Quantization Considerations of Dense Layers in Convolutional Neural Networks for Resistive Crossbar Implementation

Lei Zhang
Circuits and Systems
Fraunhofer Research Institution for Microsystems and Solid State Technologies EMFT
Munich, Germany
lei.zhang@emft.fraunhofer.de

David Borggreve
Circuits and Systems
Fraunhofer Research Institution for Microsystems and Solid State Technologies EMFT
Munich, Germany
david.borggreve@emft.fraunhofer.de

Frank Vanselow
Circuits and Systems
Fraunhofer Research Institution for Microsystems and Solid State Technologies EMFT
Munich, Germany
frank.venselow@emft.fraunhofer.de

Ralf Brederlow
Chair of Circuit Design
Technical University of Munich
Munich, Germany
r.brederlow@tum.de

***Abstract*— The accuracy and power consumption of resistive crossbar circuits in use for neuromorphic computing is restricted by the process variation of the resistance-switching (memristive) device and the power overhead of the mixed-signal circuits, such as analog-digital converters (ADCs) and digital-analog converters (DACs). Reducing the signal- and weight-resolution can improve the robustness against process variation, relax requirements for mixed-signal devices, and simplify the implementation of crossbar circuits. This work aims to establish a methodology to achieve low-resolution dense layers for CNNs in terms of network architecture selection and quantization method. To this end, this work studies the impact of the dense layer configuration on the required resolution for its inputs and weights in a small convolutional neural network (CNN). This analysis shows that carefully selecting the network architecture for the dense layer can significantly reduce the required resolution for its input signals and weights. This work reviews criteria for appropriate architecture selection and the quantization method for the binary and ternary neural network (BNN and TNN) to reduce the weight resolution of CNN dense layers. Furthermore, this work presents a method to reduce the input resolution for the dense layer down to one bit by analyzing the distribution of the input values. A small CNN for inference with one-bit quantization for inputs signals and weights can be realized with only 0.68% accuracy degradation for MNIST Dataset.**

Keywords—Convolutional Neural Network, Neuromorphic Computing Hardware, Approximate Computing, Neural Network Quantization, Resistive Crossbar, Memristive Devices

I. INTRODUCTION

The resistive crossbar circuit has received much attention in recent years due to its analog property, which performs the matrix-vector multiplication by following Kirchhoff's law and offers a sharply reduced computing time complexity of O (1) for matrix-vector multiplication in comparison to the time complexity of the digital computing method O(N^2) [1]. It means that the computation time of crossbar is independent of amount of inputs and outputs. With the benefit of using this analog property, the application of resistive crossbar circuits has been extended to not only the matrix-vector multiplication, convolution for the dense layer and convolutional layer but also the application level for Discrete Fourier Transform [2] and image processing [3].

There are still many design considerations to be taken into account during the circuit implementation due to process variations and the power overhead of mixed-signal devices. Notably, the process variation has been continuously confining the scalability of most implementations of the memristive array [4] [5]. To mitigate the effects of the process variation of devices, either the in-situ learning (e.g. [6]) or the closed-loop verification algorithm is utilized so far for finding an optimal weight mapping value for memristive devices (e.g. [7], [8]). Moreover, the power overhead of mixed-signal devices cannot be neglected during the design. In ISAAC [9], which consists of many analog cores to compute convolution and matrix-vector multiplication, ADCs and DACs contribute more than 60% to the total power consumption [9].

Recently, digital neuromorphic computing hardware with approximate computing techniques have generated considerable research interest due to sharply reduced computation complexity. The research shows that the binary neural network (BNN) [10] and ternary neural network (TNN) [11], which contain only binary weights (-1, +1) and ternary weights (-1, 0, +1) respectively, can achieve high inference accuracy and accelerate the computing speed of the digital core using digital implementations with XNOR's [12]. It indicates that a good hardware/software co-design not only improves the computation speed but also reduces the design complexity for the digital circuit implementation.

The concept of hardware/software co-design can also be performed for the analog/mixed-signal design to enable an optimal circuit implementation. Especially mixed-signal circuits and memory cells may benefit from these approximate computing techniques, by not only reducing the resolution requirements, but adding robustness to process variability, allowing for more power and area efficient design options using relatively simple circuit blocks. In our case, a hardware

accelerator for CNNs is investigated, in which the full-precise convolutional layers and the low-resolution dense layers will be implemented with digital and analog circuits, respectively. To this end, this work aims to provide some guidelines for selecting proper dense layer architecture and using quantization methods to simplify the implementation of the dense layer with crossbar circuits. In section II, this work explains roughly the basic idea of the resistive crossbar circuit. The results of experimental explorations for the impact of the dense layer's configuration are discussed in section III. In section IV, this work reviews quantization techniques for the BNN/TNN and proposes a 1-bit quantization method for inputs of the dense layer in CNNs based on analyzing its probability distribution. In the end, section V will summarize the observations as a conclusion.

II. Resistive Crossbar Circuit

The crossbar circuit is a well-known circuit architecture because of its widespread use for memory implementation. Every memory cell connects to two perpendicular wires, which are called bit-line (BL) and word-line (WL). By activating a certain BL and WL, the memory can be accessed for writing and reading. Inspired by this, densely located neurons with high parallelism can be realized with wires as axons and dendrites, and resistance-switching cells (e.g. memristor) as elementary synapses [13].

Fig.1 shows a possible arrangement of the resistive crossbar circuit for the dense layer. The matrix-vector multiplication for the dense layer can be performed by applying Ohm's law and Kirchhoff's laws on the crossbar circuit as following: Firstly, the DAC_i converts the digital signal to the analog voltage $V_{in,i}$ as input for the dense layer, where the input voltage vector V_{in} can be written as Eq. (1).

$$V_{in} = \begin{bmatrix} V_{in,1} \\ V_{in,2} \\ \vdots \\ V_{in,m} \end{bmatrix} \tag{1}$$

Then, the current, which flow through resistance-switching devices (memristors), can be calculated as the product of the input voltage vector and the memristor conductance matrix G. (Eq. (2).)

$$G = \begin{bmatrix} G_{1,1} & \cdots & G_{1,n} \\ \vdots & \ddots & \vdots \\ G_{m,1} & \cdots & G_{m,n} \end{bmatrix} \tag{2}$$

Output current I can be written as the product of the matrix-vector multiplication of the conductance matrix G and input voltage vector V_{in} by neglecting the sneak-path problem [14] and the floating voltage at the output (Eq. (3)). The output of the crossbar circuit can be fed into either an analog or a digital activation function. In latter case, additional ADC's are needed.

This project has received funding partly from the Electronics Components and Systems for European Leadership Joint Undertaking under grant agreement No 826655. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program from Belgium, France, Germany, Switzerland and the Netherlands. This project is also partly funded by German Federal Ministry of Education and Research (BMBF) under Grant No 16ESE0407.

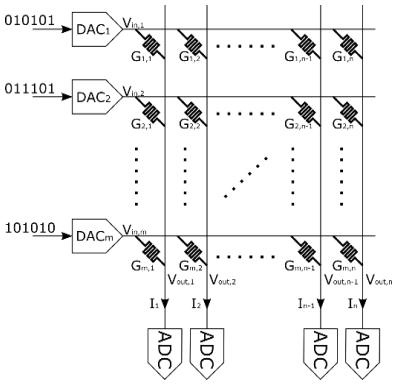

Fig. 1. The basic concept for crossbar implementation of matrix-vector multiplication, which might be used for the dense layer in our design. The DACs convert the digital signals from the convolutional layer to analog voltages, and the multiplication can be performed by applying the ohm's law for every programmable resistor (memristor). The resulted current is the product of the multiplication. If required, the output current can converted back to digital value by using ADCs.

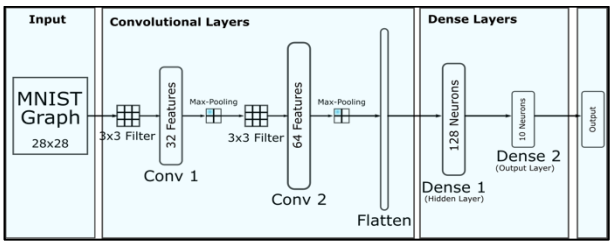

Fig. 2. The initial configuration of the convolutional neural network, which is used in this experiment and consists of two convolutional layers and two dense layers with 128 neurons and 10 neurons, respectively. Relu and Sigmoid functions are used as activation function for convolutional layers and dense layers, respectively.

$$I = G^T \cdot V_{in} = \begin{bmatrix} G_{1,1} & \cdots & G_{m,1} \\ \vdots & \ddots & \vdots \\ G_{1,n} & \cdots & G_{m,n} \end{bmatrix} \cdot \begin{bmatrix} V_{in,1} \\ V_{in,2} \\ \vdots \\ V_{in,m} \end{bmatrix} \tag{3}$$

However, the one-to-one mapping of the weight values from neural network to the crossbar circuit does not result in the expected output, because of the non-idealities of the memory cell [7]. Overcoming this problem requires an optimal mapping algorithm, which converts weights to proper conductance. Additionally, high-resolution DACs and ADCs require a significant amount of energy per conversion [9]. The low-resolution representation of the weight and input strongly simplifies the mapping algorithm, making weight storage more robust and decreasing the required resolution or rather the power overhead of the DACs and ADCs. By selecting an optimal network configuration of the dense layers, a lower-resolution implementation can be made at a limited overall inference accuracy penalty. The next section will explore the impact of architecture on the required resolution for weights and inputs of dense layers in CNNs.

III. Exploration of the Architecture

Fig.2 illustrates the initial configuration of the CNN, which consists of 2 convolutional layers with respectively 32 and 64 extracted features, a max-pooling layer for every convolutional layer and 2 dense layers with 128 and 10

TABLE I. THE CONFIGURATIONS OF THE DENSE LAYER IN CNN

Group: Depth				
Name	**Depth 0**	**Initial**	**Depth 2**	**Depth 3**
Configuration of dense layer	10	128 – 10	128 – 256 – 10	128 – 256 – 128 – 10
Group: Width				
Name	**Initial**	**Width 1**	**Width 2**	**Width 3**
# Neurons	128 – 10	256 – 10	512 – 10	1024 – 10

neurons as the hidden layer and output layer. In order to explore the impact of the dense layer's configuration on the entire network's accuracy, this CNN is trained with different widths and depths of the dense layer to recognize the handwritten digits from the MNIST dataset [15]. The width and depth denote the amount of the neurons in the hidden layer and the number of the layer as the dense layer, respectively.

Table I shows groups of configurations studied according to the width and depth of the dense layer. The following cases have been studied: The Initial dense layer's configuration 128-10 is the reference from which the two groups deviate in Depth and Width, respectively. The Depth-0 case is a special case, since it does not have any hidden layer, and the output of the last convolutional layer connects directly with a dense output layer.

In this experiment, the 3-bits means that only the first 3 bits of the original value is used, instead of using full-precision values.

A. *Accuracy vs. Width and Depth of the dense layer*

The first experiment compares the impact of the different dense layer configurations on the total accuracy of this CNN for MNIST. Fig.3 indicates that the accuracy of the entire CNN is rarely dependent on the width and depth of the dense layer. Even if the classification is performed with only one dense layer (Depth 0), the accuracy remains 99.42%, which is just 0.09% less than maximal accuracy with Width 2. Additionally, the deeper or wider dense layer does not help too much to improve the network accuracy.

B. *Accuracy vs. weight with limited resolution*

Although the accuracy of the entire CNN does not change too much, the required resolution for weights and inputs of the dense layer still varies in terms of keeping possible small accuracy degradation according to the distinctive dense layer configurations.

Fig.4(a) shows the accuracy degradation of networks with Tab.1 listed configurations if full-precise weights of dense layers are converted to 2-, 3- and 4-bits. The result illustrates that dense layers require at least 3-bits weights to maintain network works since the accuracy degradation reaches at least 22.17% with 2-bits weights by using Width 1. With the deeper network, the accuracy degradation fluctuates slightly, and no reduction of the required resolution for weights can be observed.

On the other hand, the wider structure for the 2-bits weighted dense layer does not necessarily reduce the accuracy degradation, either. As illustrated in Fig.4(a), two-layers dense layer with 256 neurons (Width 1) in the hidden layer yields the smallest accuracy degradation of 22.17%. If more neurons are arranged in the hidden layer of this configuration, such as

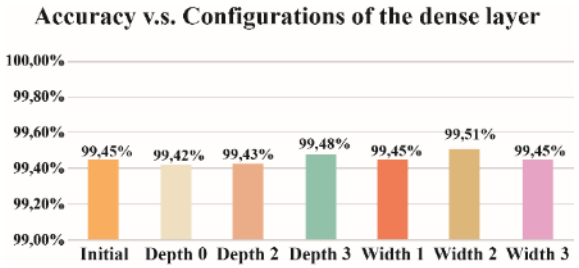

Fig. 3. A comparison for different configuration of the dense layer of CNN is shown in terms of the total accuracy of the entire CNN.

Width 2 and Width 3, the accuracy degradation grows even. It leads to that only a proper amount of parameters or neurons reduces accuracy degradation if weights are converted to low resolution.

C. *Accuracy vs. input with limited resolution*

If the weight of the dense layer has a full-precise resolution, the required resolution for the input of the dense layer is shown as Fig.4(b) according to its configurations. Fig.4(b) illustrates that Depth 0 achieves the largest accuracy degradation of 27.49% and the deeper structures (Depth 1 and Depth 2) have generally less accuracy degradation if 2-bit inputs are used for all. Similar to the previous experiment for the resolution of weights, a proper amount of neurons in the hidden layer (Width 2) can result in less accuracy degradation if a 2-layer dense layer is used.

The observation points out that it is better to pick deeper structure in order to have less required resolution for inputs.

D. *Accuracy vs. limited resolution for both weights and inputs*

In this experiment, the resolution for both weights and inputs is limited in order to investigate how few bits can be used for weights and inputs of the dense layer without huge accuracy degradation. The resolution of the weight is fixed as 3-bits, which has maximal accuracy degradation of 2.69%, is observed as Fig.4(a). However, if a 2-bits resolution limitation is applied to the input at the same time, Fig.4(c) illustrates that the minimal accuracy degradation can achieve 2.93% with Width 2.

The deeper structures Depth 2 and Depth 3 generally have less accuracy degradation around 4% in comparison to shallower structures except Width 2, as illustrated in Fig.4(c). In order to achieve less accuracy degradation, at least 3-bits inputs should be used. In this case, Depth 0 still has 14.45% accuracy degradation while other configurations have less than 1.6% accuracy degradation.

Generally, at least 3-bits weights and 3-bits inputs are required for the dense layer of this CNN for MNIST applications, in case no huge accuracy degradation shall occur. Fewer bits can be used if the structure is carefully set up.

E. *Accuracy vs. BNN and TNN*

Another way to further reducing the required resolution for weights is by considering the limited resolution during the training phase. For the BNN and TNN presented by [10] and [11], networks are trained with only binary (1-bit) and ternary (2-bits) weights, respectively. To this end, all weights with real value W_R should be mapped to either a binary W_B or a

Fig. 4. In this experimental exploration, the networks are trained with seven different configurations of dense layers. (a) and (b) shows the accuracy degradations of the networks in comparison to initial accuracy if weights and inputs of their dense layer are converted to lower resolution, respectively. In addition, while weights of dense layers are converted to 3-bits, (c) shows the accuracy degradation if inputs of dense layer are also converted to lower resolution. (a), (b) and (c) result in that dense layers of CNNs require at least 3-bits weights and 3-bits input to keep reasonable accuracy in conventional way. In order to further needed resolution for weights and inputs, the required resolution for weights can be reduced down to 1-bit by using BNN and TNN training technique. However, even if BNN and TNN are applied, (d) and (e) illustrate that at least 3-bits resolution for inputs of dense layers is still needed. (f) compares the accuracy of networks with binary and ternary dense layers to networks with conventional dense layers.

ternary W_T value by using the sign function (4) or modified threshold sign function (5).

$$W_b = sign(W_r) \tag{4}$$

$$W_T = \begin{cases} 1 & W_R \geq Threshold \\ -1 & W_R \leq -Threshold \\ 0 & otherwise \end{cases} \tag{5}$$

However, the weight updating for those networks is not similar to the normal training process, because the derivative of sign function (4) is zero for all non-zero inputs and the sign function is not differentiable at zero. Similarly, the function (5) faces the same problem that its derivative is always zero for all value except the positive and negative threshold value as the input. It means that the direct weight updating by using the derivative of those functions is unfortunately not possible. The Straight-Though Estimator (STE) [16] can solve this problem by updating only the weight with real value W_T and neglecting the sign function, which maps the real weight to binary and ternary weights.

Fig.4(f) shows that the achieved accuracy by using BNN and TNN for the dense layer is from 97.09% to 99.33%, which is very close to the achieved accuracy by using the conventional dense layer. As Fig.4(f) illustrated, the deeper binary and ternary dense layer configurations have worse performance than the shallow structure.

Fig.4(d) and (e) illustrate that the accuracy degradation can become larger if the limited resolution is applied to inputs of the binary and ternary weighted dense layer. At least a 3-bits input is needed for ternary weighted dense layer if the accuracy degradation is expected around 1% to 2%. Furthermore, this degradation becomes more obvious if the binary-weighted dense layer is used. If the accuracy degradation of the binary-weighted dense layer is not supposed to be greater than 2%, at least 4-bits input should be expected at the input because 3-bits input with BNN can still achieve more than 2% accuracy degradation.

F. Short Summary

It is shown that the architecture of dense layers for CNN applications is less critical regarding accuracy. Experiment A illustrates that Depth 0, which has only ten parameters or neurons at the output, can also perform the classification well. However, Depth 0 has the most critical resolution requirements for weights and inputs in comparison to other experiment. Experiments B, C, D indicate that the deeper structure of dense layers of CNNs can generally have fewer bits for inputs and weights. Additionally, only a proper amount of parameters or neurons of hidden layers can obtain minimal accuracy degradation with limited resolution of weights and inputs. This observation leads to the conclusion

that the neural network, which has better generalization ability, will also have a more relaxed requirement for the resolution of weights and inputs.

At least 3-bits weights and 3-bits inputs are required to keep accuracy by using the conventional method. However, the resolution of weights can be reduced to 1 bit or 2 bits with BNN and TNN technique, respectively. By using this technique, 1-/2-bits weights and 3-bits inputs can be used without large accuracy degradation according to Fig.4(d) and (e).

From a conventional digital perspective, if such configuration is implemented with a crossbar circuit and the output signal of crossbars should be recovered without any loss, an ADC with a certain resolution is needed and the required resolution can be written as Eq.(6) [9][17].

$$z = \begin{cases} log_2 d + w + l & w > 1 \; and \; l > 1 \\ log_2 d + w + l - 1 & otherwise \end{cases} \quad (6)$$

In Eq.(6), z denotes the width of the output signal, d is the row of the memristive crossbar, w and l represent the width of weights and inputs, respectively. The 1- and 2-bits resolution can be reduced for the ADC by using BNN and TNN training techniques because the weights resolution can be decreased down to 1- and 2-bits, respectively.

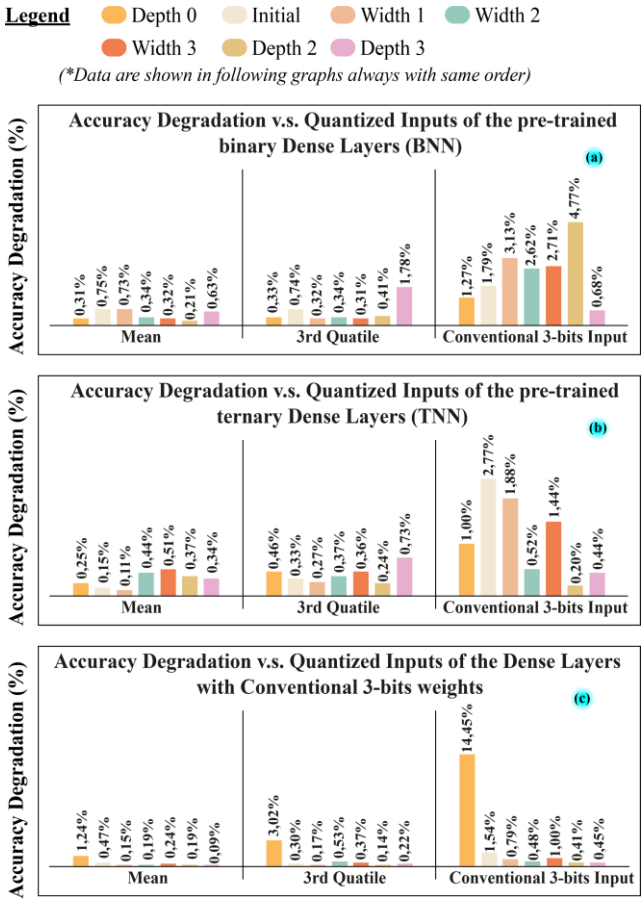

Fig. 6. The proposed quantization is used for inputs of the dense layer with different configurations. (a), (b) and (c) compare the accuracy degradation of the CNNs if those methods are applied to original pre-trained binar (BNN), ternary (TNN) and conventionally 3-bits quantized dense layer of CNNs. It results in that a fully 1-bits (both inputs and weights) dense layer of the CNN can be achieved by using porposed method and only litte accuracy degradation occurs for MNIST Dataset.

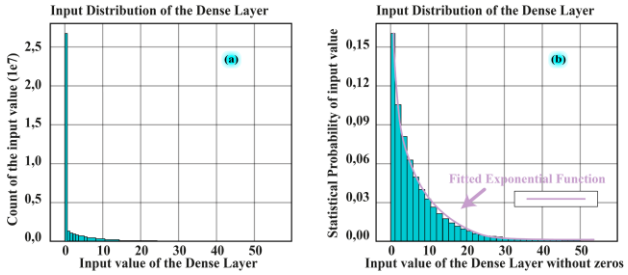

Fig. 5. The inputs of dense layer are collected. (a) shows the original distribution of values, that most values are located at 0. Then, all zeros are removed. (b) input values (without zeros) exhibit an exponentional distribution. The purple line is the resolved exponentional function.

IV. Fully One-bit Dense Layer in CNN

As observed before, the binary dense layer of CNNs may need at least 3-bits or 4-bits input to ensure that the accuracy does not decrease too much. To decrease the required resolution for inputs, this section investigates the value distribution of the input with Configuration Initial in order to find a proper threshold value to binarize inputs.

Fig.5(a) illustrates that almost one third of values locates at zero and therefore, it cannot match any meaningful probability distribution at all. To keep only useful data, all zeros should be removed because the zeros do not needed to be converted. Fig.6(b) shows that the input value for the dense layer without zeros can exhibit an exponential distribution approximately. Typically, the exponential distribution can be written as Eq. (7).

$$f_\beta(x) = \begin{cases} \frac{1}{\beta} e^{-(x-x0)/\beta} & x \geq x_0 \\ 0 & x < x_0 \end{cases} \quad (7)$$

In Eq. (7), β and x0 represent the scale parameter and location parameter of the exponential distribution, respectively. Fig.5(b) shows a fitted exponential function (purple line) with β equal to 6.3375 and x0 equal to 3.28E-04, which matches very well with the distribution of the collected input values. The mean and third quartile are selected as the threshold value to binarize inputs. The mean is the expected value of the exponential distribution and exactly equal to the scale parameter β, and the third quartile splits the amount of the value to 3:1. All input values, which are greater than the threshold value, are interpreted as one and the values, which are less than the threshold value, are converted to zero. The test accuracy remains 98.22% and 98.23% for mean and third quantile as the threshold value, respectively.

In order to verify the general application of this method, the binary, ternary, and conventional 3-bits weighted dense layer of the CNN are tested and compared, as shown in Fig.6. Fig.6(a) – (c) indicate that the binarized input can be used for any configurations of the dense layer, and the binarized input can achieve less accuracy degradation in comparison to the conventional 3-bits input.

Additionally, Fig.6 (a) and (b) show that this method can achieve maximal 1.78% accuracy degradation for the binary and ternary dense layer of CNN in comparison to 4.77% accuracy degradation with the conventional 3-bits quantized input and BNN, respectively. Fig.6 (c) illustrates that this method can also be applied for the conventionally 3-bits weighted dense layer and exhibits only 1.24% maximal

accuracy degradation for Depth 0 in comparison to 14.45% accuracy degradation with the same configuration. It indicates that the model can still keep accuracy if the binary input and binary weight are used in a dense layer, which can strongly simplify the required model for the dense layer of CNN and implementation with crossbar circuits. Furthermore, as shown in Fig.6(a) and (b), this method can gain a big advantage for the binary and ternary dense layer of CNN, because this method can generally achieve much less accuracy degradation in comparison to conventionally 3-bits quantized input.

As illustrated in Fig.6 (a) and (b), the deeper structure, Depth 3, does not have much less accuracy degradation using this method in contrast to others. The binarized input seems to introduce unavoidably noise to the model. The noise could be transferred into the next layer, leading to relatively more accuracy degradation if more layers are used.

To verify if this method could be applied not only to the MNIST dataset, further three models are built for the CIFAR-10 and CIFAR-100 dataset. Two models with six convolutional layers for 128 features and two binary weighted dense layers are trained for CIFAR-10 by using the BNN technique. The difference is that they have 128 neurons and 256 neurons in the hidden layer, respectively. A model with six convolutional layers for 192 features and two with BNN technique 2-bits weighted dense layers, which have 256 neurons and 100 neurons in the hidden layer and output layer respectively, is trained for CIFAR-100. For comparison, an MNIST's model with Configuration Initial and the fully binarized dense layer is used.

Table II shows that the model CIFAR-10(128) has the largest accuracy degradation of 3.86% and CIFAR-10(256) has 3.10% accuracy degradation. Because of limited computational resources, only a simple model is trained for CIFAR-100 and therefore, CIFAR-100 has only 32.74% accuracy in the end. For CIFAR-100, the binarized input and weight lead to only 1.96% accuracy degradation from this comparison. We can conclude that little accuracy degradation can occurs if a fully one-bit dense layer of CNN is used. At least, this should work with 6-layers convolutional neural networks for CIFAR and MNIST datasets. If such accuracy degradation is acceptable for the final implementation, this method in combination with BNN technique can be used to capture a fully 1-bit dense layer, which means dense layer with 1-bit inputs and 1-bit weights. For such dense layer, only 8-bits ADCs and 1-bit DACs are required to recover the data without any loss in comparison to 12-bits ADCs and 3-bits DACs for binary and ternary dense layer, 13-bits ADCs and 3-bits DACs for conventionally quantized dense layer if a 128 rows crossbar is assumed.

TABLE II. INPUT BINARIZATION WITH MEAN OF DISTRIBUTION

Application	MNIST	CIFAR-10 (128)	CIFAR-10 (256)	CIFAR-100
Test Accuracy with binary dense layer	98,87%	72,73%	73,08%	32,74%
Accuracy with 1-bit input	98,22%	68,87%	69,98%	30,78%

V. CONCLUSION

This work aims to provide a guideline for simplifying the implementation of dense layers of CNNs with crossbar circuits regarding selecting the proper configuration of dense layers in CNNs in combination of different quantization methods. The results show that starting with the appropriate configuration of dense layers, a trained model can be adapted to a reduced resolution model with both the weights and inputs of dense layers reduced to 3-bits and maintain reasonable accuracy without the need for retraining. Based on a BNN training technique and input value distribution analysis, the required resolution can further be reduced down to 1-bit. The result suggests that the implementation of dense layer with crossbar circuits can be strongly simplified by considering architecture selection and quantization before final implementation.

REFERENCES

[1] T. Gokmen and Y. Vlasov, Acceleration of Deep Neural Network Training with Resistive Cross-Point Devices: Design Consideration, vol.10, Frontiers in Neuroscience, 2016, pp.330.

[2] M. Hu and J. P. Strachan, Accelerating Discrete Fourier Transforms with dot-product engine, IEEE International Conference on Rebooting Computing, San Diego, CA, 2016, pp. 1-5

[3] C. Li et al, Analogue signal and image processing with large memristor crossbars, Nature Electronics, vol.1, 2019, pp. 52-59

[4] Q. Xia and J. Yang, Memristive Crossbar Arrays for Brain-Inspired Computing, Nature Materials, 18, 2019, pp.309-323

[5] I. Charkraborty, D. Roy and K. Roy, Technology Aware Training in Memristive Neuromorphic System for Nonideal Synaptic Crossbars, IEEE Transactions on Emerging Topics in Computational Intelligence, vol.2, 2018, pp. 335-344

[6] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev and D.B. Strukov, Training and Operation of An Integrated Neuromorphic Network based on Metal-Oxide Memristors, Nature 521, 2016, pp.61-64

[7] M. Hu et al, Dot-Product Engine for Neuromorphic Computing: Programming 1T1M Crossbar to Acclerate Matrix-Vector Multiplication, 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, 2016, pp. 1-6.

[8] P. Gu et al, Technological Exploration of RRAM Crossbar Array for Matrix-Vector Multiplication, the 20th Asia and South Pacific Design Automation Conference, Chiba, 2015, pp. 106-111

[9] A. Shafiee et al, ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars, ACM/IEEE 43rd Annual International Symmposium on Computer Architecture (ISCA), Seoul, 2016, pp. 14-46

[10] M. Courbariaux, I. Hubara, D. Soudry, R. EI-Yaniv and Y. Bengio, Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1, arXiv preprint arXiv:1602.02830, 2016

[11] C. Zhu, S. Han, H. Mao and W. J. Dally, Trained Ternary Quanzization, arXiv preprint arXiv:1612.01064, 2016

[12] M. Rastegari, V. Ordonez, J. Redmon, A.Farhadi, XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, arXiv prerpint arXiv:1603.05279

[13] Konstantin. K. Likharev, CorssNet: Neuromorphic Hybrid CMOS/Nanoelectronic Networks, Science of Advanced Materials, vol. 3, 2011, pp.322-331

[14] J. Zhou, K. Kim, W.Lu, Crossbar RRAM Arrays: Selector Device Requirements During Read Operation, IEEE Transactions on Electron Devices, vol.61, no.5, May 2017, pp. 1369-1376

[15] Y. LeCun, C. Cortes and C. J.C. Burges, The MNIST Database of handwritten digits, http://yann.lecun.com/exdb/mnist/

[16] Y. Bengio, N. Leonard, A. Courville, Estimating or Propagting Gradients Through Stochastic Neurons for Conditional Computation, arXiv preprint arXiv:1308.3432

[17] S. Gupta, A. Agrawal, K.Gopalakrishnan, P. Narayanan, Deep Learning with Limited Numerical Precision, arXiv preprint arXiv:1502.02551