Design of Reliable DNN Accelerator with Un-reliable ReRAM

Yun Long, Xueyuan She, Saibal Mukhopadhyay
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA
yunlong@gatech.edu, xshe6@gatech.edu, saibal.mukhopadhyay@gatech.edu

Abstract— **This paper presents an algorithmic approach to design reliable ReRAM based Processing-in-Memory (PIM) architecture for Deep Neural Network (DNN) acceleration under intrinsic stochastic behavior of ReRAM devices. We employ the dynamical fixed point (DFP) data representation format to adaptively change the decimal point location based on the data range, minimizing the unused most significant bits (MSBs). Further, we propose a device variability aware (DVA) training methodology where stochastic noise is added to the parameters during training to enhance the robustness of network to the parameter's variation. Simulations indicate that, on average, the proposed algorithms improve the computing accuracy by more than 20% considering various benchmark DNNs (convolutional and recurrent). Moreover, the proposed approach enhances robustness of the DNN to noisy input data.**

I. Introduction

Deep neural networks (DNNs) have shown great promise for a wide range of applications, including image classification [1], speech recognition [2], language processing [3], and computer games [4], to name a few. Deep learning also created demand for energy-efficient hardware accelerators. Dedicated application specific integrated circuit (ASIC) has been intensively investigated to improve the efficiency of DNN acceleration [5], [6]. However, the major computation of DNNs: vector-matrix and matrix-matrix multiplication are data intensive, making the memory access latency and energy consumption the key bottlenecks to further improve the computation efficiency with a conventional von Neumann architecture.

Direct integration of computation and memory, namely, processing-in-memory (PIM) provides a promising solution to overcome the memory barrier and enable design of DNN accelerators with orders of magnitude improvement for energy efficiency. Recently, emerging non-volatile memory (NVM), in particular, resistive random access memory (ReRAM) based mixed-signal PIM architecture have been extensively researched to design efficient machine learning accelerators due to its fast read/write speed, high density, and high on-off ratio. Several recent works have demonstrated that ReRAM based DNN (CNN and RNN) accelerator achieve promising speed/efficiency gain over GPU and ASIC solutions [7]–[10].

A major challenge for designing reliable ReRAM based DNN accelerator is the inherent unreliability of the ReRAM devices, i.e., the stochastic variations of device resistance (variation exists in both high resistance state (HRS) and low resistance state (LRS)). Unlike the digital approach where the computation can be accurately performed as long as the bit-width is precise enough, the computation inside the ReRAM crossbar (i.e. multiplication-accumulation operation (MAC)) is executed in an analog fashion. Deviation of device resistance directly leads to errors in the sum-of-products result, thereby significantly degrading computing accuracy [10], [11].

Recent papers have developed hardware-based techniques to improve robustness of DNN under ReRAM variations. For example, Chen et. al. presented a dual-reference multilevel sense amplifier (SA) to improve the sensing accuracy and ReRAM cell is used as binary device to diminish the overlapped region between two resistance levels [9]. Lin et. al. proposed a simulation framework to model the impact of noise to the accuracy of ReRAM based DNN accelerator and a workload-dependent sensing scheme is developed for better inference accuracy [12].

In this paper, we proposed a complementary, algorithm-driven approach to design reliable DNN accelerator with unreliable ReRAM devices. The paper makes following key contributions:

- We employ the dynamic fixed point (DFP) data representation [13] for mapping DNN weight matrices into the ReRAM crossbar during inference. In particular, we use different decimal point for each layers of neural network to maximize the utilization of ReRAM devices and reduce errors introduced by the unused most significant bits (MSBs).
- We propose a device-variation-aware (DVA) training methodology to enhance the robustness of neural network. By injecting random noise in the parameters during training, the trained model demonstrates high degree of resilience to parameter variation during evaluation stage.

We evaluate the proposed algorithms with CNNs (AlexNet and VGG) for image classification task and RNNs (simple RNN and stacked LSTM) for human activation recognition task. We observe that for all the benchmark applications, the computing accuracy improved by 24% on average across different levels of device variability (from 0% to 50%). The proposed algorithms also enhance the network robustness to input noise, making it suitable for deployment on low-power system with noisy sensors (such as low resolution camera). The proposed method can be used with any existing ReRAM based DNN accelerators with negligible hardware design overhead, and no inference speed/energy-efficiency drop.

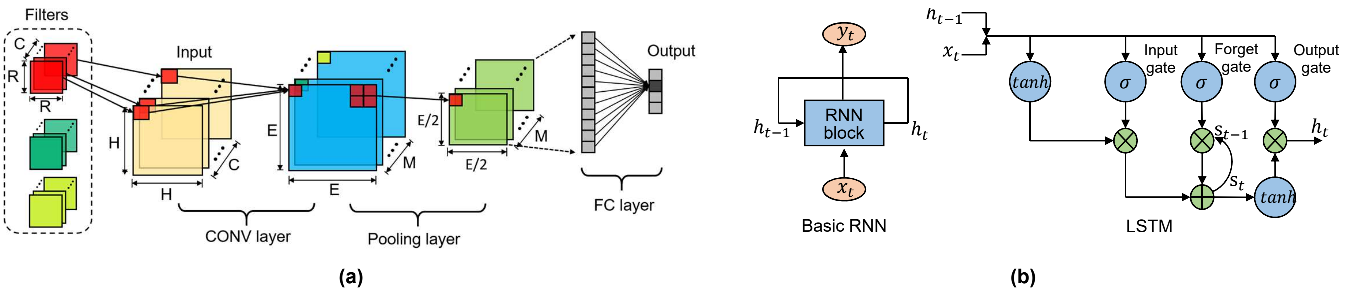


Fig. 1. (a) CNN with Conv layer, Pooling layer, and Fully-connected layer. (b) Basic RNN and LSTM.

II. Preliminaries

A. Deep Neural Network

Convolutional Neural Networks (CNNs) are the heart of modern machine learning systems due to their superior performance across a wide range of applications. Figure 1(a) shows a simple CNN structure containing 1 convolution (Conv) lyaer, 1 pooling layer and 1 fully-connected (FC) layer. The primary computation of a CNN exists in the Conv layers, which perform the convolution operation via applying a set of filters on the input feature maps (ifmaps). Pooling layer is utilized to down-sample the output feature maps (ofmaps) from the Conv layer to progressively reduce the spatial size of the representation and number of parameters. ReLU (not shown in the figure) is the abbreviation of Rectified Linear Units, which is meanly used to provide more non-linearity to the networks. FC layer typically serves as the last few layers of a CNN, where the neurons have full connections to all activations in the previous layer.

The idea behind **Recurrent Neural Network (RNN)** is to utilize the sequential information, i.e. the output not only depend on the current input but also the previous computations. With the unique feature of having *memory* for previous calculations, RNN becomes a popular models that have shown great promise in many time-dependent applications such as NLP, video prediction, dynamical system modeling, to name a few. The left side of Figure 1(b) shows the basic RNN structure. Long short term memory (LSTM) is developed to solve the gradient vanishing issues of basic RNN [14] via a gating mechanism (including input gate, forget gate and output gate), as shown in the right of Figure 1(b). LSTM (and its variant, GRU) shows better performance over a variety of problems, making them popular RNN models for deep learning.

B. ReRAM based DNN Accelerators

Figure 2(a) show the structure of ReRAM cell. The resistive switching layer (HfO_x, TiO_2, Al_2O_3 or their combinations) is sandwiched between two electrodes. The device resistance is modulated by set and reset voltage. During set process, the oxygen ions escape from the lattice and vacancies generate in-place, resulting a oxygen vacancies based conductive filament for electrons hopping, and thus, the cell state is in LRS. Reset is a reverse process which causes the rupture of conductive filament, leaving a gap between the tip of filament and the top electrode. Since there is no direct path for electron to flow through, the device is reset to HRS. Further, by controlling the gap length, device can be reset to different resistance value (i.e. multi-level cell).

The core component of ReRAM based DNN accelerator is the Vector Matrix Multiplier (VMM) engine that performs the multiplication-accumulation (MAC) operation. The VMM engine is composed of a crossbar array and its peripherals, as shown in 2(b). Neural network's parameters are first programmed as the devices conductance, input feature maps are fed as word line (WL) voltage, and the current summed at each bitline (BL) leads to the result for MAC operation. The digital-to-analog (DAC) and analog-to-digital (ADC) conversions are required in and out of the array. In practice, rather than storing a whole parameter in a single device, multiple devices connecting to the same WL are used to represent one parameter with each cell stores 1-2 bits. Similarly, to reduce the overhead of DAC, the input to the WL is divided into several segments (1-2 bits per cycle) and sequentially fed into the crossbar. The final result is summed together with a shift&add unit.

C. Variability in ReRAM

The stochastic variation of ReRAM device resistance presents as a key challenge for the design of reliable ReRAM DNN accelerator. Recent study [15] shows that the variability is one of the intrinsic feature of ReRAM due to the stochastic nature of the generation and rupture of oxygen vacancies. Beside the device-level randomness, other factors such as fluctuation of res/reset voltage, noise introduced during fabrication can also contribution to the variation. Device measurements indicate that the resistance distribution

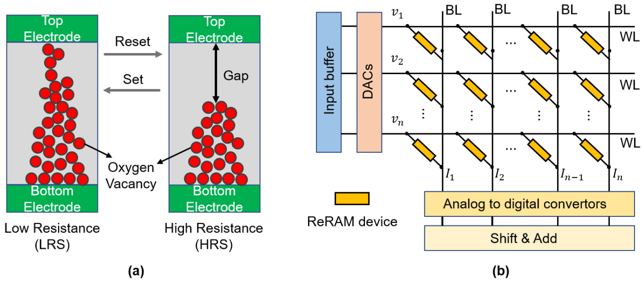


Fig. 2. (a) ReRAM structure and its resistive switching mechanism. (b) ReRAM crossbar for vector matrix multiplication.

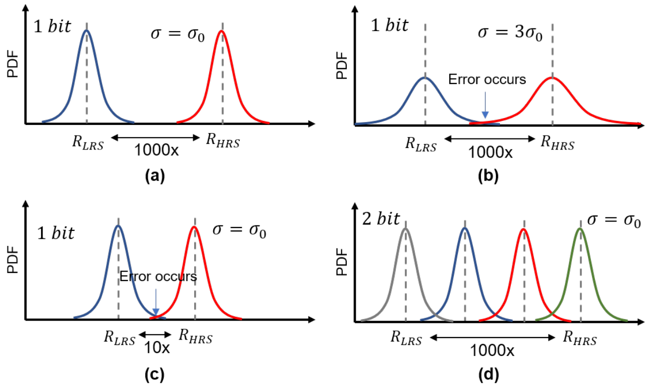


Fig. 3. Factors that can affect the computing accuracy of ReRAM based DNN accelerator.

of ReRAM (both HRS and LRS) follows normal distribution or log-normal distribution [12], [15].

The impact of device variability to the computing accuracy is determined by three factors, namely, the deviation of the statistical distribution (σ), device on/off ratio ($r_{on/off}$) and how many bits to be stored in a cell. Figure 3(a) and (b) shows that large deviation reduces the read margin and error occurs when the device is programmed into the overlapped region. Similarly, as shown in Figure 3(c), small on/off ratio squeezes the margin between different resistance states. When the device is used as MLC (Figure 3(d)), neighbouring states tends to have larger overlap, making the computation more error-prone. Recent works [9], [10], [12] use ReRAM as binary device (i.e. 1bit per cell) to improve the computing reliability. We also consider using binary ReRAM based VMM engine for the rest of the paper.

III. Proposed Methodology

A. Dynamical fixed point data representation

Dynamical fixed point (DFP) is a special data representation formation which allows us to adaptively change the location of the decimal point based on the range of data [13]. The concept of DFP is quite useful considering the fact that the range of parameters inside each layer of DNN can vary a lot. As shown in Figure 4(a), we plot the weights distribution of AlexNet trained with CIFAR-10 dataset. The weight distribution in the first Conv layer is 10x larger than the later FC layers. With conventional fixed point data format, the variation in the first few MSBs (most significant bit) can be very detrimental for small parameter value especially in the case where the device on/off ratio is not large enough.

For example, using conventional fixed point to accommodate the weights of Alexnet, we need 1 bit for sign, 2 bits for integer and 5 bits for fraction. Let's say we chose one parameter ($\hat{A} = 0.125$) and program it to ReRAM devices, as in Figure 5. The stored weight would be identical to $\hat{A}$ if the device is ideal (i.e. resistance on/off ratio $R_H : R_L = \infty$ and resistance shift $\Delta = 0$). However, if the device has limited on/off ratio (i.e. $R_H : R_L = 10$), a large readout error can be observed due to the non-zero current read from MSBs. The readout value can be further disturbed if we consider the device resistance shift (i.e. $R_H : R_L = 10$ and $\Delta = 0.1$). As illustrated in the second part of Figure 5, the final readout from a realistic ReRAM crossbar ($A = 0.505$) skews more than 300% with conventional fixed point data representation.

On the other hand, with DFP, we can left shift the decimal point position to make sure there is no unused MSBs, significantly reducing the readout error. As shown in Figure 5, with DPF, the readout from ReRAM crossbar only shows 7% shew over the original weight value.

We also evaluate the benefits of DFP under different on/off ratio statistically, as demonstrated in Figure 4(b), suggesting DFP together with higher on/off ratio can help to reduce the readout error caused by device variability. For the rest of the paper, we assume the on/off ratio is 1000, which is a realistic number from recent measurement data [15].

We should note that prior works have demonstrated that dynamic fixed-point representation is beneficial to speed up the training/inference of machine learning applications [13]. In this work, rather than accelerating DNN computing, we exploit the dynamic fixed point as a technique to reduce the impact of device variation.

B. Device-Variation-Aware Training

The training of DNN is a process to find the **optimal point** for the loss function (also called cost function) inside the parameter space. However, as shown in the top of Figure 6, a small skew from the optimal point leads to a huge

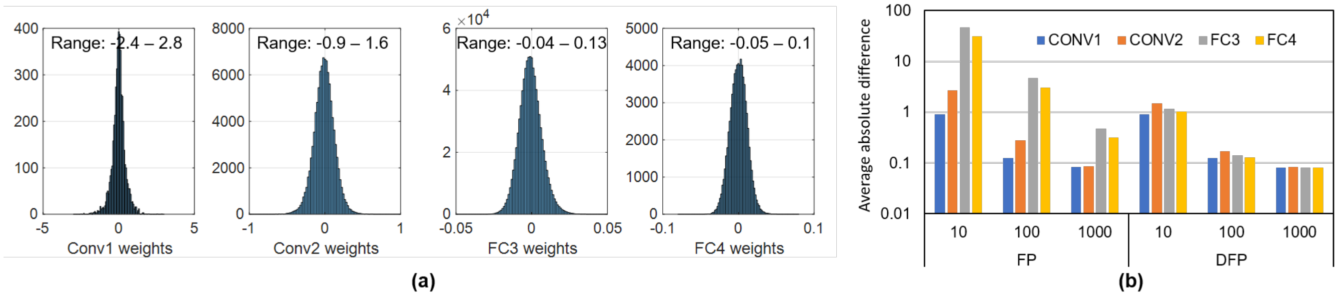


Fig. 4. (a) Parameters distribution from different layers of AlexNet. (b) Parameters readout error caused by limited on/off device ratio ($R_H : R_L = 10, 100, 1000$) and resistance variation ($\sigma = 10\%$).

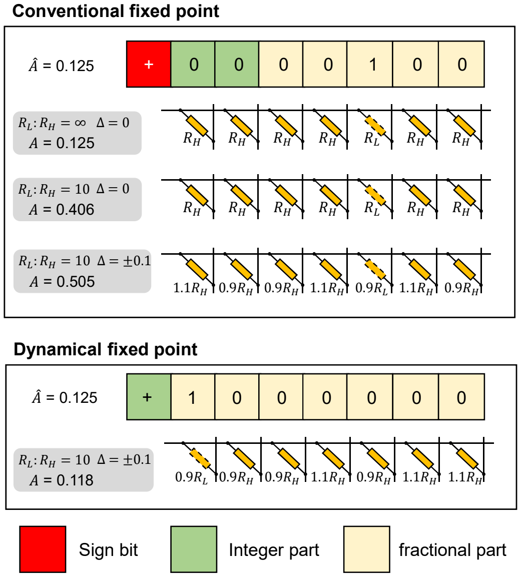
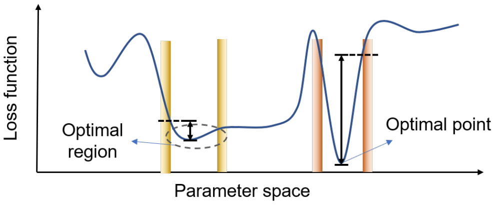


Fig. 5. Convention fixed point v.s. DFP.



```
Def Convolution_layer(input, kernel, mean, stddev):
        noise = normal_distribution(mean, stddev)
        noised_kernel = elementwise_multiplication(kernel, noise)
        conv = conv2d(input, noised_kernel, stride, padding)
        conv = add_bias(conv, bias)
        conv = relu(conv)
        Return conv
```

Fig. 6. **Top**: The comparison between optimal point and optimal region of the loss function in parameter space. **Bottom**: Pseudo code for DVA training process.

increase for the loss function, indicating the DNN accuracy is sensitive to parameter variation. On the other hand, if there is an **optimal region** in which the loss function is relatively small (may not achieve the global minimum) everywhere, we can then use the parameters inside this region to establish a noise robust network.

This inspires us to implement a device-variation-aware (DVA) training methodology where we intentionally add noise to DNN parameters during training to improve the robustness. The pseudo code in Figure 6 illustrates how we add noise to convolution layer. For each training batch, we randomly generate a noise matrix with the same size of the convolution kernel specifying the mean and deviation. Then we add the noise to the parameters by element-wise multiplication and use the noised kernel for the rest operations. Similar procedure is implemented for fully-connected layer and computation inside basic RNN/LSTM.

We have developed a noise model to correlate the device variability with the parameter noise while considering the bit-level representation of the parameters analytically. We first assume the device variation follows Gaussian distribution and the standard deviation is proportional to the mean value:

$$X \sim N(\mu, \sigma^2 = (\gamma\mu)^2) \tag{1}$$

where γ is a constant coefficient reflecting device noise level. Assuming the DNN parameter is 4-bit number $(w_3w_2w_1w_0)$, the target readout value would be:

$$S = 8w_3 + 4w_2 + 2w_1 + w_0 \tag{2}$$

Since each device follows an independent normal distribution, the distribution for the sum (S) is derived as:

$$S \sim N(8w_3 + 4w_2 + 2w_1 + w_0, \sigma^2_{parameter}) \tag{3}$$

$$\sigma^2_{parameter} = \gamma^2[(8w_3)^2 + (4w_2)^2 + (2w_1)^2 + (w_0)^2] \tag{4}$$

This analytic model helps us to statistically characterize the DNN parameter noise distribution given the device variation. For example, with device deviation coefficient $\gamma = 10\%$, the DNN parameter distribution roughly follows a normal distribution with $\sigma_P = 8\%P$ where P is the value of parameters.

In summary, if the variability in the ReRAM process is known, we develop the corresponding noise model and use that to introduce noise during DNN training. Therefore, the trained DNN become inherently aware of device variation. Besides, in the following section we also study the effect of mismatch between (i) the actual ReRAM variation experienced during inference and (ii) the ReRAM variation assumed during training.

IV. Simulation Results

We evaluate the proposed algorithms with 4 benchmarks, including AlexNet and VGG for image classification (CIFAR-10 dataset), basic RNN and LSTM for human activity recognition (UCI-HAR). We implement the DNN models with tensorflow machine learning framework and runs on a Nvidia GTX-1080Ti GPU. Recent study shows that ReRAM can achieve 1000x on/off ratio and the device variation (σ) varies from 5% to 50% of the resistance value [15]. Therefore, we run simulation based on these device characterizations and use the device as a binary cell.

A. *Variability Simulation*

Since a device only store one bit for a parameter, the device variation can't be directly interpreted as the noise of DNN parameters. The first part of Figure 7 shows the computation flow using ReRAM crossbar as the computing engine. We first need to convert the decimal parameter matrix $(P_{decimal})$ to a corresponding binary format (P_{binary}) and program each bit to an ReRAM device. Noise is added to the programmed value based on the device variability. Then we sum up the noised binary matrix $(P_{binary-noised})$ column-wisely, quantize with ADC, and shift&add the partial results together. A Monte-Carlo analysis of the preceding flow can

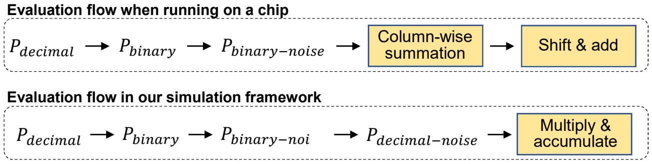


Fig. 7. Evaluation flow when running on an ReRAM chip v.s. Evaluation flow in our simulation framework.

be performed using circuit simulators to model the effect of ReRAM variation to MAC outputs.

However, the detail approach discussed above, although accurate, but computationally in-efficient when implemented in a DNN software simulation platform. Therefore, as shown in the later part of Figure 7, we propose to directly convert the noisy binary weight matrix ($P_{binary-noise}$) back to its decimal format ($P_{decimal-noise}$) and run simulation using existing software framework. One should note that these two approaches are essentially identical if the ADC is ideal. Our approach is still a good approximation even with non-ideal ADC. For example, assuming 8-bit ADC for a crossbar with 64 rows and device variation is 0.2, we randomly chose a matrix (programmed into ReRAM crossbar) and vector (used as input) to perform MAC operation. The output result after normalization for MAC operation is 1.0, 1.0146, and 1.0147 for accurate solution, detailed evaluation flow (top part of Figure 7), and our approximation (bottom part of Figure 7), respectively. This indicates the mismatch between our approximation and the detailed simulation is negligible comparing with the error introduced by device variation.

B. Accuracy Improvement under device variability

As shown in Figure 8(a), we plot the classification accuracy of AlexNet under different device variation for the baseline (i.e. no optimization), DFP enabled, and DFP+DVA enabled configurations. The benefits of DFP is more notable when the device variation is relatively small since the error caused by unused MSBs is dominating. With higher device variation, both baseline and DFP configuration show large accuracy drop, indicating the network is sensitive to parameter noise. The DFP+DVA approach demonstrates the most promising results with less than 7% accuracy drop even under 50% device variation. Similar observations are made in other networks (VGG, basic RNN, and LSTM). As shown in Figure 8(b), we plot the average accuracy across varying device variation (from 0 to 50%) for all four benchmark networks. On average, 24% accuracy improvement is achieved. One should note that for basic RNN, there is only one weight matrix, thus only one decimal point globally. Therefore, DFP configuration is essentially the same as the baseline. We also note that the improvement for simple network (such as the basic RNN) tends to be smaller. This is because with less parameters and simpler structure, the network naturally shows better robustness compared with larger and deeper counterparts.

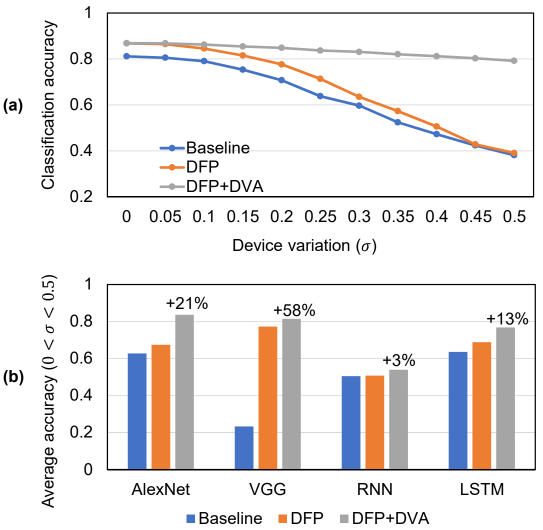


Fig. 8. Accuracy analysis: (a) Inference accuracy of AlexNet with baseline, DFP, and DFP+DVA configurations and (b) Average accuracy improvement across different device variation level for AlexNet, VGG, basic RNN, and LSTM.

C. Variation Mismatch between Training and Inference

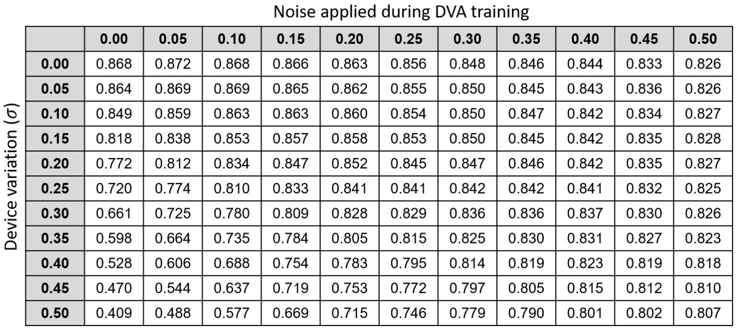
We study DNN accuracy when device variation assumed during training differs from the actual variation during inference, for example, due to lack of exact knowledge or variation changes over time (aging or temperature changes), as shown in Figure 9. With small training noise, the accuracy is approaching the ideal number (i.e. AlexNet trained on GPU has 86.8% accuracy) when the evaluation noise is small (top-left corner) but drops a lot once the evaluation variation is large (bottom-left corner), implying the network has less robustness to large variation. On the other hand, with large training noise, we observe there is a slight accuracy drop under zero evaluation noise (top-right corner) but the network is much more robust: only 2% drop when the evaluation noise increase from 0% to 50% (bottom-right corner). Therefore, if exact variation information is not available, we suggest training with a larger noise to improve reliability.

D. Impact of Non-Normal Variation

We study the classification accuracy of AlexNet under log-normal [15] device variation. Accordingly, the DVA algorithms also employ log-normal noise during training stage. Table I shows that DFP+DVA produces the best performance and robustness to device noise with 15% accuracy improvement over the baseline approach. Interestingly, the baseline shows slightly better results comparing with the case when device variation follows normal distribution (as shown

TABLE I. Accuarcy of baseline, DFP, and DFP+DVA under log-normal distribution.

	0.0	0.1	0.2	0.3	0.4	0.5	Average
Baseline	0.85	0.83	0.76	0.69	0.57	0.39	0.67
DFP	0.86	0.84	0.79	0.75	0.60	0.40	0.72
DFP+DVA	0.86	0.84	0.83	0.81	0.79	0.73	0.82



Noise applied during DVA training

Device variation (σ)	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
0.00	0.868	0.872	0.868	0.866	0.863	0.856	0.848	0.846	0.844	0.833	0.826
0.05	0.864	0.869	0.869	0.865	0.862	0.855	0.850	0.845	0.843	0.836	0.826
0.10	0.849	0.859	0.863	0.863	0.860	0.854	0.850	0.847	0.842	0.834	0.827
0.15	0.818	0.838	0.853	0.857	0.858	0.853	0.850	0.845	0.842	0.835	0.828
0.20	0.772	0.812	0.834	0.847	0.852	0.845	0.847	0.846	0.842	0.835	0.827
0.25	0.720	0.774	0.810	0.833	0.841	0.841	0.842	0.842	0.841	0.832	0.825
0.30	0.661	0.725	0.780	0.809	0.828	0.829	0.836	0.836	0.837	0.830	0.826
0.35	0.598	0.664	0.735	0.784	0.805	0.815	0.825	0.830	0.831	0.827	0.823
0.40	0.528	0.606	0.688	0.754	0.783	0.795	0.814	0.819	0.823	0.819	0.818
0.45	0.470	0.544	0.637	0.719	0.753	0.772	0.797	0.805	0.815	0.812	0.810
0.50	0.409	0.488	0.577	0.669	0.715	0.746	0.779	0.790	0.801	0.802	0.807

Fig. 9. Classification accuracy of AlexNet under the mismatch between noise used during training and variation experienced during evaluation.

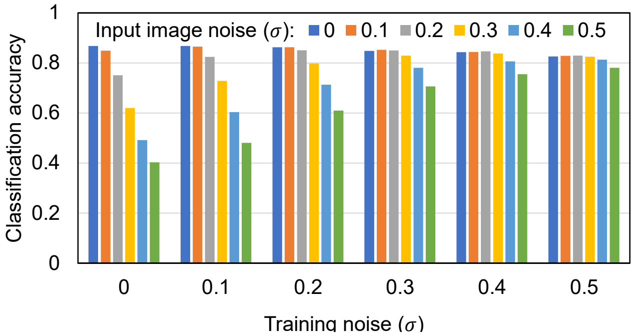


Fig. 10. AlexNet classification accuracy with varying input image noise using trained models with different level of training noise.

in Figure 8(a)). We believe this is because the log-normal distribution has a non-symmetric probability distribution function (PDF) which tends to produces more small variables than normal distribution.

E. Improving the robustness to input noise

We study whether DFP+DVA can improve robustness to noise applied to the input data during inference. We introduce random noise with normal distribution to the image and run inference with models (AlexNet) trained with different parameter noise. As in Figure 10, the accuracy drops quickly with larger input image noise without DVA training. Much better robustness is observed when a DVA trained model is used for inference. Similar with the case in Figure 9, DVA enhance the robustness but slight scarifies the accuracy (less than 4% in the case of AlexNet) for clean images. The robustness against input noise is a major advantage for deployment of DNN in power-constrained platforms for real-world applications, for example, to tolerate noise introduced by low-power and low-cost cameras.

V. CONCLUSIONS

We propose algorithmic approaches to design a reliable DNN accelerator with unreliable ReRAM devices. Our approaches are based on the dynamical fixed point data representation and device variation aware training. The experimental results demonstrate that the proposed approach not only enhance the network robustness under device variation, but also improves the accuracy when input contains noise.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation (NSF-E2CDA #1740197) and in part by the Semiconductor Research Corporation

REFERENCES

[1] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[2] A. Graves *et al.*, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649, IEEE, 2013.

[3] C. Manning *et al.*, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60, 2014.

[4] H. others, "Artificial Intelligence: Chess match of the century," *Nature*, vol. 544, no. 7651, pp. 413–414, 2017.

[5] Y. Chen *et al.*, "Dadiannao: A machine-learning supercomputer," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 609–622, IEEE Computer Society, 2014.

[6] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*, pp. 1–12, IEEE, 2017.

[7] P. Wang *et al.*, "Snrram: an efficient sparse neural network computation architecture based on resistive random-access memory," in *Proceedings of the 55th Annual Design Automation Conference*, p. 106, ACM, 2018.

[8] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Computer Architecture (ISCA), 2016 ACM/IEEE 43th Annual International Symposium on*, pp. 1–12, IEEE, 2016.

[9] W.-H. Chen *et al.*, "A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *Solid-State Circuits Conference-(ISSCC), 2018 IEEE International*, pp. 494–496, IEEE, 2018.

[10] Y. Long *et al.*, "ReRAM-Based Processing-in-Memory Architecture for Recurrent Neural Network Acceleration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 99, 2018.

[11] Y. Long *et al.*, "A Ferroelectric FET based Power-efficient Architecture for Data-intensive Computing," in *International Conference on Computer-Aided Design (ICCAD)*, 2018.

[12] M.-Y. Lin *et al.*, "DL-RSIM: A Simulation Framework to Enable Reliable ReRAM-based Accelerators for Deep Learning," in *International Conference on Computer-Aided Design (ICCAD)*, 2018.

[13] T. Na *et al.*, "Speeding up convolutional neural network training with dynamic precision scaling and flexible multiplier-accumulator," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pp. 58–63, ACM, 2016.

[14] Y. Bengio *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[15] H. Li *et al.*, "Variation-aware, reliability-emphasized design and optimization of rram using spice model," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*, pp. 1425–1430, IEEE, 2015.