Countering Uncertainties in In-Memory-Computing Platforms with Statistical Training, Accuracy Compensation and Recursive Test

Amro Eldebiky[1], Grace Li Zhang[2], Bing Li[1]
[1]Technical University of Munich, [2]TU Darmstadt
Email: {amro.eldebiky, b.li}@tum.de, grace.zhang@tu-darmstadt.de

***Abstract*—In-memory-computing (IMC) has become an efficient solution for implementing neural networks on hardware. However, IMC platforms request weights in neural networks to be programmed to exact values. This is a very demanding task due to programming complexity, process variations, noise, as well as thermal effects. Accordingly, new methods should be introduced to counter such uncertainties. In this paper, we first discuss a method to train neural networks statistically with process variations modeled as correlated random variables. The statistical effect is incorporated in the cost function during training. Consequently, a neural network after statistical training becomes robust to uncertainties. To deal with variations and noise further, we also introduce a compensation method with extra layers for neural networks. These extra layers are trained offline again after the weights in the original neural network are determined to enhance the inference accuracy. Finally, we will discuss a method for testing the effect of process variations in an optical acceleration platform for neural networks. This optical platform uses Mach-Zehnder Interferometers (MZIs) to implement the multiply–accumulate operations. However, trigonometric functions in the transformation matrix of an MZI make it very sensitive to process variations. To address this problem, we apply a recursive test procedure to determine the properties of MZIs inside an optical acceleration module, so that process variations can be compensated accordingly to maintain the inference accuracy of neural networks.**

I. Introduction

Neural networks (NNs) have achieved breakthroughs in different fields, e.g., image recognition [1] and language processing [2]. Tens of millions of weights and hundreds of millions of multiply-accumulate (MAC) operations are needed in one neural network inference step. Von Neumann architecture based on CMOS technology is energy-inefficient in implementing such operations due to the memory bottleneck. The memory bottleneck results from the need to load/store large weight matrices of NN layers and the intermediate feature maps, leading to a large power consumption.

Analog in-memory-computing (IMC) platforms based on emerging technologies have been introduced to address the challenge above. In such platforms, MAC operations are implemented by analog devices, which store the weights in NNs based on their changeable/programmable properties. The storage and computation of NNs happen in the same place, which avoids weight movement and thus reduces the power consumption significantly. Two examples of IMC platforms are optical neural networks accelerators (ONNs) based on Mach-Zehnder Interferometers (MZIs) [3], [4], and resistive RAM (RRAM) crossbar accelerators [5], [6], [7], [8].

In ONNs, computation is done by modulating phases of light through MZIs. Multiplications and additions are implemented by a grid-like layout connecting MZIs. In this architecture, a weight matrix from software training is decomposed with singular value decomposition (SVD) and mapped to phases of MZIs. In RRAM-based accelerators, matrix-vector multiplication is executed by a crossbar based on Ohm's law and Kirchhoff's current law. The weight matrix in NNs is represented by the conductance values of RRAM cells in the crossbar.

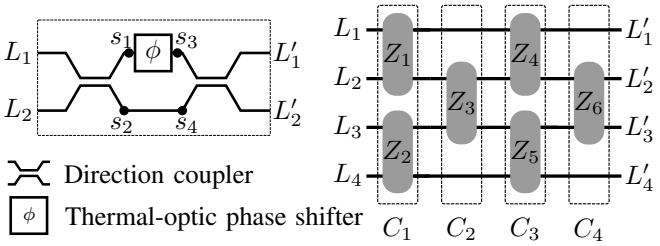
However, the inference accuracy of NNs suffers from severe degradation, when deployed on IMC accelerators, due to manufacturing process variations and noise. The inference accuracy of ONNs is susceptible to process variations and thermal effects of MZIs because weights are represented as trigonometric functions of phases in phase shifters. Accordingly, slight deviations of the phase values can cause a large error in the represented weights. For RRAMs, variations of physical parameters, e.g., cross-section area, cause deviations in their electrical properties [9]. Accordingly, when programmed, the resulting conductance value of an RRAM cell deviates from the target value. Consequently, weights in NNs are not reflected correctly, which causes a degradation in the inference accuracy.

Previous work has tackled the accuracy degradation problem due to weight variations in IMC. The approaches can be classified into three categories: ① modifying the NN training to absorb the effect of variations [10], [11], [12], ② using additional hardware computing units to enhance robustness and counter the effect of the variations [13], [14], ③ calibrating the process variations and fine-tuning the elements storing weights accordingly to restore inference accuracy [15].

In this paper, we present three different approaches from each category to tackle process variations and enhance inference accuracy for IMC accelerators. The presented approaches are demonstrated for RRAM, and ONN accelerators, namely:

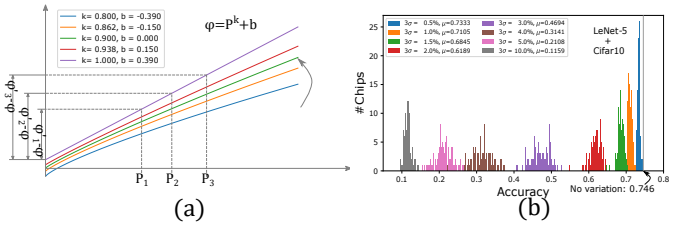
- The process variations and noise of RRAM cells are modeled as correlated random variables and incorporated into weights of NNs during statistical training.
- NNs are trained with a modified Lipschitz constant regularization to suppress error. Then error compensation is

978-3-9819263-7-8/DATE23/© 2023 EDAA



(a) (b)

Fig. 1: MZI and ONN. (a) The MZI structure. (b) The MZI array for 4×4 multiplication [16].



(a) (b)

Fig. 2: The applied power vs. the corresponding phase change in five different MZIs under process variations [16].

introduced to the necessary locations to achieve a balance between robustness and computational cost.

- Process variations of MZIs are calibrated by differential testing. The phases of MZIs are then tuned according to the obtained characteristic curves. Besides, online tuning is used to improve inference accuracy further.

The rest of this paper is organized as follows: Section II introduces the preliminaries for RRAM & ONN IMC and the challenges imposed by process variations and noise. Section III introduces three different frameworks tackling process variations and noise, and then the conclusion is drawn in Section IV.

II. Preliminaries

In this section, the computing mechanisms of RRAM, and ONN based accelerators are introduced. Besides, the challenges of process variations and the accuracy degradation of NNs, when deployed on IMC accelerators, are presented.

A. ONN Grid Architecture & Process Variations

An MZI is the basic computing unit for an ONN. As Figure 1(a) shows, an MZI has two input ports and two output ports and consists of two beam splitters and a phase shifter. The phase value of the phase shifter ϕ is programmable by modifying the temperature of the phase shifter through the application of optical power. An MZI performs a 2×2 matrix multiplication in (1), which transforms L_1^c and L_2^c to $L_1'^c$ and $L_2'^c$, where c denotes the complex representation of the light signals, and T is the unitary transformation matrix [17].

$$\begin{bmatrix} L_1'^c \\ L_2'^c \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{j}{\sqrt{2}} \\ \frac{j}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} e^{j\phi} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{j}{\sqrt{2}} \\ \frac{j}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} L_1^c \\ L_2^c \end{bmatrix} \quad (1)$$

$$= j e^{\frac{j\phi}{2}} \begin{bmatrix} \sin\frac{\phi}{2} & \cos\frac{\phi}{2} \\ \cos\frac{\phi}{2} & -\sin\frac{\phi}{2} \end{bmatrix} \begin{bmatrix} L_1^c \\ L_2^c \end{bmatrix} = \mathbf{T} \begin{bmatrix} L_1^c \\ L_2^c \end{bmatrix} \quad (2)$$

A large matrix can be implemented by a grid-like structure of MZIs as shown in Figure 1(b) with a transformation matrix in (3).

$$\mathbf{T} = \mathbf{T}_{C_4} \mathbf{T}_{C_3} \mathbf{T}_{C_2} \mathbf{T}_{C_1} \quad (3)$$

$$\mathbf{T}_{C_1} = \begin{bmatrix} \mathbf{T}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_2 \end{bmatrix}, \quad \mathbf{T}_{C_2} = \begin{bmatrix} 1 & \mathbf{0} & 0 \\ \mathbf{0} & \mathbf{T}_3 & \mathbf{0} \\ 0 & \mathbf{0} & 1 \end{bmatrix}$$

To represent the weights in NNs, the phases ϕ of individual MZIs are tuned by the application of optical power. However, due to process variations and thermal effects, the tuned value deviates from the nominal one. Trigonometric functions in the transformation matrix of an MZI make it very sensitive to process variations. Figure 2(a) shows the curves representing the relation between the applied power p and the tuned phase ϕ for five different MZIs [18]. Even for the same applied power, the tuned phases are different for different MZIs.

Figure 2(b) shows the inference accuracy degradation for ONNs under variations. The tested NN is LeNet-5 over CIFAR10 dataset. The ONN becomes unusable even at small variations levels which poses a challenge in applying ONNs in practice.

B. RRAM Crossbar & Process Variations

Figure 3 shows the architecture of an RRAM crossbar. An RRAM cell resides at each crossing point between horizontal wordlines and vertical bitlines. Voltages are applied to the wordlines representing an input vector. Conductances of RRAM cells represent the weight matrix of NNs. Accordingly, the accumulated currents over the bitlines represent the matrix-vector multiplication result.

However, due to the process variations and noise, the programmed conductance values of RRAM cells deviate from the nominal values obtained from the software training. This weight deviation leads to a degradation of the inference accuracy. Accordingly, the programmed conductance value written to an RRAM cell is represented as a random distribution rather than a deterministic value.

Figure 4 shows the inference accuracy degradation for LeNet-5 and VGG16 on MNIST, CIFAR10, and CIFAR100 datasets. The RRAM conductance is assumed to follow the lognormal distribution in (4) in this experiment, which is widely adopted for RRAMs [11], [15], [10]. The x-axis represents the standard deviation σ which defines the level of variations. The solid lines in the middle of the ranges represent the mean values of the inference accuracy and the ranges represent the standard deviation.

$$w = w_{nominal} * e^{\theta}, \quad \theta \sim N(0, \sigma^2) \quad (4)$$

Accordingly, the inference accuracy of IMC accelerators, both on ONNs and RRAM crossbar, suffers from a large degradation. The degradation is caused by the inability to reflect the weight matrices of the NN precisely. As a result, new methods should be introduced to counter such uncertainties.

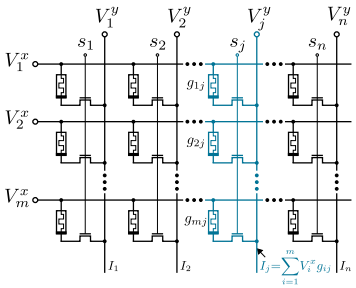


Fig. 3: The RRAM crossbar [19].

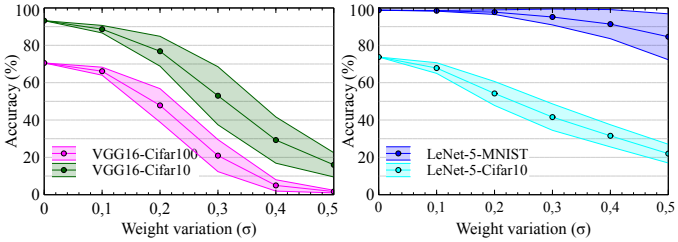


Fig. 4: Inference accuracy degradation under variations in RRAM crossbars [19].

III. Methods & Results

The effort to counter the effects of weight variations in IMC accelerators is directed in three parallel aspects. The first aspect is modifying software training of NNs to accommodate for weight variations on IMC accelerators. However, such approach works better for small variation levels and shallow NNs. The second parallel aspect is the usage of additional computations to compensate for weight uncertainties to recover the inference accuracy. This aspect incurs additional hardware cost which needs to be optimized. The last aspect is the usage of a test procedure to determine the post-deployment actual variations on chip and measure its values, so that these variations can be calibrated and further compensated by fine-tuning. In the following, we present three parallel approaches covering the aforementioned aspects and the test results on different IMC platforms.

A. Statistical Training under Weight Variations & Noise

Process variations include correlated local variations and global variations. Noise results from imprecise programming process. Process variations and noise should be modeled mathematically to be incorporated in software training. A comprehensive model is required to be developed and used for statistical training of NN considering weight variations. This model can be represented in a canonical form as follows:

$$w_i = w_{i,0} + \sum_{k=1}^{N} w_{i,k} B_k + w_{i,n} N_i \tag{5}$$

where $w_{i,0}$ is the nominal value of the NN weight, and w_i is the actual NN weight under variations. B is a set of independent random variables shared between all weights, and $w_{i,k}$ are constant coefficients. N_i represents an independent random variable for each individual weight to represent individual programming deviations and $w_{i,n}$ is the coefficient.

In order to incorporate the model in (5) in software training, the basic operations in NNs need to be modified. The basic operations in NNs include multiplication, addition, activation function and cost function. The following describes how these operations are modified according to the canonical representation of weights.

Multiplication operation is now done between two terms in a canonical form, namely a weight and an input feature. The input feature is the computation result of previous layers, so it is in the same form as (5). The multiplication is then executed as follows:

$$\begin{aligned} a_j \cdot w_i &= (a_{j,0} + \sum_{k=1}^{N} a_{j,k} B_k + a_{j,n} N_j)(w_{i,0} + \sum_{k=1}^{N} w_{i,k} B_k + w_{i,n} N_i) \\ &\approx a_{j,0} w_{i,0} + \sum_{k=1}^{N} (a_{j,0} w_{i,k} + a_{j,k} w_{i,0}) B_k + \\ &\quad + \sqrt{(a_{j,0} w_{i,n})^2 + (a_{j,n} w_{i,0})^2} N_k. \end{aligned} \tag{6}$$

In the result, only the first-order terms are kept. The high-order terms are neglected to reduce complexity. The term $a_{j,0} w_{i,n} N_i + a_{j,n} w_{i,0} N_j$ is represented as $\sqrt{(a_{j,0} w_{i,n})^2 + (a_{j,n} w_{i,0})^2} N_k$ by combining the two random variables into one N_k by matching the variances. Accordingly, the result is in the canonical form. The **Addition** of two operands in the canonical form can be described as:

$$a_i + a_j = (a_{i,0} + a_{j,0}) + \sum_{k=1}^{N} (a_{i,k} + a_{j,k}) B_k + \sqrt{a_{i,n}^2 + a_{j,n}^2} N_k, \tag{7}$$

where N_k is the single random variable combining the addition of the N_i,N_j terms and matching the variances.

Non-linear differentiable **activation functions** are expanded using Taylor series at the nominal value, $a_{i,0}$, and the high-order terms in the Taylor expansion are neglected to match the canonical form. This approach is not suitable for non-differentiable activation functions at some points, e.g., ReLU at 0.

Accordingly, the NN output is represented in the canonical form and is a distribution rather than a deterministic value. As a result the loss function used to train the NN model needs to be modified. The original cost function is as follows:

$$L = \sum_{i=1}^{M} (-\hat{Y}_i \log(Y_i) - (1 - \hat{Y}_i) \log(1 - Y_i)) \tag{8}$$

where M is the number of classes, $\hat{Y}_i$ is the true output from the dataset, Y_i is the i-th output of the NN. In statistical training, Y_i is represented as distribution in the canonical form. So the goal is to push the distribution to be sharply around the correct value $\hat{Y}_i$. This is done by pushing the mean value of Y_i to be near the correct output and punishing the distribution of

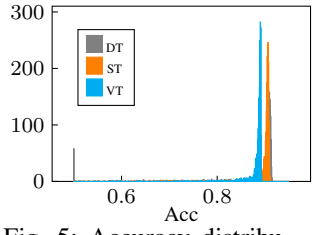


Fig. 5: Accuracy distributions [22].

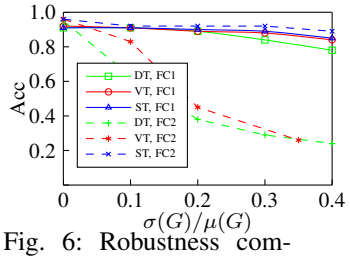


Fig. 6: Robustness comparison [22].

Y_i when widely spread away from the correct value as follows:

$$L=\sum_{i=1}^{M}(-\hat{Y}_i P^p(Y_i\leq 0.5)\log(\mu_{Y_i}) -(1-\hat{Y}_i)P^p(Y_i\geq 0.5)\log(1-\mu_{Y_i})) \quad (9)$$

The statistical training is tested on RRAM based framework. The standard deviations of the distribution of process variations and noise were set to 25% and 5% of the nominal values [20], [21], respectively. 2000 chips were simulated and tested with Monte Carlo simulations for 1-layer and 2-layer fully-connected networks denoted as FC1, FC2. The results are shown in shown in Figures 5,6. In the two figures, μ and σ are the mean and standard deviation of the inference accuracy of the simulated 2000 chips. The statistical training is denoted as ST. Compared with traditional training DT and the VT method [10], the robustness enhancement with ST is shown by the higher mean inference accuracy and the narrower distribution.

B. Error Suppression & Compensation for IMC

The modification of the training method might be sufficient to counter variations in shallow NNs. In deep NNs, the errors in the computation caused by weight variations go through deep erroneous layers causing severe accuracy degradation.

A framework is introduced combining a modified training procedure and additional computation cost to enhance robustness of IMC for NNs. First, a modified Lipschitz regularization prevents error amplification through erroneous NN layers with deviated weights. Then error compensation layers are introduced to correct erroneous feature maps at needed locations decided by reinforcement learning (RL) to minimize the additional overhead.

The Lipschitz constant defines how a difference in the input of a function is amplified or suppressed in its output as follows:

$$|f(\mathbf{x_1})-f(\mathbf{x_2})|_p\leq k|\mathbf{x_1}-\mathbf{x_2}|_p, \forall \mathbf{x_1},\mathbf{x_2}\in X \quad (10)$$

where $|\cdot|_p$ defines the p-distance between two vectors. The smallest value of k holding the inequality in (10) is the Lipschitz constant of f. For a composition of functions, the Lipschitz constant of the overall function is upper bounded by the multiplication of the Lipschitz constants of the individual composing functions as follows:

$$f=(f_l\circ f_{l-1}\circ ...\circ f_1)(x) \quad (11)$$
$$L(f)\leq k_l\cdot k_{l-1}\cdot ....\cdot k_1. \quad (12)$$

The function of an NN can be considered as a composition

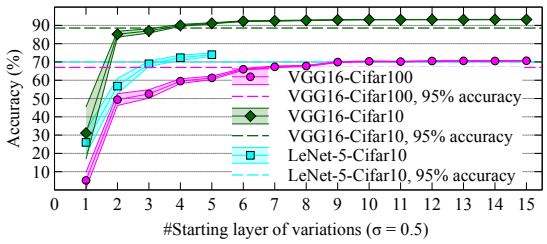


Fig. 7: Lipschitz constant regularization against variations from a given layer to the last layer [19].

of the function of each layer, which includes a matrix-vector multiplication and an activation function. Accordingly, the concept of preventing the error amplification in the NN forward path by limiting the Lipschitz constant of each layer can be adopted. The Lipschitz constant of widely used activation functions, e.g., ReLU, is bounded by 1. The problem is then reduced to bound the Lipschitz constant of the matrix-vector multiplication of each individual layer.

For the ith layer, assume the nominal input is x_1 and the input x_2 is the one affected by variation in the first $i-1$ layers when deployed in IMC accelerator. The matrix-vector computation of the ith layer can be written as $f_i'=\mathbf{w}\circ\mathbf{e}^{\boldsymbol{\theta}}\cdot\mathbf{x}+\mathbf{b}$ where $\mathbf{w}$ is the nominal weight matrix, $\mathbf{b}$ is the bias vector, and $\mathbf{w}\circ\mathbf{e}^{\boldsymbol{\theta}}$ incorporates the effect of variations into RRAM devices. To restrict the error amplification, the Lipschitz constant of f_i' need to be constrained as follows:

$$\left|\mathbf{w}\circ\mathbf{e}^{\boldsymbol{\theta}}\cdot(\mathbf{x_1}-\mathbf{x_2})\right|_p\leq k|\mathbf{x_1}-\mathbf{x_2}|_p \quad (13)$$
$$\frac{\left|\mathbf{w}\circ\mathbf{e}^{\boldsymbol{\theta}}\cdot(\mathbf{x_1}-\mathbf{x_2})\right|_p}{|\mathbf{x_1}-\mathbf{x_2}|_p}\leq k. \quad (14)$$

$$sup\left(\frac{\left|\mathbf{w}\circ\mathbf{e}^{\boldsymbol{\theta}}\cdot(\mathbf{x_1}-\mathbf{x_2})\right|_p}{|\mathbf{x_1}-\mathbf{x_2}|_p}\right)=\|\mathbf{w}\circ\mathbf{e}^{\boldsymbol{\theta}}\|_p\leq k, \quad (15)$$

where $\|\mathbf{w}\circ\mathbf{e}^{\boldsymbol{\theta}}\|_p$ cannot be reduced to a closed expression as $\boldsymbol{e^{\theta}}$ in (15) corresponds to a matrix of independent random variables. As a result, $\mu_{e^\theta}+3\cdot\sigma_{e^\theta}$ is used as a bound estimate for e^θ. Since e^θ follows a lognormal distribution, $\mu_{e^\theta}+3\cdot\sigma_{e^\theta}=e^{\frac{\sigma^2}{2}}+3\sqrt{(e^{\sigma^2}-1)e^{\sigma^2}}$, in which σ is the standard deviation of θ. (15) can be converted as follows:

$$\|\mathbf{w}\|_p\leq\lambda,\quad \lambda=\frac{k}{e^{\frac{\sigma^2}{2}}+3\sqrt{(e^{\sigma^2}-1)e^{\sigma^2}}}. \quad (16)$$

The spectral norm L^2 of $\mathbf{w}$, defined as the maximum singular value of the matrix, is used to limit $\mathbf{w}$ in (16). Accordingly, a regularization term is added to the loss function during training to limit its maximum singular value by λ as follows:

$$Loss=L_{ce}+\beta*\sum_{\mathbf{w_i}\in\mathbf{W}}\|\mathbf{w_i^T}\mathbf{w_i}-\lambda^2\mathbf{I}\|^2 \quad (17)$$

where L_{ce} is the cross-entropy loss function, $\mathbf{W}$ is the set of weight matrices of the NN, $\mathbf{w_i}$ is the weight matrix of the ith layer, and β is a regularization hyperparameter. k is set to 1 to calculate λ, so that errors will not be amplified.

To show the effect of error suppression during inference,

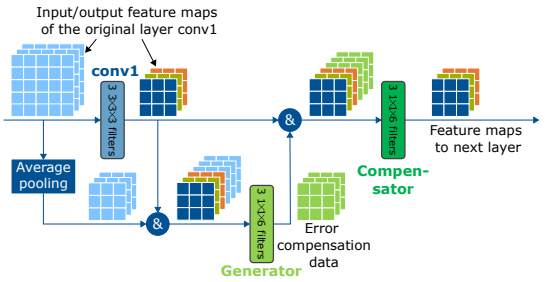


Fig. 8: Error compensation for a convolutional layer [19].

variations are added to NNs, which were trained under Lipschitz regularization, from the i-th layer till the last layer with σ=0.5. The result is shown in Figure 7. For each point, 250 samples were evaluated. The average and the standard deviation of the inference accuracy are shown using the points and the error ranges. It is clear that the regularization can counter large variations in the late layers, but the inference accuracy still remains sensitive to variations in the early layers.

In order to tackle this sensitivity to variations, light-weight compensation layers are introduced in early layers. The idea is inspired by the concept of error correction in communication system introduced as early as in [23]. The error compensation layer consists of two parts: a generator and a compensator. The generator receives both the input and the output of the intended compensated layer and then generates compensation data to correct errors. The compensator receives the compensation data and the original output of the layer and generates a corrected output.

The structures of the generator and the compensator are shown in Figure 8. The generator uses m $1\times1\times(l+n)$ filters, where l and n are the numbers of the input and output feature maps in the intended compensated layer, respectively. The usage of 1×1 filters has two advantages. First, the computational overhead is low. Second, the compensation data has the same dimensions as the original output, so that the compensator also uses 1×1 filters. The generator's number of filters m represents the number of output feature maps produced by the generator, e.g., 3 in Figure 8. A larger m indicates more computational overhead, but also a more potential robustness in the inference accuracy. The compensator also uses n $1\times1\times(n+m)$ to generate the same number of feature maps as the original output. m and the locations of compensation layers are determined with Reinforcement Learning, as described in [19].

When inserted in the original NN, the compensation layers are trained while keeping the backbone NN non-trainable and all the layers in the original NN are subject to variations.

The proposed framework, denoted as CorrectNet, is evaluated over the VGG16 and LeNet-5 NNs against three different datasets, CIFAR100, CIFAR10 and MNIST under the lognormal variation model in (4). The inference accuracy at different variations levels and the computational overhead are reported in Figure 9. The lines represent the average of the inference accuracy of 250 simulated samples, and the ranges represent the standard deviation of the inference accuracy. Results demonstrate that the inference accuracy could be

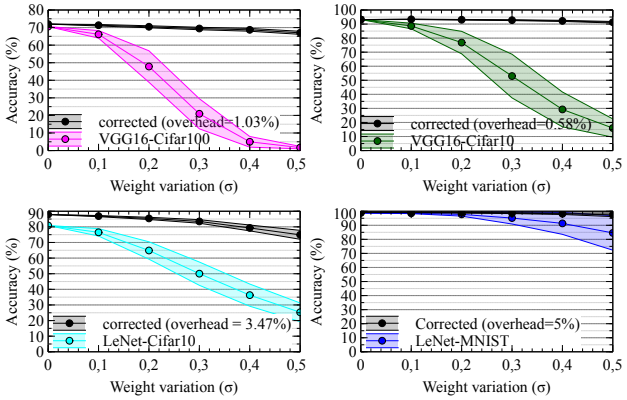


Fig. 9: Accuracy of CorrectNet under different variations [19].

recovered from as low as 1.69% to more than 95% of their original accuracy, while hardware cost is low.

C. Variation Calibration and Online Tuning for IMC

The last parallel direction to counter process variations is testing and calibration. This can be done through characterizing individual elements in IMC accelerator and then tuning the weight accordingly. In the following, a calibration process to extract characteristic curves of MZIs through the application of differential test patterns in ONNs is presented.

The testing procedure is illustrated in Figure 10, where a grid-structure composed of 4 MZI columns is presented and the column under test is assumed to be column 4. A test pattern forming an identity matrix is applied and the output vectors are stored. The stored output vectors represent the multiplication of the transformation matrix and an identity matrix. In this way, the actual transformation matrix $\mathbf{M}=\mathbf{T}_{C_4}\mathbf{T}_{C_3}\mathbf{T}_{C_2}\mathbf{T}_{C_1}$ is obtained. Next, the phases of MZIs in the column under test are modified by applying the same amount of power. After that, the identity pattern is applied again to extract the new transformation matrix $\mathbf{M}'$. Since all the individual matrices of each column is unitary, then $\mathbf{M}'\mathbf{M}^{-1}=\mathbf{T}'_{C_4}\mathbf{T}^{-1}_{C_4}=\mathbf{T}'_{C_4}\mathbf{T}^{*}_{C_4}$. This expression can be expanded using the column transformation matrix further as follows:

$$\mathbf{T}'_{C_4}\mathbf{T}^{*}_{C_4}=\begin{bmatrix}1 & \mathbf{0} & 0\\ \mathbf{0} & \mathbf{T}'_6 & \mathbf{0}\\ 0 & \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}1 & \mathbf{0} & 0\\ \mathbf{0} & \mathbf{T}^{*}_6 & \mathbf{0}\\ 0 & \mathbf{0} & 1\end{bmatrix}=\begin{bmatrix}1 & \mathbf{0} & 0\\ \mathbf{0} & \mathbf{T}'_6\mathbf{T}^{*}_6 & \mathbf{0}\\ 0 & \mathbf{0} & 1\end{bmatrix} \tag{18}$$

$$\mathbf{T}'_6\mathbf{T}^{*}_6=e^{\frac{j(\phi'-\phi)}{2}}\begin{bmatrix}\cos\frac{\phi'-\phi}{2} & \sin\frac{\phi'-\phi}{2}\\ -\sin\frac{\phi'-\phi}{2} & \cos\frac{\phi'-\phi}{2}\end{bmatrix}, \tag{19}$$

where $\mathbf{T}_6$,$\mathbf{T}'_6$ corresponds to the individual transformation matrix of the MZI at each test case. Since $\mathbf{M}$ and $\mathbf{M}'$ are known, then the phase difference $\phi'-\phi$ can be deduced by matching elements of the matrix. By repeating the process at given different optical power levels p_1,p_2,p_3, and recording the phase changes Δ_{ϕ_1},Δ_{ϕ_2},Δ_{ϕ_3}, the most appropriate characteristic curve can be determined according to Figure 2. Using the matched characterization curves, for a given required phase, the needed optical power of each individual MZI can be

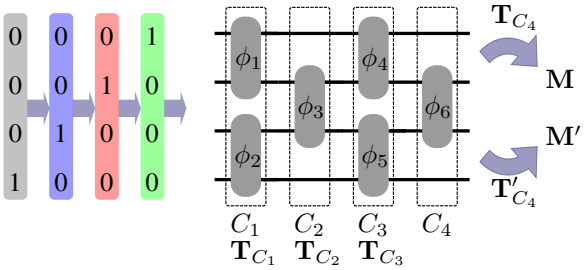


Fig. 10: Differential test for MZIs process variations [16].

TABLE I: Results of ONNs with the proposed framework [16]

		Acc. Software Training		Acc. ONN_MZI	
NN	Dataset	NN	ONN	μ	σ
FCNN	MNIST	97.40%	97.13%	92.80%	0.6%
LeNet-5	Cifar10	76.56%	75.91%	74.11%	0.26%
Aug.LeNet-5	Cifar10	82.81%	82.08%	80.80%	0.22%

obtained accurately. To further improve the inference accuracy, online tuning is performed after MZI phases are configured. The online tuning is done for a specific number of iterations. The online tuning is done based on randomized gradients stored during the offline training. In each gradient sample, a small amount of power change is applied to each individual MZI, and the direction of this tuning is determined by the gradient sample which achieves the highest inference accuracy.

The proposed framework is evaluated with 3 NNs: a 2-layer fully connected NN, LeNet-5, and Augmented LeNet-5 with 4 convolutional layers and 3 fully connected layers. The tested datasets are MNIST and CIFAR10. 100 samples were simulated for each ONN. The average and the standard deviation of the inference accuracy were evaluated. The process variations were modeled as independent Gaussian distributions for each MZI. 1000 characteristic curves were used to calibrate the process variations. The results are shown in Table I, where software training reports the accuracy of NNs when deployed on a GPU, and ONN reports the accuracy without considering variations, and the last two columns report the inference accuracy with the proposed framework under variations. The results show that the calibration and online tuning can recover the inference accuracy under variations.

IV. Conclusion

Process variations and noise have been a problem hindering the use of IMC platforms for implementing NNs efficiently due to the inference accuracy degradation. We presented three different frameworks to tackle this problem for two different IMC platforms, namely, RRAM and ONN accelerators. The frameworks cover three parallel enhancement directions ranging from modified training procedure, additional computational hardware, and online testing and calibration. The presented frameworks show the potential to rescue the inference accuracy of IMC accelerators to the accuracy after software training.

Acknowledgement

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-IDs 457473137 and 497488621.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[2] C.-C. Chiu, T. N. Sainath, Y. Wu *et al.*, "State-of-the-art speech recognition with sequence-to-sequence models," 2018.

[3] Y. Shen, N. C. Harris, S. Skirlo *et al.*, "Deep learning with coherent nanophotonic circuits," *naturephotonics*, vol. 11, pp. 441–446, 2017.

[4] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, "Optimal design for universal multiport interferometers," *Optica*, vol. 3, no. 12, pp. 1460–1465, 2016.

[5] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *International Symposium on Computer Architecture (ISCA)*, 2016.

[6] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *International Symposium on Computer Architecture (ISCA)*, 2016.

[7] S. Zhang, G. L. Zhang, B. Li, H. H. Li, and U. Schlichtmann, "Lifetime enhancement for rram-based computing-in-memory engine considering aging and thermal effects," in *International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020.

[8] ——, "Aging-aware lifetime enhancement for memristor-based neuromorphic computing," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2019.

[9] D. Niu, Y. Chen, C. Xu, and Y. Xie, "Impact of process variations on emerging memristor," in *Design Automation Conference (DAC)*, 2010.

[10] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: Variation-aware training for memristor X-bar," in *Design Automation Conference (DAC)*, 2015.

[11] Y. Long, X. She, and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," in *Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

[12] D. Gao, Q. Huang, G. L. Zhang, X. Yin, B. Li, U. Schlichtmann, and C. Zhuo, "Bayesian inference based robust computing on memristor crossbar," in *Design Automation Conference (DAC)*, 2021.

[13] G. Charan, J. Hazra, K. Beckmann *et al.*, "Accurate inference with inaccurate RRAM devices: Statistical data, model transfer, and on-line adaptation," in *Design Automation Conference (DAC)*, 2020.

[14] A. Mohanty, X. Du, P.-Y. Chen, J.-S. Seo, S. Yu, and Y. Cao, "Random sparse adaptation for accurate inference with inaccurate multi-level RRAM arrays," in *IEEE Int. Electron Dev. Meeting (IEDM)*, 2017.

[15] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar," in *Automation & Test in Europe Conference & Exhibition (DATE)*, 2017.

[16] Y. Zhu, G. L. Zhang, B. Li, X. Yin, C. Zhuo, H. Gu, T.-Y. Ho, and U. Schlichtmann, "Countering variations and thermal effects for accurate optical neural networks," in *International Conference On Computer Aided Design (ICCAD)*, 2020.

[17] M. Y.-S. Fang, S. Manipatruni, C. Wierzynski, A. Khosrowshahi, and M. R. DeWeese, "Design of optical neural networks with component imprecisions," *Opt. Express*, vol. 27, pp. 14 009–14 029, 2019.

[18] N. C. Harris, Y. Ma, J. Mower *et al.*, "Efficient, compact and low loss thermo-optic phase shifter in silicon," *Opt. Express*, vol. 22, no. 9, pp. 10 487–10 493, 2014.

[19] A. Eldebiky, G. L. Zhang, G. Böcherer, B. Li, and U. Schlichtmann, "CorrectNet: Robustness enhancement of analog in-memory computing for neural networks by error suppression and compensation," in *Automation & Test in Europe Conference & Exhibition (DATE)*, 2023.

[20] E. Amat, A. Rubio *et al.*, "Memristive crossbar memory lifetime evaluation and reconfiguration strategies," *IEEE Trans. Emerg. Topics in Comput.*, vol. 6, no. 2, pp. 207–218, 2016.

[21] S. Choi, Y. Yang, and W. Lu, "Random telegraph noise and resistance switching analysis of oxide based resistive memory," *Nanoscale*, vol. 6, no. 1, pp. 400–404, 2014.

[22] Y. Zhu, G. L. Zhang, T. Wang, B. Li, Y. Shi, T.-Y. Ho, and U. Schlichtmann, "Statistical training for neuromorphic computing using memristor-based crossbars considering process variations and noise," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.

[23] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.