Enhancing the Resiliency of Deep Neural Networks Against Analog Noise

Zhe Wan [1] Tianyi Wang [1] Yiming Zhou [1] Subramanian S. Iyer Vwani P. Roychowdhury[1]

[1]Department of Electrical and Computer Engineering, UCLA

UCLA Samueli School of Engineering

Motivation and Objective

Analog Compute-In-Memory (CIM) architectures for deep neural networks (DNNs) are *fast*, *small* and *energy-efficient*, but they are also **inaccurate**!

Objective:

- Evaluate the feasibility of analog NVM technologies for DNNs based on CIM architectures
- Enhance the resiliency without extra hardware cost

Introduction

Analog Compute-In-Memory (CIM) architectures that use analog non-volatile memories (ANVM) as synaptic weights have emerged as the hope to improve energy-efficiency and throughput for DNN operations by several orders of magnitude.

However, ANVM-based DNN accelerators suffer from the imprecision of the ANVM devices, i.e., the stochastic variations of device resistance. Deviation of ANVM device resistance due to programming error and device instability directly leads to errors in the computations and could degrade accuracy of the DNN significantly.

Although several reports have demonstrated the use of analog NVM for CIM, they are still limited in scale. It is unclear whether the uncertainties in computations and low-resolution quantization (for some CIM architectures) will become a major obstacle for large-scale applications, such as DNNs with millions of weights.

This work presents a simulation framework to evaluate the feasibility of analog NVM technologies for DNNs based on CIM architecture. Simulation results show that the DNNs trained for high-precision digital computing engines are not resilient against the uncertainty of the analog NVM devices. To enhance the resiliency of the analog DNNs, it is proposed to use the Hessian-Aware Stochastic Gradient Descent (HA-SGD) training algorithm to enhance the inference accuracy of the trained DNNs.

The proposed method comes in naturally due to the proposed hardware architecture and does not require extra overhead in the hardware design or operation. We observe that the hardware-aware methodology increase the neural network resiliency by up to about 40% increase of network accuracy depending on the application, and more than 50% if low-resolution ADCs are used in the CIM architecture.

Unique Challenges Posed by Analog Uncertainty

Inevitability Fluctuation always exists in physical processes and directly affects computation results. Because the quantities are not quantized into discrete states as in digital systems, any imprecision will contribute and accumulate.

Continuity Unlike digital bit flips, analog noise is continuously distributed. Error correction code for digital systems cannot help here either.

Stochasticity Unlike digital quantization error, analog noise is random. Extensive work has been done to enable low bit-width quantization of neural networks, but the techniques cannot be directly applied to address analog noise.

Trade-off Better precision leads to higher cost (time, energy and price). To match the raw computation accuracy of digital systems will fundamentally undermine an analog system's advantages in throughput, energy and area efficiency.

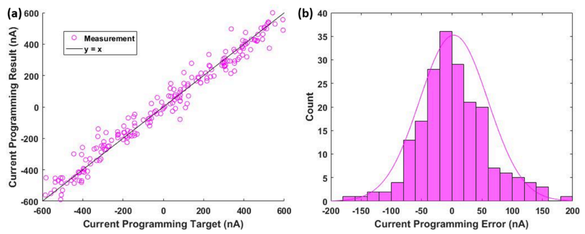

Figure 1. Typical ANVM device programming error characteristics

ANVM Compute-In-Memory Engine and Noise Modeling

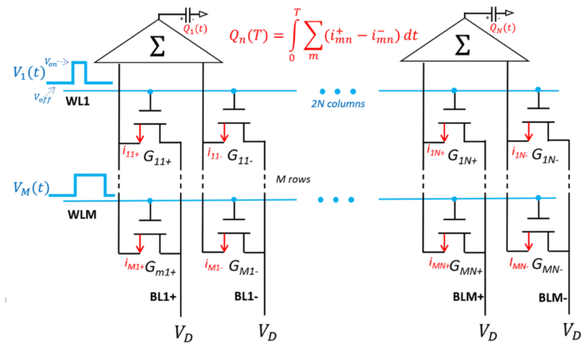

Figure 2. ANVM Compute-In-Memory (CIM) computing: one pair of transistors encodes one real number; computing happens in-place among the cross-bars; Addition: Kirchhoff's Law; Multiplication: Ohm's Law.

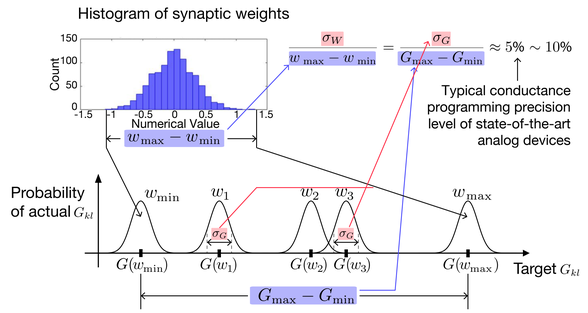

Figure 3. Programming error and conductance variations are proportional to the dynamic range of the device, and is reflected in the numerical domain through the device-programming mapping

Framework for Hardware-aware Evaluation and Training

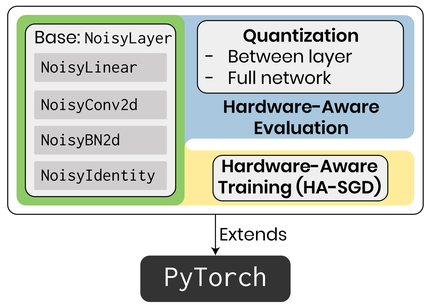

Figure 4. Analog Hardware-aware Extension Modules to PyTorch

We developed a framework for the simulation, evaluation and training of ANVM neural networks, as an extension to `PyTorch`. The simulation modeling details based on which the framework is developed is introduced in the next section.

A novel training algorithm, Hessian-Aware SGD (HA-SGD), is proposed and naturally incorporated in the framework.

Simulation Details

Basic Operations – Matrix-Vector Multiplication

$$x_{\text{out}} = \mathbf{W}x_{\text{in}} + b \quad \Rightarrow \quad x_{\text{out}} = \begin{bmatrix}\mathbf{W} & b/s\end{bmatrix}\begin{bmatrix}x_{\text{in}} \\ s\end{bmatrix} := \widetilde{\mathbf{W}}\,\widetilde{x}_{\text{in}}$$

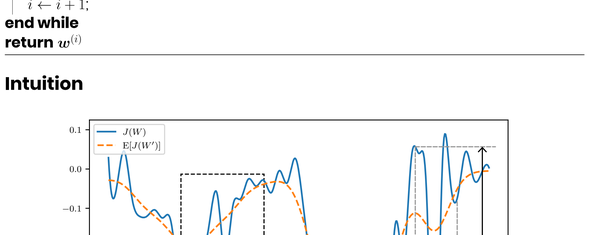

Figure 5. Matching the range of $\mathbf{W}$ and b: To ensure that both weights and biases use the full dynamic range of the device and minimize the effect of the device noise, adjust s to match the range of bias and weights.

Perturbation Model

$$\widetilde{W}_{ij} \longrightarrow \widetilde{W}_{ij} + \nu_{ij}, \quad \nu_{ij} \overset{iid}{\sim} p(\nu)$$

$p(\nu)$: the distribution of the analog perturbation. Determined by hardware characteristics and device-programming mapping. Approximately Gaussian.

Hessian-Aware Stochastic Gradient Descent (HA-SGD)

Algorithm 1: Hessian-Aware Stochastic Gradient Descent (HA-SGD)

Result: Perturbation-resilient parameters of the DNN, $\boldsymbol{w}^*$

Hyper-parameters: L: number of samples to take per step;
  $q(\cdot)$: the perturbation distribution;
  a gradient-based optimizer;
  hyper-parameters of the optimizer;
initialization: $\boldsymbol{w}^{(0)}$, $i \leftarrow 0$;
while *not converge* **do**
  sample mini-batch $\boldsymbol{x}^{(i)}$ from $\mathcal{X}$;
  for $\ell \leftarrow 1$ **to** L **do**
    generate perturbation $\Delta\boldsymbol{\omega}^{(\ell)} \sim q(\Delta\boldsymbol{\omega})$;
    calculate $\nabla_{\boldsymbol{w}} J(\boldsymbol{w}^{(i)} + \Delta\boldsymbol{\omega}^{(\ell)}; \boldsymbol{x}^{(i)})$ by forward-backward propagation;
  end for
  $\bar{\boldsymbol{g}}^{(i)} \leftarrow \frac{1}{L}\sum_{\ell=1}^{L} \nabla_{\boldsymbol{w}} J(\boldsymbol{w}^{(i)} + \Delta\boldsymbol{\omega}^{(\ell)}; \boldsymbol{x}^{(i)})$;
  use the optimizer and $\bar{\boldsymbol{g}}^{(i)}$ to update $\boldsymbol{w}$: calculate $\boldsymbol{w}^{(i+1)}$;
  $i \leftarrow i + 1$;
end while
return $\boldsymbol{w}^{(i)}$

Intuition

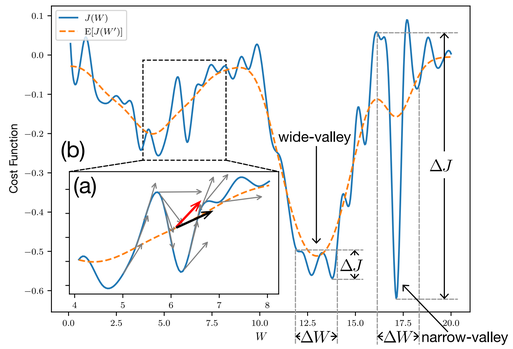

Figure 6. A (doubly-)stochastic optimization, minimizing the expected loss $\mathbb{E}_{p(\Delta\boldsymbol{w})}[J(\boldsymbol{w} + \Delta\boldsymbol{w}; \mathcal{X})]$ (when taking $q(\cdot) = p(\cdot)$).

Results

Analog Uncertainty with Different Variance

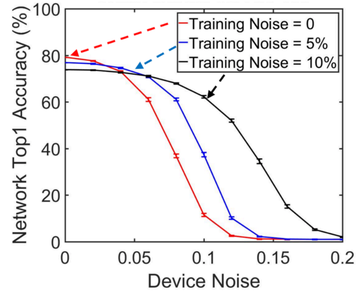

Figure 7. Top-1 accuracy for the WideResNet-28 tested for CIFAR-100. The analog perturbations are assumed to be zero-mean.
A higher training noise level leads to better resiliency that can stand stronger analog uncertainty, but also sacrifices a little of zero-noise inference accuracy.

Analog Uncertainty with Different Mean

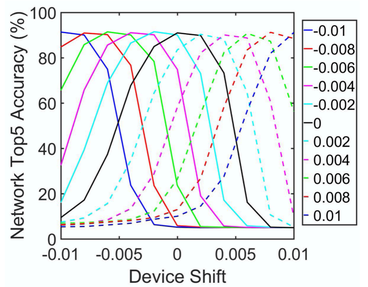

Figure 8. The degradation of network due to analog device shift: Wide-ResNet-28 for CIFAR-100 is trained and tested. Different networks are trained with different values device shift (showed in the legend). All networks are trained with 5% device noise and tested with 5% variance device noise. The mean shift of the device can therefore be compensated by using it as a prior knowledge during training.

Boosting Quantization Resiliency

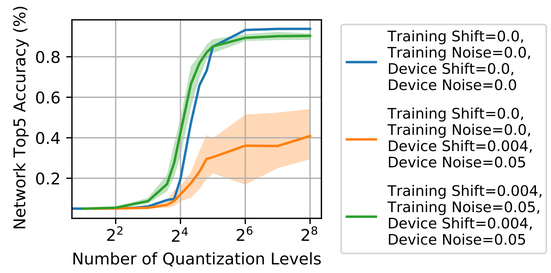

Figure 9. Effect of quantization levels for WideResNet-28 trained using different HA-SGD parameters on CIFAR-100 dataset. HA-SGD can also help in mixed-signal Compute-In-Memory inference engines. If we digitize the output of each layer with low bitwidth (which is necessary for some implementations of convolutional neural networks), the performance will suffer, and HA-SGD can reduce that performance drop as an interesting side-effect.