Memristor-NN, Technion Project A.

Background

Artificial neural networks (ANN) became a common solution for a wide variety of problems in many fields, such as control and pattern recognition. Many ANN solutions reached a hardware implementation phase, either commercial or with prototypes, aiming to accelerate its performance. Recent work has shown that hardware implementation, utilizing nanoscale devices, may increase the network performance dramatically, leaving far behind their digital and biological counterparts, and approaching the energy efficiency of the human brain. The background of these advantages is the fact that in analog circuits, the vector-matrix multiplication, the key operation of any neuromorphic network, is implemented on the physical level. The key component of such mixed-signal neuromorphic networks is a device with tunable conductance, essentially an analog nonvolatile memory cell, mimicking the biological synapse. There have been significant recent advances in the development of alternative nanoscale nonvolatile memory devices, such as phase change, ferroelectric, and magnetic memories. In particular, these emerging devices have already been used to demonstrate small neuromorphic networks. However, their fabrication technology is still in much need for improvement and not ready yet for the large-scale integration, which is necessary for practically valuable neuromorphic networks. This project investigates a network prototype based on mature technology of nonvolatile floating-gate memory cells.

Project definition and goals

- Simulate single layered fully connected neural network and evaluating network performance for comparison.
- Simulate the same layer now using Manhattan Rule weights update and evaluating the rule influence on the network predictions.

- Simulating and implement the latter network now using weights being summed by positive and negative blocks as mentioned in the Manhattan Rule.
- Creating a Y-Flash neuron class to to simulate the network with small signal theory.
- Using bias difference method for time evaluation.

Research ideas:

- To test what will happen if:
    - Neuron dies.

Description of the algorithm

Manhattan Rule:

Weight updates according to Manhattan rule training for memristive crossbar circuit pattern classifiers

Output calculation:

First a bias voltage is inserted as input to the system and being recorded. Then an observation to be classified is entered to the NN and the output being recorded. The resulting output is the observation output - bias output. The procedure of calculating bias output being done after every weight update.

Architectural design of the selected solution

Network architecture:

```
reference model see wikipedia MNIST or
http://yann.lecun.com/exdb/mnist/
2-layer NN, 800 HU, Cross-Entropy Loss
```

Y-Flash device

- Physical behavior:
- Program and erase:

Status

- [x] Choosing appropriate dataset for project - **MNIST Dataset**
- [x] Simulating device in Virtouso
- [x] Simulate single layered fully connected neural network and evaluating network performance for comparison.
    - [x] Code Manhattan rule weight update function.
    - [x] Code positive and negative weights.
    - [x] Quantize input images to appropriate input resolution dictated by physical constraints.
    - [x] Train and evaluate network performance
    - [ ] Compare to state of the art networks.

Step 1: Choosing dataset and simulating device behavior:

Choosing dataset:

The simplest image dataset which is the older brother of the ImageNet dataset is the MNIST dataset. Being a benchmark dataset for NN, we have decided to choose it for first estimation since the problem can be solved without the need for CNN. We want to test the memristor array without CNN involved in order to asses the network performance on the least complicated NN.

Simulating device:

Step 2: Simulate MNIST literature neural network architecture and evaluating network performance with memristor constraints.

1. We chose to work with pytorch as our neural network platform because of the robustness and the available modification we can implement in order to simulate the Y-Flash device behavior.

Schedule for the remaining part

Summary

Appendix:

Appendix A: MNIST Dataset

From Wikipedia:

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

MNIST sample images Sample images from MNIST test dataset The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. There have been a number of scientific papers on attempts to achieve the lowest error rate; one paper, using a hierarchical system of convolutional neural networks, manages to get an error rate on the MNIST database of 0.23%. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support-vector machine to get an error rate of 0.8%. An extended dataset similar to MNIST called EMNIST has been published in 2017, which contains 240,000 training images, and 40,000 testing images of handwritten digits and characters.