Time taken on CPU: 185.020000
error: 4.473346e-01 Time taken on CPU: 189.040000
error: 7.380794e-01 Time taken on CPU: 194.070000
error: 7.402201e-01 Time taken on CPU: 199.090000
error: 4.769174e-01 Time taken on CPU: 202.110000
error: 6.943619e-01 Time taken on CPU: 207.140000
Final Time - 207.140000
Error Rate: 69.44% Accuracy Rate: 81.53%
bender /home/cegrad/sarya/GPUFinalCode/GPU-Final-Project $