2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC) | 978-1-6654-9005-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/VLSI-SoC54400.2022.9939619

Towards Energy Efficient DNN accelerator via Sparsified Gradual Knowledge Distillation

Foroozan Karimzadeh* and Arijit Raychowdhury*
*School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332
Email: fkarimzadeh6@gatech.edu, arijit.raychowdhury@ece.gatech.edu

***Abstract*—Artificial intelligence (AI) is becoming increasingly popular in many applications. However, the computation cost of deep neural network (DNN) , which is a powerful form of AI, calls for efficient DNN compression technique to make energy efficient networks. In this paper, we proposed SKG, a method to jointly sparsify and quantize DNN models to ultra-low bit-precision using Knowledge Distillation and gradual quantization (SKG). We demonstrated that our method can preserve the accuracy more than 20% for uniform quantization with 2 bit-width compared to the baseline methods on ImageNet and ResNet-18. In addition, our method can achieve up to 2.7x lower energy consumption using compute-in-memory (CIM) architecture compared to a traditional 65nm CMOS architecture for both pruned and unpruned network during inference and eventually enabling using DNN models on resource constrained edge devices.**

***Index Terms*—DNN model, Knowledge distillation, DNN compression, quantization, low bit precision.**

I. Introduction and Background

EDGE intelligence is the new trend toward computing on the edge devices such as drones and self driving cars. Edge computing provides low latency data transfer and increase the privacy as there is no need to transfer the data to the cloud for computations [1]. DNN methods are a powerful form of artificial intelligence (AI) which can perform accurate computation in many complex tasks such as healthcare [2] and computer vision [3]. However, since edge devices are resource constrained in terms of memory and battery, these powerful computing methods cannot be easily deployed on edge devices due to their large network size and intensive computations.

DNN compression techniques such as pruning [4] and quantization [5] offer solutions to enable DNNs to be run on the edge devices. Several works in the literature focus on quantization methods for DNN compression [5]. Low-precision operation reduce the required memory size and computation while achieving analogous accuracy to the floating point computation [6]. The multiply-and-accumulate (MAC) operations in quantized network can be then replaced by simple bit-wise operations cause huge reductions in computations and required energy. Eventually, quantization enhances the energy efficiency and speed up the computation during inference. On the other hand, information loss and significant accuracy drop incur as a result of quantization to lower precision numbers [6], especially in complex data like ImageNet. The goal is to quantize the network while achieving comparable accuracy as the original network with floating point operation.

Recently, several papers have investigated techniques to quantize weights and activation during training to achieve higher accuracy [5]–[7]. Incremental Network Quantization (INQ) [8] introduces a method to efficiently quantize the weights of any pre-trained full-precision convolutional neural network (CNN) models into a low-precision version through three interdependent steps, including weight partition, group-wise quantization and re-training. However, the activation remains in full-precision which makes it hard to deploy it on an on-chip system. DOREFA-NET [9] is another method to quantize both weight and activation of a CNN model using low bit-width parameter gradients by stochastically quantize the parameter gradients to low-precision values during back propagation. Additive Powers-of-Two (APoT) [5] quantize the bell-shaped distribution of weights and activation non-uniformly. In this method, all quantization levels are constrained to be the sum of Powers-of-Two. SYQ [6] is a symmetric quantization method which decreases the loss by learning a symmetric codebook for particular weight subgroups. Quantization to lower precision such as 4-bit or ternary quantization is also harder since the range of the numbers are very limited and a huge information loss might occur [5].

Knowledge distillation (KD) is a method to transfer knowledge from another pre-trained network as a teacher to a target student network, letting the student network to mimic the teacher network performance [10]. KD can be used as the DNN compression technique, for instance, to better prune the network [10]. [11] transformed knowledge from multiple teacher. [12] leveraged KD method and Kullback-Leibler (KL) divergence as the loss function for quantization. To tackle the problem of training DNNs with low-precision, [13] transfer the knowledge from the feature map and the logits output to better train the network. PQK method pruned and quantizad the network at the same time in an interactive manner [14] while in our paper, we prune and quantize the network in the separate steps.

From the hardware perspective, several specialized CMOS based hardware for DNN accelerators has been proposed to enable compressed DNNs running on edge devices. However, new architectures based on traditional CMOS still face the fundamental technological limitations of CMOS. On the other hand, Von Neumann architecture suffers from prohibitive power dissipation incurred by massive data transfer between the PEs and memory. Accessing DRAM memory requires three order of magnitude higher power than a simple multiplication [15]. On the other hand, compute-in-memory (CIM) architecture has emerged to solve the aforementioned issues

by performing the computation in the memory [16], [17].

In this paper, we propose a method to jointly prune and quantize the model to ultra low bit-width, allowing DNNs to run on the power and battery constrained edge devices. The method has three steps including: (1) pruning (2) quantization using KD and (3) gradual quantization to lower bit precision. During inference, we also use both traditional CMOS based accelerator and CIM architecture for comparison. We train the pruned network while gradually reduces the bit-precision using KD, and demonstrated that we can preserve the accuracy better in very low bit precision. We also calculated the energy required during inference using a 2 bit/cell CIM architecture and 65nm CMOS technology. The advantage of using a CIM architecture is to achieve low-latency and low-power computation scheme since the computation is conducted in the memory as opposed to a traditional Von Neumann architecture which suffers from the latency and power dissipation caused by intra-chip data communication.

II. Method

The proposed method illustrated in figure 1 consisting of three steps: pruning, gradual quantization and knowledge distillation scheme to achieve an ultra-low bit-precision DNN model. The goal is to achieve an energy-efficient computation while preserving the accuracy during inference. The training starts with the initialization using a full precision (i.e. float-32) pre-trained model. In this work, we used both pruned and unpruned network for the full precision network. For pruning, we have utilized a threshold based method as explained in [15] to sparsify the full precision network. In the next step, the network is quantized to n-bit fixed-point precision (e.g. 8-bit) during training using knowledge distillation to achieve similar accuracy. Next, the model with quantized weight and activation is saved as the pre-trained model for the lower bit precision training. This process is continued to decrease the bit-width gradually one by one until the result for the desired lowest bit-precision (e.g. 2-bit) is achieved. The goal is to quantize the network gradually to gain higher accuracy and compensate for the loss of information as a result of ultra-low bit quantization. We used uniform and power-of-two (PoT) quantization methods in this paper but other methods of quantization can be applied. During inference, we also utilized traditional CMOS and CIM architectures to evaluate and compare the energy efficiency of the system.

A. Preliminaries

Matrix-vector multiplication between weight (W) and its input values (X) defines as a basic building block in DNN models. The result is then pass through an element-wise nonlinear activation function which is usually a Rectified Linear Unit (*ReLU*). The formula is shown in Eq. 1 where T is a transpose function, b is the bias.

$$a = ReLU(W^T X + b) \tag{1}$$

To quantize the model, suppose $W \in R^{C_{out} \times C_{in} \times k \times k}$ is a $4D$ tensor representing a convolutional layer's kernels, where C_{out} and C_{in} and k are the number of output channel, input channels and the kernel size, respectively. We can define the weight quantization formula [5] as follow:

$$W_q = \alpha \Pi_{Q(1,b)} \lfloor \frac{W}{\alpha}, 1 \rceil, \tag{2}$$

where b is the bit-precision, α is the scaling factor and the scaling function $\lfloor ., 1 \rceil$ scales the weights to the range of $[-1, 1]$. Then, the scaled W is projected by $\Pi_{(.)}$ in an element-wise manner to the defined quantization levels, $Q(1, b)$.

For uniform quantization, $Q_U(1, b)$ defines as Eq. 3. Moreover, PoT quantization (Eq. 4) quantized the networks by constraining quantization levels to be powers-of-two values or zero.

$$Q_U(1, b) = \{0, \frac{\pm 1}{2^{b-1} - 1}, \frac{\pm 2}{2^{b-1} - 1}, ..., \pm 1\}, \tag{3}$$

$$Q_{POT}(1, b) = \{0, \pm 2^{-2^{b-1}+1}, \pm 2^{-2^{b-1}+2}, ..., \pm 1\}, \tag{4}$$

Quantization maps each element of the full precision weight matrix to a b bit-width fixed-point representation. The result of convolution against quantization level is then re-scaled by multiplying to the scaling factor (α). α is generally a floating-point number [5]. However, in this paper, we round it to the nearest int8 power of two number to have fully fixed-point computation during inference.

During backward path, the modified version of Straight-Through Estimator (STE) [18] is adopted for the projection operation [5]. The gradients of α are calculated as bellow:

$$\frac{\partial W_q}{\partial \alpha} = \begin{cases} sign(W) & if\ |W| > \alpha \\ \Pi_{Q(1,b)} \frac{W}{\alpha} - \frac{W}{\alpha} & if\ |W| \leq \alpha \end{cases} \tag{5}$$

After quantization, the arithmetical calculation of DNN models can be performed in an on-chip system with low-precision fixed-point operations, which is much more efficient in terms of the required memory and energy than their floating-point equivalent [19].

B. Knowledge Distillation and Gradual Quantization

In this paper, we explore the idea of KD and gradual quantization for a pruned full-precision network. Knowledge distillation are mainly divided into three types [20]: relation-based knowledge, feature-based knowledge and response-based knowledge. In this work, we apply the response-base knowledge where the network is learned to use quantized weights and activation using KD during training while using the logits layer values of a higher-precision network as the ground-truth (Figure 1). First, we use both pruned and unpruned full-precision (float-32) model as a pre-trained model. The pruning is performed using a threshold based method as explained in [15] where the weights less than a defined threshold are removed during the training. Since the value of pruned weights are zero, they remain zero during quantization in the forward path. We also applied a mask to make sure

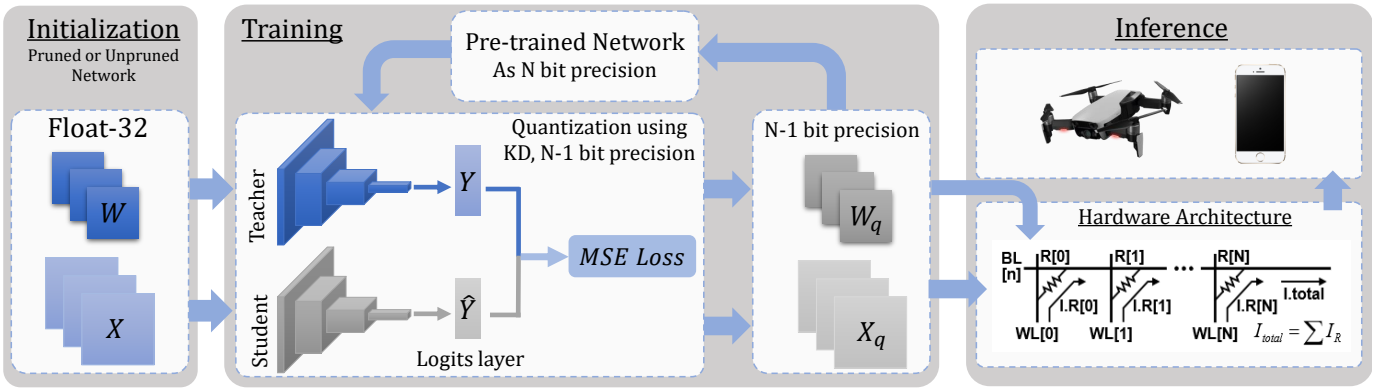


Fig. 1: An overview of the proposed method consisting of pruning, knowledge distillation (KD) and gradual quantization steps for DNN compression during training. The quantized weight and activation will then be used during inference to increase the energy efficiency of the system while preserving the accuracy.

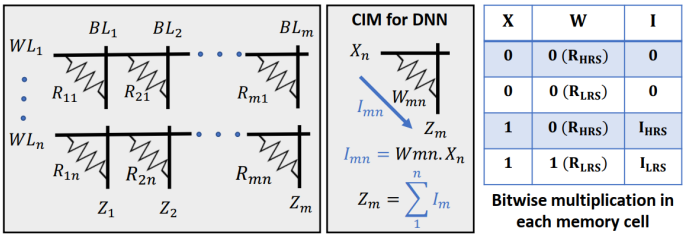
the percentage of pruning remain the same until the end of gradual quantization. We then use the last activation of the full-precision network as the ground-truth and teacher to quantize the lower precision like 4-bit as the student network. For KD method, minimum square error (MSE) is used as a loss function, L, as shown in Eq. 6, where Y and $\hat{Y}$ are the last logits layers of the original full-precision network and the quantized network, respectively.

$$L(Y, \hat{Y}) = (Y - \hat{Y})^2 \tag{6}$$

Afterward, the final model with 4-bit precision is saved and used as the pre-trained model for training the network with 3-bit precision. We continue this gradual reduction in bit-precision until we reach the target bit-precision, for example 2-bit precision. It should be mentioned that we quantize all the layers of the network except the last layer and both the weight and activation to the desired bit-width. Last layer is quantized to 8-bit precision as it is used for classification. The reason of using gradual quantization is that if we quantize the network from full-precision to the target low bit-precision, the accuracy drops drastically because of losing huge information. However, using gradual quantization, we can better preserve the accuracy for lower bit-precision like 4 or 2 bit-precision.

C. Hardware

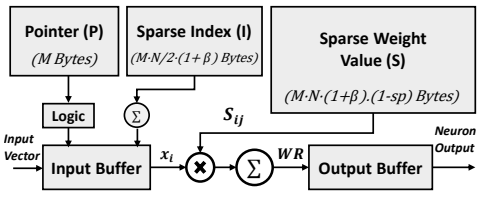
In this paper, we used two different hardware architectures, CIM and 65nm conventional CMOS technology, during inference to evaluate the energy efficiency. CIM architecture utilizing resistance of memory cells is illustrated in figure 2a where a memory cell itself serves as a PE and memory simultaneously. The memory cell holding the weights as the resistance generates the current which is the result of bit-wise multiplication between the inputs and weights. Then, currents are summing up as a result of a current-summing BL structure in a memory array. Therefore, the intermediate



X	W	I
0	0 (R_{HRS})	0
0	0 (R_{LRS})	0
1	0 (R_{HRS})	I_{HRS}
1	1 (R_{LRS})	I_{LRS}

Bitwise multiplication in each memory cell

(a) CIM architectures exploiting resistance of memory cells.



Technology Node TSMC	65nm
Frequency	1GHZ
Voltage	1 V
Temperature	25 • C
Memory Bank	256B, 512B 1KB, 4KB

Hardware Parameters

(b) The hardware architecture for sparse DNN computation using a conventional 65nm CMOS and the its characteristics.

Fig. 2: The high-level hardware architectures of CIM and conventional CMOS for inference.

data are accumulated immediately. As a result, an energy-efficient architecture is achieved especially for deep learning computations. In particular, the CIM architectures utilizing emerging memory such as RRAM have achieved importance in performing energy-efficient computing thanks to its inherent multiply-and-accumulate (MAC) functionality in BL structures, non-volatility and compatibility to CMOS process. In this paper, we utilized the voltage-sensing multi-bit RCIM architecture proposed in [17], [21]. The detailed description of the architecture is explained in [21]. In this architecture, a 2-bit encoding is employed where with the 2-bit-encoded RRAM cells (11, 10, 01, and 00) and the ADC-based readout circuits, the measured energy per bit during CIM is 0.83, 0.47, 0.28, and 0.15 pJ/bit, respectively. We have used these actual measurements to estimate the required energy of the networks.

The proposed method also synthesized with 65nm CMOS technology to measure hardware metrics. Implementation parameters are shown in 2b. In this architecture, the sparse weight matrix is compressed in three vectors and saved in the memory: (1) the non-zero values of the weights (S), (2) location of the non-zero weights (I) and (3) a pointer vector to point to the start of each column in the weight matrix (P). In addition to the mentioned three vectors, each entry bit-width of S and I is designed to be equal to the bit-precision, and additional memory usage ratio resulted from limited index representation is denoted by β [22], [23].

III. Result

In this section, we validate our proposed method, SKG, on ResNet-18 network and ImageNet-ILSVRC2012 dataset with 1000 classes [24]. ImageNet dataset consists of 1.2M training and 50K validation images. Before starting the training step, the images are randomly cropped and resized to 224×224. Apart from a single crop and normalization, we have not done any other pre-processing on images. The original full-precision ResNet model is implemented using PyTorch official implementation and initialized from the released pre-trained model. The results have compared with various baseline methods including uniform quantization, PoT quantization, additive powers-of-two (APoT) [5] and XNOR-Net [25]. Training is carried out on Nvidia GTX 1080 Ti GPUs. In this paper, we quantize both weight and activation with the same method to the target bit-precision for all the layers except the last layer of the network to gain fully quantized DNNs. Like other baseline methods, the last layer is quantized to 8 bit-width since it is used for classification. Moreover, stochastic gradient descent (SGD) with the momentum of 0.9 is employed for parameter optimization. MSE (Eq. 6) is also applied as the loss function for KD method between Y, the last logits layer of the original full-precision network, and $\hat{Y}$ the last logits layer of the quantized network.

In the proposed method, we start with training a ResNet model on Imagenet using float-32 format as a pre-trained and teacher network. Instead of using the Imgenet labels, we utilized the last logits layer of the original full-precision network as the teacher and ground-truth. In this work, both pruned and unpruned networks are utilized as the teacher network. Since the edge devices are resource constrained in terms of battery and energy, the pruned network as the teacher can help to further compress the DNN models. As illustrated in figure 2, multiplying to the sparse wight matrix (i.e. zero values) decrease the energy consumption [22]. Specifically, in CIM architecture, the weight values with more zeros in their binary representations are desirable since multiplying to the bit with zero value requires 6x less energy. Therefore, sparsifying the network will help to decrease the energy consumption during inference. Eventually, the goal is to quantize the network to ultra-low bit-precision as a student network using a KD technique and gradually decrease the bit-precision.

First, we compare the accuracy of the proposed method with different baseline methods including APoT, XNOR-Net. First we evaluated our method using an original ResNet model with no pruning. We implement KD and gradual quantization method on two commonly used quantization method, (1) uniform quantiozation and (2) PoT, and quantize them gradually to the lower bit precision by using the model from the previous bit-width as the pre-trained model and its last logits layer as the ground truth. The two state-of-the-art baseline methods (APoT and XNOR-Net) are quantize using FL32 model as the pre-trained network. XNOR-Net is the method for 2-bit quantization. Therefore, XNOR-Net result is just presented for 2 bit-width. The results for 4, 3, 2 bit-width, illustrated in figure 3 show that our proposed method helps the network to preserve the accuracy for ultra low bit-widths.

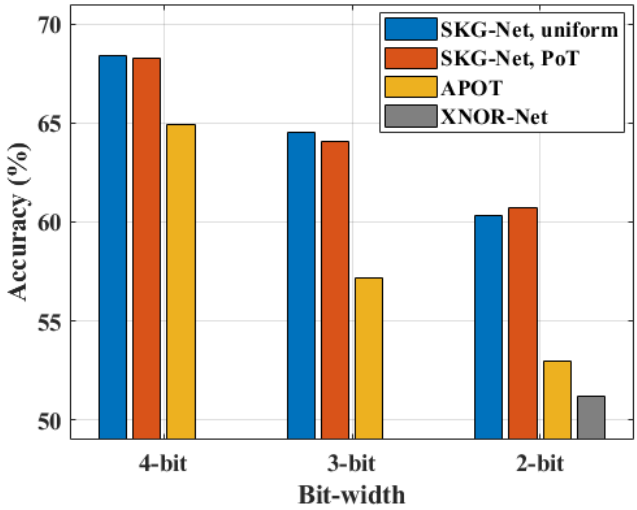


Fig. 3: Accuracy vs bit-width (4-2 bits) for uniform and PoT method with KD and gradual quantization and baseline methods including APoT and XNOR-Net with FL32 as the pre-trained model on ResNet-18 and ImageNet dataset.

To show the effectiveness of gradual quantization, for 8-bit to 2-bit precision is implemented using ResNet-18 on Imagenet dataset. We then compare the accuracy for different bit precision with the accuracy of a network trained and quantized to different bit precision using fl-32 network with no pruning as the pre-trained model. Figure 4 illustrates the accuracy vs bit-width from 8-bit to 2-bit for the proposed method and the baseline using uniform quantization. No gradual quantization is applied for the baseline method. As demonstrated, gradual quantization can preserve the accuracy better for lower bit precision. The accuracy for 5, 4, 3, 2 bit-widths are 69.14%, 68.41%, 64.52%, 60.34%, respectively. While the accuracy for the mentioned bit-widths for the baseline method are 67.81%, 60.11%, 58.31%, 40.51%, respectively, which are lower than the accuracy of our proposed method.

In addition, we explored the idea of pruning and quantization to increase the energy efficiency by removing unimportant connections and quantize the rest to the ultra-low bit precision. To do so, we start with a prunned original float-32 ResNet-18 model and used that as the ground-truth for the KD method to train a network with 4-bit precision. Next, the last logits layer of the trained 4-bit network is utilized as the label to train the

TABLE I: The accuracy for our method and a baseline method for different sparsity percentage.

Method	Our method			baseline		
Sparsity \ Bit-width	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit
40%	68.2	64.7	59.1	64.5	56.2	51.2
70%	68.5	63.1	57.9	63.4	53.2	50.6
90%	67.2	61.2	60.9	54.9	51.2	48.7

TABLE II: The energy (J) of our proposed method for different sparsity percentage using CIM and CMOS architectures.

Method	CIM			65nm CMOS		
Sparsity \ Bit-width	4-bit	3-bit	2-bit	4-bit	3-bit	2-bit
40%	204.5	150.2	55.6	551.3	412.8	338.4
70%	156.2	98.5	42.8	477.9	364.1	251.7
90%	102.6	65.1	29.4	328.2	218.9	158.1

network with 3-bit precision and so on. Table I shows that our method gains higher accuracies in different sparsity rates and bit-precision. For example, in 70% sparsity rate, SKG achieves 68.5%, 63.1% and 57.9% accuracy in 4-bit, 3-bit and 2-bit precision, respectively.

Moreover, we compared the required energy to run our method for different sparsity rates and bit-precision based on CIM and 65nm CMOS architecture illustrated in figure 2. We have used the actual measurement to estimate the total energy required in CMOS and CIM architectures. The results are demonstrated in table II. The estimated energy for CIM includes the RRAM array, the ADC, the controller, and other peripheral circuits except for the voltage reference (VREF) generator, and for CMOS architecture includes memory, multiplier, accumulator and input/output buffer. In the CIM architecture, a 2-bit encoding is employed where with the 2-bit-encoded RRAM cells (11, 10, 01, and 00) and the ADC-based readout circuits, the measured energy per bit during CIM is 0.83, 0.47, 0.28, and 0.15 pJ/bit, respectively. We have used these actual measurements to estimate the required energy of the networks with different sparsity rates. The result shows that the energy consumption estimated using CIM architecture are lower than traditional CMOS architecture.

Since CIM architecture achieved lower required energy, we compare the energy consumption of our method and baseline method using only the 2bit/cell RRAM based CIM architecture during inference (figure 5). The results show that SKG with PoT method and gradual quantization can achieve the best accuracy while consuming the least energy among other methods for ultra-low bit precision. The energy saving of gradual PoT quantization compare to the APoT for 4, 3 and 2 bit-width are 75.98%, 86.96% and 21.97%, respectively.

IV. Conclusion

Using powerful DNN algorithm on the resource constrained edge devices requires us to develop compression techniques to make efficient DNNs in terms of energy consumption and size. In this paper, we proposed SKG to jointly prune and quantize ResNet models for highly compressed networks that can be deployed on edge devices. we demonstrated that using KD technique along with the gradual quantization help the network to better preserve the accuracy in ultra-low bit-precision and the accuracy can be increased up to 20% compared to the baseline methods. In addition, SKG can achieve higher compression rates by using the sparse float-32 network as the ground-truth. We also compared the energy consumption during inference using CIM and traditional 65nm CMOS technologies. The results show that CIM architecture can achieve 2x less energy. Moreover, using KD and gradual PoT quantization and CIM architecture, we can reduce the energy consumption more than other methods during inference. Using

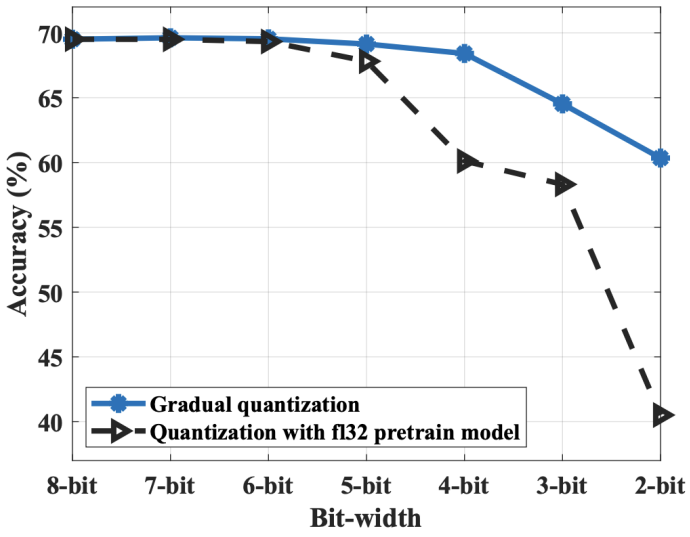


Fig. 4: Accuracy vs bit-width (8-2 bits) for uniform quantization with and without gradual quantization on ResNet-18 and ImageNet dataset.

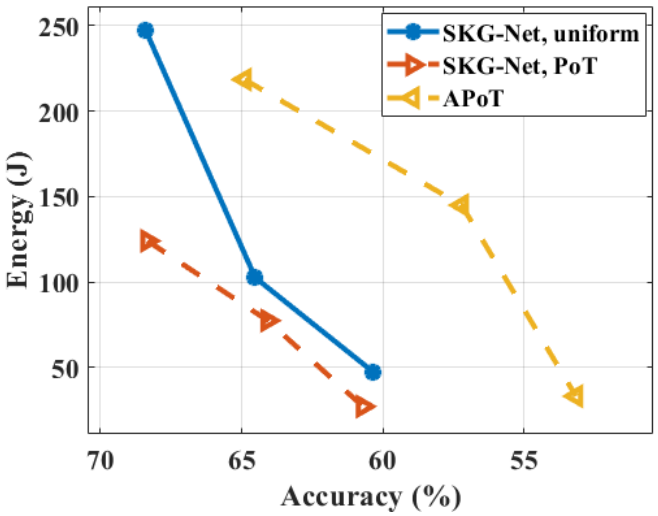


Fig. 5: The total estimated energy achieved by the proposed method for Uniform and PoT compared with the baseline method, APoT, with FL32 pre-trained model for ResNet-18 on ImageNet. From left to right: 4-bit, 3-bit, 2-bit precision.

CIM architecture for 4 to 2-bit-width quantization, gradual PoT quantization saves 75.98%, 86.96% and 21.97% energy compare to the APoT baseline method and achieve 68.3%, 64.1% and 60.7% accuracy, respectively which is higher than the baseline methods with full-precision model as the pre-trained network.

ACKNOWLEDGMENT

This project was supported by the Semiconductor Research Corporation under grant JUMP CBRIC task ID 2777.004, 2777.005 and 2777.006.

REFERENCES

[1] F. Karimzadeh and A. Raychowdhury, "Memory and energy efficient method toward sparse neural network using lfsr indexing," in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*. IEEE, 2020, pp. 206–207.

[2] R. Boostani, F. Karimzadeh, and M. Nami, "A comparative review on sleep stage classification methods in patients and healthy individuals," *Computer methods and programs in biomedicine*, vol. 140, pp. 77–91, 2017.

[3] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Computational intelligence and neuroscience*, vol. 2018, 2018.

[4] F. Karimzadeh, N. Cao, B. Crafton, J. Romberg, and A. Raychowdhury, "Hardware-aware pruning of dnns using lfsr-generated pseudo-random indices," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[5] Y. Li, X. Dong, and W. Wang, "Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks," *arXiv preprint arXiv:1909.13144*, 2019.

[6] J. Faraone, N. Fraser, M. Blott, and P. H. Leong, "Syq: Learning symmetric quantization for efficient deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4300–4309.

[7] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[8] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv preprint arXiv:1702.03044*, 2017.

[9] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[10] L. Chen, Y. Chen, J. Xi, and X. Le, "Knowledge from the original network: restore a better pruned network with knowledge distillation," *Complex & Intelligent Systems*, pp. 1–10, 2021.

[11] S. You, C. Xu, C. Xu, and D. Tao, "Learning from multiple teacher networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1285–1294.

[12] S. Shin, Y. Boo, and W. Sung, "Knowledge distillation for optimization of quantized deep neural networks," in *2020 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2020, pp. 1–6.

[13] B. Zhuang, M. Tan, J. Liu, L. Liu, I. Reid, and C. Shen, "Effective training of convolutional neural networks with low-bitwidth weights and activations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[14] J. Kim, S. Chang, and N. Kwak, "Pqk: Model compression via pruning, quantization, and knowledge distillation," *arXiv preprint arXiv:2106.14681*, 2021.

[15] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[16] J. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "29.1 a 40nm 64kb 56.67 tops/w read-disturb-tolerant compute-in-memory/digital rram macro with active-feedback-based read and in-situ write verification," in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64. IEEE, 2021, pp. 404–406.

[17] F. Karimzadeh, J.-H. Yoon, and A. Raychowdhury, "Bits-net: Bit-sparse deep neural network for energy-efficient rram-based compute-in-memory," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2022.

[18] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[19] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM computing surveys (CSUR)*, vol. 23, no. 1, pp. 5–48, 1991.

[20] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[21] J.-H. Yoon, M. Chang, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, and A. Raychowdhury, "A 40nm 100kb 118.44 tops/w ternary-weight computein-memory rram macro with voltage-sensing read and write verification for reliable multi-bit rram operation," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2021, pp. 1–2.

[22] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2016, pp. 243–254.

[23] F. Karimzadeh, N. Cao, B. Crafton, J. Romberg, and A. Raychowdhury, "A hardware-friendly approach towards sparse neural networks based on lfsr-generated pseudo-random sequences," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2020.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

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