# Vortex: Variation-aware Training for Memristor X-bar

Beiye Liu[1], Hai Li[5], Yiran Chen[6]
ECE, University of Pittsburgh
Pittsburgh, PA, 15261
{bel34[1], hal66[5], yic52[6]}@pitt.edu

Xin Li[2]
ECE, Carnegie Mellon University
Pittsburgh, PA, 15213
xinli@cmu.edu

Qing Wu[3]
Air Force Research Laboratory
Rome, NY, 13441-4505
qing.wu@us.af.mil

Tingwen Huang[4]
Texas A&M University
Doha, Qatar
tingwen.huang@qatar.tamu.edu

## ABSTRACT

Recent advances in development of memristor devices and crossbar integration allow us to implement a low-power on-chip neuromorphic computing system (NCS) with small footprint. Training methods have been proposed to program the memristors in a crossbar by following existing training algorithms in neural network models. However, the robustness of these training methods has not been well investigated by taking into account the limits imposed by realistic hardware implementations. In this work, we present a quantitative analysis on the impact of device imperfections and circuit design constraints on the robustness of two popular training methods – "close-loop on-device" (CLD) and "open-loop off-device" (OLD). A novel variation-aware training scheme, namely, ***Vortex***, is then invented to enhance the training robustness of memristor crossbar-based NCS by actively compensating the impact of device variations and optimizing the mapping scheme from computations to crossbars. On average, Vortex can significantly improve the test rate by 29.6% and 26.4%, compared to the traditional OLD and CLD, respectively.

## 1. INTRODUCTION

Computer systems have been experiencing great revolutions in two fundamental areas: semiconductor manufacturing and computing architecture. On the one hand, the scaling of conventional CMOS devices is approaching the limit [1]. Emerging devices, such as spintronic [2] and resistive devices (memristor) [3], have been extensively investigated and studied; On the other hand, the von Neumann architecture faces well known "memory wall" challenge [4]. The gap between CPU performance and memory bandwidth keeps increasing in many emerging applications, e.g., deep neural networks that can easily have more than 1 billion training parameters [5]. These challenges motivated the investment on new computing architectures, including neuromorphic computing systems (NCS) inspired from human brains.

The invention of memristor crossbar offers a highly integrated cost-efficient solution for implementing programmable connections in neural networks [1][2][6][8]. Every input neuron can connect to all output neurons via a crossbar, and a synaptic weight is represented by the resistance of the memristor at the corresponding cross-point [3][4]. By piggybacking the training algorithms of neural network models, many training schemes were also developed to program the resistance of the memristors in the crossbar.

One popular NCS training scheme is "Close-Loop on-Device" (CLD) method, which directly implements a gradient descent training algorithm on the memristor crossbar by repeating the loop of "programming and sensing" [9]. Another popular scheme is "Open-Loop off-Device" (OLD) method, which pre-calculates the programming pulse width/magnitude of each memristor based on the target resistance value and then programs every device according to the calculations [10]. CLD is able to adaptively adjust the training inputs to tolerate the variability of memristors, though it requires accurately sensing the memristor (output current from the crossbar) in the real-time. Instead, OLD does not need to monitor the obtained output current and eliminates the costly feedback control and high-resolution analog-digital converter (ADC), etc. However, the quality of OLD is vulnerable to memristor device variations because these variations are unknown when pre-calculating the training pulses.

The computations in a memristor crossbar-based NCS are executed in analog form, which makes the computational accuracy of the NCS very sensitive to some realistic hardware design factors, e.g., device variability, IR-drop, sneak paths, peripheral circuit constraints, etc. In this work, we perform an insightful analysis on the impacts of hardware design factors on the training quality of NCS. Based on our analysis, we propose a novel and robust variation-aware off-device training scheme, namely, ***Vortex***: Vortex first modifies the programming pre-calculation algorithm to compensate the impact of memristor variations. It then introduces an adaptive mapping process that can selectively map the synapses having large impacts on network output onto the memristors with small variations. Experimental results show combining these two techniques can significantly improve the training quality of NCS.

## 2. PRELIMINARY

### 2.1 Memristor Basics

Predicted by Prof. Leon Chua, the memristor is the fourth fundamental circuit element uniquely defining the relationship between magnetic flux and electrical charge [3]. In 2008, HP Labs reported that the memristive effect was realized by moving the doping front along a $TiO_2$ thin-film device [11]. The resistance of a memristor can be controlled by adjusting the magnitude and pulse width of programmed current/voltage, as shown in Fig. 1 (a) [12].

A nanoscale memristor device suffers from two types of variabilities: parametric variation and switching variation. The parametric variation is a device-to-device variation caused by fabrication imperfection. In Fig. 1 (c), for example, if we program multiple memristors to low resistance state (LRS), the programmed resistances of different memristors follow a lognormal distribution [14]. The switching variation, on the other hand, denotes the cycle-to-

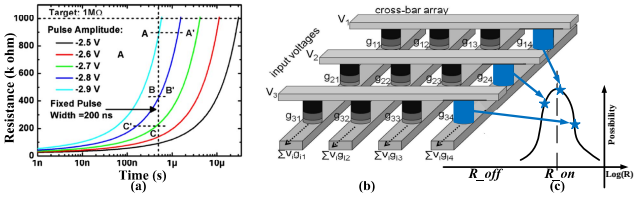


Fig. 1: (a) Analogue resistance switching of memristor [12]. (b) Memristor crossbar for [4]. (c) Parametric variation.


DAC '15, June 07 - 11, 2015, San Francisco, CA, USA

ACM 978-1-4503-3520-1/15/06 $15.00
http://dx.doi.org/10.1145/2744769.2744930

cycle variation of a single device. Programming a memristor from the same initial state to the same target state may end up with different resistances. The switching variations are mainly caused by driving circuit design. The small fluctuation in programming pulses usually makes the switching variations negligible compared with the parametric variations [14]. In this paper, our design only considers, but is not restricted to the parametric variations.

Fig. 1(b) depicts a memristor crossbar where a memristor with conductivity $w_{ij}$ is sandwiched between two metal wires, i.e., the *i-th* horizontal and the *j-th* vertical wires [2]. Input of the memristor crossbar is the digital or analog voltage applied on every horizontal wire while its output is the current flowing on every vertical wire. According to different targeted functions, the output current of the crossbar will be processed by different sensing devices, such as a comparator [7][2], "sign function" circuit [2], and an ADC [9]. Without loss of generality, in this work, we adopt the setup of digital input voltage and ADC sensing device. Note that the sensing device may be shared by multiple memristor columns, as the operations of each column are independent.

## 2.2 Memristor Crossbar Based NCS

### *2.2.1 Computing process*

In the ideal case, when setting input voltages to certain values representing a vector *x*, the output current *y* shows the product of a vector-matrix multiplication $y = x \cdot \boldsymbol{W}$, which is widely used in the operations of the weighted connections in neural networks. Here $\boldsymbol{W}$ is the conductance matrix of the memristor crossbar. Since the elements of $\boldsymbol{W}$ could be either positive or negative, $\boldsymbol{W}$ can be represented by two crossbars, which correspond to the absolute values of the positive and negative weights, respectively [9].

### *2.2.2 Program mechanism*

The memristors of a crossbar can be programmed by applying proper voltage pulses on the metal wires. For example, when programing the memristor $\boldsymbol{w_{ij}}$, we may apply a positive voltage *V* on the *i-th* horizontal wire and ground the *j-th* vertical wire. To avoid interference to other memristors, all the unselected wires are connected to *V*/2 so that only the memristor $\boldsymbol{w_{ij}}$ has a full voltage bias *V* while all the other memristors in the crossbar are half selected with a voltage of *V/2* [13]. Because of the nonlinearity of the voltage dependency in memristor programming, the resistances of the half-selected memristors remain almost unchanged during the programming. As shown in Fig. 1(a) [12], when the programming voltage reduces merely from 2.9V (point 'A') to 2.8V (point 'B') at a pulse width of 0.5µs, the programmed memristor resistance changes from 900KΩ to 400KΩ; if the programming voltage reduces down to the half selected one, i.e., 1.45V, the incurred memristor resistance change becomes negligible. Once the targeted memristor resistance value and the programming voltage magnitude are decided, the required programming pulse width can be obtained by referring to the switching model like Fig. 1(a).

### *2.2.3 CLD training and OLD training*

In this work, the memristor crossbar-based NCS is trained to implement a neural network for classification tasks. Similar to a network model in machine learning theory, NCS is trained and tested by two separate groups of data samples. Training process tunes the connection weights (memristor resistances) of a neural network so that certain percentage (training rate) of training samples can be successfully classified ("fitted"). In CLD [9], the gradient descent training (GDT) algorithm can be directly applied on the crossbar hardware by iteratively comparing the difference between the actual output and the target output, and adjusting the training inputs accordingly until the output converges to the target. In OLD [10], programming pulse width and magnitude are pre-calculated and applied to each memristor based on the characterized memristor switching model. The trained NCS will be then tested by the test samples. The probability of successfully classifying the test samples is denoted as "test rate".

# 3. UNDERSTANDING NCS TRAINING

Hardware training of a memristor crossbar based NCS is subject to many realistic factors and constraints. In this section, we will investigate the impacts of these limitations on the robustness of different hardware training methods. Here, the "robustness" is quantitatively measured as the test rate of a memristor crossbar-based NCS trained by a specific method.

## 3.1 Impact of Device Variation

The main difference between CLD and OLD is that CLD adaptively adjusts the programming signal during the iterations based on the sensed output current of the memristor crossbar. Memristor device variations can be naturally tolerated in this process. On the contrary, OLD determines the programming pulse magnitude and width before accessing the devices. Hence, device variations inevitably incur the discrepancy between the targeted memristor resistance and the actual programmed value.

To illustrate the impact of device variations on the training of memristor crossbars, we performed CLD and OLD on a column of 100 memristors. The nominal on- and off-state resistances of the memristor are set to 10kΩ and 1MΩ, respectively. Here we assume the memristor device variation follows a lognormal distribution [14]. It means that for an on-state memristor, its resistance $r \rightarrow e^{\theta} \cdot 10\text{k}\Omega$, where $\theta \sim N(0, \sigma^2)$. The training goal is to ensure when the input wires are all connected to 1V, the memristor column shall generate an output current of 1mA. Fig. 2 shows 1000 runs Monte-Carlo simulation results when the standard deviation $\sigma$ changes. Following the increase of $\sigma$, CLD constantly maintains a small discrepancy between the trained output and the target output while this output discrepancy keeps growing in OLD.

## 3.2 Impact of IR-drop

The resistance of the connection wires in a memristor crossbar causes the IR-drop, and consequently, the degradation of the pro-

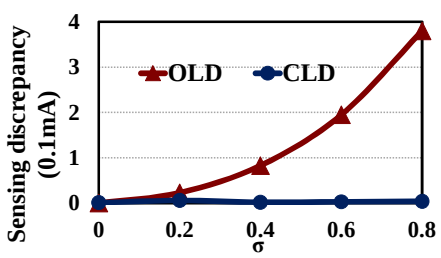


Fig. 2: Impact of process variations.

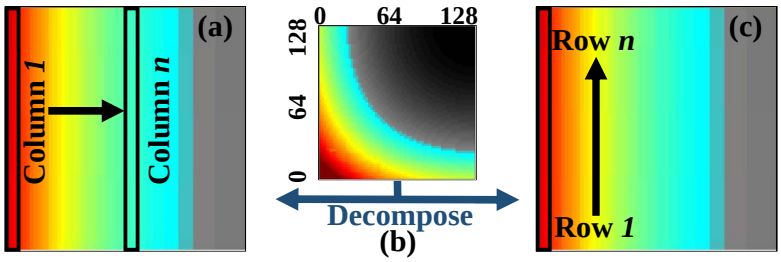


Fig. 3: IR-drop impact on CLD training: (a) Horizontal degradation. (b) Original degradation. (c) Vertical degradation.

gramming voltage reaching the memristors [10]. Fig. 3(b) conceptually shows the degradation trend of the programming voltage in a memristor crossbar by assuming all memristors are set to LRS. This trend is symmetric to the diagonal line (from left-bottom to right-top) because the crossbar is driven symmetrically from the left to the bottom, as shown in Fig.1 (b).

IR-drop shows different impacts on CLD and OLD. For OLD, IR-drop induced insufficient programming can be partially compensated with the technique proposed in [10]. Although CLD can intrinsically tolerate small irregular voltage degradation, the impact of large skewed IR-drop can still cause systematic error. We propose to mathematically decompose this trend into a combination of vertical and horizontal degradations and study their individual impact on GDT algorithm, which can be described by:

$$\boldsymbol{W} \coloneqq \boldsymbol{W} - \alpha \cdot \frac{\partial y}{\partial \boldsymbol{W}} \cdot (\hat{y} - y). \quad (1)$$

Here the output $y$ is the sensed current from the vertical wires and $\hat{y}$ is the target output pattern. $\alpha$ is the learning step. When taking into account the impact of IR-drop, the GDT algorithm for the *j-th* column of the crossbar can be expressed as:

$$\boldsymbol{W_j} \coloneqq \boldsymbol{W_j} - (\beta \cdot \alpha) \cdot \left(\frac{\partial y_j}{\partial \boldsymbol{W_j}} \cdot \boldsymbol{D}\right) \cdot (\hat{y}_j - y_j). \quad (2)$$

Here $\beta$ ($\beta < 1$) is a coefficient representing the impact of horizontal IR-drop on the learning step $\alpha$ in the GDT algorithm, as virtualized in Fig. 3(a). Since each column is trained separately, the reduction in effective learning step only changes every column's own convergence speed. Different from horizontal degradation, vertical degradation results in a gradually increased programming voltage reduction from bottom to top on the same column, as shown in Fig. 3(c). This impact can be represented by a diagonal matrix $\boldsymbol{D}$, which changes the converging direction of the GDT algorithm in Eq. (2). Here the $d_{ii}$ (the *i-th* diagonal element of $\boldsymbol{D}$, $i$=*1…n*) keeps decreasing when $i$ raises. The skewness of $\boldsymbol{D}$, i.e., $d_{nn}/d_{ii}$, increases when the scale of the crossbar $n$ rises. For example, in the worst case that all memristors are at LRS, $d_{11}/d_{nn} > 2$ when $n$>128. A high $d_{nn}/d_{ii}$ indicates a large difference between the voltage drops along the crossbar column. As discussed in Section 2.2.2, such an IR-drop induced programming voltage difference will cause significant difference in the achieved memristor resistance change during the training due to the nonlinearity of the memristor switching. Our simulation shows that in some extreme cases, the resistance change of the first memristor ($\Delta w_{1j}$) on one column can be less than 1/1000 of that of the last memristor ($\Delta w_{nj}$) on the same column during a training iteration. Note that here the value of $\boldsymbol{D}$ is determined by not only the interconnect resistance but also the memristor resistance.

## 3.3 Impact of Sensing Resolution

The nature of analog computation makes memristor crossbar-based NCS susceptible to the resolution of sensing circuitry. Sensing resolution affects not only the computational accuracy (reflected as the precision of the sensed currents) but also the training quality of the close-loop schemes. In GDT algorithm based CLD, for example, connection weights keep being updated until the sensed output equals (or becomes close enough) to the target output of the crossbar. In such a case, sensing resolution directly affects the convergence criterion and consequently, the training quality. In OLD, the training quality is irrelevant to sensing resolution, as the programming signals are pre-calculated.

# 4. VORTEX

The low hardware cost and the capability to compensate the IR-drop in the pre-calculation stage makes OLD an attractive solution in memristor crossbar-based NCS designs. However, compared with CLD, the disadvantage of OLD is also obvious: the open-loop scheme intrinsically lacks the mechanism to tolerate memristor device variations. In this work, we propose *Vortex* – a variation-aware robust training scheme that can tolerate the memristor device variations using the following two techniques:

- Variation-aware training (VAT) – an off-device training method that models the device variations and adjusts the training goal to tolerate the impact of the variations in the pre-calculation stage;
- Adaptive mapping (AMP) – a method to pre-test all devices and then adaptively map the synaptic connections to physical devices based on the actual memristors' variations.

## 4.1 Variation-aware Training (VAT)

### *4.1.1 Algorithm*

For a neural network, the goal of the conventional GDT algorithm is to find the connection weights $\boldsymbol{W}$ that can successfully classify the training samples as many as possible. The computation of the network can be expressed as $y = x \cdot \boldsymbol{W}$. Here $x$ is a $1\times n$ input feature vector, $\boldsymbol{W}$ is a $n\times m$ weight connection matrix and $y$ is a $1\times m$ output vector that corresponds to $m$ classes. Since each column of $\boldsymbol{W}$ is trained separately, the training procedure of the *r-th* column ($W_r$) can be summarized as the following optimization process:

$$Min \sum_i^s \varepsilon^{(i)} \quad S.T.\ \hat{y_r}^{(i)} \cdot \left(x^{(i)} \cdot W_r\right) \geq 1 - \varepsilon^{(i)}; \quad \varepsilon^{(i)} \geq 0. \quad (3)$$

Here the *i-th* training sample contains input feature vector $x^{(i)}$ and target output $\hat{y_r}^{(i)}$($\hat{y_r}^{(i)} \in \{-1,1\}$). $s$ is the total number of the training samples. The optimization process minimizes the difference between the actual output ($x^{(i)} \cdot W_r$) and the target put $\hat{y_r}^{(i)}$. Here we use "1 *vs. all*" method in the output neuron design: $\hat{y_r}^{(i)} = 1$ only when a training sample is labeled as class *r*.

In a memristor crossbar-based NCS, weight matrix $\boldsymbol{W}$ is represented by the crossbar. When memristor variations are taken into account, the actual programmed weight matrix $\boldsymbol{W'}$ may be different from the target $\boldsymbol{W}$ even we can perfectly control the programming voltage pulse width and magnitude. Similar to Section 3.1, here we assume the memristor device variation follows log-normal distribution [14] as $W_r' \to diag(e^{\theta}) \cdot W_r$, and $\theta_i \sim N(0, \sigma^2), i = 1 \ldots n$. The optimization constraint of Eq. (3) then becomes:

$$\hat{y_r}^{(i)} \cdot \sum_{q=1}^{n} x_q^{(i)} \cdot W_{r_q} \cdot e^{\theta_q} \geq 1 - \varepsilon^{(i)}. \quad (4)$$

As $\theta$ is a small variation with a mean of zero, we can simplify the Eq. (4) using a linear approximation as:

$$\hat{y_r}^{(i)} \cdot \sum_{q=1}^{n} x_q^{(i)} \cdot W_{r_q} \cdot (\alpha_0 + \alpha_1 \cdot \theta_q) \geq 1 - \varepsilon^{(i)}. \quad (5)$$

Eq. (5) can be further reorganized as:

$$\overbrace{\alpha_1 \cdot \hat{y_r}^{(i)} \cdot \sum_{q=1}^{n} x_q^{(i)} \cdot W_{r_q} \cdot \theta_q}^{penalty\ of\ variation} + \overbrace{\alpha_0 \cdot \hat{y_r}^{(i)} \cdot \sum_{q=1}^{n} x_q^{(i)} \cdot W_{r_q}}^{conventional\ training} \geq 1 - \varepsilon^{(i)}. \quad (6)$$

The second term of Eq. (6) is also used as the constraint in the conventional GDT algorithm. We call the first term as "penalty of variations" because it represents the sum of the crossbar output deviation induced by device variations. However, the optimization process cannot be performed with random variable $\theta_q$. Therefore, we estimate the upper bound of the penalty of variations by:

$$\sum_{q=1}^{n} x_q^{(i)} \cdot W_{r_q} \cdot \theta_q \leq \|\theta\|_2 \cdot \left\|V^{(i)}\right\|_2 \quad (7)$$

Here, $V^{(i)} = [W_r \cdot x_1^{(i)} \cdots W_r \cdot x_n^{(i)}]^T$, $\theta = [\theta_1 \cdots \theta_n]$.

$\|\theta\|_2$ is the 2-norm of a vector of random variables that follow normal distributions. At a certain confidence level, we can restrict $\|\theta\|_2 \leq \rho$ based on Chi-square distribution with degree of free-

dom $n$. Then the modified training process under the consideration of weight variations can be expressed as:

$$Min \quad \sum_i^s \varepsilon^{(i)} \text{ S.T.} \tag{8}$$

$$\begin{cases} V^{(i)} = \left[x_1{}^{(i)} \cdot W_r \cdots x_n{}^{(i)} \cdot W_r\right]^T, t^{(i)} \geq \left|V^{(i)}\right|, \varepsilon^{(i)} \geq 0 \\ \rho \cdot \left|\alpha_1 \cdot \hat{y_r}^{(i)}\right| \cdot t^{(i)} - \alpha_0 \cdot \hat{y_r}^{(i)} \cdot \sum_{q=1}^n x_q{}^{(i)} \cdot W_{r_q} + 1 - \varepsilon^{(i)} \leq 0 \end{cases} \tag{9}$$

We refer to this technique as VAT (variation-aware training).

### 4.1.2 Variation tolerance vs. Training rate

The introduction of the estimated penalty of variations makes the training procedure be aware of device variations at a predetermined degree and include them in the training constraints. The trained memristor crossbar, hence, becomes more robust in tolerating device variations during computations, allowing us to obtain the desired output even there are variations in the programmed weights. However, such a method applies a tighter constraint to the training process and results in potentially lower training rate.

To evaluate the tradeoff between the training rate and the variation tolerance of the NCS under VAT, we vary the estimated penalty of variation in Eq. (9) by multiplying a scalar $\gamma (0<\gamma<1)$:

$$\gamma \cdot \rho \cdot \left|\alpha_1 \cdot \hat{y_r}^{(i)}\right| \cdot t^{(i)} - \alpha_0 \cdot \hat{y_r}^{(i)} \cdot \sum_{q=1}^n x_q{}^{(i)} \cdot W_{r_q} + 1 - \varepsilon^{(i)} \leq 0, \tag{10}$$

and simulate the corresponding training rate and test rate. When $\gamma$ changes from 0% to 100% (none to the maximum estimated penalty of variations), the training rate keeps reducing, as depicted in Fig. 4.

The left side of Fig. 4 shows test rate (w/ variation) is significantly lower than test rate (w/o variation), indicating significant impact of device variations on the training quality in this range. When $\gamma$ rises, test rate (w/o variation) continues to decreases due to the disturbance of the introduced estimated penalty of variations to the optimization process. Test rate (w/ variation), however, raises to a peak first when $\gamma$ increases to 0.2. It clearly shows the efficacy of VAT to tolerate device variations in the training. Continue increasing $\gamma$, however, may not further improve the variation tolerance. The disturbance to the training process starts to dominate and results in a decrease of the test rate.

### 4.1.3 Self-tuning and validation

Fig. 4 shows that for a specific memristor crossbar based NCS, there exists an optimal $\gamma$ that ensures the maximum test rate (but not corresponding to the highest training rate). Hence, we propose a self-tuning process that is very similar to the regularization used in regressions to prevent over-fitting and maximize the test rate [15]. The details of the proposed process are illustrated in Fig. 5. Instead of training the neural network only once with all training samples, we separate the training samples into two groups (one is large and one is small). The large group is used as the actual "training samples" while the small group is used for "validation". After training, a validation step will be launched: we first model the memristor variations and inject them into the weight matrix $\boldsymbol{W}$

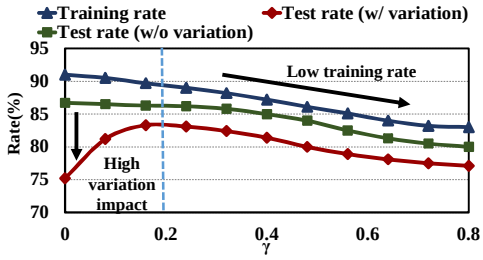


Fig. 4: Tradeoff between variation tolerance and training rate.

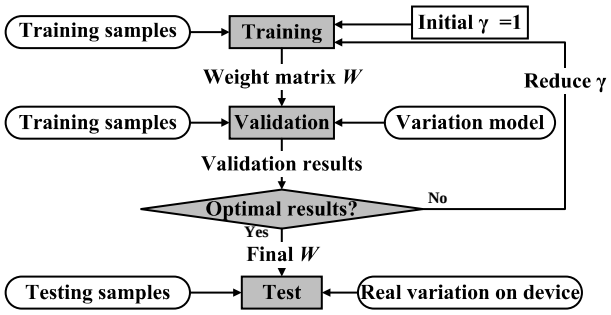


Fig. 5: Self-tuning process in training.

trained by the training samples. Then the training quality of the NCS is tested by the validation samples under a fixed $\gamma$. We repeat such training-validation loops by scanning the value of $\gamma$ until achieving the maximum test rate over all the validation samples and the corresponding $\gamma$ will be selected in the final training process. Note that the efficacy of such a self-tuning loop varies when the memristor device variation model changes. In this paper, we use the lognormal distribution as our memristor device variation model [14]. However, our proposed techniques are not restricted to any particular variation models.

## 4.2 Adaptive Mapping (AMP)

VAT aims to optimize the training algorithm to tolerate the impact of device variations. In this section, we propose adaptive mapping (AMP) – a hardware solution to mitigate the impact of device variations by optimizing the mapping scheme of the computation to the crossbar and leveraging the design redundancy.

### 4.2.1 Basic steps of AMP

**Pre-testing –** After a memristor crossbar is manufactured, we program every memristor targeting a certain resistance state and then sense the device resistance to get the distributions of memristor resistance in a crossbar (we may need to sense multiple times to eliminate the impacts of switching variations). To minimize the impact of IR-drop and sneak paths, we perform pre-testing on each individual memristor and keep all other memristors at high-resistance state (HRS). The obtained distribution should follow lognormal distribution [14].

**Sensitivity analysis** – Because of device variations, different memristors may affect the computation accuracy of a NCS differently. To identify the memristors that have large impacts on the NCS computation accuracy and need better control of device variations, a sensitivity analysis can be performed. In an $m \times n$ crossbar, the sensitivity of the $j$-$th$ output $y_j$ to the device variation of a specific memristor ($e^{\theta_{ij}}$) is:

$$\partial y_j / \partial e^{\theta_{ij}} = \frac{\partial\left(\sum_{i=1}^n x_i \cdot \left(w_{ij} \cdot e^{\theta_{ij}}\right)\right)}{\partial \theta_{ij}} = x_i \cdot w_{ij}. \tag{11}$$

Eq. (11) shows that the impact of a memristor's variations on the NCS computation accuracy is proportional to the product of the input and the weight that the memristor represents. Since the weight matrix $W$ of a neural network is often highly skewed (e.g., $\max(w_{ij})$ is easily >1000× $\min(w_{ij})$), the memristors with a low resistance and a high input demand for a better variation control.

**Mapping** – To reduce the impact of device variations on the NCS computation accuracy, we may replace a memristor that has a resistance significantly deviating from the nominal value and high impact on NCS computation accuracy with a memristor that has smaller variation. It is hard to physically replace a memristor with another after a crossbar is fabricated. However, changing the mapping relation between elements in the weight matrix and me-

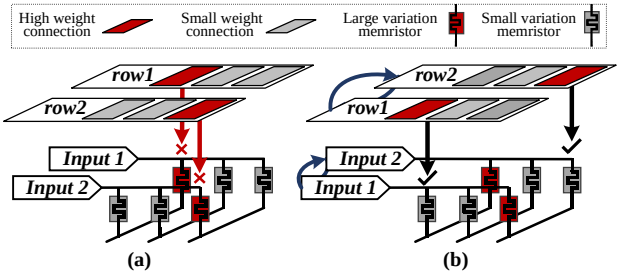


Fig. 6: Illustration of active remapping (AMP).

mristors in the crossbar can be possibly done. In observation of the multiplication of permutations of vector-matrix multiplication, switching two rows in weight matrix together with their inputs does not change the output of the multiplication. Hence, if one row (e.g., row1 in Fig. 6) in the crossbar has one memristor with large variation that matches a high weight connection, we can assign the input signals originally on row1 to the input of another row (e.g., row2 in Fig. 6) and program row2 to the original weight of row1. In this case, high weight connections and large variation memristors are intentionally mismatched.

### *4.2.2 Greedy mapping algorithm*

To further reduce the impact of memristor device variations across the whole crossbar, we could adopt a greedy mapping algorithm in AMP to determine the mapping relations between the weight matrix and the crossbar, as depicted in Algorithm 1. The whole mapping process can be summarized as follows:

We first calculate the impact of device variations by mapping the *p-th* row of weights matrix onto the *q-th* row of the memristor crossbar. As discussed in sensitivity analysis, such an impact can be measured by "summed weighted variations (SWV)" as:

$$SWV_{pq} = \sum_{j=1}^{n} \left| w_{pj} \cdot (e^{0} - e^{\theta_{qj}}) \right| = \sum_{j=1}^{n} \left| w_{pj} \cdot (1 - e^{\theta_{qj}}) \right|. \quad (12)$$

Here we assume both the crossbar and weight matrix $\boldsymbol{W}$ have total $n$ columns. $w_{pj}$ is a connection weight at the location $(p,j)$ in $\boldsymbol{W}$ and $e^{\theta_{qj}}$ represents the device variation of the memristor at the location $(q,j)$ in the crossbar. $w_{pj} \cdot (e^{0} - e^{\theta_{qj}})$ shows the difference between the ideal weight $(w_{pj} \cdot e^{0})$ and the actual weight represented by the memristor $((1 - e^{\theta_{qj}}))$. Here $\theta \sim N(0, \sigma^2)$.

The mapping starts with the row of $\boldsymbol{W}$ with the largest device variation sensitivity calculated in Eq. (11) and maps it to the row of the crossbar with the smallest SWV. After a row is mapped, its original row from $\boldsymbol{W}$ and the mapped row in the crossbar will be removed from the queue of the to-be-mapped rows. AMP will repeat this process until all the rows are properly mapped. If redundant design is allowed, we may also leverage additional memristor columns/rows to further improve the efficacy of the mapping. The mapping algorithm remains almost the same expect that there are more memristor rows are available.

Defective cell is another reliability issue in the fabrication of memristor crossbars, causing the device resistance stuck at HRS or LRS. Such defective cells can be detected as memristors with large variations and replaced by following the similar AMP process. Note that here greedy mapping algorithm is just one example of the possible AMP schemes. Other optimization algorithms can also be applied to the mapping process.

**Algorithm 1: Greedy mapping algorithm**

```
1  Xbar = abs(Ones(m×n) - Xbar_variation (m×n));
2  For (row#=1, row#<m, row#++){
3  |  Target_row# = 1;
4  |  Min_SWV = abs (Weight (row#,:) × Xbar (1,:)';
5  |  For(scaner=1, scanner<n, scanner#++){
6  |  |  If Min_SWV > abs (Weight (row#,:) × Xbar (scaner,:)');
7  |  |  |  Target_row#=scanner#;
8  |  |  End
9  |  End
10 |  Pair(row#) = Target_row#;
11 |  Remove Xbar (Target_row#);
12 End
```

## 4.3 Integration of VAT and AMP

VAT and AMP are two complementary techniques that can be seamlessly integrated while the efficacies of them are also stackable: For example, if effective device variations of the memristor crossbar have been reduced by AMP, this reduction shall be captured by the memristor device variation model that being used in the self-tuning process of VAT. As a result, a smaller penalty of variation will be introduced in VAT, leading to potentially higher training rate and test rate.

# 5. EXPERIMENTS

To evaluate our proposed *Vortex* scheme, we implement a two-layer neural network on a memristor crossbar-based NCS for the famous MNIST digits classification task [11]. The input signals of the crossbars are digital voltages corresponding to the pixels of the original benchmark images. The output signals are the currents sensed from the ten vertical wires of the crossbar; each of them represents one class from '0' to '9'. "1 *vs. all*" method is still used in output neuron designs. Each benchmark image has 28×28 pixels, requiring a 784×10 crossbar for the computation. The nominal on-state and off-state resistances of memristors used in our experiment are 10kΩ and 1MΩ, respectively. Benchmark may need to be under-sampled to fit into the memristor crossbars with difference sizes in the relevant evaluations.

## 5.1 Effectiveness of AMP

Fig. 7 illustrates the training rate of VAT and test rates of the crossbar before and after applying AMP. As expected, after AMP is applied, the impact of device variations of the crossbar decreases, resulting in the improvement of test rate w.r.t. the case before AMP is applied. Besides, the optimal $\gamma$ also reduces from 0.4 (before AMP) to 0.2 (after AMP).

## 5.2 ADC Resolution

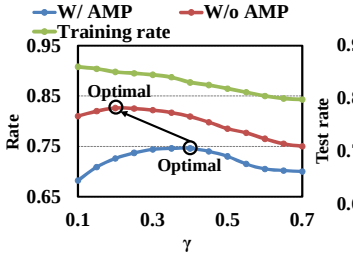


Fig. 7: Effectiveness of AMP.

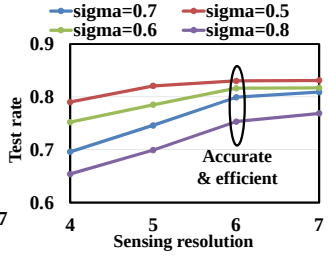


Fig. 8: ADC resolution VS test rate.

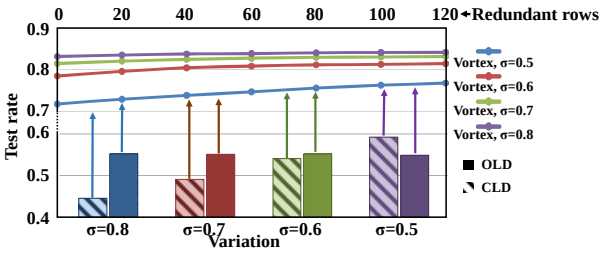


Fig. 9: Overhead vs. Test rate.

The resolution of analog-digital converters (ADC) is an important factor that affects the efficacy of AMP by influencing the memristor resistance pre-testing accuracy. We analyze the impact of different resolutions on NCS computation robustness (test rate). No redundancy is added in this analysis. Fig. 8 shows the test rates of the NCS with different ADC resolutions under different device variations. Low resolution (4-bit/5-bit) significantly limits the computation robustness. The test rates of the NCS with different variations start to saturate when a 6-bit ADC is applied. Further improving the ADC resolution gives us very marginal computation robustness enhancement. Hence, we fix the ADC resolution at 6-bit in the following experiments.

## 5.3 Design Redundancy

We analyze the tradeoff between design redundancy and NCS computation robustness using the same experiment setup in Section 5.1. When memristors have a large variation ($\sigma$=0.8) and there are no redundant rows, the test rate is generally low, i.e., 71.8%. To improve the computation robustness, we may add extra $p$ rows. Fig. 9 shows the test rates of the crossbar with different extra rows under different training schemes.

In general, increasing the redundancy ($p$) helps to improve the test rates. However, the test rates are primarily determined by the device variations rather than the redundancy, and the help of redundancy is more prominent when the device variations are large. For comparison purpose, Fig. 9 also depicts the test rates under conventional OLD and CLD without design redundancy. On average, Vortex achieves 29.6% and 26.4% higher test rates compared to OLD and CLD, respectively, even without redundant rows. Here, the theoretical maximum test rate in this configuration is ~85%, which is determined by the nature of the adopted neural network model. In the following experiments, we choose 100 redundant rows and $\sigma$=0.6 as our default setup.

## 5.4 Different Sizes

As described in Section 3, CLD has intrinsic tolerance to device variations. However, CLD is also sensitive to the effect of IR-drop in the crossbar, especially when the crossbar is large. Hence, we compare the training quality of Vortex and CLD on the memristor crossbars with different sizes. Besides 28×28 pixels, the benchmark images are also sampled to 14×14 and 7×7 pixels. The corresponding classifiers are also implemented with the memristor crossbars whose sizes are scaled according to the size of under-sampled benchmark images. All networks are trained by 4000 training samples, which are sufficient for preventing over-fitting under both Vortex and CLD. All the results here are tested by 2000 test samples.

Our experimental results are summarized in Table 1. If we do not consider the IR-drop in CLD ("CLD w/o IR-drop"), both the test rate and training rate of the NCS keep decreasing when the resolution of the benchmark images reduces. It is because of losing the features of the images during under-sampling. However, when the IR-drop is considered in CLD ("CLD w/ IR-drop"), e.g., the wire resistance $r_{wire}$=2.5Ω, the test rate of the largest crossbar (784×10) drop significantly down to 33.7%. This fact shows the susceptibility of CLD to the IR-drop. Decreasing the size of the crossbar will improve the test rate of CLD with IR-drop, however, requiring the degradation of the image resolution.

**Table 1. "Vortex" *vs.* CLD at different crossbar sizes.**

| | Test rate (%) | | | Training rate (%) | | |
|---|---|---|---|---|---|---|
| **Number of rows** | **784** | **196** | **49** | **784** | **196** | **49** |
| **CLD w/ IR-drop** | **33.7** | **54.9** | **64.5** | **35.7** | **58.9** | **72.75** |
| **Vortex w/ IR-drop** | **84.9** | **79.8** | **56.53** | **94.7** | **90.4** | **68.5** |
| **CLD w/o IR-drop** | **89.4** | **84.9** | **67.8** | **94.3** | **89.1** | **69.7** |

In contrast, Vortex has relatively low test and training rates when the size of the crossbar is small (49×10), i.e., 56.53% compared to 68.5% in CLD, respectively. When the crossbar size increases to 784×10, both test rate and training rate improve substantially to 84.9%, showing very minimum impact of IR-drop.

# 6. CONCLUSION

In this paper, we try to understand the training robustness of memristor crossbars by quantitatively analyzing the influences of some hardware limitations, e.g., IR-drop, device variation, and sensing resolution. Based on our analysis, "Vortex" – a variation-aware off-device training scheme is then developed to better tolerate device imperfections and design constraints. Experimental results show that Vortex achieves significantly improved training quality, i.e., 29.6% higher test rate, w.r.t. conventional open-loop off-device training. Vortex also shows better training quality than close-loop on-device training for large-size crossbars. Finally, as an open-loop training method, Vortex is not affected by the sensing resolution.

**Acknowledgement**

This work was supported in part by AFRL FA8750-15-2-0048, NSF CNS-1116171, NSF CNS-1342566, NSF CNS-1253424, and HP Lab Innovation Research Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of grant agencies or their contractors. Received and approved for public release by AFRL on 03/04/2015, case number 88ABW-2015-0830. Any Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or its contractors.

**REFRENCE**

[1] A. Mahesri and V. Vardhan, "Power Consumption Breakdown on A Modern Laptop," *International Power Aware Computer Systems Workshop*, 2004.

[2] M. Sharad et. al, "Ultra Low Power Associative Computing with Spin Neurons and Resistive Crossbar Memory," *DAC*, 2013.

[3] L. Chua, "Memristor-the missing circuit element," *IEEE Trans. On Circuit Theory*, 1971.

[4] A. Sally, "Reflections on the Memory Wall," *1st Conference on Computing Frontiers*, 2004.

[5] Q. Le, M. Ranzato et. al., "Building High-level Features Using Large Scale Unsupervised Learning," *ICML*, 2012.

[6] M. Hu et. al., "Hardware Realization of BSB Recall Function using," *DAC*, 2012.

[7] B. Liu et. al.,"Digital-assisted noise-eliminating training for memristor crossbar-based analog neuromorphic computing engine," *DAC*, 2013.

[8] F. Alibart et. al, "Pattern classification by memristive crossbar circuits usingex situandin situ training," *Nature*, 2013.

[9] M. Hu et. al., "BSB training scheme implementation on memristor-based circuit," in *CISDA*, 2013.

[10] B. Liu et. al., "Reduction and IR-drop Compensations Techniques for Reliable Neuromorphic Computing Systems," *ICCAD*, 2014.

[11] D. B. Strukov et. al., "The Missing Memristor Found," *Nature*, 2008.

[12] S. Yu et. al. "Investigating the switching dynamics and multilevel capability of bipolar," *APL*, 2011.

[13] J. Liang et. al, "Cross-Point Memory Array Without Cell Selectors—Device Characteristics and Data Storage Pattern Dependencies," *TED*, 2010.

[14] S. R. Lee, "Multi-level switching of triple-layered TaOx RRAM with excellent reliability for storage class memory," *VLSIT*, 2012.

[15] Y. LeCun, Y. B. L. Bottou and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE, 86(11):2278-2324*, 1998.

[16] D. W. Hosmer, S. L. Jr. and R. X. Sturdivant, Applied Logistic Regression, 3rd Edition, Wiley, 2013.