Digital Object Identifier

Design Exploration of ReRAM-Based Crossbar for AI Inference

YASMIN HALAWANI, (Member, IEEE), BAKER MOHAMMAD, (Senior Member, IEEE) and HANI SALEH, (Senior Member, IEEE).
System-on-Chip Center (SoCC), Department of Electrical and Computer Engineering, Khalifa University, Abu Dhabi, UAE

Corresponding author: Baker Mohammad (e-mail: baker.mohammad@ku.ac.ae).

This publication is based upon work supported by the Khalifa University Competitive Internal Research Award (CIRA) under Award No. [CIRA-2019-026], and System-on-Chip Center Award No. [RC2-2018-020].

ABSTRACT ReRAM-based crossbar designs utilizing mixed-signal implementation has gained importance due to their low power, small size, low cost, and high throughput especially for multiply-and-add operations in AI-related applications. This paper provides a framework with associated code for analyzing the impact of ReRAM device variation and post-training analog conductance quantization that has not been fully explored for pre-trained network accelerators. A detailed study with end-to-end implementation is presented, ranging from mapping the pre-trained DNN weights to quantized crossbar conductance values and into final classification with the presence of variation. Monte Carlo analysis was performed to better analyze the impact of the different parameters on the final accuracy without being device-specific. The work assumes different conductance value variations, different conductance dynamic ranges, and various device quantization levels (QLs).

MNIST and CIFAR-10 data sets were used in this study for ANN and CNN, respectively. Results show that for simple ANN, the accuracy drop due to quantization was $\sim 2\%$ at 64-QLs. While for CNN, the decrease in classification accuracy was around 10% with the same number of levels. Moreover, weight variation might cause a $\sim 5\%$ and $\sim 8\%$ drop in classification accuracy for ANN with 5% and CNN with 3% variation, respectively. The study confirms that increasing the number of levels with small variation results in near-optimal accuracy. However, the increase in accuracy saturates at an upper limit. The amount of distortion propagated through the layers is different in the two cases. It is dependent on the complexity of input data and network structure, such as the size of the neurons in each layer, the number of layers, and the number of channels and filters at each stage. This contribution is the first to provide the framework to explore the implication that emphasizes on device-independent post-training DNN quantization and weight variation on classification accuracy. This helps explore design trade-offs especially for edge devices in cases where there is no access to the training set to the end-user due to security or cost issues for pre-trained networks.

INDEX TERMS Device variation, in-memory computing, neural network, post-training, quantization, ReRAM.

I. INTRODUCTION

EMERGING applications related to Artificial Intelligence (AI) rely on complex machine learning (ML) algorithms to offer high performance in executing segmentation and classification tasks. Machine learning models have two phases: learning and inference that can be both computationally and memory-intensive [1].

Graphics processing units (GPUs) were introduced with hundreds or thousands of smaller cores to enable massively parallel computing to increase throughput in realizing ANN applications. However, it still suffers from considerable bus access time, high energy consumption, and memory bottleneck , which is a big issue for edge devices. Recently, embedded field-programmable gate arrays (FPGAs) attract more attention as a low power solution for ML inference [2]. 3D integration of memory processing is also investigated. The consensus in the hardware community is that training phase for edge devices need to be done offline and with access to high performance and large memory system. However, there is a need for inference to be implemented at the edge due to limited bandwidth, requirements for low latency and in some cases, security [3]. This presents a new challenge

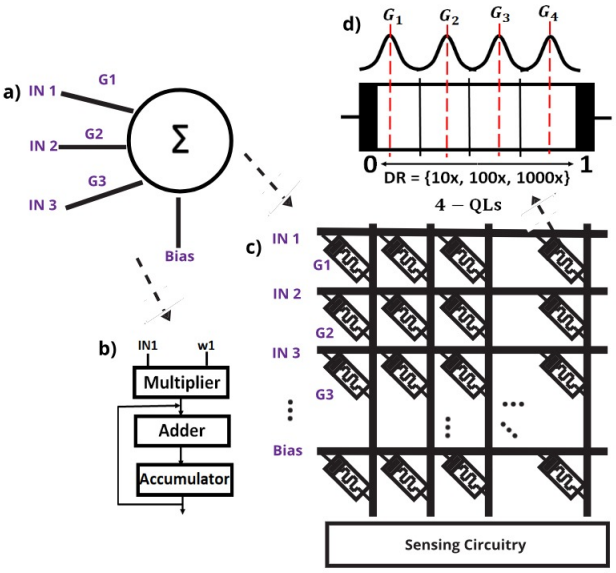

FIGURE 1. Multiply-and-add operations that are found in applications such as ANN, shown in a), can be realized by a multiplier, an adder, and a memory as demonstrated in b). While in c) a ReRAM-based crossbar architecture is presented where a single device can substitute the three blocks from CMOS implementation. Each device in the crossbar represents a weight from the layer of the network. Moreover, the device may have certain quantization levels (QLs) in its dynamic range (DR), as illustrated in d). Due to intrinsic device variation, the target weight value programmed (G_1, G_2, G_3 and G_4) on the ReRAM will have a $\pm$ certain deviation, which impacts the overall result.

to implement inference on the edge due to their limited performance and power budget [4]. Luckily, most of edge device inference applications can tolerate accuracy loss to some extent, and this can be exploited for hardware trade-off to attain speed and power advantages.

Neural network accelerator is the main core design in an AI inference engine. Their main building blocks consist of multiply-and-add (MAC) blocks and memory. Many architectures use in-memory computing (IMC) with both traditional and emerging technologies [4]–[6].

ReRAM is a promising technology for building efficient IMC architectures for accelerating artificial neural networks, as demonstrated in Fig. 1 [7], [8]. Besides computation, the sensitivity of the device's resistivity to operating conditions and surrounding environment provides great potential in the sensing and security applications [9], [10]. These devices suffer from intrinsic variation that can degrade the performance of the application. ReRAM-devices are built in a crossbar fashion where they inherently support parallel computations, and can naturally realize vector-matrix operations with significant savings in energy, area and achieve faster execution time. Moreover, these devices can be configured to support analog or mixed-signal type IMC operations [11], [12]. Furthermore, the low power requirement for computing with ReRAM devices makes them an ideal candidate for resource-constrained Internet-of-Things (IoTs) nodes.

An essential aspect in the design and implementation of a neural network (NN) is the sensitivity of the output to the variations in its parameters. The sensitivity of the results are affected by the structural configuration such as the number of layers and the number of neurons per layer. Also, the hardware implementations of NNs are susceptible to imprecision and unavoidable variations in the inputs and weights of the network due to the physical limitation of digital and/or analog components.

The main contribution of this work is to provide a methodology that incorporates the device's intrinsic variation, dynamic range and available conductance states into DNN model to study the impact of devices characteristics and post-training quantization on the classification accuracy of analog accelerators. The work is inspired from [13], so in our case we envision a lookup table with resistance-level approach based on the measurements data of our mentioned device. To the best of our knowledge, no work that emphasizes on device-independent post-training NN quantization has been reported before. This helps when there is no access to the training sets to the end-user for security issues and in generic cases where the network has already been trained. This will support a vertical market where testing set can be used for multiple devices with different requirements for accuracy, power, and speed.

The rest of the paper is organized as follows: Section II provides insight into the quantization technique as an efficient solution for Hardware-based inference at the edge. Section III describes the work-flow utilized from building and training the neural network to statistically evaluating the quantized ReRAM-based NN. After that, Section IV presents a full example of a pre-trained ANN following the flowchart process from training the network to evaluating the post-quantized ReRAM ANN. Then, the obtained ANN classification results from exploring the different design points from the device's dynamic range, and weight variation effect are demonstrated in Section V. The work is then expanded to study more complex network structure by evaluating CIFAR-10 data set on a pre-trained CNN in Section VI. Finally, the paper is concluded in section VII and provide some future work.

II. OVERVIEW OF EFFICIENT INFERENCE FOR EDGE DEVICE

In addition to selecting the right algorithm for the target application, further optimization to reduce computing complexity and achieve lower power happens at all levels. Some techniques to reduce the number of operations, such as pruning and quantization, are proposed to achieve efficient hardware inference on the devices [14].

This paper focuses on utilizing the quantization approach and emerging ReRAM crossbar architecture to achieve power and area efficiency. It is another compression technique besides pruning, where inputs and/or weights for the NN can be represented with a lower number of bits. Quantization is powerful as it requires a smaller number of bits to perform a significant amount of MAC operations in the neural network. Inputs, weights, and activation functions can be quantized to

achieve savings in memory, computational resources, speed-up, and number of MAC operations. Utilizing an extremely compact representation such as 2-bits has shown 58× reduction in convolution operations and 32× savings in memory. This comes at the cost of 12.5% loss in accuracy compared to the full-precision floating-point [15], [16]. However, aggressive quantization generally entails a severe penalty in terms of accuracy and often requires retraining the network or resorting to higher precision quantization. Trained quantization can help fine-tune the weights to the available representation, but this comes at the expense of execution time and resources. Hence, in [17] a trade-off between accuracy and number of bits used for the inputs is presented. They showed that a 7% reduction in accuracy compared to the full-precision version was obtained by using ternary inputs and binary weights. Also, higher precision is required during the weight update step, which means that other specific hardware units might be needed. In [16], authors indicated the need for a quantization-aware approach to reduce the quantization error. This is based on the statistical characteristics of weight distribution.

A PArameterized Clipping acTivation (PACT) technique uses a parameter that is optimized during training to find the right quantization scale [18]. They focus on quantizing the activation functions instead of the weights as they are relatively larger in size. A Hardware-Aware Automated Quantization framework is demonstrated in [19]. It is based on the concept that each layer in a CNN pipeline requires a different optimal bitwidth resolution for the weights and the activation function for efficient implementation over different hardware architectures.

PytorX, a simulation framework that considers Stuck-At-Fault, IR-drop, thermal noise, shot noise, and random telegraph noise, is proposed in [20]. Also, they show injecting noise during training gives better results as the NN adapts to it.

In [21] RxNN, an extension to the Caffe learning framework, is presented to evaluate large-scale DNNs on resistive crossbar systems. It splits and maps the network's computations into crossbar operations and evaluates them using a fast crossbar model. It takes into account the non-linear DAC/ADC converters and the equivalent non-ideal conductance matrix for the crossbar. Their results show that retraining can partly restore the degradation in accuracy.

An end-to-end framework for data-dependent crossbar modeling along with a PyTorch-based functional simulator considering tiling, and bit-slicing is proposed in [22]. Data-dependent non-idealities can have a pronounced effect on the crossbar outputs, particularly at higher operating voltages. The neural network was trained to learn the transfer characteristics of the non-ideal crossbar. Mitigation techniques for such non-idealities strongly depend upon the modeling approach and requires retraining of the neural network weights. They for example show that lower ON/OFF conductance ratio leads to high relative error between the ideal output and the non-ideal obtained output. This is due to the fact that for a given ON resistance, the average resistance in the crossbar is low for lower ON/OFF ratio.

In this paper, authors demonstrate why, how, and where to prune neural networks for superior exploitation of the crossbar's underlying parallelism model [23]. Computation costs can be improved by aligning the network design space with the underlying hardware's parallelism model. The key take-away from these deductions is that it is possible to achieve both higher accuracies and lower computation costs by considering the target hardware's parallelism model along with the network architecture search space.

In [24], the authors have studied the impact of the ReRAM device on the training/inference accuracy of CIFAR-10 using VGG-Net by integrating the device properties in the TensorFlow. The examined characteristics included the device's non-linearity and asymmetry of conductance tuning, conductance variations, endurance, and the data retention for the inference phase. One of their findings is that the conductance range variation does not degrade the training accuracy; instead, a small variation can even reduce the accuracy loss introduced by asymmetry.

Authors in [25], presents analytical solutions to quantize both activations and weights while minimizing quantization error at the tensor level. They utilize a 4-bit post-training quantization approach for faster deployment. A framework for optimized deployment of NN on embedded devices where a sensitivity analysis of each layer for quantization is performed [26]. This helps in quantizing each layer differently than the other to help maintain the performance of the NN and at the same time gain savings in the memory storage and traffic. A piecewise linear quantization method enables accurate post-training quantization for a low number of bits in [27]. Their approach divides the Gaussian distribution of the NN weights into two regions; the condensed center and the tail.

Several works in literature included circuit-level details in their frameworks. In [28] and [29], details from the interconnects to the IOs and ADC/DACs were taken into consideration in evaluating the accuracy, power, area and latency metrics. Moreover, in [30], authors presented a micro-architecture simulator whose core component functionality is verified by the corresponding circuit simulation of a real chip design that is taped-out under 130nm process. Their case study focused on the effect of pruning on the evaluation of the framework's accuracy and only for Cifar-10 with 15 convolution layers. They also highlighted the training algorithm for retraining and/or fine-tuning with different setups like quantization and non-linearity of the IO.

ReRAM devices play a vital role in accelerating NN computations, especially in IoT nodes and edge computing devices. However, it is hard to get to the exact analog value with very high precision in memristors due to the intrinsic variations that can occur due to process variations such as oxygen concentration in switching behavior [31], [32]. Moreover, successive writing and reading operations might cause the device to wear out due to low training endurance.

Moreover, the memristor's continuous analog resistance

range is quantized into discrete levels to decrease the write circuitry's complexity. Non-uniform quantization where the low resistance state contributes significantly to mapping the parameters and the high resistance state has little contribution to the magnitude of conversion [33]. Hence having more accurate mapping of low resistance is more important than the high resistance.

Recent work on a simple two-layer ANN for MNIST classification has been utilized for training and testing [34]. Input pixels, as well as the weighted sum of each neuron, are truncated to 1-bit. Hence, the inference is realized with low-precision. They showed that at least 6-bits are required for training and at most 2-bits for testing. Moreover, the authors extended their work for CNN in [35]. Data retention and ADC quantization are the key factors for inference accuracy degradation in their CIM inference engine. They evaluate their impacts and benchmark across technologies on an accelerator design based on VGG-8, for the CIFAR-10 dataset, with 8-bit weight and 8-bit activation precision for 2%, 6%, and 10% state drift. However, both frameworks were more hardware-oriented. Not many details are disclosed on the crossbar's architecture such as how negative weights are presented as conductance values, and the weight-to-device mapping algorithm. While in [36], an alternating direction method of multipliers (ADMM) is incorporated into the network's training phase. They include weight pruning, linear quantization, conductance range, the mismatch between weight value and real conductance values. They use different crossbars to represent positive and negative weights. And another crossbar to represent the higher and the lower bits. This causes area overhead. In our proposed work, only a single column is concatenated to the original crossbar to be able to remove the effect of weight shifting as a mean to represent negative conductance values.

Moreover, in [37], the author showed that post-training quantization of neural network weights only can have a different impact on the reported drop of accuracy depending on the chosen CNN structure. For example, for Mobilenet, the accuracy goes down to 0.001 from the full floating-point of 0.707. While Inception and ResNet can still provide near-optimal classification accuracy.

Our study shows the details of the architecture's mapping to take into account the negative neural weights. Also, the full range of pixels as well as the image dimension were considered.

In addition to the desire for universal training models and not device-specific models, the limited capabilities of edge devices to update weights make it necessary to perform post-training quantization that can minimize the hardware requirements and maintain acceptable accuracy. As a consequence, in the following section, a proposed work-flow for studying the effect of post-quantization, weight variation, and dynamic range on ReRAM-Based NN Inference is presented.

III. PROPOSED WORK-FLOW FOR RERAM-BASED NEURAL NETWORK INFERENCE

In this section, the post-training quantization work-flow for weights of a pre-trained NN with full precision (FP) is analyzed for target ReRAM-crossbar implementation. As an example, a simple two-layer ANN and more complex CNN have been used to demonstrate the approach. Using ReRAM implies quantization of weights due to the physical limitations on the number of resistance states an analog memristor device can exhibit. Several works have shown that memristor devices can exhibit stable multi-level resistance states that are captured in the device models [38], [39]. Authors in [40] fabricated a Al-doped HfO_2-based memristor with 20-levels. Also in [41] reported on a 12-levels stable resistance states using HfO_{2-x}. Varying the electrode material used can vary the number of stable resistance states to be achieved [42]. Some works even have demonstrated 64-conductance levels [43], [44].

Fig. 2, demonstrates the proposed flow utilized in our analysis. The flow first starts with block (1), where the NN is trained using MATLAB 2019b or Caffe deep learning framework. Then these weights are mapped into memristor conductance values in a specific conductance range in (2). After that, in block (3), a certain number of QL is assumed for the specific conductance range. Next, variations are added into the quantized conductance weights sampled from a normal distribution as in step (4). Finally, in block (5), the input data set is mapped into voltage levels and applied to the NN to evaluate the testing classification accuracy. The whole process can be repeated for various conductance range, QLs, and weight variations. Outputs are multiplied by a scaling factor as a way to reduce the effect of mapping and quantization.

IV. MAPPING OF SIMPLE ANN UNDER THE PROPOSED FLOW

In here, we start with the ANN example. The structure of the ANN for MNIST classification is shown in Fig. 3. The 28×28 input image is first flattened to 1D, resulting in 784 inputs. This is followed by 32 hidden nodes and 10 output nodes to classify the input images to classes from 0-9. The ANN was trained for 200 epochs with 0.01 learning rate resulting in 94.58% classification accuracy. Further training can achieve higher accuracy. However, we chose to stop at this level and consider it our baseline. This ANN structure was then mapped into two memristor crossbars demonstrated in Fig.4. The hidden and output crossbars have the following dimensions respectively 785×33 and 33×11. The last row in both crossbars is for the bias connection values and has an input of 1V. While the additional last column is used to hold the minimum negative value from the FP weights. The weight-to-conductance mapping algorithm from [45] has been considered in this paper. It considers the actual physical range G_{off} to G_{on} of the memristor [13]. As a start, a 10× dynamic range was assumed. The DR can later be changed to 100× and then 1000× after going through the whole flow

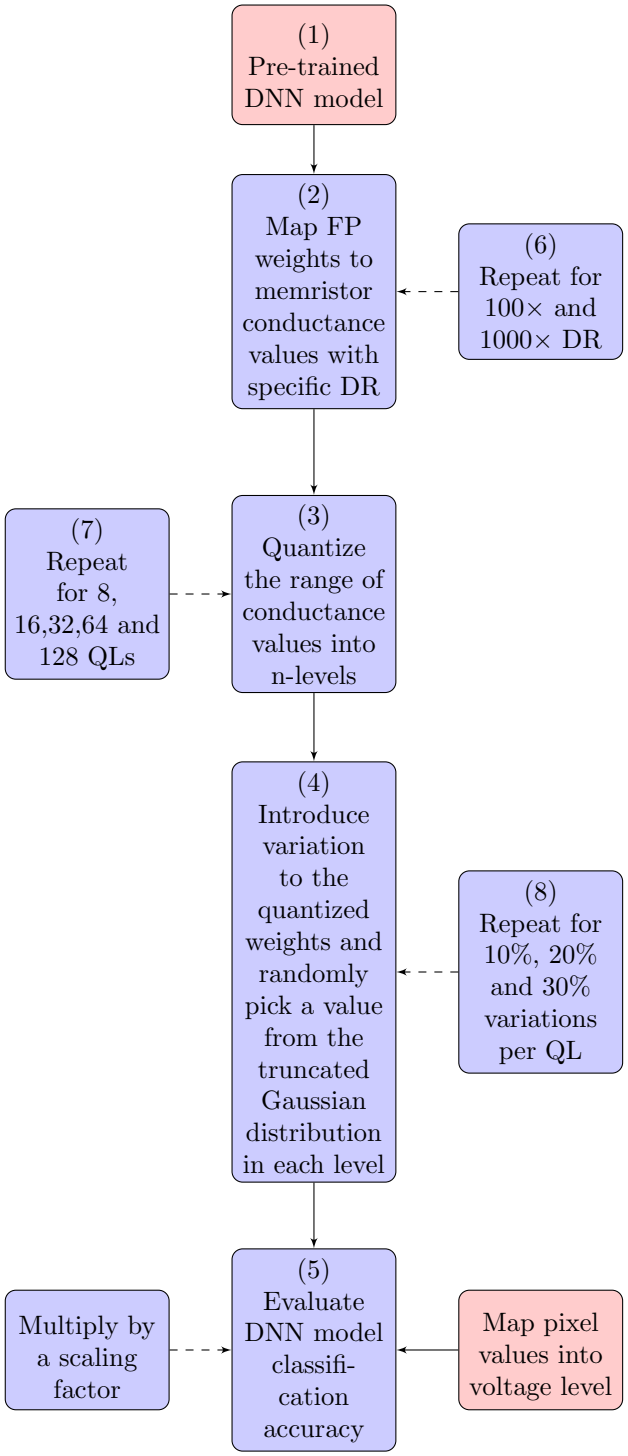

FIGURE 2. Flow chart showing the undergone method to perform statistical analysis on a pre-trained DNN for several quantization levels and evaluating their classification accuracies.

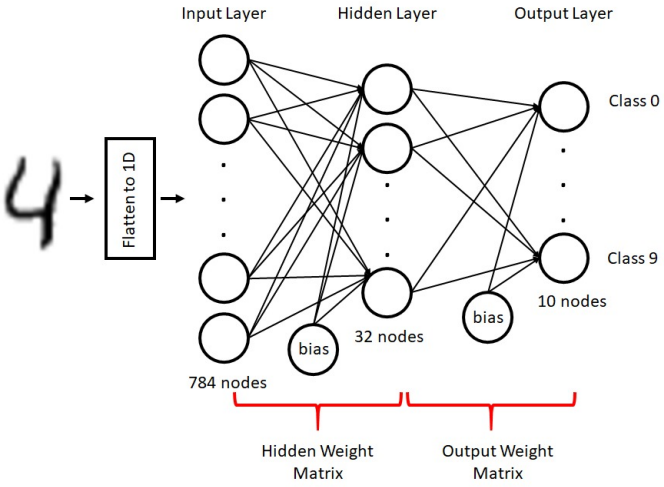

FIGURE 3. ANN structure for MNIST data-set classification utilizing 32 nodes for the hidden layer and 10 nodes for the output layer.

as depicted by block (6). The pre-trained ANN weights are first shifted by the minimum negative value and then converted into conductance values via the mapping algorithm. The shifted mapped conductance values achieve 94.58% classification accuracy. In order to remove the contribution of the shifting, the summation of all inputs multiplied by the minimum weight is subtracted from each output column before passing it to the non-linear activation function. In addition, the outputs at this stage are multiplied by a scaling factor obtained by realizing the relation between the shifted output values from a neural network and the output from memristor-based crossbar architecture. This step is required to compensate for the mapping between the neural weight values and memristor conductance. Moreover, state disturbance with time can be considered as part of the conductance variations from the target value. It can be compensated via the programmable scaling factor. And in case of device failure, memory redundancy can be considered. Our goal is to provide designers with a way to analyze the variation and quantify its impact on system performance. After that, the output is passed through the non-linear activation function. The operation is expressed by the following, where output currents are produced by multiplying input voltages with the network's conductance values

$$\begin{bmatrix} \mathbf{I_1} & \mathbf{I_2} & \mathbf{I_3} \end{bmatrix} = \begin{bmatrix} \mathbf{V_{in1}} & \mathbf{V_{in2}} & \mathbf{bias} \end{bmatrix} \begin{bmatrix} \mathbf{w_1} + |\mathbf{w_{min}}| & \mathbf{w_2} + |\mathbf{w_{min}}| & |\mathbf{w_{min}}| \\ \mathbf{w_3} + |\mathbf{w_{min}}| & \mathbf{w_4} + |\mathbf{w_{min}}| & |\mathbf{w_{min}}| \\ \mathbf{w_5} + |\mathbf{w_{min}}| & \mathbf{w_6} + |\mathbf{w_{min}}| & |\mathbf{w_{min}}| \end{bmatrix},$$

where I_i is the output current per column i, the whole term $\mathrm{w}_j + |w_{min}|$ is the mapped conductance value after shifting by the minimum NN weight value w_{min}, and V_{in} is the input voltage.

This is followed by removing the effect of the shifting by subtracting the following term: $\sum_{i=1}^{n} = \mathrm{V}_{in} \times |w_{min}|$, where V_i is the input to the hidden/output layer and n is the number of

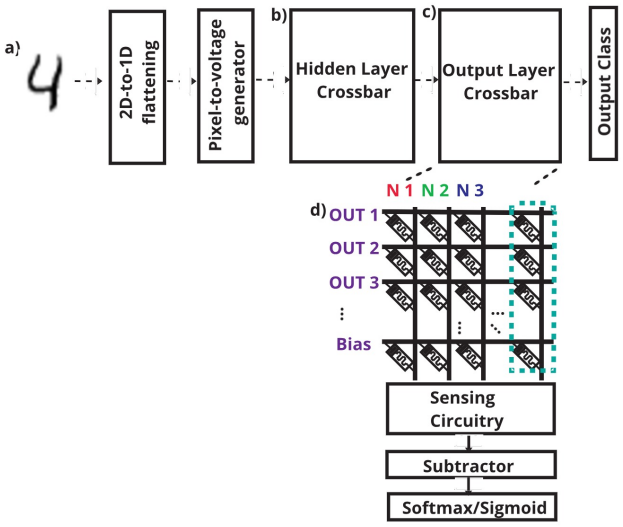

FIGURE 4. Memristor-based ANN structure for MNIST data-set classification utilizing a hidden layer and an output layer crossbars. a) Shows an input image that will be flattened to 1D and then converted to voltage. b) and c) both demonstrate the crossbars followed by a sensing circuitry that will sense the output current from all columns and then convert it to voltage. This is then followed by a subtractor to remove the effect of weight shifting. After that the output is ready to pass through a non-linear activation function and d) instance for both hidden and output crossbars.

input nodes. The whole process of mapping the FP weights into memristor conductance value is captured by block (2) in Fig. 2. It is worth mentioning that the matrix values to memristor conductance mapping should take into account the polarity and wire resistance. Inaccurate mapping of matrix coefficient values or neural network weights to conductance values might cause the mapped parameter to go beyond the device's actual range. Moreover, the memristor state may get saturated because of the small dynamic range of the parameters. Another point that should be noticed is if the crossbar's mapped weights are so high, the summation of the output voltage might exceed the operational range of the output amplifier [46]. A simple linear and fast 1:1 approximation mapping from matrix coefficients to memristor conductance values is defined in [47].

A simple, linear mapping of conductance values in a physical memristive crossbar does not perform well as the array size gets larger [45]. Hence the authors in [48] illustrated that the coefficient value in the mapping depends on the conductivity of the specific memristor $G_{i,j}$ and all the other memristors in the column in a non-linear manner.

According to block (3) in the flowchart in Fig. 2, the next step is to define the number of quantization levels (QL) for the inference phase with the specific physical memristor conductance range. The following QL were considered: 4, 8, 16, 32, 64, and 128. First, we started with 4-QLs, meaning 2 bits of information. Then for each one of those levels, a truncated Gaussian distribution was created. It is worth mentioning that the weights in a trained network often follow Normal distribution that is widely used to fit a wide range

TABLE 1. Classification accuracy for a simple 2-layer ANN when only the effect of quantization is considered for a 10× DR.

Quantization-Levels (QL)	Bits of information	Accuracy
Full ANN (baseline)	FP	94.58%
128-QL	7	94.43%
64-QL	6	92.46%
32-QL	5	84.46%
16-QL	4	50.6%
8-QL	3	9.28%
4-QL	2	10.09%

of the variations of real data applications [49]. Besides, the trained weights have values around zero, which means they can be roughly fitted to Normal distribution. Truncated Gaussian distribution was assumed for the following reasons: a) memristor's mapped weight values should be strictly positive, and if the distribution was not truncated, then a negative weight might appear even after the mapping and shifting processes, and b) avoid having extreme values [-inf,+inf] even though the actual probability of an extreme event will be very low; hence, truncate the range is made finite at both ends [50]. Nonetheless, other distributions can be utilized to replace the normal distribution. Each mapped weight from the hidden and output matrices was substituted by a random value drawn from its target truncated Gaussian distribution of the target conductance range with a specific variation starting with 5% as in block (4). Eventually, the newly created hidden and output matrices were evaluated by a forward pass to examine the classification accuracy. This is repeated 1000 times, and in each time, new random values are drawn from each respected range. Assuming that each sample resembles a memristor crossbar. After that, another 1000 samples are generated for 10%, 20% and 30% each as illustrated by block (10) in the flow. The MATLAB code used in analyzing the weight variation effect on the accuracy is available in the Appendix. Next, an 8-QLs is considered, and the whole process will be repeated. And so on and so forth until all QL are tested and evaluated as in block (5).

V. ANN WEIGHT QUANTIZATION AND VARIATION FOR DIFFERENT DYNAMIC RANGES

A major limitation in memristor devices is the intrinsic programming variation that occurs to a specific conductance value due to the movement of the ions and other device limitations [51]. Designing memristor devices with higher dynamic ranges can relax the need for precise writing circuitry in analog computations as the noise margin between successive states will be larger compared to devices with smaller dynamic ranges [52]. Hence, the output result is affected by both the state drift and the accuracy of conductance tuning.

This section shows three case studies on the multilayer perceptron where the QL, device's dynamic range, and weight variations are the variables.

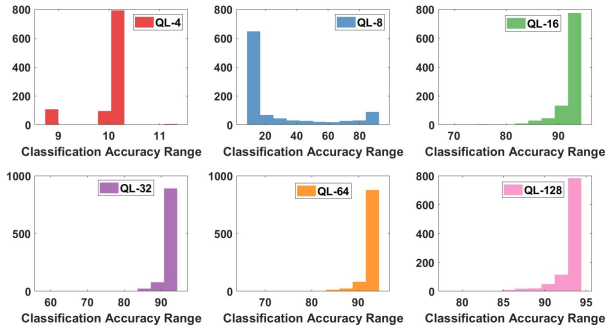

FIGURE 5. Histogram for six quantization levels ranging from 2-bits to 7-bits representation with a 100× off/on resistance ratio and 10% weight variation for MNIST data set classification on 2-layer ANN.

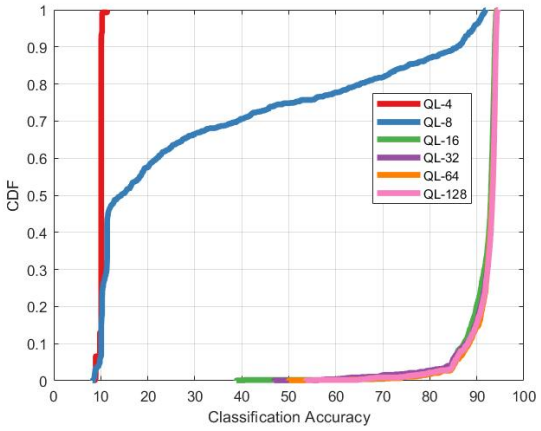

FIGURE 6. CDF showing the probability of having a less than or equal a target classification accuracy for 10× dynamic range for a simple 2-layer ANN with MNIST classification.

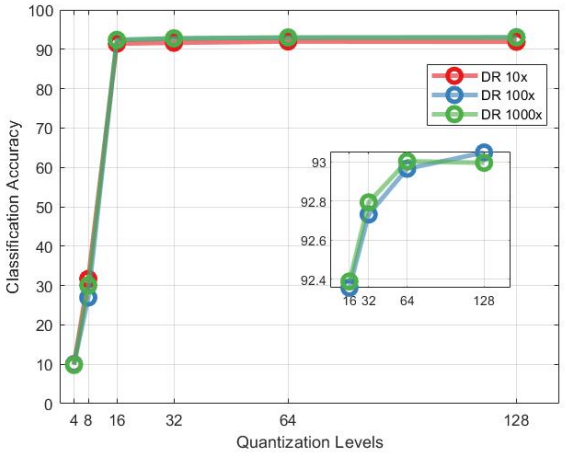

FIGURE 7. Classification accuracies for different memristor device dynamic ranges as a function of various QLs with 10% variation for MNIST classification on ANN.

Quantization Table 1 demonstrates the obtained classification accuracies for each one of the considered quantization levels if each value was substituted by the mean of the range it resides in. The accuracy is significantly affected at low-bit representation with around 10% classification accuracy. The accuracy starts to increase with 16 levels of quantization until it reaches an acceptable level at 64 levels or higher. The classification values in the table were obtained with 10× DR.

Dynamic Range Memristor devices have shown a large off/on resistance ratio that can reach up-to 10^4 [42]. Hence, in our simulations, three cases with 10×, 100×, and 1000× were considered. By examining the histograms for all these ranges, it shows that the distribution for the 4-QL among all DRs is narrow and concentrated mainly between 9%-11% accuracies for the 1000 samples. We only demonstrate the histogram for the 100× DR in Fig. 5 since the other two follow the same trend. It can be noticed that with 8-QLs, the distribution of accuracies starts to spread. Higher QLs are observed to have a right-skewed distribution with most occurrences having >90% classification accuracies. It should be noted that all these plots were obtained using 10% weight variation around the mean.

The curves in Fig. 6 present the cumulative distribution function (CDF) and shows the probability that a particular sample takes in classification accuracy. For example, by examining the blue curve in Fig. 6, the probability of having a classification accuracy of 50% or less is about 75% for 8-QL. Moreover, the probability of having a 10% for 4 levels of quantization is almost the same. While for higher QLs, the probability of obtaining an accuracy <90% is around 10%. But for 93% it grows to around 90%. CDF was also calculated for the 100× and 1000×, and it was concluded that regardless of the dynamic range of the memristor device, all QL levels follow the same curve.

The mean of the 1000 samples for each quantization level and over all three different dynamic ranges were calculated. Fig. 7 shows how the mean of accuracies per level is small for low number of quantization levels (low number of bits). And intuitively, as the number of bits increases (QL), the accuracy increases. But there is an upper limit at 16-QLs where the accuracy saturates and no apparent benefit for going to higher number of states. Since there is a jump from 8-QLs to 16-QLs, another intermediate value which is 9-QLs was tested and achieved an 88.98% classification accuracy. It should be noted that the simulated 9-QLs had 10% variation and 10× dynamic range.

Moreover, although increasing the dynamic range of a Memristor device helps to relax the noise margin and the precision of the writing circuitry, in our simulations, it did not show much of a difference in accuracy. That is mainly due to limiting the truncated Gaussian distribution to a certain range where weights can vary within.

Weight Variation For further analysis of different weight variations, the 16-QLs case was selected to proceed with as it gives the best accuracy with the lowest states. The boxplot representation for 5%, 10%, 20% and 30% variations of the

TABLE 2. CIFAR-10 classification accuracy for both original CNN and Memristor-based CNN. Simulations were carried on for a layer-by-layer quantization as well as quantizing the whole CNN pipeline with 16-QLs and 10× DR. Also, the effect of variation on the fully quantized Memristor-based CNN as well as the layer-by-layer quantization is demonstrated. Results were compared to the baseline network accuracy.

	Original CNN	Memristor-based CNN (No variation)	Memristor-based CNN (with variation)
Full CNN (baseline)	57.12%	57.12%	-
Quantized CNN	30.52%	31.12%	(1%/ 3%/ 10%) 29.88%/ 23.50%/ 15.40%
ConV1	46.14%	46.84%	(10%) 36.03%
ConV2	41.00%	42.22%	(10%) 39.67%
ConV3	47.73%	47.66%	(10%) 45.33%
FC1	52.84%	41.39%	(10%) 29.26%
FC2	56.07%	54.28%	(10%) 55.11%

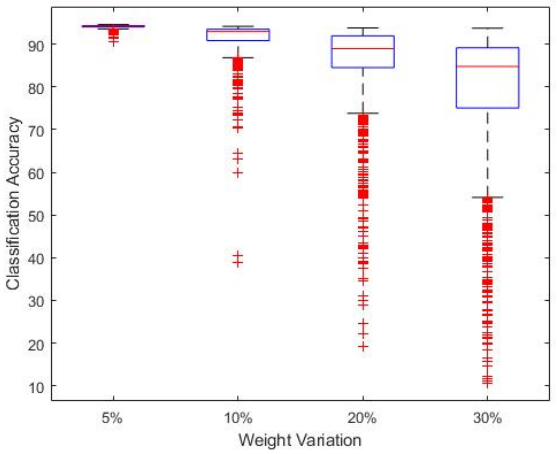

FIGURE 8. Boxplot representation for 5%, 10%, 20% and 30% of ANN weight variation cases for MNIST classification. It shows that for smaller variations the spread is narrow and contained in the whiskers. While for higher variations the spread of the output results is large with many outliers.

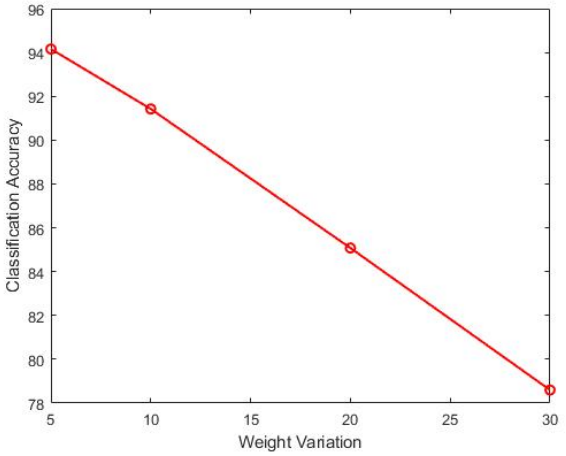

FIGURE 9. The mean of the variations 5%, 10%, 20% and 30% respectively, showing a decrease in the classification accuracy of MNIST data set as the programing variation increase.

weight around the mean is shown in Fig. 8. It demonstrates that for smaller variations, the majority of accuracies for the 1000 samples have a narrow distribution and are >90%. While the conductance variation increases, this causes the distribution of the classification accuracy of the 1000 samples to go wider, causing many outliers, and the accuracy of these samples is skewed. Consequently, the mean of the accuracies drops as the device variation increases; this is presented in Fig. 9.

VI. CNN WEIGHT QUANTIZATION AND VARIATION

In this section, the work has been extended to account for other DNNs such as the state-of-the-art CNN. The goal is to provide a predictive performance of any pre-trained CNN on the dedicated Memristor-based hardware accelerator. The dedicated accelerator is constrained by the physical number of quantization (conductance) levels the memristor device can exhibit and the expected amount of variations in its conductance. In order to follow the same flowchart demonstrated in Fig. 2 and convert the convolution operation to vector-matrix multiplication, the following must be performed: Every channel within the layer resembles a memristive crossbar. The filters in every channel are flattened to 1D and concatenated with a number of columns equal to the number of filters in that channel as demonstarted in Fig. 10. For example, if there are 16 filters with a dimension of 3×3, then there will be 16 minimum values one from each filter. Hence, the crossbar will have a size of 9×32. These additional 16 columns hold the absolute minimum value of each filter in

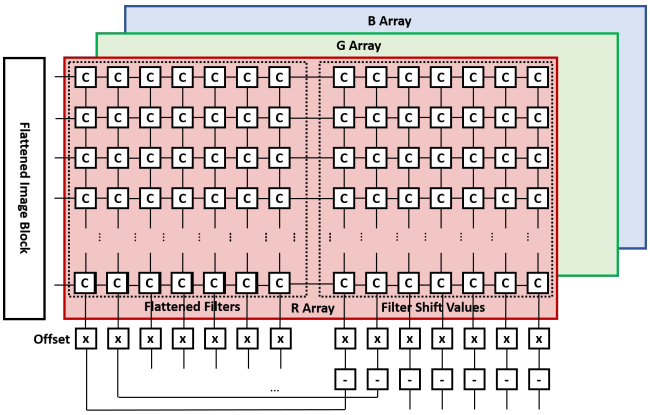

FIGURE 10. Architecture of the RRAM-based convolution layers organized by concatenating the shifting columns, addition of offset multipliers, output subtractors, and array-wise adders. The lump of the effect of non-idealities is included in the offset value.

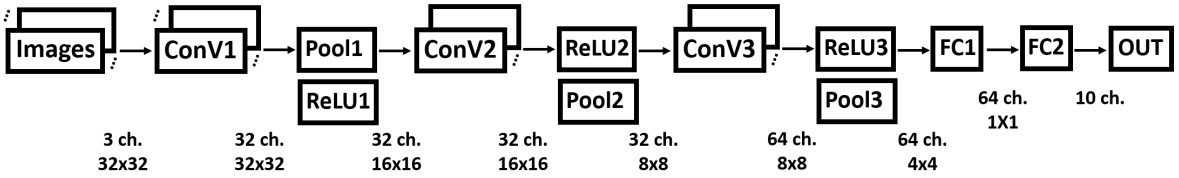

FIGURE 11. A CNN structure consisting of 3-convolution and 2-fully connected layers. CIFAR-10 data set was used on this structure to study the impact of post-training quantization and weight variation.

that channel. After that, every image block is flattened to 1D, converted to voltage, and applied at the crossbar's terminals. Then same procedure highlighted in Section III is followed. In the following simulation results and discussions, 16-QLs and a dynamic range of 10× were assumed.

As an example to demonstrate the proposed framework on CNN, a 3 ConV and 2 FC (3C2F) layer neural network is presented in Fig. 11 where CIFAR-10 data-set was trained on Caffe deep learning framework. After that, the network was quantized to 16 levels in a layer-by-layer fashion to study the effect of quatization and weight variation separately. Table 2 presents the classification accuracy of the baseline CNN, fully-quantized network and layer-by-layer quantized CNN. It shows that the latter layers are more robust to the quantization from the earlier ones. Then weight variation of 10% was introduced to the layer-by-layer quantized network as demonstrated in column 4 in Table 2. Moreover, the weights of a quantized network were varied by 1%, 3%, and 10% to analyze the sensitivity to the variation on the fully quantized network. Results are highlighted in the second row and fourth column of the Table. The presented outcomes show that the effect of quantization is more dominant as long as the conductance variation is below 3%.

It is worth mentioning that layer-by-layer mapping of the quantized weights to the quantization levels (physical available conductance states) performed better than a one time mapping of the pre-trained weights to the range. For example, for original CNN when quantizing the whole network with one maximum, the accuracy was 9.96%. While when each layer was quantized depending on its maximum, the accuracy was 30.52%. It should be noted that the classification accuracy for 64 QLs is 47.18%.

In addition to the classification accuracy, confusion matrix showing precision and recall factors are presented. Fig. 12 illustrates how the images are classified for a baseline memristor-based CNN assuming full-precision can be achieved without quantization and conductance variation. It gives an insight into the predicted labels compared to the true ones through calculations of the true positives, false positives, true negative and false negative for each class. With quantization and variation, the network seems to classify the majority of images to airplane and horse as Fig. 13 depicts.

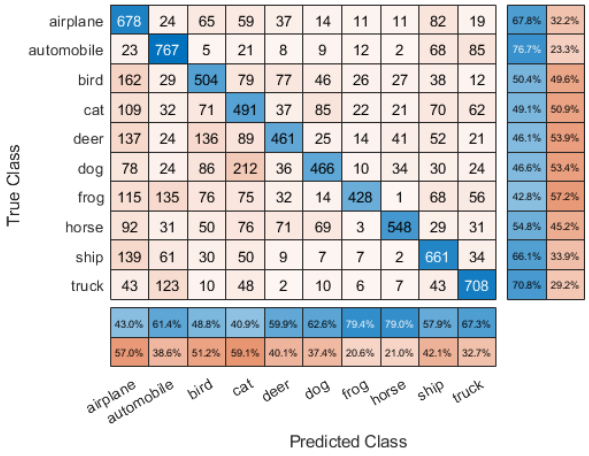

FIGURE 12. Confusion Matrix showing precision and recall values for CIFAR-10 on Memristor-based CNN without quantization and variation.

VII. CONCLUSIONS AND FUTURE WORK

This paper presents a methodology for investigating the impact of post-quantization, variation, and dynamic range in ReRAM-based pre-trained DNN for MNIST and CIFAR-10 classification. The framework is application-applicable, where the user can explore other sets. Also, the framework helps when there is no access to the training data sets to the end-user for security reasons and in generic cases where the network has already been trained. Our simulation results show that the accuracy depends on the DNN structure and complexity of inputs. For example, for the simple multi-layer perceptron, 4-bits of information is enough to represent distinct conductance weights and achieve near-optimal accuracy ∼90% compared to full-precision accuracy 94.58%. While in the state-of-the-art CNN, memristor devices with a higher number of stable conductance states are required. Moreover, performing a layer-by-layer quantization instead of quantizing the network as a whole resulted in better accuracy: 30.52% versus 9.96%. The effect of weight variation on CNN is less than 8% when the variation is $< 3\%$.

In addition, a large dynamic range of the devices (HRS/LRS) is desirable to enable multi-level conductance representation for the NN weight matrices values. Due to intrinsic variations, reduced noise margin between successive resistance levels will affect the operation's reliability and correctness. However, in our simulations larger dynamic ranges did not affect the classification accuracy. A detailed MATLAB code is reported in the appendix together with GitHub codes available on: https://github.com/YasmineHalawani/Statistical-

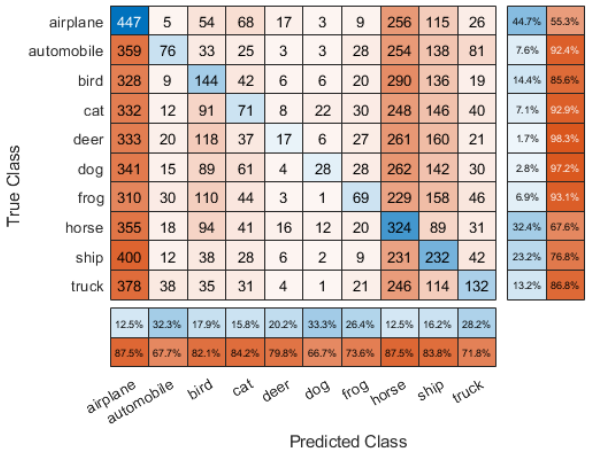

FIGURE 13. Confusion matrix showing precision and recall for CIFAR-10 classification on a quantized Memristor-based CNN (all layers) with variation = 10%.

Analysis-ReRAM-Neural-Network.

In the future, a compensation technique to mitigate the accuracy drop due to post-training quantization and variation for memristor-based NN structures targeting edge devices will be utilized. Also, a better network-to-crossbar mapping without the need for re-training.

APPENDIX.

The following code can be used after training the neural network (NN) and obtaining all the trained weights for all layers. After that, the NN network weights are mapped into conductance values given the available physical range of the ReRAM device and that all weights should be positive. Also, the input testing patterns from the data set are mapped into appropriate voltages. The number of inputs, outputs and layers are data set and design dependent. The number of samples, can be considered as the number of wafers, are decided by the user. Also the variation of the mapped weight around the mean is controlled by the variable `var`. The number of quantization levels assumed is allocated by

```
% This code is for testing the variatoins in
% the values of the weights/conductance
% versus the classification accuracy with
% certain QLs.

% MNIST images flattened to 1D
n_inputs = 784;
% number of classes to classify
n_outputs = 10;
% number of hidden nodes
n_hidden = 32;

% Load hidden layer NN weights
% Load output layer NN weights
% Load testing patterns
% Load testing labels
% Perform one hot encoding for the labels

% Shift the NN weights in both hidden and
% output layer by the minimum value
% Specify the conductance dynamic range
Goff = 1/900; %Siemens
Gon = 1/110; %Siemens

% Map the shifted weights to
% conductance values


%% Monte-Carlo Simulations
n_samples = 1000;
% conductance variation from the mean 5%
var = 0.05;

% for each wafer test all input data
for k = 1:n_samples
    %N-level quantization
    q_level = 4;
    % Check the ranges
    [N,edges] = histcounts(conductance_h,
        q_level);
    for i = 1:size(conductance_h,1)
        for j = 1:size(conductance_h,2)
            if conductance_h(i,j) >= edges
                (1) && conductance_h(i,j) <
                edges(2)
                m = [edges(1) edges(2)];
                % Normal dist. with mean and
                    variation
                pd = makedist('normal','mu',
                    mean(m), 'sigma', mean(m
                    ) + (var*mean(m)));
                upper = mean(m) + (var*mean(
                    m));
                lower = mean(m) - (var*mean(
                    m));
                t = truncate(pd, lower,upper
                    );
                % Substitite the value of
                    the conductance_h by a
                    random
                % number from the range
                conductance_h(i,j) = random(
                    t,1,1);

            elseif conductance_h(i,j) >=
                edges(2) && conductance_h(i,j
                ) < edges(3)
                m = [edges(2) edges(3)];
                pd = makedist('normal','mu',
                    mean(m), 'sigma', mean(m
                    ) + (var*mean(m)));
                upper = mean(m) + (var*mean(
                    m));
                lower = mean(m) - (var*mean(
                    m));
                t = truncate(pd, lower,upper
                    );
                conductance_h(i,j) = random(
                    t,1,1);

            elseif conductance_h(i,j) >=
                edges(3) && conductance_h(i,j
                ) < edges(4)
                m = [edges(3) edges(4)];
                pd = makedist('normal','mu',
                    mean(m), 'sigma', mean(m
```

```
                    ) + (var*mean(m)));
                upper = mean(m) + (var*mean(
                    m));
                lower = mean(m) - (var*mean(
                    m));
                t = truncate(pd, lower,upper
                    );
                conductance_h(i,j) = random(
                    t,1,1);

            elseif conductance_h(i,j) >=
                edges(4) && conductance_h(i,j
                ) <= edges(5)
                m = [edges(4) edges(5)];
                pd = makedist('normal','mu',
                     mean(m), 'sigma', mean(m
                    ) + (var*mean(m)));
                upper = mean(m) + (var*mean(
                    m));
                lower = mean(m) - (var*mean(
                    m));
                t = truncate(pd, lower,upper
                    );
                conductance_h(i,j) = random(
                    t,1,1);
            end
        end
    end

    % Perform the same quantization and
        variation procedure for the output
    % layer matrix

    % Once the hidden and output matrices
        are ready move to ANN testing
    % phase
     for i = 1 : size(testData,1)  %All
         testing patterns
        % Perform forward pass for the ANN
        % 1) Vector matrix multiplication
            between the voltage inputs and
            the
        % hidden conductance matrix
        % 2) Pass the outputs through a non-
            linear activation function
        % 3) Vector matrix multiplication
            between the output of the hidden
        % layer and the output conductance
            matrix

    end

    % Evaluate classification accuracy
end
```

ACKNOWLEDGMENT

This publication is based upon work supported by the Khalifa University Competitive Internal Research Award (CIRA) under Award No. [CIRA-2019-026], and System-on-Chip Center Award No. [RC2-2018-020].

REFERENCES

[1] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE journal of solid-state circuits, vol. 52, no. 1, pp. 127–138, 2016.

[2] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. Ong Gee Hock, Y. T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra, et al., "Can FPGAs beat GPUs in accelerating next-generation deep neural networks?," Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp. 5–14, 2017.

[3] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: On-demand accelerating deep neural network inference via edge computing," IEEE Transactions on Wireless Communications, 2019.

[4] J.-M. Hung, X. Li, J. Wu, and M.-F. Chang, "Challenges and trends in developing nonvolatile memory-enabled computing chips for intelligent edge devices," IEEE Transactions on Electron Devices, vol. 67, no. 4, pp. 1444–1453, 2020.

[5] Y. Halawani, B. Mohammad, M. A. Lebdeh, M. Al-Qutayri, and S. F. Al-Sarawi, "ReRAM-based in-memory computing for search engine and neural network applications," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 2, pp. 388–397, 2019.

[6] H. Huang, L. Ni, K. Wang, Y. Wang, and H. Yu, "A highly parallel and energy efficient three-dimensional multilayer cmos-rram accelerator for tensorized neural network," IEEE Transactions on Nanotechnology, vol. 17, no. 4, pp. 645–656, 2017.

[7] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," Nature Nanotechnology, 2020.

[8] M. Le Gallo and A. Sebastian, "An overview of phase-change memory device physics," Journal of Physics D: Applied Physics, 2020.

[9] H. Abunahla, B. Mohammad, L. Mahmoud, M. Darweesh, M. Alhawari, M. A. Jaoude, and G. W. Hitt, "Memsens: Memristor-based radiation sensor," IEEE Sensors Journal, vol. 18, no. 8, pp. 3198–3205, 2018.

[10] Y. Pang, B. Gao, D. Wu, S. Yi, Q. Liu, W.-H. Chen, T.-W. Chang, W.-E. Lin, X. Sun, S. Yu, et al., "25.2 a reconfigurable rram physically unclonable function utilizing post-process randomness source with< 6× 10- 6 native bit error rate," IEEE International Solid-State Circuits Conference-(ISSCC), pp. 402–404, 2019.

[11] Y. Halawani, B. Mohammad, M. Al-Qutayri, and S. F. Al-Sarawi, "Memristor-based hardware accelerator for image compression," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 12, pp. 2749–2758, 2018.

[12] Y. Halawani, M. A. Lebdeh, B. Mohammad, M. Al-Qutayri, and S. F. Al-Sarawi, "Stateful memristor-based search architecture," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 12, pp. 2773–2780, 2018.

[13] H. Abunahla, Y. Halawani, A. Alazzam, and B. Mohammad, "NeuroMem: Analog graphene-based resistive memory for artificial neural networks," Scientific Reports, vol. 10, no. 1, pp. 1–11, 2020.

[14] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-net: Imagenet classification using binary convolutional neural networks," in European conference on computer vision, pp. 525–542, Springer, 2016.

[16] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks," in Proceedings of the 2nd SysML Conference, 2019.

[17] D. Wan, F. Shen, L. Liu, F. Zhu, J. Qin, L. Shao, and H. Tao Shen, "Tbn: Convolutional neural network with ternary inputs and binary weights," in Proceedings of the European Conference on Computer Vision (ECCV), pp. 315–332, 2018.

[18] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," arXiv preprint arXiv:1805.06085, 2018.

[19] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8612–8620, 2019.

[20] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," Proceedings of the 56th Annual Design Automation Conference, pp. 1–6, 2019.

[21] S. Jain, A. Sengupta, K. Roy, and A. Raghunathan, "RxNN: A framework for evaluating deep neural networks on resistive crossbars," arXiv preprint arXiv:1809.00072, 2018.

[22] I. Chakraborty, M. F. Ali, D. E. Kim, A. Ankit, and K. Roy, "Geniex: A generalized approach to emulating non-ideality in memristive xbars

using neural networks," in 2020 57th ACM/IEEE Design Automation Conference (DAC), pp. 1–6, IEEE, 2020.
[23] M. A. Hanif, A. Manglik, and M. Shafique, "Resistive crossbar-aware neural network design and optimization," IEEE Access, 2020.
[24] X. Sun and S. Yu, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 9, no. 3, pp. 570–579, 2019.
[25] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in Advances in Neural Information Processing Systems, pp. 7948–7956, 2019.
[26] M. de Prado, M. Denna, L. Benini, and N. Pazos, "Quenn: Quantization engine for low-power neural networks," in Proceedings of the 15th ACM International Conference on Computing Frontiers, pp. 36–44, 2018.
[27] J. Fang, A. Shafiee, H. Abdel-Aziz, D. Thorsley, G. Georgiadis, and J. Hassoun, "Near-lossless post-training quantization of deep neural networks via a piecewise linear approximation," arXiv preprint arXiv:2002.00104, 2020.
[28] L. Xia, B. Li, T. Tang, P. Gu, P.-Y. Chen, S. Yu, Y. Cao, Y. Wang, Y. Xie, and H. Yang, "Mnsim: Simulation platform for memristor-based neuromorphic computing system," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 37, no. 5, pp. 1009–1022, 2017.
[29] W. Zhang, X. Peng, H. Wu, B. Gao, H. He, Y. Zhang, S. Yu, and H. Qian, "Design guidelines of rram based neural-processing-unit: A joint device-circuit-algorithm analysis," 56th ACM/IEEE Design Automation Conference (DAC), pp. 1–6, 2019.
[30] H. Liu, J. Han, and Y. Zhang, "A unified framework for training, mapping and simulation of ReRAM-based convolutional neural network acceleration," IEEE Computer Architecture Letters, vol. 18, no. 1, pp. 63–66, 2019.
[31] Y. Sun, X. Yan, X. Zheng, Y. Liu, Y. Shen, and Y. Zhang, "Influence of carrier concentration on the resistive switching characteristics of a ZnO-based memristor," Nano Research, vol. 9, no. 4, pp. 1116–1124, 2016.
[32] B. Mohammad, M. A. Jaoude, V. Kumar, D. M. Al Homouz, H. A. Nahla, M. Al-Qutayri, and N. Christoforou, "State of the art of metal oxide memristor devices," Nanotechnology Reviews, vol. 5, no. 3, pp. 311–329, 2016.
[33] B. Li, P. Gu, Y. Wang, and H. Yang, "Exploring the precision limitation for RRAM-based analog approximate computing," IEEE Design & Test, vol. 33, no. 1, pp. 51–58, 2016.
[34] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," in IEEE International Electron Devices Meeting (IEDM), pp. 6–1, 2017.
[35] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+ NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in IEEE International Electron Devices Meeting (IEDM), pp. 32–5, 2019.
[36] G. Yuan, X. Ma, C. Ding, S. Lin, T. Zhang, Z. S. Jalali, Y. Zhao, L. Jiang, S. Soundarajan, and Y. Wang, "An ultra-efficient memristor-based DNN framework with structured weight pruning and quantization using ADMM," in 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 1–6, IEEE, 2019.
[37] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," arXiv preprint arXiv:1806.08342, 2018.
[38] W. Sun, B. Gao, M. Chi, Q. Xia, J. J. Yang, H. Qian, and H. Wu, "Understanding memristive switching via in situ characterization and device modeling," Nature communications, vol. 10, no. 1, pp. 1–13, 2019.
[39] H. Abunahla, B. Mohammad, D. Homouz, and C. J. Okelly, "Modeling valance change memristor device: Oxide thickness, material type, and temperature effects," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 63, no. 12, pp. 2139–2148, 2016.
[40] L. Wu, H. Liu, J. Li, S. Wang, and X. Wang, "A multi-level memristor based on al-doped hfo 2 thin film," Nanoscale research letters, vol. 14, no. 1, pp. 1–7, 2019.
[41] W. He, H. Sun, Y. Zhou, K. Lu, K. Xue, and X. Miao, "Customized binary and multi-level hfo 2- x-based memristors tuned by oxidation conditions," Scientific reports, vol. 7, no. 1, pp. 1–9, 2017.
[42] A. A. Minnekhanov, A. V. Emelyanov, D. A. Lapkin, K. E. Nikiruy, B. S. Shvetsov, A. A. Nesmelov, V. V. Rylkov, V. A. Demin, and V. V. Erokhin, "Parylene based memristive devices with multilevel resistive switching for neuromorphic applications," Scientific reports, vol. 9, no. 1, pp. 1–9, 2019.
[43] C. Li, M. Hu, Y. Li, H. Jiang, N. Ge, E. Montgomery, J. Zhang, W. Song, N. Dávila, C. E. Graves, et al., "Analogue signal and image processing with large memristor crossbars," Nature Electronics, vol. 1, no. 1, p. 52, 2018.
[44] J. Park, M. Kwak, K. Moon, J. Woo, D. Lee, and H. Hwang, "Tio x-based rram synapse with 64-levels of conductance and symmetric conductance change by adopting a hybrid pulse scheme for neuromorphic computing," IEEE Electron Device Letters, vol. 37, no. 12, pp. 1559–1562, 2016.
[45] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6, 2016.
[46] B. Li, Y. Shan, M. Hu, Y. Wang, Y. Chen, and H. Yang, "Memristor-based approximated computation," Proceedings of the International Symposium on Low Power Electronics and Design, pp. 242–247, 2013.
[47] M. Hu, H. Li, Y. Chen, Q. Wu, G. S. Rose, and R. W. Linderman, "Memristor crossbar-based neuromorphic computing system: A case study," IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 10, pp. 1864–1878, 2014.
[48] P. Gu, B. Li, T. Tang, S. Yu, Y. Cao, Y. Wang, and H. Yang, "Technological exploration of RRAM crossbar array for matrix-vector multiplication," The 20th Asia and South Pacific Design Automation Conference, pp. 106–111, 2015.
[49] M. Jiang and G. Gielen, "The effects of quantization on multi-layer feedforward neural networks," International journal of pattern recognition and artificial intelligence, vol. 17, no. 04, pp. 637–661, 2003.
[50] J. Burkardt, "The truncated normal distribution," Department of Scientific Computing Website, Florida State University, pp. 1–35, 2014.
[51] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," Nature materials, vol. 18, no. 4, pp. 309–323, 2019.
[52] Y. Halawani, B. Mohammad, D. Homouz, M. Al-Qutayri, and H. Saleh, "Modeling and optimization of memristor and STT-RAM-based memory for low-power applications," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 3, pp. 1003–1014, 2015.