Brain Criticality Hypothesis Simulation - Implementation

Drew Smith

March 17, 2023

1 Overview

1.1 Summary

The NN will be implemented in C++ and CUDA. Activations will all happen concurrently, but not asynchronously. This means that on every 'tick', all of the activations will occur at the same time in terms of simulation time. Each tick will include several steps. These steps are enumerated below along with the hardware they will run on:

1. Assign input values (CPU)
2. Apply excitatory postsynaptic potentials decay (GPU)
3. Feed-forward (GPU)
4. Determine activations (GPU)
5. Pass output values to simulation (CPU)
6. Adjust connection weights (GPU)
7. Determine then kill dead neurons (GPU)
8. Create new neurons (CPU)
9. Do simulation tick (CPU)

In this document, I will explain the implementation details specific to the NN. The game simulation details will be explained in another document. Each of the steps above will be explained further in its own section below.

2 NN Architecture

2.1 Neurons

On the GPU, neurons will store their activation threshold, an array of connections, the current excitory level, and a true/false value for if the neuron has been activated on this tick. On the CPU neurons will store their location and a temporary array of connections used to kill neurons CPU-side. The array of connections will have a fixed size (Preferably a multiple of only 2).

Lastly, neurons will store an array of reception values. This is to eliminate race conditions during feed-forward. This will be elaborated further in the Neuron subsection in this section.

2.2 Connections

Each connection will have a source neuron (implicitly by its location in memory), a target neuron (explicitly), and an index in the array of its target neuron's reception array. Upon feed-forward, a kernel will be created with a thread for each connection in each neuron. This connection will take the activation of its source and multiply it by the connection's weight and update the target neuron's reception array.

3 Applying Excitatory Postsynaptic Potentials Decay

3.1 Kernel

A kernel will be called with a thread for each neuron. Each neuron's excitory level will be decreased by a constant value.

4 Feed-Forward

4.1 Kernel

A kernel will be called with a thread for each connection. Each connection will update its target neuron's reception array with the source neuron's activation times the weight of the connection.

5 Adjusting Connection Weights

5.1 Kernel

Connection weights will be updated based on the activations of the target neurons. The equation for neuron updates is in the idea.pdf.

6 Determine Activations

6.1 Kernel

A kernel will be called with a thread for each item in the reception array. A reduction algorithm will be implemented to sum all of the items in the reception array. The output of the reduction will be added to the neuron's previous

excitory level. This final output will be the neuron's exitory level for the next tick.

Another kernel will be called with a thread for each neuron. If the neuron's excitory level exceeds the neuron's activation threshold, the neuron will activate.

7 Determining Then Killing Dead Neurons

7.1 Kernel

Neurons will be considered dead if they have fired fewer than 10 times in the last 1000 ticks. A neuron is marked dead by the GPU.

8 Creating New Neurons

8.1 Kernel

NA

8.2 CPU

The CPU will be responsible for creating new neurons. New neurons will be spawned in partitions within the simulated 3D brain space with fewer neurons than its maximum capacity. It will favor partitions with fewer neurons to encourage even growth. New neurons will select, at random, target and source connections. The weights of these connections will also be randomly generated. NOTE: it may be more efficient to call the adjusting connection weights kernel to balance the new connections.