

Introduction

Neural network, especially Deep neural network (DNNs), has been recognized to be effective machine learning process, applicable from automated driving to medical devices. However, their training can be time and energy-intensive. One class of DNNs, known as the Spiking neural networks (SNN), mimics the biological brain and produces discrete spikes based on to transfer of information between neurons. To further simulate biological networks, STDP SNN network with LIF and Dense processes can be trained.

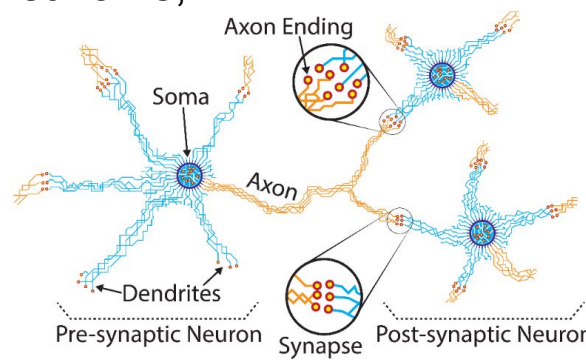


Fig1. STDP applied to neural network

Objective & Impact of Professor's Research

Professor Pedram's lab focuses on power awareness in VLSI circuits and systems with current research on Spiking neural networks (SNNs) to achieve trainable neuron circuits. As part of the Discover Expedition project, the lab also aims to develop and demonstrate superconductor electronics (SCE) and superconductive computing technology to reduce the energy requirements of national computing infrastructure significantly. SuperSoCC stands for the superconductive system of cryogenic (computing) cores, the first step toward realizing an exascale superconductive supercomputer. The goal is, therefore, to build a fully operational SuperSoCC to execute applications at scale.

Acknowledgements

I would like to thank Professor Pedram for giving me the opportunity to research in S.P.O.R.T lab as well as my mentors Sasan Razmkhah, Zeynep Ucpinar and Altay Karamuftuoglu for guiding me through my project. Additionally, I would like to thank Marcus Gutierrez, my Center Mentor for supporting me through the program in my difficult times. SHINE program.

Research & Learning Process

Firstly, I program one single neuron because the generalized network requires intense programming and advanced concepts.

$$\hat{y} = g \left(w_0 + \sum_{i=1}^m x_i w_i \right)$$

Fig2. architecture equation of a single neuron
* g = activation function

Then, I performed SNNs on the Iris dataset to classify three types of iris. I learned how to use the Numpy library to transform and input the dataset there.

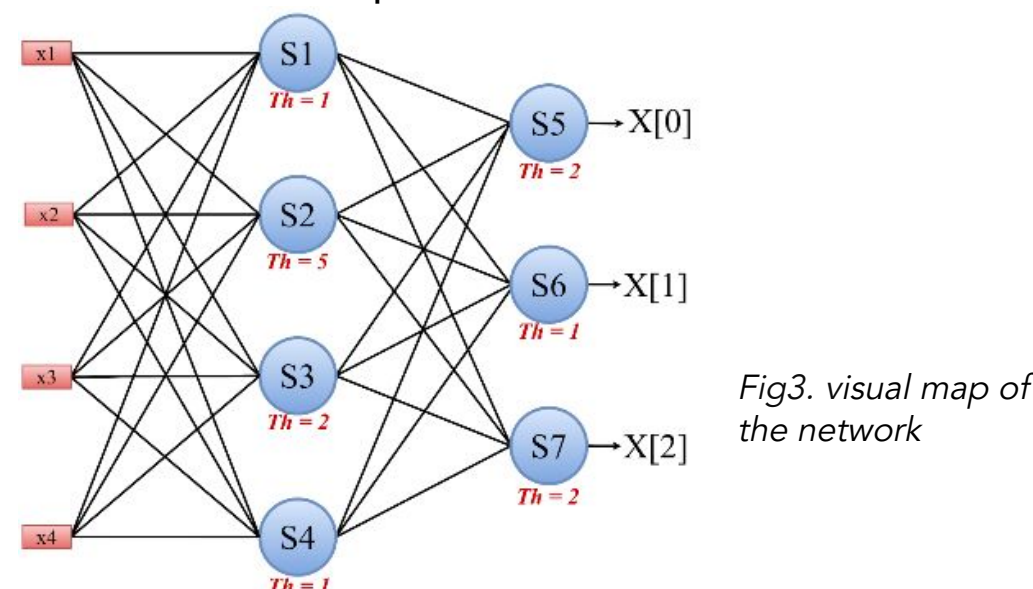


Fig3. visual map of the network



Iris Versicolor Iris Setosa Iris Virginica

Fig4. 3 types of iris

To allow any data to pass into the NN, I generalized the previously hard-coded NN and applied one hot encoding for classification comparison under the scikit-learn library. The network runs on Pycharm, distributed by Anaconda, Ubuntu, a system I had to learn.

Then I learned/previewed various machine learning models' implementation, bias-variance tradeoff, and TensorFlow (Keras) in NN for classification.

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(16, activation='relu'),
    tf.keras.layers.Dense(1)
])
```

Fig5. Creating layers with Keras

They all led up to my final learning, the Lava framework, in order to construct unsupervised learning: MNIST Digit Classification and the Spike-timing Dependent Plasticity (STDP) phenomenon. STDP is a spike-based formulation of a Hebbian learning rule, where synaptic weights increase proportionally to two neighboring neurons' activation/deactivation. As for MNIST hyperparameter tuning, I trained and then pruned new parameters.

Methods & Results

Iris Neural Network:

- Quantize parameter, sepal length(x1), sepal width(x2), petal length(x3), petal width(x4), into int.
- Layer1: Plug int parameters into the below operations, which give sums S1-4; a spike is released for every S if its S is higher than the Threshold value, which returns 1, or return 0 if S is lower. Repeat the process for Layer 2.

Layer1:

$$S1 = x1 \oplus x2 \oplus x4$$

$$S2 = 2x1 \oplus x2 \oplus x3 \text{ weight}$$

$$S3 = x3 \oplus x4$$

$$S4 = x1 \oplus x2 \oplus x3 \oplus x4$$

Layer2:

$$S5 = 2s1 + s2 - 2s3 + s4$$

$$S6 = -2s1 - s2 + s3$$

$$S7 = s1 + s2 + s3 - 2s4$$

- S5-S7 makes up for a number set of three 0/1s
- read dataset txt file by line,
 - create an empty string to extract numbers and classes, then convert it to a list separately

5. Result:

```
150
116
Accuracy: 0.7733333333333333
```

Generalized Network:

- Pass input, weight (w), and threshold as NumPy arrays.
- Sum: For loop w*input until the length of the weight array
- Spike: Compare the sum with the Threshold value to return 0 or 1.
- Take the last column as a class and reshape it into a column vector.
- Perform one-hot encoding on class column vector.
- Return data and corresponding classes.

MNIST Digit Classification hyperparameter tuning:

- Pre-training verification for available GPU
- Training using SNN torch and Lava
 - adjust the number of layers and neurons in each layer in the program file
 - import weight when finished
- Pruning
 - satisfy limitations on the hardware
 - calculate final_sparsity to fit desired fan in (case1: 32, case2: 64) and limit synapses going into a neuron
 - import & test if weights satisfy the fan in cases.
 - if not, then explore the floating points of final_sparsity and re-prun

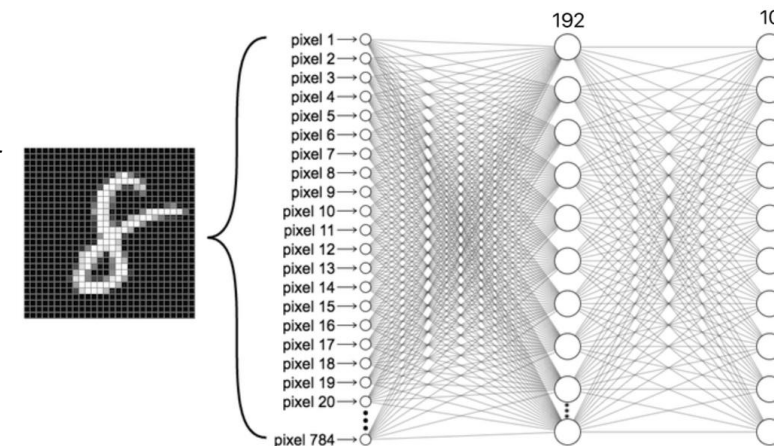
4. Result

This network's throughput is 10GHz, which stays constant for all layers since it directs one layer to another, but the latency (latency = #layer*100 ps) increase by 100ps per addition of a single layer.

Results Analysis

The accuracy increase proportionally with layers, neurons, and fan. However, since fanins or synaptic weights increase power consumptions significantly, along with the other two parameters, should be lowered. Thus the balance point for power optimization need to established.

Fig6. neural network of classifying a 3 layer MNIST digit.



Next Steps for You & Advice to Future SHINE participants

- I plan to further my research of SNN and STDP in robotics as I've read about their ability to accomplish a multi-task autonomous learning paradigm, applying many biological inspirations. DNNs' potential in automatic combat recovery, speech processing, and adapting high-dimensional data are essential to the future robotic community.
- I advise future SHINE participants to display their earnest passion and get to know their mentors without hesitation to ask questions.

Citations

- [1] A. Bozbey, M. A. Karamuftuoglu, S. Razmkhah, and M. Ozbayoglu, "Single Flux Quantum Based Ultrahigh Speed Spiking Neuromorphic Processor Architecture", <http://arxiv.org/abs/1812.10354>
- [2] Fisher, R. A. (1988). Iris. UCI Machine Learning Repository. <https://doi.org/10.24432/C56C76>.
- [3] Mouha, Radouan Ait. "Deep Learning for Robotics." *Journal of Data Analysis and Information Processing*, vol. 09, no. 02, May 2021, pp. 63–76, <https://doi.org/10.4236/jdaip.2021.92005>.