A Quantized Training Framework for Robust and Accurate ReRAM-based Neural Network Accelerators

Chenguang Zhang
School of Information Science and Technology,
ShanghaiTech University
zhangchg@shanghaitech.edu.cn

Pingqiang Zhou
School of Information Science and Technology,
ShanghaiTech University
zhoupq@shanghaitech.edu.cn

ABSTRACT

Neural networks (NN), especially deep neural networks (DNN), have achieved great success in lots of fields. ReRAM crossbar, as a promising candidate, is widely employed to accelerate neural network owing to its nature of processing MVM. However, ReRAM crossbar suffers high conductance variation due to many non-ideal effects, resulting in great inference accuracy degradation. Recent works use uniform quantization to enhance the tolerance of conductance variation, but these methods still suffer high accuracy loss with large variation. In this paper, firstly, we analyze the impact of the quantization and conductance variation on the accuracy. Then, based on two observation, we propose a quantized training framework to enhance the robustness and accuracy of the neural network running on the accelerator, by introducing a smart non-uniform quantizer. This framework consists of a robust trainable quantizer and a corresponding training method, and needs no extra hardware overhead and compatible with a standard neural network training procedure. Experimental results show that our proposed method can improve inference accuracy by 10% ∼ 30% under large variation, compared with uniform quantization method.

CCS CONCEPTS

• **Hardware → Process, voltage and temperature variations**; **Emerging technologies**.

KEYWORDS

ReRAM, Neural Network, Variation, Robust, Quantize

ACM Reference Format:
Chenguang Zhang and Pingqiang Zhou. 2021. A Quantized Training Framework for Robust and Accurate ReRAM-based Neural Network Accelerators. In *26th Asia and South Pacific Design Automation Conference (ASPDAC '21), January 18–21, 2021, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3394885.3431528

1 INTRODUCTION

Neural networks (NN), especially the deep neural networks (DNN), have achieved great success and is seen as the most promising and effective method in the broader artificial intelligence field. But both training and inference processes of NN are computing-consuming and bandwidth-intensive, because large amounts of data processing and data movements between computing units and memories are required. However, conventional CPU and GPU platforms are not able to accommodate such processes due to the well-known "power wall" and "memory wall".

Resistive Random Access Memory (ReRAM) is an emerging memory featuring high speed, low power and large capacity [1]. The crossbar-structured ReRAM can not only store the weights of the NN but also perform analog matrix-vector multiplications (MVMs) efficiently in memory. With these features, the time complexity of the MVM ($n \times n$ matrix) is reduced from $O(n^2)$ to $O(1)$, thereby achieving high-density computation with reduced data movement. With these merits, the ReRAM-based neural network accelerator has been widely studied and is proved to be a promising alternative to Von-Neumann architecture [2] [3] for this type of application.

Despite these advantages, just like other nano-scale devices, ReRAM also suffers some challenges inevitably. Non-ideal effects, such as fabrication imperfection, switching stochastic, limitation of programming facilities etc., lead to conductance variation (including device-to-device variation and pulse-to-pulse variation). That is to say, the conductance of ReRAM device always deviates from its ideal value[4]. In general, neural networks need to maintain stable and high-precision weight to guarantee high accuracy due to its high sensitivity to weight. Consequently, conductance variation drastically degrades the inference accuracy of NN that executed on the ReRAM-based accelerator[5]. For instance, with 10% maximum deviation, the recognition accuracy of ReRAM-based SVM significantly drops from 90% to only 53% [6].

To overcome conductance variation of the ReRAM devices, and achieve high reliable accurate NN computing, various solutions have been proposed, which are categorized as following two types. **Weight compensation:** Some works [5, 7] cooperate mask retraining and re-mapping methods to compensate the conductance deviation measured on a specific ReRAM crossbar. However, these methods require careful learning rate tuning, complex mask design and device-by-device conductance reading, which are either computing-intensive or time-consuming, making them highly inefficient. Besides, re-mapping methods may not work, when IR-drop effect is taken into consideration. Other works [8] [9] try to train NN on the ReRAM crossbar hardware directly with stochastic gradient decent algorithm (online train). But online training heavily relies on the accuracy of the cell-by-cell fine-tuning and measurement. Commonly, these types of methods does not tolerate stochastic noise very well. With stochastic noise taken into consideration, the number of iterations in training will be greatly increased; thus, leading to high energy cost and device endurance issue. The most

ASPDAC '21, January 18–21, 2021, Tokyo, Japan

ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431528

serious common issue of these device-level algorithm is that the cell-by-cell weights compensation of one crossbar is not reproducible in another one, the processes have to be re-conducted on another new ReRAM crossbar. Obviously, inherent robustness of NN is not exploited.

Digital crossbar: In [10–12], a high-precision value is represented by several single-bit crossbars, which is also called "ReRAM crossbar in digital mode". And [13, 14] try to incorporate quantization with training to eliminate the quantization error. Their results show improvement in conductance variation tolerance. Nevertheless, there is still a significant accuracy drop with large variation compared with the baseline model. For example, in [14] accuracy drops 7.09% (from 94.11% to 87.02%) for MNIST dataset with 10% conductance variation.

To eliminate accuracy loss brought by conductance variation, in this paper, we present a framework including the design, evaluation and training process of the quantization scheme, for NN accelerator using multi-bit crossbar. The main contributions of this paper include:

(1) We propose a quantized training framework for ReRAM-based Neural Network accelerators. The model trained by this framework, can be executed on a ReRAM-based accelerator robustly and accurately. This training framework is variation-aware, the inherent robustness of a well-trained NN is enhanced by an improved quantizer.
(2) To evaluate the robustness and accuracy of a quantization scheme, we design a metric to measure quantization error and sensitivity of a quantizer to conductance variation and make a trade-off between them.
(3) Instead of using fixed and uniform conductance level, our quantizer is designed to be trainable, based on the metric mentioned above, thereby realizing much more flexibility for exploiting accuracy-robustness trade-off.
(4) We design a corresponding heuristic optimization algorithm for the quantizer that can be integrated in standard network training procedure. The quantizer can be updated during each training phase with little extra training overhead.

2 BACKGROUND AND MOTIVATION

2.1 Computing unit with uniform quantization

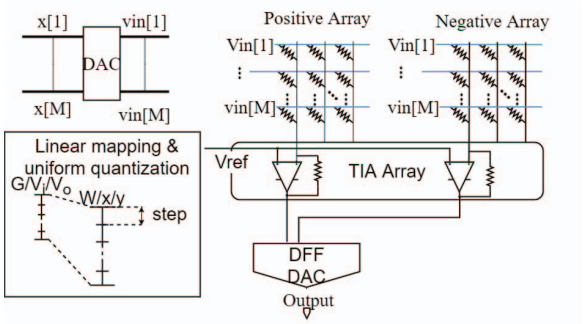


Figure 1: Computing unit of ReRAM NN accelerator[15].

The primary operation of the NN is matrix-vector multiplications. It is processed by the basis processing element of the ReRAM-based NN accelerator that consists of ReRAM crossbar arrays and peripheral digital circuits, such as Analog-to-Digital Converters (ADCs), Digital-to-Analog Converters (DACs), and Sensing Amplifier (SAs), output buffers, switch matrices, etc., as shown in Fig. 1. Matrix-vector multiplication can be written as:

$$\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{x}, \tag{1}$$

where $W \in R^{M\times N}, x \in R^M, y \in R^N$ are weight matrix, input vector and output vector respectively. In order to accelerate matrix vector multiplication in analog domain, the weight matrix and input vector need to be converted to the ReRAM conductance matrix and the input voltage vector $G \in R^{M\times N}$ and $V \in R^M$. The matrix-vector multiplication is achieved by current accumulation along each bit-lines. Specially speaking, current of the ReRAM cells on the n-th bit-line is accumulated though Kirchhoff's current law as Eq. 2.

$$I_{ADC,n} = (\boldsymbol{G}^T \boldsymbol{V})_n = \sum_{m=1}^{M} (G^+_{m,n} - G^-_{m,n})(V_{in,m} - V_{ref}) \tag{2}$$

Finally, the current vector from bit-lines is converted to output voltage by transimpedance amplifiers (TIAs) and converted to digital output vector by ADCs. Weight matrix (W) is rescaled by S_w to match the conductance range.

$$G_{m,n} = G^+_{m,n} - G^-_{m,n} = \frac{w_{m,n}}{S_w}\frac{g_{max} - g_{min}}{\eta_w} \tag{3}$$

$$G^-_{m,n} = \begin{cases} \frac{|x_{m,n}|}{S_w}\frac{g_{max} - g_{min}}{\eta_w} + g_{min} & w < 0, \\ g_{min}, & w > 0 \end{cases}$$

$$G^+_{m,n} = \begin{cases} g_{min} & w < 0 \\ \frac{|x_{m,n}|}{S_w}\frac{g_{max} - g_{min}}{\eta_w} + g_{min} & w > 0 \end{cases}$$

Input vector is rescaled by S_x to range $[-V_{dd}/2, V_{dd}/2]$ and then shifted to be positive by the bias $V_{ref} = V_{dd}/2$ added to all entries. g_{min}, g_{max} are the minimal and the maximum value of the conductance.

$$V_{in} = \frac{x}{S_x}\frac{V_{dd}}{2} + V_{ref} \tag{4}$$

For a uniformly quantized system, continuous values of input vector, weight matrix and output are discretized with uniform distance, and a k-bit quantization scheme can be performed as Eq. 5.

$$Q(t) = \hat{t} = r(\frac{t}{S_t} \cdot (2^N_t - 1)) \cdot \frac{2^N_t - 1}{S_t} \tag{5}$$

Here, N_t is the quantization bit-width and S_t is the scaling factor of t and $r(\dot{})$ is the round function. The value range is evenly divided into $2^N_t - 1$ intervals called **quantization step**. The input vector and weight matrix are quantized during mapping, while output vector is quantized during current sensing out. In this paper, we note bit-width of input vector, weights and output vector as N_x, N_w, N_y respectively.

2.2 Modeling and impact of conductance variation

2.2.1 Variation modeling. When mapping the weight matrix to ReRAM array, conductance variation including device-to-device

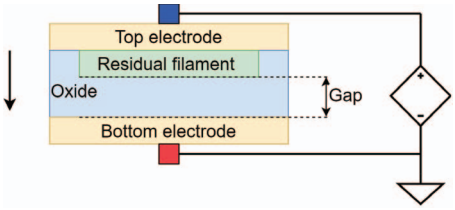


Figure 2: Structure of the ReRAM cell.

variation and pulse-to-pulse variation, is commonly observed and is remarkable. In [16], the I-V relationship of the ReRAM devices and approximate conductance are modeled as :

$$I = I_0 \cdot exp(-d/d_0) \cdot sinh(V/V_0), \tag{6}$$

$$G \approx \frac{I_0}{V_0} \exp(-\frac{d}{d_0}). \tag{7}$$

Here, d is average tunneling gap distance shown in Fig. 2, I_0 (∼ 1 mA), g0 (∼ 0.25 nm) and V_0 (∼ 0.25 V) are fitting parameters. Device-to-device variation is caused by fabrication variation and pulse-to-pulse variation is caused by stochastic evolution of the gap distance [17].

$$d\big|_{t+\Delta t} = d\big|_t + \Delta d_{(ideal)} + \delta d \tag{8}$$

$\delta d \sim N(0, \sigma^2)$ is Gaussian noise that implies the random variation during gap distance changing. And it can be easily deduced that the distribution of the actually programmed conductance value (G_{actual}) after programming:

$$G_{actual} \approx G_{target} \cdot e^{\theta}. \tag{9}$$

Where, θ follows normal distribution. Obviously, conductance exhibits a lognormal distribution [18], i.e, $G \sim lognorm(\mu_{G_{ideal}}, \sigma_G)$. This is also supported by experimental results [19].

2.2.2 Impact of the quantization and conductance variation. State-of-the-art ReRAM device can achieve up to 7-bit precision for a single device [20], by iterative read-verify-write operations until the resistance converges to the desired value. However, to achieve such high tuning accuracy, complex algorithms and circuits for programming signal generation are required, which means energy and timing inefficiency. In many conditions, only low precision tuning can be achieved because of limited time and facilities. That is to say, conductance variation may considerable. To avoid a sharp accuracy loss, a quantization scheme with lower quantization error and higher conductance variation tolerance is necessary. Based on the device model and the conventional uniform quantization methods, there are two observations:

Observation 1: The fewer the devices with high conductance, the better the robustness of the neural network running on the accelerator.

It is observed that the effect of process variations, the effect is greater near the high conductance region [21]. And with the same θ in Eq. 9, the ReRAM device with large conductance value suffers more serious variation.

Observation 2: Quantization introduces accuracy loss. By incorporating quantization and training procedures, we can minimize the quantization error and recover the accuracy. Fig. 3 shows the accuracy degradation of a three-layer NN under different variation amplitudes and bit-width configurations.

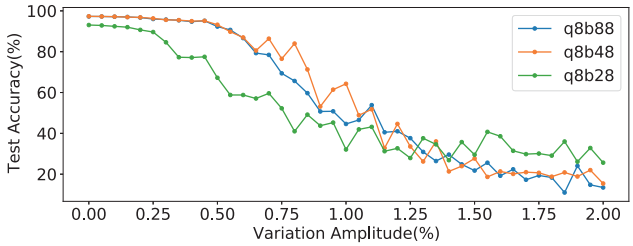


Figure 3: Test accuracy w.r.t. variation amplitude.

Here, the *abbreviation* "$\mathbf{qN_xbN_wN_y}$" represents the model quantized with N_x bits input vectors, N_w bits weights and N_y bits output vectors mentioned in Section 2.1. Although high bit-width quantization introduces less quantization error and has better inference accuracy without considering variation, low bit-width quantization has better robustness under variation. There should be a **trade-off between accuracy and robustness** when selecting quantization bit-widths.

If a ReRAM device is used to represent K bit unsigned number, the conductance range is divided evenly. By "3 sigma criterion", we assume the actual conductance $g_{(k,actual)}$ ranges from $[g_{(k)} - 3\sigma_{g(k)}, g_{(k)} + 3\sigma_{g(k)}]$. Uniform quantization method can be interpreted by a **naive intuition**: decreasing the bit-width enlarges the margin between two conductance levels, numerical overlapping that caused by actual conductance variation can be avoided. As a result, quantization method can achieve better robustness.

3 ROBUST TRAINABLE QUANTIZATION FOR RERAM-BASED NN ACCELERATOR

3.1 Robust quantization

The two observations mentioned above show that it is make sense to find such a quantization scheme that can maintain accuracy and enhance robustness. Firstly, A good quantizer should minimize the quantization error. Quantization introduces quantization error and affects the inference accuracy of a NN. In conventional "digital-style" quantization methods, quantization levels are fixed and uniformly distributed in full range. However, weight distributions differ across layers, and during the training process, the distribution of weights changes all the time, a predefined quantizer can not always be optimal [22].

Besides, conductance variation is not uniformly distributed in conductance range, with uniform quantizer, the sensitivity of the model to conductance variation is not controllable. Hence, a good quantizer should be adapted to suit weights distribution and variation distribution, maintain the accuracy of the full-precision system and improve the robustness of a NN during inference process. Basing on the fact mentioned above, a learnable quantizer is necessary.

3.1.1 Learnable quantizer. The target of quantization is to map the full-precision input values in a large set to the output values in a smaller countable set of discrete values. It can be reformulated as a **1-dimensional clustering problem**. In this paper, full precision weights in each layer are grouped into K **clusters** $C_1, ...C_K$. The weights grouped into C_k are represented by **representative value** q_k and allocated the **label** k, that indicates the mapping relationship. Clustering operation is performed by quantizer which can be

modeled as:

$$Q(w) = q_k, \quad if\ w \in C_k \quad k \in \{1, 2, ..., K\}, \tag{10}$$

w is the full precision weight value and $Q(\cdot)$ is the transform function of the quantizer. The representative value and the cluster boundary can be determined according to the system requirement.

3.1.2 Quantizer metric: robustness and accuracy. To find a good quantizer, the clustering metric should also be defined.

Based on the two observations about the quantization error and the variation sensitivity, for a quantized NN implemented on ReRAM-based, the weight deviation can be broken down into two parts: quantization error ($Q(\mathbf{w}) - \mathbf{w}$) between quantized and the full-precision value, and the device variation ($\mathbf{Q(w)_{actual}} - Q(\mathbf{w})$) between actual value represented by ReRAM ($Q(w)_{actual}$) and ideal quantized value ($Q(w)$).

Firstly, we propose clustering metric consists of two optimization targets. Then, we make a trade-off between the two optimization targets to balance robustness and accuracy under conductance variation of the neural network running on the accelerator.

A. Minimize the quantization error. To ensure each quantizer has a good performance, the quantization error minimization should be guaranteed. The first optimization goal is quantization error as below:

$$L_1 = \sum_{j=1}^{M} \left\| \mathbf{w}^{(j)} - Q(\mathbf{w}^{(j)}) \right\|_F^2, s.t.\ Q(\mathbf{w}^{(j)}) \in \{q_1, ..., q_K\}^N.$$

Consider an M-layers neural network, the weight matrix of j-th layer is denoted by $\mathbf{w}^{(j)}$, which consists of full-precision weights. $Q(\cdot)$ denote the non-uniform quantizer, and $q_1, ..., q_K$ denote the quantizer values that the weights in j-th layer are mapped to and $q_i \in [w_{min}, w_{max}]$.

B. Minimize the variation. To ensure the robustness of the NN, total weight perturbation under conductance variation should be small enough. To measure the impact of conductance variation, we implement this metric "accumulated squared error (ASE)" in [23] on quantized NN to be the second optimization goal as:

$$L_2 = ASE = \sum_{j=1}^{M} \left\| Q_{actual}(\mathbf{w}^{(j)}) - Q_{ideal}(\mathbf{w}^{(j)}) \right\|_F^2,$$
$$s.t.\ Q_{ideal}(\mathbf{w}^{(j)}) \in \{q_1, ..., q_K\}^N. \tag{11}$$

This metric measures the accumulated deviation between the quantized values and actual value represented by ReRAM devices. By optimizing this target, each quantized level (q_k) is encouraged to be selected in region that has lower probability of suffering large conductance variation. As a result, cumulative probability that the model is affected by the conductance fluctuation is under control.

C. Modeling the robustness (variation sensitivity). Conductance variation is randomly distributed, which is not suitable for optimization. In Eq. 9 conductance level with large G and $\sigma_{g(k)}$ suffers more serious variation. Here we use conductance range upper bound defined by "3σ criterion" mentioned in Section 2.2.1, to represent the sensitivity of each conductance level q_k to conductance variation. Here we suppose q_k is converted to $g_{(k)}$. According to the mapping relationship between weight w and conductance in Eq. 3, we can get the function of sensitivity of each weight quantization level $\mathcal{S}(\cdot)$:

$$\mathcal{S}(q_k) = q_k \cdot \frac{3\sigma_{g_{(k)}}}{g_{(k)}} \tag{12}$$

Then accumulated squared error (ASE) can be replaced by:

$$L_2 = \sum_{j=1}^{M} \left\| \mathcal{S}(Q(\mathbf{w}^{(j)})) \right\|_F^2, \quad s.t.\ Q(\mathbf{w}^{(j)}) \in \{q_1, ..., q_K\}^N. \tag{13}$$

D. Optimization target. Two optimization targets are combined by introducing a coefficient γ. Here γ controls the importance of the robustness in clustering. Then the quantization problem is reformulated into choosing the proper K discrete representative values $q_1, ..., q_K$, and learning the mapping from full-precision weights to quantization levels for quantizer $Q(\cdot)$, that is, minimize the target function:

$$q_1, ..., q_K = \arg\min_{q_1,...,q_K} \sum_{j=1}^{M} \left\| \mathbf{w}^{(j)} - Q(\mathbf{w}^{(j)}) + \gamma \mathcal{S}(Q(\mathbf{w}^{(j)})) \right\|_F^2. \tag{14}$$

3.2 Training algorithm

3.2.1 EM-style clustering method. It is complex and not efficient to solve the Eq. 14 directly with brute-force search, therefore, an EM-style [24] layer-by-layer quantizer optimization algorithm is designed for this clustering problem.

A: Latent variable introduction. Latent variable A_{nk} is introduced to represent the **clustering pattern**, i.e, another form of the label. It is called "latent", because it was never intended to be known in the first place.

$A_{nk} = 1$ When the n_{th} example is assigned to the k_{th} cluster, and 0 otherwise. Then we can reshape the loss function Eq. 14:

$$L(A, q_1, ..., q_K) = \sum_{n=1}^{N} \min_k [(x_n^{(j)} - q_k^{(j)})^2 + \gamma \mathcal{S}^2(q_k^{(j)})] \tag{15}$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} A_{nk} [(x_n^{(j)} - q_k^{(j)})^2 + \gamma \mathcal{S}^2(q_k^{(j)})] \tag{16}$$

$$s.t. \sum_{k=1}^{K} A_{nk} = 1$$

Instead of solving A_{nk} and q_k simultaneously, in our EM-style algorithm, A_{nk} and q_k are solved alternately.

B: Clustering pattern A_{nk} update (E-step). Fix A_{nk} and update q_k. Minimize target loss function Eq. 16 by finding the stationary points that $\frac{\partial L}{\partial q_k} = 0$.

$$q_k + \gamma \mathcal{S}(q_k) \frac{\partial \mathcal{S}(q_k)}{\partial q_k} = \frac{\sum_{k=1}^{K} A_{nk} x_n}{\sum_{n=1}^{N} A_{nk}} \tag{17}$$

In this paper, for simplicity, we adapt the assumption in [16] that $3\sigma_{g_{(k)}}/g_{(k)}$ is a constant, i.e. $\mathcal{S}(q_k) = \alpha q_k$. Then we have:

$$q_k = \frac{\sum_{n=1}^{N} A_{nk} x_n}{(1 + \gamma\alpha^2) \sum_{n=1}^{N} A_{nk}} \tag{18}$$

C: Representative value $\mathbf{q_k}$ update (M-step). Fix q_k and update A_{nk}. Minimize Eq. 16 Find the cluster pattern by find which cluster

x_n should belong to.

$$A_{nk} = \begin{cases} 1 & \hat{k}_n = \arg\min_k [(x_n^{(j)} - q_k^{(j)})^2 + \gamma \mathcal{S}^2(q_k^{(j)})] \\ 0 & else \end{cases} \quad (19)$$

After T times iterations of E/M-steps, we can get a sub-optimal solution $(A, q_1, ..., q_K)$ for the quantization problem. At the beginning of the training, $q_1, ..., q_K$ are chosen uniformly.

3.2.2 Network training. As discussed in Section 2.2.2, quantizer training and weight training should work together to minimize the quantization error.

The training process of the NN quantizer by EM-style clustering method is conducted during the **forward propagation**. During the **backward propagation**, the parameters of the quantizer remain unchanged, only the weights of the NN are updated. With continuously inputs, the quantizer produces discrete outputs, which means the transform function of the quantizer would always have zero gradient with respect to its input mathematically. Backward propagation can not work. To make the gradient-decent feasible, "straight-through-estimator (STE)" method in [25] [26] is adopted to compute the gradients:

$$\textbf{Forward}: r_o = Q(r_i) \quad (20)$$

$$\textbf{Backward}: \frac{\partial E}{\partial r_0} \overset{STE}{=} \frac{\partial E}{\partial r_i}\bigg|_{|r_i|\le 1} \Longrightarrow \frac{\partial r_o}{\partial r_i}\bigg|_{|r_i|\le 1} = 1 \quad (21)$$

Here, r_i is variable that is normalized to the interval $[-1, 1]$, and the gradient of the quantizer transform function is set to 1 for values between $[-1, 1]$, and 0 elsewhere.

During the training process, every several times backward propagation, parameters of the quantizer is updated by the EM-style clustering method during next forward propagation, based on the result in the previous forward propagation. The overall training procedures are summarized in the algorithm 1.

Algorithm 1 Training and testing the quantizer

Input: Weight matrix **w** of an N-layer NN, number of cluster K
Output: $Q(\mathbf{w})$

```
for j = 1 → M do
    if in the forward propagation then
        Initial the q_1^(j), ..., q_K^(j);
        for t = 1 → T do
            Compute q_1^(j), ..., q_k^(j) with A_nk^(j) by Eq. 18
            Compute A_nk^(j) with q_1^(j), ..., q_K^(j) by Eq. 19
        end for
    else
        Compute label k̂_n^(j) with q_1^(j), ..., q_K^(j) by Eq. 19
        Compute Q(w) by Eq. 10.
    end if
end for
```

4 EXPERIMENTAL RESULTS

4.1 Experimental setup

To evaluate our proposed training algorithm, we build a comprehensive simulation framework based on PytorX [15], a simulator based on the PyTorch and implemented by Python. It models the crossbar computation including input/output signal conversion, current accumulation, the weight splitting/mapping and conductance variation. With the help of PyTorch API, the simulator framework can train and test the model on multi-GPU with flexible network structures and training configurations. We experiment on the MNIST hand-writing digit dataset [27] with LeNet-300-100 (2 layers perceptron) [28] (listed in Table 1). LeNet-300-100 is a fully connected network with two hidden layers, with 300 and 100 neurons each, which achieves 1.6% error rate on MNIST. Crossbar size is 64×64. The remaining settings of hardware parameters are same as PytorX.

Table 1: Parameters of the LeNet-300-100

Layer	LeNet-300-100
input	Linear 784×300, Relu
output	Linear, 300×10

4.2 Tolerance to conductance variation and quantization error

In this section, firstly, we train the model with different bit-width values and coefficient values (γ in Eq. 14) combinations. All the models are trained from **scratch**. Then the performance of well-trained models is tested by sweeping conductance variation amplitude (σ) from 0 to 2.0 as shown in Fig. 4. Monte Carlo simulation is performed to model resistance variation of the device, and average cases are plotted. Here, the *abbreviation* "$\mathbf{qN_xbN_wN_y}$" represent

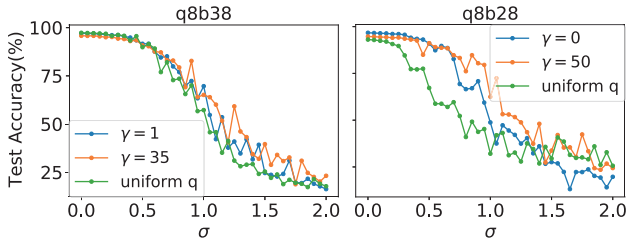
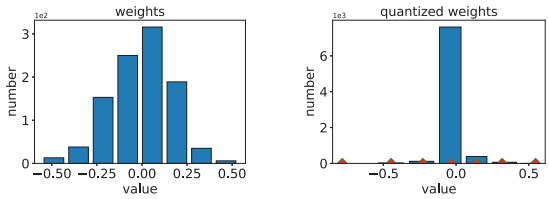


Figure 4: Test accuracy of the models with different configurations and values of γ.

the model quantized with N_x bits input vectors, N_w bits weights and N_y bits output vectors mentioned in Section 2.1. Curves with *label* "$\gamma = \mathbf{t}$" represent the performance of model trained with our method and $\gamma = t$, Curve with *label* "**uniform q**" represents the performance (inference accuracy) of model trained with uniform quantization.

As shown in Fig. 4, With proper γ setting, as variation amplitude increasing, prediction accuracy of the models trained with our method drops slower. For conventional uniform quantization method, as σ increases from 0 to 0.6, 1.2 and 2.0, accuracy of model drops from 93.07% to 58.79%, 32.67% and 25.64%. In contrast, our method can achieve accuracy of 94.69%, 88.7%, 51.65% and 24.05% at most, when σ is equal to 0, 0.6, 1.2 and 2.0, respectively. It means that our method can improve robustness of model under conductance variation. Considering the situations without device variation, our method can achieve at least similar prediction accuracy compared to the original uniform quantization scheme, and even better in



(a) (b)

Figure 5: (a). Weight distribution before quantization; (b). Weight distribution after quantization with our method.

some cases. This implies that the quantization error can be minimized by our method. We compare our method with variation tolerant scheme proposed in [14]. As σ increases from 0 to 0.6, 1.2 and 2.0, accuracy of model drops from 94.11% to 87.02%, 48.16% and 22%. Although Song's method shows similar performance with conductance variation, but our method shows better accuracy without variation and better flexibility.

Fig. 5 shows the statistical distribution of weights before and after quantization of the input layer of the two-layer perceptron trained by our method. It is obviously observed that the region nearby zero (suffer less variation) and with higher weight density have more quantization levels. i.e., quantization step in our method is not a constant. The target of our optimization algorithm is achieved.

4.3 Optimal hyper parameter selection

As discussed in Eq. 14, the hyper-parameter γ is introduced to balance the robustness and accuracy under conductance variation. In Fig. 6, we vary the values of γ when training the NN for hunting for the peak accuracy. It can be seen that models with different values of σ and γ differ in test accuracy, and a peek value can be identified.

Besides, the models trained with larger γ shows better robustness, but at the cost of some accuracy degradation under small variation. This phenomenon verifies the effectiveness of the γ and indicates that γ should be tuned depending on actual variation distribution.

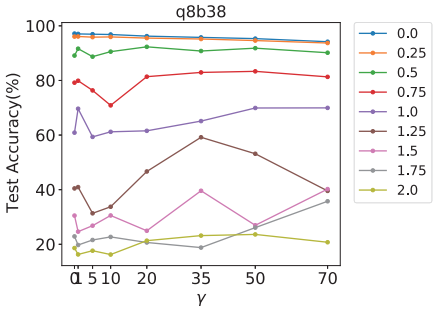


Figure 6: Scan coefficient γ to search the best accuracy.

5 CONCLUSION

In this paper, we analyze the impact of the quantization and conductance variation on neural network executed on the ReRAM-based NN accelerator. The observations reveal the trade-off between robustness and accuracy. By introducing a framework including the design, evaluation and training process of the quantizer, we can minimize the quantization error and the probability that the model is affected by the conductance variation. With our method, we can train a model with robustness and high accuracy, and make a well balance between them. The framework does not need extra hardware overhead and need little extra training efforts, and the trained-NN can be mapped to any ReRAM-based NN accelerator maintains its accuracy as much as possible without more optimization. Both pulse-to-pulse variation and device-to-device variation can be well tolerated.

REFERENCES

[1] H. . P. Wong *et al.*, "Metal–oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.
[2] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *SIGARCH Comput. Archit. News*, vol. 44, no. 3, p. 14–26, 2016.
[3] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *ISCA*, vol. 44, no. 3, 2016, pp. 27–39.
[4] C. Wang *et al.*, "Cross-point resistive memory: Nonideal properties and solutions," *TODAES*, vol. 24, no. 4, 2019.
[5] S. Jin *et al.*, "A variation tolerant scheme for memristor crossbar based neural network designs via two-phase weight mapping and memristor programming," *Future Generation Computer Systems*, vol. 106, pp. 270–276, 2020.
[6] Peng Gu *et al.*, "Technological exploration of RRAM crossbar array for matrix-vector multiplication," in *ASP-DAC*, 2015, pp. 106–111.
[7] L. Chen *et al.*, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar," in *DATE*, 2017, pp. 19–24.
[8] L. Xia *et al.*, "Fault-tolerant training with on-line fault detection for RRAM-based neural computing systems," in *DAC*, 2017, pp. 1–6.
[9] M. Cheng *et al.*, "TIME: A training-in-memory architecture for memristor-based deep neural networks," in *DAC*, 2017, pp. 1–6.
[10] Z. Zhu *et al.*, "A configurable multi-precision cnn computing framework based on single bit RRAM," in *DAC*, 2019, pp. 1–6.
[11] L. Ni *et al.*, "On-line machine learning accelerator on digital RRAM-crossbar," in *ISCAS*, 2016, pp. 113–116.
[12] B. Li *et al.*, "Merging the interface: Power, area and accuracy co-optimization for RRAM crossbar-based mixed-signal computing system," in *DAC*, 2015, pp. 1–6.
[13] Y. Cai *et al.*, "Low bit-width convolutional neural network on RRAM," *TCAD*, pp. 1414–1427, 2020.
[14] C. Song *et al.*, "A quantization-aware regularized learning method in multilevel memristor-based neuromorphic computing system," in *NVMSA*, 2017, pp. 1–6.
[15] Z. He *et al.*, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *DAC*, 2019.
[16] S. Yu *et al.*, "A neuromorphic visual system using RRAM synaptic devices with sub-pj energy and tolerance to variability: Experimental characterization and large-scale modeling," in *IEDM*, 2012, pp. 10.4.1–10.4.4.
[17] S. Yu, X. Guan, and H.-S. P. Wong, "On the switching parameter variation of metal oxide RRAM—part ii: Model corroboration and device design strategy," *TED*, pp. 1183–1188, 2012.
[18] M. Hu *et al.*, "Leveraging stochastic memristor devices in neuromorphic hardware systems," *JESTCS*, vol. 6, no. 2, pp. 235–246, 2016.
[19] A. Fantini *et al.*, "Intrinsic switching variability in HfO^2 RRAM," in *IEEE International Memory Workshop*, 2013, pp. 30–33.
[20] F. Alibart *et al.*, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, p. 075201, 2012.
[21] J. Rajendran *et al.*, "Improving tolerance to variations in memristor-based applications using parallel memristors," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 733–746, 2015.
[22] D. Zhang *et al.*, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *ECCV*, 2018, pp. 365–382.
[23] B. Liu *et al.*, "Vortex: variation-aware training for memristor x-bar," in *DAC*, 2015, pp. 1–6.
[24] G. J. McLachlan and T. Krishnan, *The EM algorithm and extensions*, 1996.
[25] Y. Bengio *et al.*, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.
[26] G. Hinton *et al.*, "Neural networks for machine learning," *Coursera, video lectures*, vol. 264, no. 1, 2012.
[27] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[28] Y. LeCun, "1. s. denker, and sa solla. optimal brain damage," *Advances in Neural Information Processing Systems (NIPS)*, 1989.