GPU Accelerated Neuromorphic Training Framework

Sponsor: Jackson Mowry (Senior Design student)

August 25, 2025

Contents

1 Background

One of the largest unsolved challenged in neuromorphic computing is the ability to efficiently train networks to perform complicated tasks. One proposed solution to this issue is evolutionary optimization, which takes inspiration for our biological world and exposes many different candidate networks to the same scenario, keeping only the best performers.

Neuromorphic computing is a novel technique that utilizes neurons and synapses to perform arbitrary computations. These networks store information both in their structure and the parameters that define the connections between neurons. Any tasks that can be solved with traditional machine learning can also be solved with neuromorphic computing, and often with much less computational power. One of the major reasons for this is that we train the structure of the network at the same time as the parameters, meaning neurons can be eliminated at any point in the training process if they are providing no benefit to the desired outcome.

Evolutionary optimization works by trying hundreds of networks simultaneously on a single task evaluating their fitness. This means that the only bottleneck is simulator speed, which is the component responsible for applying input to a network and simulating how spikes flow between different neurons. I have previously written a CPU only simulator that takes advantage of vector instructions to improve training speed by around 3x, which proves that this pathway should yield results far better than baseline performance.

Writing a more GPU based simulator would not only massively improve performance, but it would also allow us to take advantage of GPU hardware that is present on nearly every modern device. These accelerators generally sit idle when simulating neuromorphic networks, and should be taken advantage of. Additionally, lower power devices are often built with accelerators supporting OpenCL (Raspberry Pi Zero 2W), which can offer speeds orders of magnitude higher than CPU only simulation on low SWaP devices.

2 Project Needs

1. Access to devices supporting OpenCL (or other competing GPGPU computation technology)
2. Ability to utilize the TENNLab corpus of software

3 Project Scope

1. Software Development: A general purpose neuroprocessor support both fallback CPU computation and plugable GPU computation that is configurable at runtime to utilize the most performant hardware available

4 Potential Technical Considerations

1. GPU kernels should have the ability to be written in any applicable language, while the driver/wrapper application will likely be written in C++

5 Deliverables

1. A simple user-friendly application utilizing the TENNLab framework suite of software to enable training networks with all available hard-

ware, likely GPUs

2. Training videos showing how to effectively utilize the application
3. Documentation for the application