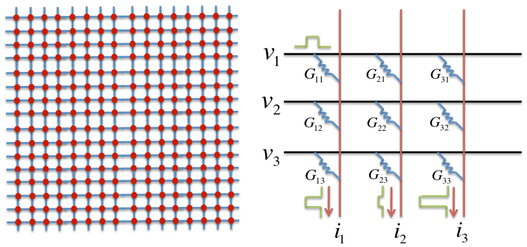
Neural Network with Binary Activation for Efficient Neuromorphic Computing

Weier Wan, Ling Li

Background

Artificial neural network has been adopted to achieve state-of-the-art performance across many tasks. However, the deployment of ANN on conventional hardware such as CPU and GPU remains highly inefficient in terms of both power and speed, largely limiting its usage in systems with tight power budget and real-time processing requirement. The main issue is the frequent data movement between memory (where data and parameters are stored) and logic (where computations are done), which is very expensive with the modern semiconductor technology. Therefore, we want a novel hardware architecture that can maximally eliminate data movement.

Non-volatile Memory (NVM) Crossbar Array



Left: structure of NVM crossbar array with red dots representing NVM devices
Right: Use NVM crossbar to perform affine transformation in O(1) time

Affine Transformation using NVM crossbar:
Encode input vector V as voltage pulses applied on each row, and parameter matrix G as conductance of NVM elements. Using Ohm's law, we can compute the matrix vector multiplication: I = G*V in O(1) time, not depending on the input vector size. Moreover, all computation happens right at where the parameters are stored. No data movement – highly energy efficient.

ANN with Binary Activations

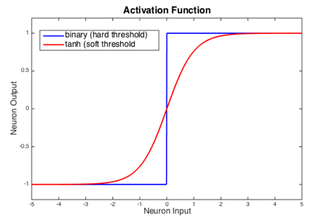
Converting summed current into a multi-bit value requires an ADC that consumes large power and area. Therefore, we propose to constrain the inputs and the activations of network to binary value -1 and 1, which can be implemented with a simple integrator and comparator on hardware

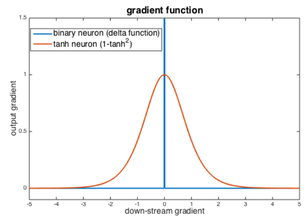
Problem with using binary activation function:

1. **Non-differentiable, gradient = 0 everywhere except at x=0**
2. **In real world, most inputs take real value**

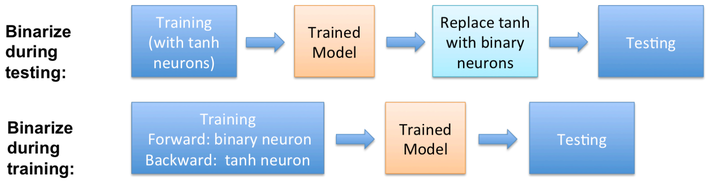
Techniques for Binarizing ANN

1. Tanh Neuron for Approximating Gradient and Encouraging Binarization





2. Binarizing While Training (in contrast to binarizing while testing)



3. Stochastic *Multi-sampling for Real-value Input*

Normalize real-value input to be within the range [-1 1]. Treat the normalized input as expectation value of Bernoulli random variable x:

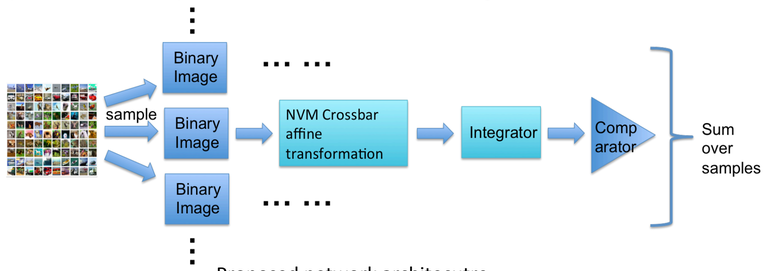
$$x = \begin{cases} +1 & with\ prob = \varphi \\ -1 & with\ prob = 1-\varphi \end{cases}$$

$$E(x) = \varphi - (1-\varphi) \Rightarrow \varphi = (E(x)+1)/2$$

Sample multiple times from the distribution and sum at output.

4. *Bounded Weights (Clip weights at large value)*

Conductance for real devices are bounded, use two devices per parameter to implement both positive and negative values.

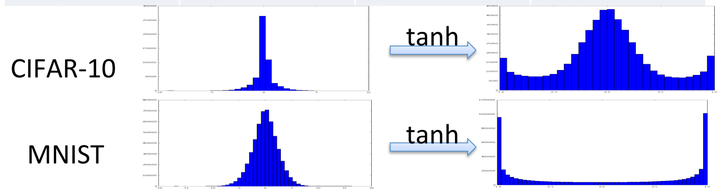


Proposed network architecutre

Results

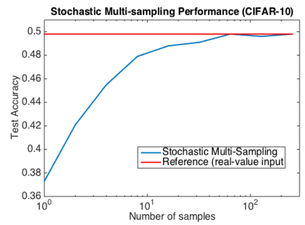
1. Approximated Gradient

	Tanh w/o binarization	Train and Binarize	Binarize and Train
CIFAR-10	51.6	30.3	49.8
MNIST	97.7	80.7	97.4

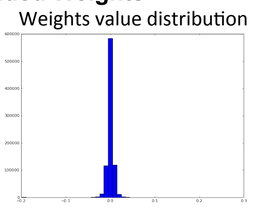


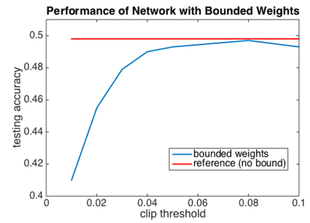
2. Stochastic Multi-Sampling

	Binarize and Train	Deterministic Binarize input	Stochastic Multi-sampling (8)	Stochastic Multi-sampling (64)
CIFAR-10	49.8	35.6	47.9	49.8
MNIST	97.4	97.0	97.5	97.7



3. Bounded Weights





Conclusion

With the four techniques proposed above, we can implement a multi-layer neural network with binary activation with no or little loss in testing performance. Such network can be efficiently deployed onto the proposed NVM-crossbar based hardware architecture.

Contact

Weier Wan
PhD candidate @ Dept. Electrical Engineering
Stanford University
Email: weierwan@stanford.edu

Ling Li
PhD @ Dept. Electrical Engineering
Stanford University
Email: lingli6@stanford.edu