ENERGY-AWARE DNN QUANTIZATION FOR PROCESSING-IN-MEMORY ARCHITECTURE

A Dissertation
Presented to
The Academic Faculty

by

Beomseok Kang

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2022

ENERGY-AWARE DNN QUANTIZATION FOR PROCESSING-IN-MEMORY ARCHITECTURE

Approved by:

Dr. Saibal Mukhopadhyay, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Shimeng Yu
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Tushar Krishna
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: April 29, 2022

ACKNOWLEDGEMENTS

First, I am deeply grateful to my advisor Dr. Saibal Mukhopadhyay for his continuous guidance and advice. His passion to research keeps me motivated and I was able to enjoy the first step to research with his assistance. I would also like to thank my thesis committee members Dr. Shimeng Yu and Dr. Tushar Krishna for their invaluable advice during the research. I would also like to thank especially Anni Lu, Yun Long, Daehyun Kim, and other GREEN Lab members for their contribution to the research.

At last, I would like to extend my sincere thanks to my family, dog, and friends for their continuous supports.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **iii**

LIST OF TABLES **v**

LIST OF FIGURES **vi**

SUMMARY **vii**

CHAPTER 1. Introduction **1**

CHAPTER 2. Background **4**
2.1 Processing-In-Memory (PIM) Architecture **4**
2.2 Energy Consumption in PIM **5**

CHAPTER 3. Related Works **7**
3.1 Prior Quantization Algorithms for PIM **7**
3.1.1 Energy-unaware Quantization Algorithms 7
3.1.2 Energy-aware Quantization Algorithms 8
3.1.3 Chanllenges in Prior Algorithms 9

CHAPTER 4. Proposed Approach **10**
4.1 Genetic Algorithm based Quantization **10**
4.1.1 Overall Quantization Flow 10
4.1.2 Linear Quantization 11
4.2 Energy-aware Fitness Function **12**

CHAPTER 5. Experimental Results **16**
5.1 ResNet Quantization **16**
5.2 VGGNet Quantization **23**
5.3 Comparison with Prior Works **26**
5.4 Other Low-Complexity Network Quantization **30**

CHAPTER 6. Conclusion **32**

APPENDIX A. Properties of Proposed Approach
A.1 Run-time **33**
A.2 Weighting Factors **33**

APPENDIX B. NeuroSim

REFERENCES **37**

LIST OF TABLES

Table 1 – Energy-aware and Energy-unaware Quantization Results of ResNet-18. 17

Table 2 – Layer-wise Bitwidth of ResNet-18 Compressed by Energy-aware and Energy-unaware Quantization. 19

Table 3 – Energy Efficiency Comparison between VGG-19 with and without Fully-Connected Layers. 24

Table 4 – Layer-wise Bitwidth of VGG-19 Compressed by Energy-aware and Energy-unaware Quantization. 24

Table 5 – Comparison against Various PIM-aware Quantization Algorithms. 27

Table 6 – Comparison against Various Quantization Algorithms for ResNet-50. 28

Table 7 – Comparison against Various Quantization Algorithms for MobileNet-V2 and SqueezeNet. 30

Table 8 – Energy Efficiency of VGGNet, ResNet with Balanced Weighting Factors. 34

LIST OF FIGURES

Figure 1	– (a) SRAM-based subarray in PIM. (b) dynamic energy of circuit components in PIM.	4
Figure 2	– Data flow and weight mapping of CNN in PIM.	5
Figure 3	– (a) Overall flow of EGQ. (b) and (c) Upper and lower bound in ResNet-18. (d) Description of the first iteration in EGQ.	10
Figure 4	– Dynamic energy comparison between EGQ and NeuroSim.	16
Figure 5	– (a) and (c) Effectiveness of gamma in EGQ for ResNet-18 and ResNet-50. (b) and (d) Dynamic energy comparison of ResNet-18 and ResNet-50 with different gamaa. (c) and (g) Effectiveness of alpha and beta to the number of ADC access in ResNet-18 and ResNet-50. (d) and (h) Weight and activation bitwidth of ResNet-18 and ResNet-50 from (c) and (g).	16
Figure 6	– Distribution of weight and activation bitwidth in ResNet-18.	19
Figure 7	– Distribution of the number of ADC access in ResNet-18.	20
Figure 8	– (a) Dynamic energy and energy efficiency of ResNet-50 with different memory technologies. (b) Dynamic energy and energy efficiency of ResNet-50 with single-bit and multi-bit eNVMs.	21
Figure 9	– (a) Effectiveness of gamma for VGG-19. (b) Dynamic energy comparison of VGG-19 with different gamma. (c) Effectiveness of alpha and beta to the number of ADC access in VGG-19. (d) Weight and activation bitwidth of VGG-19 from (c).	23
Figure 10	– (a) Dynamic energy and energy efficiency of VGG-19 with different memory technologies. (b) Dynamic energy and energy efficiency of VGG-19 with multi-bit eNVMs.	25
Figure 11	– Changes of fitness value during EGQ iterations.	33

SUMMARY

With increasing computational cost of deep neural network (DNN), many efforts to develop energy-efficient intelligent system have been proposed from dedicated hardware platforms to model compression algorithms. Recently, hardware-aware quantization algorithms have shown further improvement in the energy efficiency of DNN by considering hardware architectures and algorithms together. In this work, a genetic algorithm-based energy-aware DNN quantization framework for Processing-In-Memory (PIM) architectures, named EGQ, is presented. The key contribution of the research is to design a fitness function that can reduce the number of analog-to-digital converter (ADC) access, which is one of the main energy overhead in PIM. EGQ automatically optimizes layer-wise weight and activation bitwidth with negligible accuracy loss while considering the dynamic energy in PIM. The research demonstrates the effectiveness of EGQ on several DNN models VGG-19, ResNet-18, ResNet-50, MobileNet-V2, and SqueezeNet. Also, the area, dynamic energy, and energy efficiency in the compressed models with various memory technologies are analyzed. EGQ shows 15%-103% higher energy efficiency with 2% accuracy loss than other PIM-aware quantization algorithms.

CHAPTER 1. INTRODUCTION

Processing-In-Memory (PIM) is an attractive hardware architecture for energy-efficient deep neural network (DNN) inference since processing units are integrated into memory [1]-[3]. Along with the efforts in the dedicated hardware for DNN, algorithmic approaches such as weight and activation quantization are studied to reduce the model complexity of DNN [4]-[8]. Recently, hardware-aware quantization algorithms have shown that the energy efficiency of DNN inference can be further improved by considering hardware architectures and algorithms together [9], [10]. It is important to consider data movement such as memory access in conventional CMOS-based architectures to effectively improve the energy efficiency [11]. However, PIM architectures avoid the large number of memory access, and rather analog-to-digital converters (ADC) access is the main energy overhead. This motivates energy-aware quantization for PIM to differentiate from the quantization for CMOS-based architectures.

Uniform bitwidth quantization is widely used and simple to adapt for various DNN models. However, significant accuracy loss is observed at very low bitwidth [12]. Sun et al. [13] shows uniform bitwidth quantization with negligible accuracy loss, but it needs to re-train models after the quantization. Flexible bitwidth quantization is a promising technique to improve compression ratio and/or reduce accuracy loss compared to uniform quantization [6], [8]. It is important to note that the bitwidth of DNN can be differently determined depending on quantization methods while showing the same accuracy level. In other words, optimizing the bitwidth only with the constraint of accuracy does not guarantee the optimal energy efficiency. However, most of the flexible quantization

algorithms for PIM do not explicitly consider energy consumption [14], [15]. There are few works on energy-aware quantization algorithms for PIM with simplified energy cost [13], [16]. As the energy consumption of hardware components is non-uniform in different layers [17], different bitwidth should be assigned with the consideration of the energy distribution to effectively improve the energy efficiency. It is still required to consider hardware components such ADC in flexible bitwidth quantization algorithms for PIM.

In this research, a genetic algorithm-based energy-aware DNN quantization framework for PIM architectures (EGQ) is presented. EGQ optimizes weight and activation bitwidth for each layer and supports flexible bitwidth quantization by predicting layer-wise dynamic energy consumption in PIM platforms. The predicted dynamic energy is directly utilized to evaluate and search bitwidth candidates that achieve high energy efficiency with low accuracy loss. As the large amount of dynamic energy is consumed by ADC in PIM [1], we mainly focus on the number of ADC access to predict the layer-wise dynamic energy consumption. EGQ only requires the basic model information such as activation sizes (i.e. dimensions) and kernel sizes in convolution neural network (CNN) to predict dynamic energy. Also, EGQ does not need re-training, which means any pre-trained DNN models can be simply compressed by EGQ.

EGQ uses genetic algorithm to automatically search appropriate layer-wise weight and activation bitwidth. The concept of using genetic algorithm for flexible bitwidth quantization was recently proposed [7]. However, it does not optimize activation bitwidth, and more importantly, it is a hardware-agnostic algorithm as ADC cost related to the PIM performance is not considered. The main contribution of this research is to develop an automatic framework for searching optimal layer-wise weight and activation bitwidth

considering the number of ADC access and the associated dynamic energy. The research mainly focus on designing a fitness function to estimate the number of ADC access, and genetic algorithm is used as an optimization method to prove that overall energy efficiency in PIM can be improved by the presented fitness function. It is important to note that EGQ is not limited to genetic algorithm as the fitness function can be applied to other problem solvers such as reinforcement learning [16].

EGQ is mainly evaluated with three CNN models, VGG-19 [18], ResNet-18, and ResNet-50 [19]. The effect of EGQ on the area, dynamic energy, and energy efficiency in PIM architectures are analyzed by NeuroSim [17]. As PIM platforms can be combined with various memory devices such as SRAM and embedded non-volatile memories (eNVMs), how the effectiveness of EGQ changes for PIM designs with different memory technologies is also studied. Finally, EGQ is compared with previous works on PIM-aware quantiztion and genetic algorithm-based quantization [7], [12], [13].

The research presents EGQ is an effective approach to reduce dynamic energy in various PIM designs with SRAM, RRAM, and FeFET technologies. In particular, embedded non-volatile memories (NVMs) show the higher improvement of energy efficiency with EGQ due to low leakage energy. EGQ improves the energy efficiency of ResNet-18 by 6.5 times compared to the 16-bit model. Also, EGQ shows 15% - 103% higher energy efficiency than the existing quantization algorithms for PIM with 2% accuracy loss.

CHAPTER 2. BACKGROUND

2.1 Processing-In-Memory (PIM) Architecture

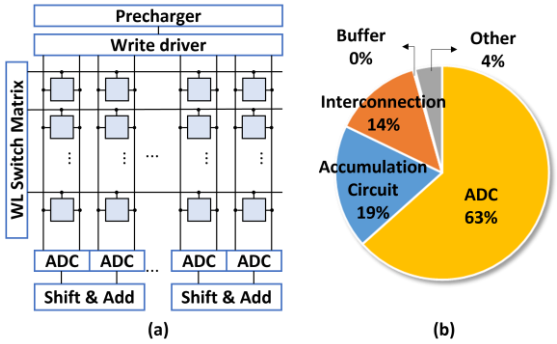

Figure 1 – (a) SRAM-based subarray in PIM. (b) Dynamic energy of circuit components in PIM. Blue squares are SRAMs. 16-bit ResNet-18 is used for dynamic energy simulation.

PIM-based DNN accelerators with various memory devices have attracted many interests due to high energy efficiency [1], [2], [24]. Most of the PIM architectures accelerate vector and matrix multiplication (VMM) operation based on parallel analog computations by bit-line current summation [1], [2], [24]. Figure 1(a) shows the basic schematic of a SRAM subarray in PIM architectures [25]. We also assume the weight within the subarray is stored as multi-bit words, where each cell represents a single bit. We assume the input to a word-line is a serial bit-wise binary value which does not require DAC [26]. Once the input vector is applied to the word-lines, the current flowing through the bit-line is determined by weight values stored in the SRAM. The current is converted to a digital value by ADC without row-by-row access, and then the converted value is used as an input feature map (IFM) of the next layer. The parallel operation significantly increases computational efficiency in PIM architectures.

2.2 Energy Consumption in PIM

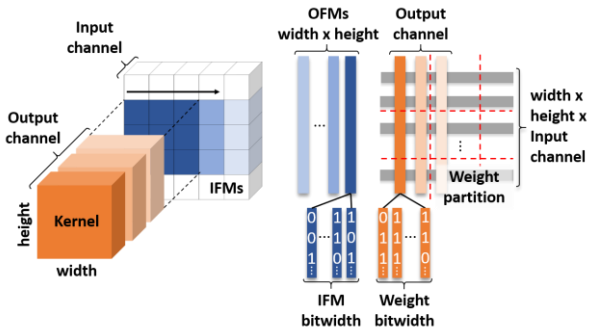

Figure 2 – Data flow and weight mapping of CNN in PIM. The left figure shows the schematic of convolution operation with blue IFMs and orange kernels. The right figure shows the weight mapping on subarrays and the serial bit-wise input of IFMs to subarrays.

Most energy consumption in PIM comes from read peripheral circuits, mainly, ADC. Figure 1(b) shows the dynamic energy consumption of circuit components in a PIM platform estimated by modified NeuroSim [17]. We use ResNet-18 with ImageNet dataset for the inference simulation. 128×128 subarray size, 5-bit ADC, and 7nm SRAM technology are used for the simulation. More details about modified NeuroSim are described in the next section. We observe that ADC consumes about 63% of total dynamic energy. As ADC dynamic energy is proportional to the number of ADC access, the lower number of ADC access can effectively reduce the total dynamic energy. EGQ assumes the hardware design parameters such as the size of subarrays and ADC precision are fixed and find optimal bitwidth for each layer to reduce the total number of ADC access.

The number of ADC access in PIM architectures is determined by two factors, the length of input vectors and the number of subarrays to store weight matrices. All weight parameters are assumed to be stored on a single chip, and each layer uses different subarrays. Figure 2 shows the schematic of IFMs and kernels for convolution operation.

First, the number of subarrays for a weight matrix (i.e. kernel) is related to weight bitwidth and a weight dimension. Each kernel is unrolled along the columns in subarrays. The number of required rows are same with the multiplication of the width, height, and input channels of a kernel. (see gray rows in Figure 2.) The number of required columns for each kernel is same with weight bitwidth. Thus, the total memory size is determined by each kernel size, the number of kernels, and weight bitwidth. As we assume the size of subarrays is not flexible, the weight matrix should be partitioned by the subarray size. Red-dot lines in Figure 2 indicate subarray boundaries. Next, the length of input vectors is related to the bitwidth of IFMs. IFMs with high precision need to be converted to long serial binary vectors that proportionally increase the number of ADC access. For convolution layers, the length of input vectors also relies on an output feature map (OFM) size. Kernels capture partial information from IFMs, and each capture constitutes one OFM. Thus, the number of captures is determined by the OFMs size, not the IFMs size. Blue bars in Figure 2 show that the length of input vectors is proportional to IFM bitwidth and OFMs size.

CHAPTER 3. RELATED WORKS

3.1 Prior Quantization Algorithms for PIM

In this section, the details of related works on PIM-aware quantization are introduced [13], [15], [16], [20], [21], and they are compared with EGQ to state the differences between EGQ and these works.

3.1.1 Energy-unaware Quantization Algorithms

The paper by Z. Zhu et al. [15] showed flexible weight and activation quantization for a RRAM-based PIM architecture with the consideration of storage and latency. This RRAM-aware quantization quantifies the storage based on the number of crossbars and the latency by the number of digital-to-analog converters (DAC) access as optimization objectives. Each quantity is related to weight and activation bitwidth, respectively. The storage and latency can indirectly reduce energy consumption, but energy is not reflected into a loss function in the research. Also, re-training is required after every iteration of the weight and activation quantization.

C. Zhang et al. [21] proposed a robust RRAM-based quantization framework by developing the quantization algorithm to minimize RRAM variation error. They calculate the numerical error between full-precision weight and weight with the noises introduced by quantization error and device variation. The error is used as a loss function to utilize gradient-descent and back-propagation algorithms. As the loss function is only related to weight parameters, activation bitwidth is not considered. Also, energy is not related to the loss function. Thus, the algorithm is only suitable for weight quantization.

Reinforcement learning is a widely used algorithm in AutoML. S. Qu et al. [21] used a reinforcement learning based deep deterministic policy agent to search flexible weight bitwidth to maximize memory utilization. The memory utilization is estimated based on the weight mapping in PIM subarrays. Bitwidth candidates that have the high memory utilization and accuracy give the agent high rewards, so the agent automatically searches an optimal bitwidth candidate. However, the memory utilization does not consider activation bitwidth. As the large amount of energy consumption is resulted from activtion [22], flexible weight and activation bitwidth are required for energy-aware quantization.

3.1.2 Energy-aware Quantization Algorithms

H. Sun et al. [13] proposed uniform bitwidth activation quantization for an energy-efficient PIM accelerator. They show a non-linear quantization scheme to reduce required ADC bitwidth. As ADC results in critical energy overhead in analog PIM, decreasing ADC bitwidth is an effective method to enhance the energy efficiency [23]. However, changing ADC bitwidth depending on CNN models requires variable resolution ADC macros that are difficult to design. Also, energy-aware regularization in a loss function was introduced in the paper. The energy consumption is estimated with the simple multiplication of the voltage and current at RRAM devices in PIM subarrays. However, the improvement due to the energy-aware regularization is marginal since the dynamic energy consumed by eNVMs is not critical.

S. Huang et al. [16] proposed reinforcement learning based flexible weight and activation quantization for PIM. A cost function is defined with weight, activation, and ADC bitwidth to incorporate hardware performance into the quantization algorithm. Two

fractions are mainly considered in the cost function. One is the fraction of the number of parameters over total parameters, and the other one is the fraction of bitwidth over full precision. ADC bitwidth is multiplied to the fractions as an exponential term to reflect the large energy overhead of high-precision ADC. However, as the fractions only represent the compression ratio of weight and activation parameters, the cost function cannot accurately estimate the layer-wise energy consumption in PIM architectures.

3.1.3 Challenges in Prior Algorithms

Prior quantization methods for PIM show the storage-aware, latency-aware, or utilization-aware algorithms. Although there are some energy-aware algorithms, they require complex multi-precision ADCs or used overly simplified energy cost without considering the details of data layout/flow for weight mapping and input data transfer. An energy-aware, flexible, and automated quantization algorithm for PIM designs is still missing. In contrast, EGQ is an energy-aware algorithm for PIM and uses a fitness function that includes weight and activation compression ratio, the number of ADC access, and accuracy loss. The fitness function estimates energy using the number of ADC access which needs to consider the weight mapping and input data transfer in PIM. Accordingly, EGQ is a more suitable quantization framework for energy-efficient inference in PIM.

CHAPTER 4. PROPOSED APPROACH

4.1 Genetic Algorithm based Quantization

Genetic algorithm is a well-known algorithm for complex search and optimization. As flexible bitwidth quantization is also a search problem, EGQ uses genetic algorithm to determine appropriate bitwidth for each layer.

4.1.1 Overall Quantization Flow

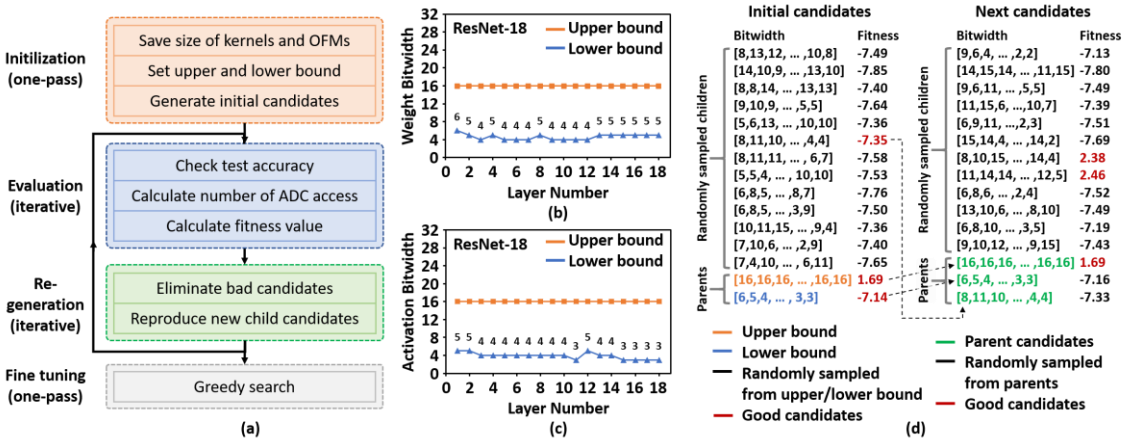

Figure 3 – (a) Overall flow of EGQ. (b) and (c) Upper and lower bound in ResNet-18. (d) Description of the first iteration in EGQ. Upper bound is heuristically determined. Left half of (d) is the result from the initialization step, and the right area means next candidates after the first evaluation and re-generation step.

Figure 3 shows the overall flow and details of each stage in EGQ. The initialization step prepares weight and activation bitwidth candidates. The size of kernels and OFMs are stored at the initialization step as EGQ uses them repeatedly at the evaluation step. Figure 3(b) and Figure 3(c) are the lower and upper bound of ResNet-18 which constrains the range of initial candidates. The candidates are randomly sampled based on the two boundaries. Detail process to determine the upper and lower bound is explained in the

previous study [7]. The candidates are evaluated based on fitness values at the evaluation step. Figure 3(d) shows 15 candidates and the fitness values of each candidate. The red fitness values mean three good candidates, and they become parent candidates at the re-generation step. Other candidates are eliminated. The re-generation step includes crossover and mutation. Two parents from the three are randomly selected, and children candidates are again randomly sampled in the range of the two parents. EGQ repeatedly evaluates and reproduces candidates searching appropriate weight and activation bitwidth candidates. Final bitwidth can be further compressed by the greedy search based fine tuning [7].

4.1.2 Linear Quantization

Linear quantization is a widely used scheme for DNN quantization. EGQ also uses linear quantizer for both weight and activation. The quantization of weight with n-bit precision is defined by

$$Q(x) = \text{round}\left(\frac{x(2^{n-1}-1)}{\max(|x|)}\right) \times \frac{\max(|x|)}{2^{n-1}-1} \quad \textbf{(1)}$$

Weight range is first normalized and expanded to 2^n range. As weight generally includes negative values, the range is from -2^{n-1} to $-2^{n-1}-1$. Round function makes the expanded float values to be integers, thus introduces quantization error. The other multiplication term is for dequantization from the integer range to the float range again. For activation quantization, range can be different depending on the location of a quantizing layer. If activation quantization is directly connected to the OFMs, activation quantization uses same formula with weight quantization. However, if activation

quantization is after ReLU function, there are no negative values. Thus, the range becomes from 0 to $2^n - 1$. EGQ quantizes activation after ReLU function.

4.2 Energy-aware Fitness Function

In this subsection, we present the fitness function of EGQ and compare it with the fitness function in the prior genetic algorithm-based quantization method, named Q-PIM [7]. The fitness function of EGQ is given by

$$F(C) = \alpha \cdot C_W + \beta \cdot C_A + \gamma \cdot C_{ADC} + \delta \cdot \text{Accuracy} \quad \textbf{(2)}$$

where C represents a bitwidth candidate; C_W, C_A, and C_{ADC} are the compression ratio of weight, activation (i.e. IFMs), and the number of ADC access, respectively. α, β, γ, and δ are weighting factors. EGQ optimizes the fitness function to find a bitwidth candidate that has high compression ratio and energy efficiency with low accuracy loss. The fitness function of Q-PIM is given by [7]

$$F(C) = -\alpha \cdot \sum_{i=1}^{N} W_i \cdot P_{W,i} - \beta \cdot \text{Error} \quad \textbf{(3)}$$

where W_i is the weight bitwidth of ith layer; $P_{W,i}$ is the number of weight parameters; Error represents accuracy loss. There are two main differences between EGQ and Q-PIM. First, the fitness function of Q-PIM does not consider the activation compression ratio. As equation (3) has only two terms related to the weight compression ratio and accuracy loss, it cannot be used for flexible activation quantization. Next, equation (3) does not have the connection between the quantization algorithm and hardware platforms, which is the main

contribution of EGQ. The fitness function of EGQ considers the number of ADC access in PIM platforms. Thus, EGQ is more suitable for energy efficient quantization for PIM.

C_{ADC} represents the compression ratio of the number of ADC access compared to the 32-bit model. It is assumed that the dynamic energy consumption is proportional to the number of ADC access as discussed above, so C_{ADC} reflects the expected total dynamic energy to the fitness function. C_{ADC} is defined by

$$C_{ADC} = 1 - \frac{\sum_{i=1}^{N} ADC_i(W_i,\ A_i)}{\sum_{i=1}^{N} ADC_i(32,\ 32)} \quad \textbf{(4)}$$

$$ADC_i = S_i \times OFM_{w,i} \times OFM_{h,i} \times A_i \quad \textbf{(5)}$$

$$S_i = \left\lceil \frac{K_{IC,i} \times K_{w,i} \times K_{h,i}}{s} \right\rceil \times \left\lceil \frac{K_{OC,i} \times W_i}{s} \right\rceil \quad \textbf{(6)}$$

where ADC_i is the number of ADC access; W_i and A_i are weight and activation bitwidth; S_i is the number of subarrays. $OFM_{w,i}$ and $OFM_{h,i}$ are the width and height of a OFM. $K_{IC,i}$, $K_{w,i}$, $K_{h,i}$, $K_{OC,i}$ are the input channel, width, height, and output channel of a kernel, respectively. s is a subarray size such as 128. As EGQ reduces weight and activation bitwidth, the ratio in equation (4) also becomes smaller. The ratio is subtracted from 1 to make good candidates have high fitness values. $OFM_{w,i}$ and $OFM_{h,i}$ are varying depending on layers. Fully connected layers do not have width and height dimensions, so we regard both $OFM_{w,i}$ and $OFM_{h,i}$ as 1. S_i is the multiplication of the row and column number of subarrays. The first operand in S_i is the row number of subarrays, and the second operand is the column number of subarrays. Each operand is in a ceil function in equation (6). As

ADC_i changes depending on candidates and iterations, the size of OFMs and kernels should be saved in EGQ. For the accuracy in the fitness function, we set an accuracy threshold that can give the penalty value of -10, if a candidate shows lower accuracy than the threshold. All quantization experiments here use 2% accuracy threshold. The weighting factors would change the behavior of EGQ, but they are set to 1 as default. We compare the impact of the weighting factors in the next section.

C_W and C_A are determined by the ratio of the number of weight and activation parameters. Each of them are given by

$$C_W = 1 - \frac{\sum_{i=1}^{N} W_i \times P_{W,i}}{\sum_{i=1}^{N} 32 \times P_{W,i}} \quad \textbf{(7)}$$

$$C_A = 1 - \frac{\sum_{i=1}^{N} A_i \times P_{A,i}}{\sum_{i=1}^{N} 32 \times P_{A,i}} \quad \textbf{(8)}$$

where $P_{W,i}$ and $P_{A,i}$ mean the number of weight and activation parameters. The total number of the binary bits for weight parameters is obtained by the product of $P_{W,i}$ and weight bitwidth, W_i. We assume weights in batch normalization layers are floating point, so weights in convolution layers and fully connected layers are mainly compressed. As W_i at the denominator is 32, the ratio in equation (7) represents how much the weight of i^{th} layer is compressed compared to the 32-bit model. C_A can be explained in the same way as C_W. EGQ calculates the compression ratio of each candidate every iteration. Hence, the number of layer-wise weight and activation parameters should be stored in the initialization step.

EGQ is chosen to be layer-wise granular to limit the search space and hence, improve convergence time. EGQ can be modified to support channel-wise optimization, but search space significantly increases. For example, ResNet-50 has 50 layers and 22,720 channels. The granularity of EGQ can be switched if equation (5), (8), and (8) are modified. N in the each equation represents the number of layers. EGQ estimates the compression ratio of weight, activation, and ADC access with total N number of precision assigned to each layer. If N is modified to represent the number of channel, EGQ can be adapted to be channel-wise granular.

CHAPTER 5. EXPERIMENTAL RESULTS

5.1 ResNet Quantization

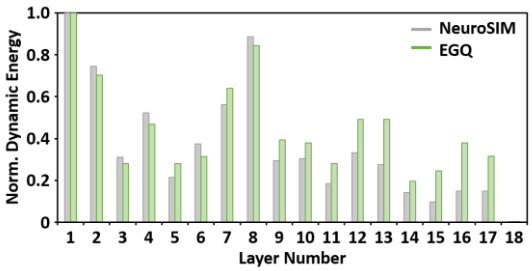

Figure 4 – Dynamic energy comparison between EGQ and NeuroSim. The prediction from EGQ is based on the number of ADC access. Both results are obtained from ResNet-18 with same precision.

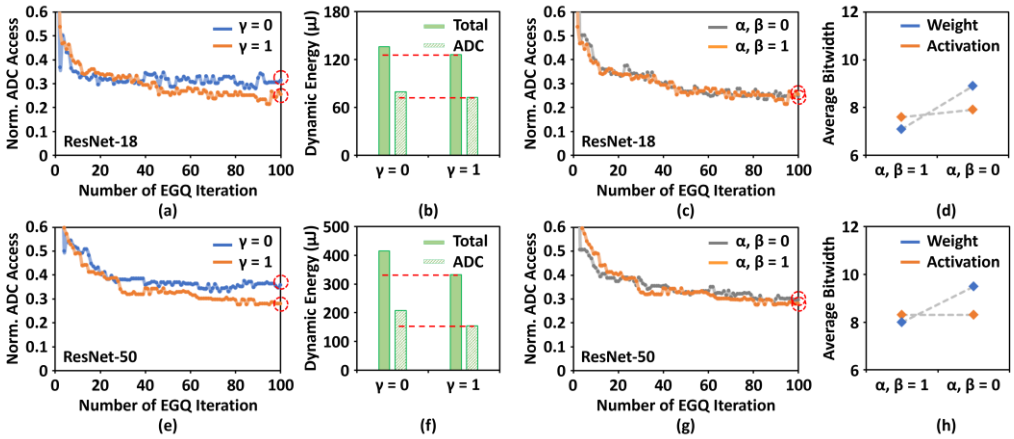

Figure 5 – (a) and (c) Effectiveness of γ in EGQ for ResNet-18 and ResNet-50. y-axis means the normalized number of ADC access. (b) and (d) Dynamic energy comparison of ResNet-18 and ResNet-50 with different γ. (c) and (g) Effectiveness of α, β to the number of ADC access. (d) and (h) Weight and activation bitwidth of ResNet-18 and ResNet-50 from (c) and (g). Weighting factors not mentioned in each figure are set to 1.

Figure 4 shows the layer-wise normalized dynamic energy of quantized ResNet-18 with flexible bitwidth. The prediction from EGQ based on the normalized number of ADC access closely matches the results from NeuroSim.

We compare ResNet-18 and ResNet-50 with energy-unaware (i.e. $\gamma = 0$) and energy-aware (i.e. $\gamma = 1$) cases in Figure 5. Also, compression-unaware (i.e. $\alpha, \beta = 0$) and compression-aware (i.e. $\alpha, \beta = 1$) cases are compared in the same figure. As C_{ADC} is the ratio of the number of ADC access, the weighting factor of C_{ADC} (γ) in the fitness function controls the energy awareness of EGQ. Similarly, α and β control the compression awareness for weight and activation, respectively. Other weighting factors in both experiments are set to 1. Normalized ADC access in Figure 5(a), (c), (e), and (g) means how much the number of ADC access is compressed compared to the 16-bit model.

Table 1 – Energy-aware and Energy-unaware Quantization Results of ResNet-18

Model	**ResNet-18**		**ResNet-50**	
Parameter	$\gamma = 0$	$\gamma = 1$	$\gamma = 0$	$\gamma = 1$
Normalized Number of ADC Access	0.30	0.26	0.36	0.28
Weight Bitwidth	8.1	7.1	8.1	8.0
Activation Bitwidth	7.6	7.6	8.0	8.3
Total Dynamic Energy (µJ)	136.2	125.9	414.5	333.2
ADC Dynamic Energy (µJ)	79.3	72.3	207.4	154.4
Total Leakage Energy (µJ)	54.9	40.0	724.1	673.5
Energy Efficiency (TOPS/W)	27.9	32.2	9.0	10.2
Area (mm^2)	18.7	17.5	49.1	47.7
Baseline Accuracy (%)	69.8	69.8	76.2	76.2
Compressed Accuracy (%)	67.6	68.0	74.3	74.2

In both ResNet-18 and ResNet-50, the energy-aware cases effectively reduce the number of ADC access than the energy-unaware cases showing the effectiveness of including C_{ADC} in the fitness function of EGQ. Figure 5(b) and (f) show the dynamic energy of the circled candidates in Figure 5(a) and (e). Both figures show the improved total dynamic energy in the energy-aware cases. As the difference in the normalized number of ADC access at the 100^{th} iteration is larger in ResNet-50, we observe that the total dynamic energy of ResNet-50 is more reduced than ResNet-18. Table 1 summarizes the total

dynamic energy, total leakage energy, and energy efficiency of the ResNet models. As we assume all weight parameters in each model are mapped to on-chip memory in PIM platforms, large leakage energy is observed in both models during the layer-wise dynamic energy simulation. In particular, Table 1 shows the total leakage energy is even higher than the total dynamic energy in ResNet-50. The energy efficiency is improved by 15.4% for ResNet-18, and 13.3% for ResNet-50, but the two cases show almost same accuracy (Table 1).

We observe that compression awareness does not play a significant role in the number of ADC access in Figure 5(c) and (g). As both cases include $\gamma = 1$, the normalized ADC access is reduce to the similar level of the energy-aware case. Figure 5(d) and (h) compare the average weight and activation bitwidth at the circled candidates in Figure 5(c) and (g). The compression-aware cases show lower weight and activation bitwidth. In particular, the weight bitwidth of the compression-aware cases is 1.5 bit lower in ResNet-18 and 1.8 bit lower in ResNet-50. Activation bitwidth is 0.3 bit difference in ResNet-18, and same in ResNet-50. We expect similar dynamic energy consumption in both compression-unaware and compression-aware cases, but the larger weight bitwidth will increase required memory footprint and leakage energy. Thus, α and β should be considered together with γ in the fitness function to achieve compact and energy efficient models.

Table 2 – Layer-wise Bitwidth of ResNet-18 Compressed by Energy-aware and Energy-unaware Quantization

ResNet-18	$\gamma = 0$	Weight	12,12,10,7,12,11,15,13,14,12,8,7,11,10,9,5,6,12
		Activation	8,9,7,9,11,5,3,7,8,6,5,5,10,10,6,5,9,10
	$\gamma = 1$	Weight	10,9,6,10,11,10,7,10,8,12,10,7,7,7,7,6,5,13
		Activation	8,9,6,6,3,8,13,12,7,9,4,10,10,8,5,9,9,8
ResNet-50	$\gamma = 0$	Weight	13,7,11,8,6,10,15,4,15,10,13,7,6,14,7,9,8,11,6,11,10,14,11,9,12,8,6,11,13,10,12,12,14,6,11,6,7,12,14,10,5,7,9,10,5,7,8,8,7,7
		Activation	11,11,11,7,6,10,8,7,11,11,7,4,6,8,13,7,5,11,15,5,13,8,6,6,11,11,12,8,14,11,7,8,10,9,5,12,12,8,11,9,10,8,13,12,10,13,5,13,10,8
	$\gamma = 1$	Weight	8,14,9,9,8,7,4,12,8,10,5,12,9,13,6,12,11,12,11,10,13,6,14,6,7,5,8,10,10,7,14,8,10,10,10,11,10,13,5,14,9,6,6,12,6,12,13,4,5,6
		Activation	9,9,11,10,7,8,4,14,11,15,8,5,8,12,8,8,4,7,13,7,8,11,6,8,11,5,7,6,11,6,8,4,9,4,6,10,6,8,4,5,5,12,10,4,11,5,8,8,5,9

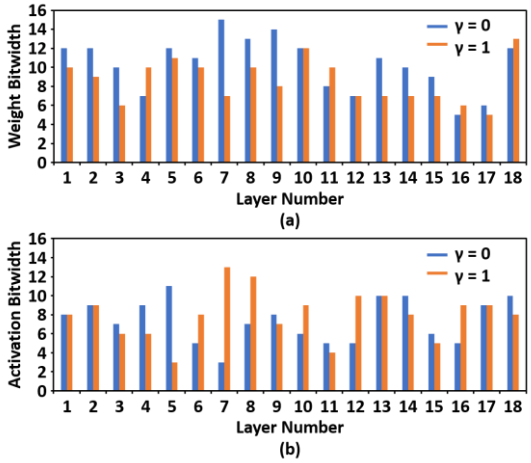

Figure 6 – Distribution of weight and activation bitwidth in ResNet-18. Energy-aware case and energy-unaware case are compared.

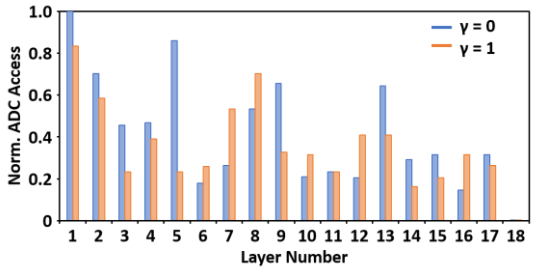

Figure 7 – Distribution of the normalized number of ADC access in ResNet-18. Energy-aware case and energy-unaware case are compared.

Table 2 summarizes the layer-wise weight and activation bitwidth for the energy awareness experiment. Figure 6 visualizes the layer-wise weight and activation bitwidth in the energy-aware and energy-unaware cases of ResNet-18. Figure 7 shows the normalized number of ADC access for both cases. The 5th layer shows huge difference in the normalized ADC access. This is because the 5th layer has the much higher activation precision in the energy-unaware case while the weight bitwidth is similar. The 7th layer has also higher activation precision in the energy-unaware case, but the smaller weight precision mitigates the effect of the activation precision to the normalized ADC access. We observe that most of the normalized ADC access is smaller in the energy-aware case due to the lower weight bitwidth. Thus, the improved energy efficiency in the energy-aware case is reasonable.

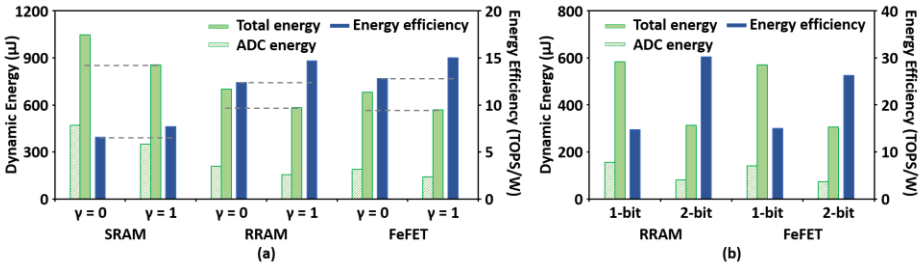

Figure 8 – (a) Dynamic energy and energy efficiency of ResNet-50 with different memory technologies. Single-bit SRAM, RRAM, and FeFET are applied. Energy-unaware and energy-aware cases are compared with the memory devices. (b) Dynamic energy and energy efficiency of ResNet-50 with single-bit and multi-bit eNVMs. Energy-aware case is used for the comparison in (b).

Figure 8 shows the performance comparison between the energy-unaware and energy-aware cases for ResNet-50 with various memory devices. We use single-bit SRAM, RRAM [27], and FeFET [28] in 22nm technology node in Figure 8(a). We do not consider non-ideality and stochastic variations of the memory devices for NeuroSim simulation and accuracy analysis. The SRAM case shows higher total dynamic energy than the RRAM and FeFET cases. Also, ADC dynamic energy is smaller in the RRAM and FeFET cases. We observe that the amount of the total dynamic energy differences are similar with the one of ADC dynamic energy. This indicates that RRAM and FeFET energy efficiently access ADC than SRAM decreasing the total dynamic energy. As we use same the ResNet-50 model with same weight mapping and data transfer, the performance difference is mainly determined by memory device characteristics. The read dynamic energy of ADC is related to the resistance of the reading column in subarrays. Thus, the column resistance of RRAM and FeFET subarrays will be much smaller than SRAM subarrays.

The proportion of the ADC dynamic energy over the total dynamic energy in the RRAM and FeFET cases are small, but EGQ assumes ADC consumes most of the dynamic energy. However, the RRAM and FeFET cases also show better energy efficiency in the

energy-aware cases. This is because the number of ADC access is related to the number of VMM operations in PIM. As read peripheral operations including ADC access are always followed by VMM operations, the energy-aware cases indirectly reduce the number of VMM operations and access to the other read peripheral circuits. The reduced total dynamic energy helps increasing the energy efficiency in the energy-aware cases, but total leakage energy also plays a significant role in the energy efficiency for large models such as ResNet-50. eNVMs can significantly reduce the total leakage energy. For these reasons, Figure 8(a) shows significant increases in the energy efficiency with RRAM and FeFET compared to the amount of the dynamic energy improvement.

Figure 8(b) shows the dynamic energy and energy efficiency of the energy-aware cases for ResNet-50 with multi-bit memory devices. Total dynamic energy and ADC dynamic energy is significantly reduced in 2-bit RRAM and FeFET. This will be due to the lower number of required subarrays. As 2-bit memory devices use the half number of columns in subarrays, the number of required subarrays can be almost half of one for 1-bit memory devices. The number of ADC access is proportional to the number of subarrays. Hence, 2-bit memory devices achieve approximately two-times higher energy efficiency compared to 1-bit memory devices. Multi-bit eNVMs benefit not only the footprint of PIM platforms but also significantly the energy efficiency gain of EGQ. The key observations in the ResNet experiments are summarized:

- The number of ADC access is an effective approach to predict dynamic energy consumption in PIM.
- Including C_{ADC} in fitness function decreases the number of ADC access improving overall energy efficiency.

- Including C_W and C_A in fitness function reduces weight and activation bitwidth.
- EGQ is effective in SRAM, RRAM, and FeFET-based PIM

5.2 VGGNet Quantization

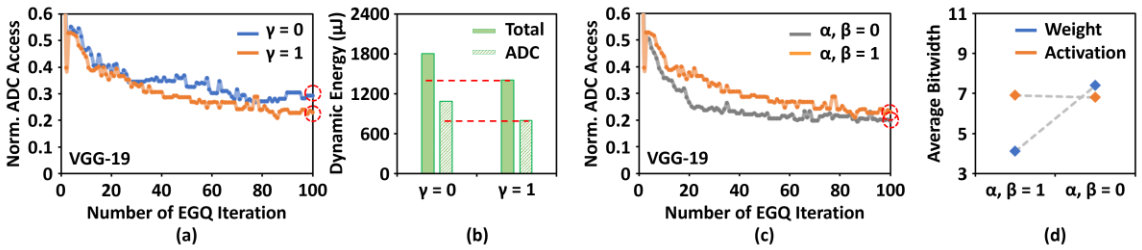

Figure 9 – (a) Effectiveness of γ in EGQ for VGG-19. (b) Dynamic energy comparison of VGG-19 with different γ. (c) Effectiveness of α, β to the number of ADC access in VGG-19. (d) Weight and activation bitwidth of VGG-19 from (c). The weighting factors not mentioned in each figure are set to 1.

Figure 9(a) and (b) show the comparison between the energy-unaware and energy-aware cases of VGG-19. EGQ shows the consideration of γ is also effective in VGG-19. Figure 9(c) shows the compression-unaware case effectively reduces the number of ADC access at the early stage of iterations. However, the saturated level of the number of ADC access is similar with the compression-aware case at the 100th iteration. The average bitwidth of weight and activation is similar to the ResNet cases. Weight bitwidth is much smaller in the compression-aware case, but the difference in activation bitwidth is marginal.

Table 3 – Energy Efficiency Comparison between VGG-19 with and without Fully-Connected Layers

Model	**VGG-19**		**VGG-19 without FC**	
Parameter	$\gamma = 0$	$\gamma = 1$	$\gamma = 0$	$\gamma = 1$
Normalized Number of ADC Access	0.28	0.22	-	-
Weight Bitwidth	4.4	4.1	-	-
Activation Bitwidth	7.0	6.9	-	-
Total Dynamic Energy (μJ)	1801.1	1403.4	1796.6	1258.1
ADC Dynamic Energy (μJ)	1088.0	795.0	1086.0	792.0
Total Leakage Energy (μJ)	8160.8	9510.0	1662.3	766.4
Energy Efficiency (TOPS/W)	3.9	3.6	11.3	19.3
Area (mm^2)	101.4	94.9	46.8	39.4
Baseline Accuracy (%)	72.4	72.4	-	-
Compressed Accuracy (%)	71.2	71.1	-	-

Table 4 – Layer-wise Bitwidth of VGG-19 Compressed by Energy-aware and Energy-unaware Quantization

VGG-19	$\gamma = 0$	Weight	12,8,10,9,5,8,14,6,15,9,13,9,13,13,13,8,3,4,8
		Activation	12,5,13,8,7,6,4,6,12,9,6,7,9,12,14,7,9,9,11
	$\gamma = 1$	Weight	14,4,7,7,11,7,7,13,11,6,4,9,7,13,11,11,3,4,7
		Activation	13,7,8,7,5,6,5,7,15,6,5,6,12,5,12,5,14,6,14

Table 3 summarizes the energy comparison between the two cases including total leakage energy. Layer-wise weight and activation bitwidth are summarized in Table 4. We observe that VGG-19 consumes tremendous leakage energy. The total leakage energy of VGG-19 is much larger as fully connected layers have way more weight parameters than the ResNet models. Specifically, the first fully connected layer in VGG-19 has 25088×4096 parameters, but one in ResNet-18 has 512×1000. Table 3 shows the energy efficiency of the energy-aware case is slightly smaller than the energy-unaware case. As EGQ does not consider leakage energy, unexpected and too high leakage energy introduces the error to the energy efficiency.

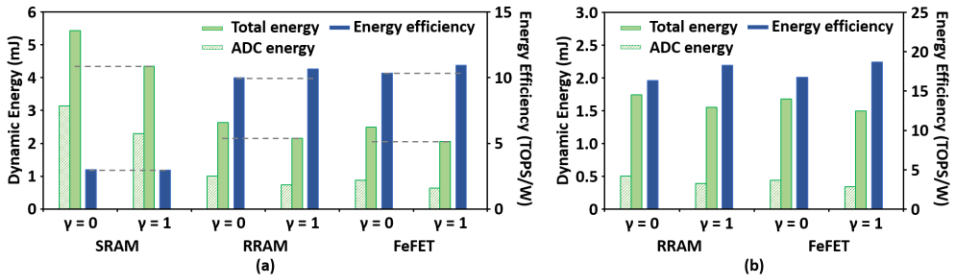

Figure 10 – (a) Dynamic energy and energy efficiency of VGG-19 with different memory technologies. Single-bit SRAM, RRAM, and FeFET are applied. (b) Dynamic energy and energy efficiency of VGG-19 with multi-bit eNVMs. Energy-unaware and energy-aware cases are compared with the memory devices.

We also compare the quantized VGG-19 and quantized VGG-19 without fully connected layers in Table 3 to clarify the impact of fully-connected layer to EGQ. VGG-19 without FC in Table 3 uses same models with the energy-aware and energy-unaware case, but the fully connected layers are eliminated. Table 3 shows that the energy efficiency is improved by 70.8% in the energy-aware case without the fully connected layers. The energy efficiency tendency is flipped mainly because the total leakage energy is largely reduced. There is no meaningful change in the total dynamic energy as the length of input vectors is 1 in fully connected layers indicating the number of ADC access is negligible compared to convolution layers.

Figure 10(a) shows the comparison between the energy-unaware and energy-aware cases with various memory devices. We use the same 22nm technology node and hardware setting in Figure 10. SRAM cases show energy-aware case has no longer lower energy efficiency than energy-unaware case. The energy efficiency in the SRAM cases is both 3.0 TOPS/W. Compared to the 7nm technology node in Table 3, we observe slightly smaller energy efficiency due to the larger dynamic energy of 22nm technology node. However, the SRAM cases with the 22nm technology node shows lower total leakage energy than

the 7nm technology node. The RRAM and FeFET cases also decrease the total leakage energy significantly showing better the energy efficiency in energy-aware cases. By using eNVMs, we expect to reduce EGQ error to energy efficiency in VGG-like models.

Figure 10(b) shows the comparison between the energy-unaware and energy-aware cases with multi-bit RRAM and FeFET. As multi-bit eNVMs can effectively reduce the number of subarrays, we observe much lower dynamic energy and leakage energy. Energy efficiency is improved by 12.3% in RRAM and 11.3% in FeFET. Large models including VGG-19 require the large number of subarrays to store them in on-chip memory, that increases leakage energy more than dynamic energy. Multi-bit eNVMs can be promising solution to effectively reduce the footprint of subarrays and the total leakage energy. As recent CNN models do not have large fully connected layers, we expect EGQ will not experience such error from the leakage energy in modern CNN models. The key observations in the VGGNet experiments are summarized:

- Considering ADC access and weight/activation bitwidth in the fitness function, reduces dynamic energy in PIM for VGG-19, similar to the ResNet models.
- Large fully connected layers in VGG-19 like network and appreciable leakage energy of SRAM reduce overall energy-efficiency of EGQ.
- eNVMs mitigates the error and makes EGQ effective by decreasing leakage energy.

5.3 Comparison with Prior Works

Table 5 compares EGQ with recent quantization algorithms for PIM [12]-[14]. A flexible bitwidth quantization algorithm is used in the paper by Vasquez et al. [14], and the others apply uniform bitwidth quantization algorithms [12], [13]. All algorithms are based

on ResNet-18, but the dataset for test accuracy are different. Energy efficiency in the table means the improvement of the energy efficiency compared to ResNet-18 with 16-bit precision. EGQ achieves 6.5× higher energy efficiency compared to the 16-bit model with ImageNet dataset. Vasquez et al. shows small accuracy loss, but the improvement of energy efficiency is also small. The uniform quantization algorithms [12], [13] show very low bit quantization with CIFAR-10 dataset. Sun et al. [13] achieves 4 bit for both weight and activation. Cai et al. [12] shows more compressed model with 2-bit weight and 4-bit activation. However, huge accuracy loss of 7% is observed [12]. EGQ compresses weight and activation to 2.5 bit and 3.9 bit without huge loss of accuracy. Also, EGQ achieves lower energy consumption and memory size compared with the algorithm in [13].

Table 5 – Comparison against Various PIM-aware Quantization Algorithms

Algorithm	**EGQ**	**EGQ**	**[14]**	**[13]**	**[12]**
Model	**ResNet-18**	**ResNet-18**	ResNet-18	ResNet-18	ResNet-18
Dataset	**ImageNet**	**CIFAR-10**	CIFAR-100	CIFAR-10	CIFAR-10
Weight Bitwidth	**6.1**	**2.5**	flexible	4	2
Activation Bitwidth	**6.3**	**3.9**	Flexible	4	4
Energy Efficiency	**6.5×**	**18.6×**	3.2×	17.7×	29.8×
Baseline Accuracy (%)	**69.8**	**88.6**	70.9	88.9	89.1
Compressed Accuracy (%)	**67.5**	**86.4**	70.5	86.6	81.2

Table 6 – Comparison against Various Quantization Algorithms for ResNet-50

Algorithm	Re-training	Biwidth (W, A)	Baseline Accuray (%)	Compressed Accuracy (%)	Energy Efficiency (TOPS/W)
Dorefa [4]	One-pass	2, 2	76.9	67.1	257.1
PACT [5]	One-pass	4, 4	76.9	76.5	83.0
OMSE [29]	No need	4, 32	76.0	75.0	9.3
OCS [30]	No need	8, 7	76.1	74.5	14.5
EGQ (2%)	**No need**	**6.8, 7.2**	**76.2**	**74.4**	**25.3**
ACIQ [31]	No need	8, 4	76.1	71.5	25.9
EGQ (5%)	**No need**	**6.2, 6.3**	**76.2**	**71.5**	**31.7**

We also compare EGQ with five different quantization methods in ResNet-50. Energy efficiency for each method is estimated in 7nm SRAM based PIM by NeuroSim. There are several well-known works on extremely low-precision quantization such as Dorefa [4] and PACT [5]. We observe that 2-bit weight and activation by Dorefa achieves the best energy efficiency, but it shows highest accuracy loss (9.8%). PACT shows 4-bit weight and activation with only 0.4% of accuracy loss with high energy efficiency. We expect that models compressed by these methods will show higher energy efficiency as compression ratio is generally much higher than EGQ. However, most of them require at least one-pass of re-training, large training dataset, or handcrafted tuning to reduce accuracy loss. For this reason, we mainly compare EGQ with re-training free quantization algorithms such as OMSE [29], OCS [30], and ACIQ [31].

OMSE achieves 4-bit weight and 32-bit activation. It successfully reduces the large leakage energy of ResNet-50, but 32-bit activation introduces high read dynamic energy. Energy efficiency of OMSE is 9.3TOPS/W which is the lowest among the algorithms. OCS achieves 8-bit weight and 7-bit activation, and it shows relatively balanced leakage energy and read dynamic energy. Hence, higher energy efficiency of 14.5TOPS/W is observed.

EGQ shows 6.8-bit weight and 7.2-bit activation. Though average activation bitwidth is a bit higher than OCS, we observe that lower and more balanced leakage energy and read dynamic energy. EGQ achieves 25.3TOPS/W of higher energy efficiency with almost same accuracy compared to OMSE and OCS. We further investigate the performance of EGQ with ACIQ. ACIQ shows 8-bit weight and 4-bit activation with higher accuracy loss (4.6%). Energy efficiency of ACIQ is 25.9TOPS/W. We set the 5% of accuracy threshold for EGQ to achieve similar accuracy loss. We observe overall energy efficiency is improved by decreasing the accuracy threshold. EGQ shows 31.7TOPS/W energy efficiency and same accuracy with ACIQ. EGQ can automatically and efficiently find the optimal weight and activation bitwidth for energy efficient PIM.

5.4 Other Low-Complexity Networks Quantization

Table 7 – Comparison against Various Quantization Algorithms for MobileNet-V2 and SqueezeNet

Algorithm	Integer-only [34]	Per-channel [35]	EGQ	OMSE	EGQ
Model	MobileNet-V2			SqueezeNet	
Weight Bitwidth	6	8	7.8	4	**6.7**
Activation Bitwidth	6	8	7.5	32	**7.9**
Baseline Accuracy (%)	-	71.9	71.8	58.0	**58.0**
Compressed Accuracy (%)	70.9	69.7	69.7	55.4	**55.5**
Baseline Size (MB)	13.4	13.4	13.4	4.8	**4.8**
Compressed Size (MB)	2.5	3.3	3.3	0.6	**1.0**
Normalized Number of ADC Access	0.16	0.25	0.23	0.47	**0.26**
Energy Efficiency (TOPS/W)	39.7	23.1	24.5	11.9	**32.2**

We study the performance of EGQ with MobileNet-V2 [32] and SqueezeNet [33]. We also compare with the performance of several other quantization algorithms on these networks [29], [34], [35]. Table 7 summarizes the comparison for the two models. Hardware architecture is assumed to be 7nm SRAM based PIM same as preceding experiments. Normalized number of ADC access represents the number of ADC access of the compressed model divided by the one of 16-bit model. As NeuroSim does not support these models, we approximately calculate the total leakage energy and total read dynamic energy for the energy efficiency based on the energy model of NeuroSim.

For MobileNet-V2, Integer-only compresses weight and activation more than EGQ and Per-channel, and energy efficiency is also higher. However, it requires retraining while Per-channel and EGQ are retraining free quantization algorithms. EGQ shows lower weight and activation bitwidth than Per-channel with same accuracy. Also, normalized number of ADC access is lower than Per-channel, hence we observe higher energy efficiency in EGQ. EGQ shows relatively slight improvement of energy efficiency compared to ResNet-50. We observe that early layers in MobileNet-V2 is relatively more sensitive to quantization errors compared to VGGNet and ResNet models. Though early layers consume more number of ADC access, EGQ does not assign lower bitwidth to the layers to avoid the huge accuracy loss, and it makes relatively small improvement of energy efficiency in MobileNet-V2. As EGQ is based on linear quantization scheme with maximum clipping threshold to simplify the algorithm, we expect that more advanced clipping method or non-linear quantization scheme will enhance the performance of EGQ in MobileNet like models. For SqueezeNet, OMSE shows 4-bit weight quantization with floating-point activation. EGQ achieves 6.7-bit weight and 7.9-bit activation with similar accuracy. Similar to the comparison between OMSE and EGQ in Table 6, we observe that EGQ shows more balanced leakage energy and read dynamic energy. Also, EGQ achieves much lower number of ADC access with higher energy efficiency. Effectiveness of EGQ over the energy efficiency improvement can be variable depending on the models, but EGQ is still effective in MobileNet-V2 and SqueezeNet.

CHAPTER 6. CONCLUSION

This thesis presents a genetic algorithm-based energy-aware quantization method (EGQ) to run DNN models energy efficiently on PIM platforms. We discuss how to combine energy efficiency to a genetic algorithm-based quantization method. EGQ predicts the dynamic energy consumption for inference based on the number of ADC access. ResNet-18, ResNet-50, and VGG-19 are used to evaluate the effectiveness of EGQ. We observe the ResNet models are more suitable for EGQ, and VGG-19 shows unexpected behavior due to too high leakage energy from large fully connected layers. However, the error in EGQ introduced by the leakage energy is mitigated by eNVMs, consequently showing better energy efficiency. As most of modern CNN models do not have such large fully connected layers, EGQ can effectively search quantization bitwidth of various CNN models for energy-efficient PIM. For future works, the consideration for more hardware details such as memory device types and ADC precision will further improve the energy efficiency of EGQ. We use a genetic algorithm due to the flexibility of the fitness function design. Other problem solvers can be adapted for future studies. Also, understanding the effect of device variation or non-ideality on mixed-precision design will be explored in the future.

APPENDIX A. PROPERTIES OF PROPOSED APPROACH

A.1 Run-time

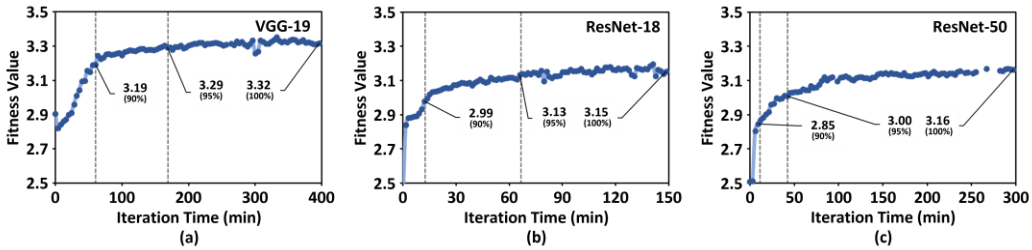

Figure 11 – Changes of fitness value during EGQ iterations. Energy-aware cases are used for VGG-19, ResNet-18, and ResNet-50. The range of iteration time in three models is different, but the total number of EGQ iteration is same as 100. 90% and 95% of final fitness value are marked.

We observe one EGQ iteration takes 1.5 minutes for ResNet-18, 3.0 minutes for ResNet-50, and 4.0 minutes for VGG-19 in GTX 1080Ti and i7-7700K environment. Figure 11 shows saturation behavior of fitness values of three models. Fitness value abruptly decreases in VGG-19 and ResNet-50 sometimes because we set large penalty if the best candidate does not satisfy accuracy threshold. This time is mainly consumed at candidate evaluation step. Accuracy cost in the fitness function is estimated by average accuracy of 3,000 test images. The number of test data is heuristically determined, and it can be controlled depending on the required run-time. We can reduce the amount of the test data so that decrease the inference time proportionally. However, too small test dataset will not be desirable to correctly evaluate the average accuracy.

A.2 Weighting Factors

Weighting factors (α, β, and γ) reflect the relative importance of weight compression, activation compression, and ADC access. We determine the weighting factors as 0 or 1 in the previous section only to consider compression awareness and energy awareness. However, the overall energy efficiency can be affected by both weight, activation compression and ADC access. For example in Table 3, we observe large fully-connected layers in VGG models lead to tremendous leakage energy. In this case, higher α, β will be desirable as it can increase compression ratio and decrease leakage energy.

Table 8 – Energy Efficiency of VGGNet, ResNet with Balanced Weighting Factors

Model	**VGG-19**	**ResNet-18**	**ResNet-50**
Parameter (α, β, γ)	**1.49, 1.49, 0.02**	**0.86, 0.86, 1.28**	**1.30, 1.30, 0.40**
Weight Bitwidth	4.2	7.1	8.0
Activation Bitwidth	6.2	6.8	7.8
Total Dynamic Energy (μJ)	1422.0	104.8	308.1
ADC Dynamic Energy (μJ)	824.2	56.9	144.0
Total Leakage Energy (μJ)	7263.1	43.0	526.9
Energy Efficiency (TOPS/W)	4.5	36.1	12.3
Baseline Accuracy (%)	72.4	69.8	76.2
Compressed Accuracy (%)	70.9	67.5	74.2

We explore other weighting factors for VGG-19, ResNet-18, and ResNet-50. They are determined based on the ratio of leakage energy and read dynamic energy of the 16-bit models. Specifically, we assume that the desirable ratio of the weighting factors follows leakage energy: read dynamic energy = α: γ = β: γ. We constrain the sum of the weighting factors to 3 as the sum in initial energy-aware case is also 3. For example, 16-bit ResNet-50 shows 1.01mJ for leakage energy and 3.35mJ for read dynamic energy in 7nm SRAM based PIM. We set α, β, and γ for the model to 1.3, 1.3, and 0.4 as 3.35: 1.01 (mJ) = 1.3(α): 0.4(γ) = 1.3(β): 0.4(γ). The weighting factors of VGG-19 and ResNet-18 can be similarly

balanced. Table 8 shows the energy efficiency of the three models with balanced weighting factors. We observe that VGG-19 shows slightly higher ADC dynamic energy while lower total leakage energy compared to energy-aware case in Table 3. ResNet-50 also shows similar results with VGG-19. However, ResNet-18 shows lower ADC dynamic energy with a bit higher total leakage energy. All three models prove that balancing weighting factors can further improve the overall energy efficiency compared with the energy-aware cases.

APPENDIX B. NEUROSIM

DNN+NeuroSim is an integrated framework to benchmark processing-in-memory (PIM) accelerators for deep neural networks, with hierarchical design options from device-level, to circuit-level and up to algorithm-level [17]. It supports flexible PIM array design options with different device technologies (from SRAM to eNVMs) with various peripheral circuitry modules, which have been validated with SPICE simulations and actual device data. The framework supports automatic algorithm to hardware mapping for diverse models, and evaluates chip-level performance like area, latency and energy consumption. The automatic floorplan and dataflow design of NeuroSim is inherently compatible with various algorithm and device precisions. NeuroSim is originally designed to simulate the uniform bit-width model. To accommodate the flexible bit-width, layer-wise quantization is employed to extract the real trace data to feed the hardware performance estimation, and the different activation and weight precisions are assigned to corresponding layers. The digital resources like shift-add, accumulator and activation are designed to accommodate the maximum precision. We set a subarray size to 128×128, and ADC precision to 5 bit for all simulation.

REFERENCES

[1] A. Shafiee et al., “Isacc: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars,” ACM SIGARCH Computer Architecture News, vol. 44, no. 3, pp. 14–26, 2016.

[2] P. Chi et al., “Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory,” ACM SIGARCH Computer Architecture News, vol. 44, no. 3, pp. 27–39, 2016.

[3] C. Eckert et al., “Neural cache: Bit-serial in-cache acceleration of deep neural networks,” in 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2018, pp. 383–396.

[4] Z. Shuchang, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” in arXiv preprint arXiv:1606.06160, 2016.

[5] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, “Accurate and efficient 2-bit quantized neural networks,” in SysML 2019, 2019.

[6] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, “Haq: Hardware-aware automated quantization with mixed precision,” in Computer Vision and Pattern Recognition (CVPR). IEEE/CVF, 2019, pp. 8612–8620.

[7] Y. Long, E. Lee, D. Kim, and S. Mukhopadhyay, “Q-pim: A genetic algorithm based flexible dnn quantization method and application to processing-in-memory platform,” in 2020 57th ACM/IEEE Design Automation Conference (DAC), 2020, pp. 1–6.

[8] Y. Zhou, S. M. Moosavi-Dezfooli, N. M. Cheung, and P. Frossard, “Adaptive quantization for deep neural network,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

[9] M. Rusci, A. Capotondi, and L. Benini, “Memory-driven mixed low precision quantization for enabling deep network inference on microcontrollers,” in arXiv preprint arXiv:1905.13082, 2019.

[10] C. Ding, N. L. S. Wang, K. Xu, Y. Wang, and Y. Liang, “Req-yolo: A resource-aware, efficient quantization framework for object detection on fpgas,” in Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2019, pp. 33–42.

[11] T. J. Yang, Y. H. Chen, and V. Sze, “Designing energy-efficient convolutional neural networks using energy-aware pruning.” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5687–5695.

[12] Y. Cai, T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, “Low bit-width convolutional neural network on rram,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 7, pp. 1414–1427, 2019.

[13] H. Sun, Z. Zhu, X. Chen, Y. Wang, and H. Yang, “An energy-efficient quantized and regularized training framework for processing-in-memory accelerators,” in 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2020, pp. 325–330.

[14] K. Vasquez, Y. Venkatesha, A. Bhattacharjee, A. Moitra, and P. Panda, “Activation density based mixed-precision quantization for energy efficient neural networks,” in arXiv preprint arXiv:2101.04354, 2021.

[15] Z. Zhu et al., “A configurable multi-precision cnn computing framework based on single bit rram,” in 2019 56th ACM/IEEE Design Automation Conference (DAC). IEEE, 2019, pp. 1–6.

[16] S. Huang et al., “Mixed precision quantization for reram-based dnn inference accelerators,” in 2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2021, pp. 372–377.

[17] 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 2019 IEEE International Electron Devices Meeting (IEDM), 2019, pp. 32.5.1–32.5.4.

[18] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in arXiv preprint arXiv:1409.1556, 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, 2016, pp. 770–778.

[20] S. Qu, B. Li, Y. Wang, D. Xu, X. Zhao, and L. Zhang, “Raqu: An automatic high-utilization cnn quantization and mapping framework for general-purpose rram accelerator,” in 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020, pp. 1–6.

[21] C. Zhang and P. Zhou, “A quantized training framework for robust and accurate reram-based neural network accelerators,” in 2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2021, pp. 43–48.

[22] K. Samal, M. Wold, and S. Mukhopadhyay, “Attention-based activation pruning to reduce data movement in real-time ai: A case-study on local motion planning in autonomous vehicles,” IEEE Journal of Emerging and Selected Topics in Circuits and Systems, vol. 10, no. 3, pp. 306–319, 2020.

[23] S. Yin, Z. Jiang, J. S. Seo, and M. Seok, “Xnor-sram: In-memory computing sram macro for binary/ternary deep neural networks,” IEEE Journal of Solid-State Circuits, vol. 55, no. 6, pp. 1733–1743, 2020.

[24] Y. Long et al., “A ferroelectric fet based power-efficient architeture for data-intensive computing,” in Proceedings of the International Conference on Computer-Aided Design (ICCAD). ACM, 2018, pp. 1–8.

[25] X. Si et al., “24.5 a twin-8t sram computation-in-memory macro for multiple-bit cnn-based machine learning,” in 2019 IEEE International Solid- State Circuits Conference (ISSCC). IEEE, 2019, pp. 396–398.

[26] X. Peng, R. Liu, and S. Yu, “Optimizing weight mapping and data flow for convolutional neural networks on processing-in-memory architectures,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 67, no. 4, pp. 1333–1343, 2020.

[27] W. Wu et al., “A methodology to improve linearity of analog rram for neuromorphic computing,” in 2018 Symposium on VLSI Technology. IEEE, 2019, pp. 103–104.

[28] K. Ni et al., “In-memory computing primitive for sensor data fusion in 28 nm hkmg fefet technology,” in 2018 International Electron Devices Meeting (IEDM). IEEE, 2018, pp. 16–1.

[29] R. Zhao, Y. Hu, J. Dotzel, C. D. Sa, and Z. Zhang, “Improving neural network quantization without retraining using outlier channel splitting,” in International conference on machine learning. PMLR, 2019, pp. 7543–7552.

[30] C. Y, E. Kravchik, F. Yang, and P. Kisilev, “Low-bit quantization of neural networks for efficient inference,” in ICCV Workshops, 2019, pp. 3009–3018.

[31] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, “Aciq: Analytical clipping for integer quantization of neural networks,” 2018.

[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

[33] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size,” in arXiv preprint arXiv:1602.04354, 2016.

[34] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, A. Hartwig, 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.

[35] R. Krishnamoorthi, “Quantizing deep convolutional networks for efficient inference: A whitepaper.” in arXiv preprint arXiv:1806.08342, 2018.