Optimizing Weight Value Quantization for CNN Inference

Wakana Nogami[1,2] Tsutomu Ikegami[1] Shin-ichi O'uchi[1] Ryousei Takano[1] Tomohiro Kudoh[2]
[1] *National Institute of Advanced Industrial Science and Technology*
Tsukuba, Japan
[2] *The University of Tokyo*
Tokyo, Japan

***Abstract*—The size and complexity of CNN models are increasing and as a result they are requiring more computational and memory resources to be used effectively. Use of a lower bit width numerical representation such as binary, ternary or several bit width has been studied extensively so as to reduce the required resources. However, the representation capability of such extremely low bit width is not always sufficient and the accuracy obtained for some CNN models and data is low. There are some prior studies that use moderate lower bit width with well-known numerical representations such as fixed point or logarithmic representation. It is not apparent, however, whether those representations are optimal for maintaining high accuracy. In this paper, we investigated the numerical quantization from the ground up, and introduced a novel "Variable Bin-size Quantization (VBQ)" representation in which quantization bin boundaries are optimized to obtain maximum accuracy for each CNN model. A genetic algorithm was employed to optimize the bin boundaries of VBQ. Additionally, since the appropriate bit width to obtain sufficient accuracy cannot be determined in advance, we attempted to use the parameters obtained by a training process using higher precision representation (FP32), and used quantization in inference only. This reduced the required large computational resource cost for training. During the process of tuning VBQ boundaries using a genetic algorithm, we discovered that the optimal distribution of bins can be approximated by an equation with two parameters. We then used simulated annealing for finding the optimal parameters of the equation for AlexNet and VGG16. As a result, AlexNet and VGG16 with our 4-bit quantization achieved top-5 accuracy at 74.8% and 86.3% respectively, which were comparable to 76.3% and 88.1% obtained by FP32. Thus, VBQ combined with the approximate equation and the simulated annealing scheme can achieve similar levels of accuracy with less resources and reduced computational cost compared to other current approaches.**

***Index Terms*—Convolutional Neural Network, Quantization, Optimization, Genetic Algorithm, Simulated Annealing**

I. Introduction

The increasing size and complexity of CNN models is requiring more computational power and memory in order for the models to be used more widely. Some of the larger models can only be run on limited high-end servers [1]–[3]. For example, ResNet-50 takes 29 hours using 8 Tesla P100 to train ImageNet [4]. In the latest research, Mikami et al. have conducted the same training in 224 seconds, but they used up to 2,176 Tesla V100 [3]. More complex models also require more computation resources for inference,

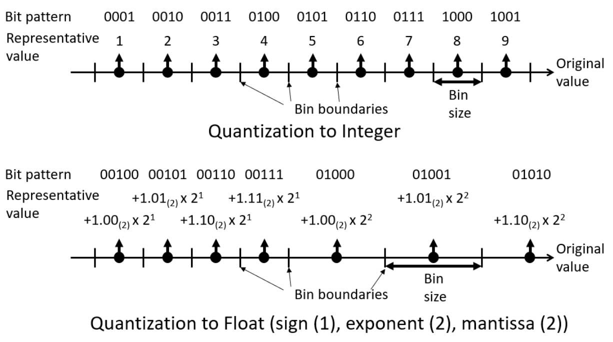

Fig. 1. Quantization examples

approximately a third of the resources needed for training. However, inference typically runs on edge devices or small servers for hosting applications, and these devices typically have much less number of cores and memory. Thus, it is necessary to compress the complex model to adapt to those environments.

Compression techniques to reduce the trained model size are extensively studied. Reducing bit width in quantizing numbers is one promising approach for the compression. With the reduced bit width, both computation and memory footprint are expected to be reduced. It is even possible to save energy for inference, by preparing specially tailored hardware. There are two different approaches to bit width reduction: one includes training and the other does not. In the former approach, a quantized model is built from scratch, which is then trained [5]–[9]. With this approach, improved accuracy may be achieved compared to the corresponding floating point 32bit (FP32) results [8], and the compression rate is generally very high. However, building and tuning the new models require a lot of effort, and are very time consuming. The latter approach aims at reducing the bit width of already-trained models without affecting their accuracy [10]–[12]. Although this approach is difficult because quantization error can accumulate, existing models with established trained data can be reused without the costly training process.

This work described in this paper focuses on the second approach. Different from prior studies, we make no assumption about the quantization format, such as fixed point, floating point, or logarithmic representations. Instead, we find an "optimal" quantization under a given bit width. The quantization can be defined as mapping a continuous value onto a numerical representation of finite bit width. In more detail, a continuous value range is divided into bins (sections) indexed by the given bit width, and any numbers in each bin are replaced by a representative value assigned to the bin. Quantization schemes are depicted in Fig . 1 for two widely used representations: the fixed point format and the floating point format. These schemes are designed to have systematically determined bin sizes and representative values, and to be easily processed mathematically. However, they are not necessarily "optimal" in maximizing the accuracy of CNN under the given bit width. By arranging bin boundaries and representative values more flexibly, it will be possible to improve the accuracy, or to reduce the bit width further. The contributions of this research are as follows:

- We proposed a scheme called variable bin size quantization (VBQ) and applied it to CNN.
- The bin sizes (or the representative values) are optimized by using the genetic algorithm.
- We derived a formula to describe the distribution of the representative values, which smoothly connects the fixed point-like quantization and the floating point-like quantization.
- We optimized parameters in the formula using simulated annealing and succeeded in reducing the weight of all layers to 4 bits with only a slight reduction in top-5 accuracy of 1.84% in AlexNet and 1.50% in VGG16 respectively.

II. Related Works

There are several studies to reduce the bit width of the weight during training [13]–[15]. These studies use MNIST as data sets, and a multilayer perceptron consisting only of fully connected layers. The techniques of reducing the bit width of CNN weights for tasks of 10 class classifications such as MNIST and CIFAR-10 are also studied [5], [6], [16]. These techniques succeeded in binarizing weights and activations without degrading the inference accuracy. Additionally, several researchers studied reducing the bit width of the weight and activation in CNN for large-scale data sets such as ImageNet [7], [10], [17]. One key difference with these studies and our current study is that they needed to train models with binarized weights and activations.

On the other hand, some studies [10]–[12] aim at reducing bit width for already-trained models, similar to our study. Lin et al. compressed the model by optimizing the bit width of each layer so as to minimize the fixed point quantization error [11]. They applied the compression only to the convolution layers because fully connected layers dominate the model size there, but results in the ImageNet model being not well compressed. Conversely, the CIFAR-10 model can be compressed by $> 20\%$, where the convolution layers dominate. Kamiya et al. decomposed real-valued weight vectors by a set of binary basis vectors, and quantize activation [12]. This replaces real-valued inner-product computation with binary inner-product, which accelerates inference and decreases model size. Miyashita et al. showed that it is possible to reduce the weight of the fully connected layers to 4 bits and the weight of the convolutional layers to 5 bits without a large reduction in accuracy, by expressing the values using logarithms based on 2 or $\sqrt{2}$ [10].

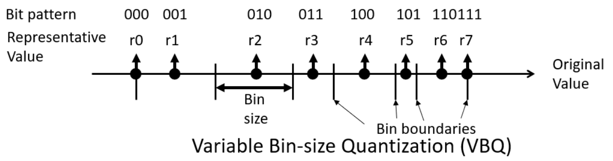

Fig. 2. Variable Bin-size Quantization example

In signal compression, adaptive quantization schemes are widely used, in which an appropriate bin size is chosen for each sample to be quantized. This idea was applied to CNN in [18], [19], though the fixed point representation is implied and bin sizes are not variable.

Other approaches to handle the complexity of large CNN models include (a) training a small model with the output of a large and high accuracy model [20], [21] (b) sharing weights using a feature hash [22] and (c) combining pruning and quantization during training [23]. These methods are complementary to our proposed approach, and further compression is expected by using them together.

III. Optimizing Quantization

A. Variable Bin-size Quantization

When converting a continuous value to a digital value, quantization is used. Quantization is an operation that converts a range of values to a representative value. We denote the range by bin, the width of the range by bin size, and the boundary of the range by bin boundary. When n bits are available for the quantization, the number of bins becomes 2^n

In order to optimize the quantization bins more freely, we propose "Variable Bin-size Quantization (VBQ)" as shown in Fig . 2. VBQ is a type of adaptive quantization where representative values can be changed independently, so as to optimize bin locations and bin sizes. Here, the bin boundary is taken at the middle of two adjacent representative values.

B. Genetic Algorithm

A genetic algorithm is used to optimize the representative values v_i of VBQ by taking the accuracy of CNN to be the fitness score. Genetic algorithms [24] are heuristic algorithms that search for an optimal solution by repeating operations such as reproduction, crossover, and mutation on multiple individuals with various genomes. In this study, each "genome" is a set of representative values $\{v_i\}$, and is a candidate of an optimized numerical representation for CNN. In the following, procedures of the employed algorithm are outlined.

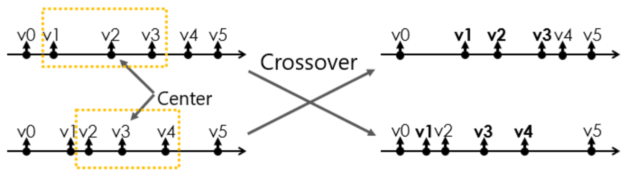

Fig. 3. Diagram illustrating the modified crossover method implemented in the genetic algorithm.

1) Reproduction: Reproduction is an operation to pass superior genomes to the next generation. In the present algorithm, we combine the elite selection and the roulette selection as follows. First, we select an individual with the highest fitness, and reproduce it in the next generation. From the rest of the individuals, $R_r\%$ is selected according to a probability proportional to the fitness.

2) Crossover: Crossover (also known as recombination) is an operation that combines information from two individuals to generate offspring by mating. There are various methods for the crossover operation, but the two-point crossover is widely used, where two crossover points are randomly chosen and a section of genomes between them are exchanged. In the present algorithm, we choose two sets of $R_c\%$ of individuals randomly, and apply a variation of the two-point crossover operation on each pair of them as follows.

In an ordinary two-point crossover, indexes of the genetic sequence are taken as the crossover points. However, in the VBQ optimization, where the genetic sequence is an array of representative values, the range between two crossover indexes $[v_i, v_j]$ may not overlap at all between two individuals. Therefore, instead of specifying two crossover indexes, we specify a center of the range and the number of representative values to be exchanged around the center. In this manner, genetically comparable information can be exchanged as shown in Figure 3.

3) Mutation: Mutation is an operation to randomly change a part of a gene and serves to promote genetic diversity. This then avoids local optimal solutions by preventing the genes from becoming too similar. We implemented mutation by randomly moving the position of the randomly selected representative values. The amount of the movement is regulated such that the value does not exceed the representative values on both sides. We choose $R_m\%$ of individuals randomly, and apply the mutation above.

C. Outline of Experiments

The outline of the system used in this experiment is shown in Fig. 4. The system consists of three parts: GA, Quantizer, and CNN. The flow of the experiment was as follows.

1) CNN: is trained with FP32 a priori, to prepare a set of trained weights.
2) GA: prepares a set of genomes, and sends each of them to the Quantizer.

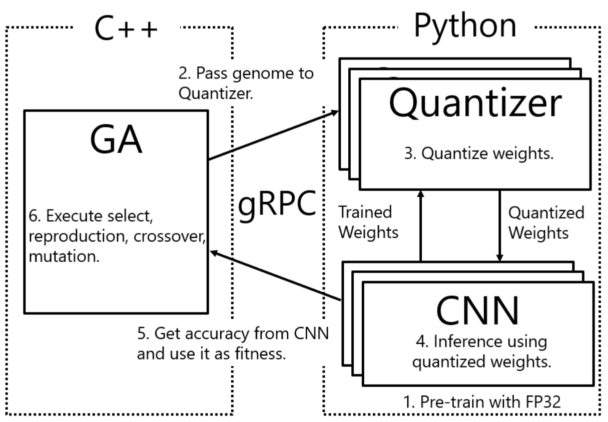
C++
Python
2. Pass genome to Quantizer.
GA
6. Execute select, reproduction, crossover, mutation.
gRPC
Quantizer
3. Quantize weights.
Trained Weights
Quantized Weights
CNN
4. Inference using quantized weights.
5. Get accuracy from CNN and use it as fitness.
1. Pre-train with FP32

Fig. 4. Diagram showing the overall steps in the experimental system.

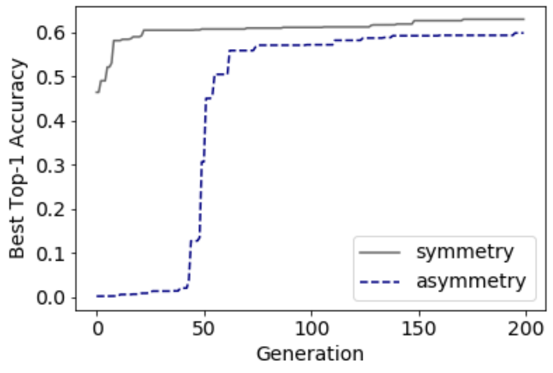

Fig. 5. Changes in the accuracy for each generation

3) Quantizer: quantizes the trained weights based on the representative values of the genome.
4) CNN: performs inference using the quantized weights to obtain the Top-1 accuracy.
5) GA: collects the Top-1 accuracies of genomes from CNN, which are used as the fitness scores.
6) GA: performes the genetic algorithm in Section III-B to create the next generation genomes.
7) Repeat $1 \sim 6$

The GA was implemented in C++, and the Quantizer and CNN were implemented in Python. We used gRPC for the communication between these three parts. We used Keras and Tensorflow for the CNN implementation. We experimented with VGG16 using the ILSVRC2012 validation dataset. Initial genomes were randomly generated, and the number of genomes was 50. The probability of reproduction (R_r), crossover ($2R_c$), and mutation (R_m) were 30%, 50%, and 20% respectively, chosen for the simulation to proceed smoothly.

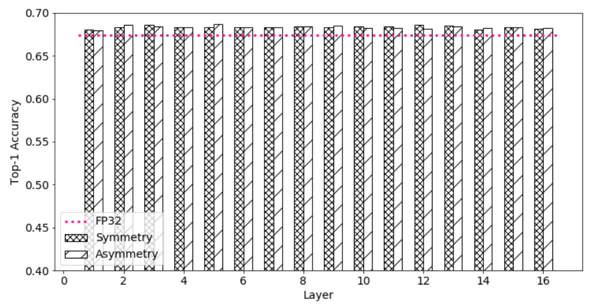

Fig. 6. The accuracy of quantizing and optimizing only one layer with other layer using FP32

D. Experimental Results and Discussion

In the VGG16 experiment, we used the trained weights available from Keras library, which gives 67.38 % Top-1 accuracy with FP32 for the validation data set of ILSVRC2012. The number of bins, or the number of the representative values to be optimized, is $N = 16$, which corresponds to 4-bit quantization. There are two simulation modes in VBQ: asymmetric and symmetric. In the asymmetric simulation, we place no assumptions on the distribution of v_i. On the contrary, in the symmetric simulation, v_i are distributed symmetrically about the origin, $v_{N-1-i} = -v_i$.

First, we applied the same VBQ representation on all layers of VGG16, and optimized v_i by GA. History of the best Top-1 accuracy among genomes is compared between the asymmetric and symmetric simulations in Fig. 5.

It is apparent that the symmetric simulation attains the faster convergence and the better accuracy, although both fail to reach the accuracy at FP32. In principle, the asymmetric simulation can reach the same accuracy as the symmetric simulation, but the convergence is slow due to the larger degree of freedom.

Next, we applied VBQ on each layer, treating the rest of layers at FP32. The best score after 200 generations is plotted in 6. In this case, both the asymmetric and symmetric simulations identically give the better accuracy than FP32. These results indicate that the optimal VBQ representation differs from layer to layer.

To investigate the layer-wise difference of the optimal VBQ representation, we conducted nonlinear regression analysis on the distribution of the representative values. We heuristically deduced the regression formula,

$$V(x) = \mathrm{sign}(x-d) \times b \times (a^{|x-d|} - 1) + c, \tag{1}$$

where x is a bin index normalized to $[-0.5, 0.5]$, a is an exponential factor, b is a scaling factor, c is an offset, d is a bias in x-direction, and $V(x)$ gives a representative value for the bin indexed by x. Note that c and d become zero for the symmetric distribution. This formula is characteristic in smoothly connecting the linear distribution ($a \simeq 1$) and the exponential distribution ($a >> 1$). Fitting results are illustrated for the first and tenth layers in Fig. 7, for both the asymmetric and symmetric simulations.

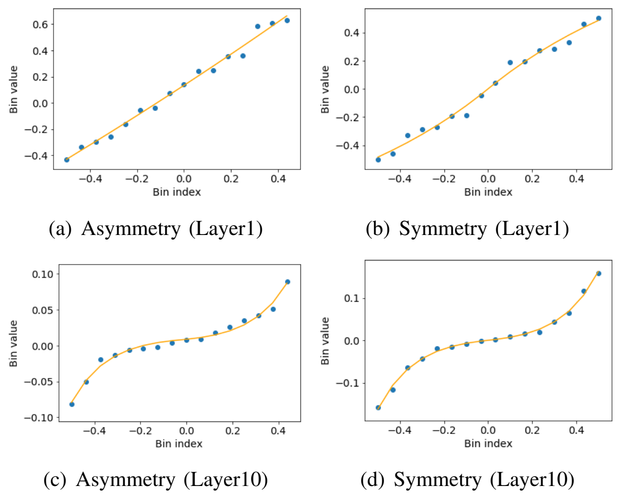

Fig. 7. Distributions of representative values and the regression results

In both cases, a linear and an exponential distributions are observed for layer 0 and layer 10, which are reflected in the fitted parameter of $a = 0.313$ and 385, respectively.

By using the parameters obtained from the regression, VBQ is set up based upon Eq.1, and the Top-1 accuracy is calculated. The results are shown in Fig. 8 along with the R^2 scores of the regression. Thus calculated Top-1 accuracy almost reproduce the FP32 results, except for the 14th layer of the asymmetric simulation. In this case, judging from the R^2 score, the distribution of v_i is not well described by Eq.1. Because the symmetric counterpart gives a satisfactory result, however, the failure is due to non-convergence of GA. From Fig. 8, the symmetric cases generally give a better R^2 scores, indicating that the symmetric distribution of v_i is preferable. Hereafter, we replace Eq.1 by the simpler form,

$$V(x) = \mathrm{sign}(x) \times b \times (a^{|x|} - 1). \tag{2}$$

Therefore, we decided to experiment only in the case of placing representative values in positive/negative symmetry hereafter.

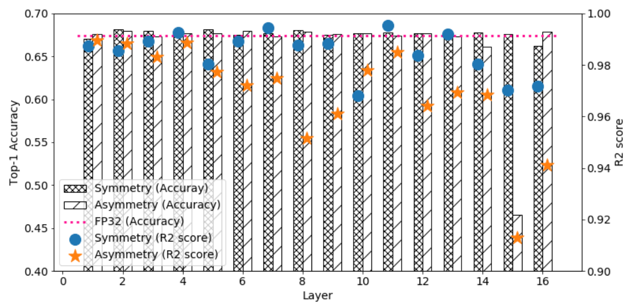

Fig. 8. The accuracy of quantizing each layer using regression results

In the case of optimizing the representative values itself, the accuracy exceeds that of FP32 int all layers. On the other hand, in the case of taking the representative values among the regression function, the accuracy has fallen below that of FP32 in some layers. However, in the case of symmetry, the decrease is only about 1.2%. Besides, when represent values are placed along Eq.2, It is enough to optimize two values(a, b) for each layer, regardless of the number of representative values. So, the number of parameters to optimize can be greatly reduced especially when the number of representative values is large. Therefore, instead of optimizing the representative values itself, it is useful to optimize a, b and take representative values along the Eq.2. In the next chapter, we will consider how to optimize a, b directly without regression.

IV. Optimizing Fitting Parameters

Relying on Eq. 2, we can set up VBQ of any number of bins by just choosing parameters a and b. This greatly reduces the optimization cost of VBQ, especially when the number of bins is large. In this section, we discuss how to optimize a and b.

A. Grid Search

First, we draw heatmaps of the Top-1 accuracy against parameters (a, b), to catch what the search space looks like. For better survey, we employ $\tilde{a} = a^{1/N}$ instead of a in constructing the uniform grid, where N is the number of bins. Logarithmic grid is also employed for b. The results are summarized in Fig. 9. The accuracy at FP32 is taken as the baseline, and those points deviated more than 1 % from the baseline are omitted from the plots. The red and blue regions in the plots correspond above and below the baseline, respectively. As shown in the figure, the shape of the heatmap is similar among layers, though the the size of the high accuracy region differ. Especially, the high accuracy regions are wide for the fully connected layers (Layer 13–15), indicating that the accuracy is insensitive to the quantization parameters. For these layers, it may be possible to reduce the quantization bit width further. On the contrary, the convolutional layers are sensitive to the quantization parameters, especially for the Layer 1. It is probably because the weights are heavily reused in the convolutional layer, which causes a slight quantization error to accumulate. These results indicate that the bit width should also be optimized from layer to layer, though it is beyond the scope of this paper.

B. Simulated Annealing

In the heatmap drawn in the previous section, we notice a lot of local maximums. To optimize the parameters a and b avoiding those local maximums, we employ the simulated annealing (SA) algorithm. SA is an optimization algorithm that simulates annealing in metal engineering [25], [26]. SA utilizes an artificial thermal fluctuation to prevent from sticking at the local optimum solutions. Setup of the SA simulation is simple: just replace the GA portion in Fig.4 by SA. One iteration of our algorithm goes as follows.

1) SA: selects neighborhood a', b' of a, b
2) SA: generates a genome with $v_i = V(x_i|a', b')$ and sends it to Quantizer
3) Quantizer and CNN: perform inference to obtain the Top-1 accuracy e'
4) SA: updates a, b by a', b' with the probability $P(e, e', T)$
5) Repeat $1 \sim 6$ for each layer
6) SA: lowers the temperature T

In step 1, we prepare Δa and Δb randomly in the range of $[-\frac{1}{2}aT, \frac{1}{2}aT]$ and $[-\frac{1}{2}bT, \frac{1}{2}bT]$, respectively, and generate three neighborhoods, $(a+\Delta a, b)$, $(a, b+\Delta b)$, and $(a+\Delta a, b+\Delta b)$. In step4, probability $P(e, e', T)$ is defined as

$$P(e, e', T) = \begin{cases} 1 & (e' \geq e) \\ \exp\left(\frac{(e'-e)\times 100}{T}\right) & (e' < e) \end{cases}, \tag{3}$$

where e and e' are the Top-1 accuracies calculated with (a, b) and (a', b'), respectively. The initial values of (a, b) are taken to be $(1.25, 0.05)$ for all layers, based on the heatmap, and the initial temperature is taken at $T = 1$ with the temperature reduction rate of 0.95. The simulation is stopped when (a, b) of all layers are not updated for consecutive 30 iterations.

The SA simulations are performed by varying the number of bins N, which is taken to be the same among layers. The obtained Top-1 accuracies are plotted as a function of N in Fig.10. The accuracy improves as N increases, and becomes comparable to the FP32 result at $N = 16$ (4 bits). Comparing with the result shown in Fig. 5, where the same VBQ is applied on all layers, the layer-wise optimization is shown to be important to reproduce the FP32 accuracy; this can only be achieved by combining Eq.2 and SA.

In the experiments so far, we used the same set of 5000 images for the SA optimization and the final evaluation. It is more practical, however, to use a different set of images for the final evaluation. Minimum number of images required for the optimization is also concern, because it is directly proportional to the optimization cost. In Fig. 11, the achieved Top-1 accuracies are plotted as a function of the number of images used for the SA optimzation. The Top-1 accuracies are evaluated by using a different set of 5000 images, and the number of bins is taken to be $N = 16$. The figure indicates that the optimization can be performed with as small as 200 images, and no further improvement is expected after 1000 images.

C. Evaluation with another CNN model

To check the generality of the proposed approach, we have applied our method to AlexNet [27]. AlexNet consists of five convolutional layers and three fully connected layers. It is used for evaluation in many previous studies [10], [12]. We refer [28] for the implementation and the trained data. The Top-1 accuracy at FP32 is 0.5244 in our environment.

The experiments are performed by varying the number of bins N. The SA optimizations are performed for VGG16 and AlexNet by using 1000 images, which are evaluated with a

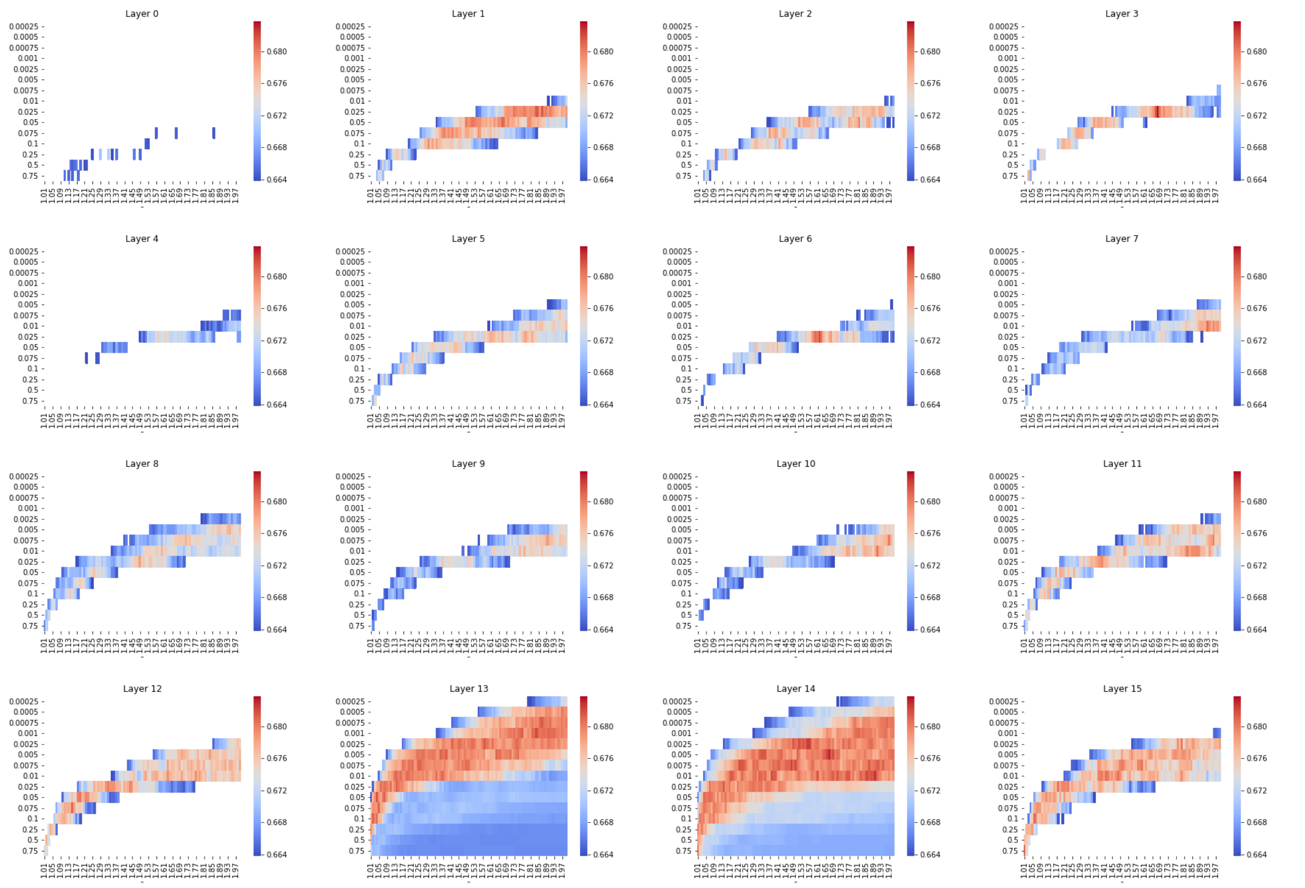

Fig. 9. Heatmap of a, b in 8 represent values(3 bit)

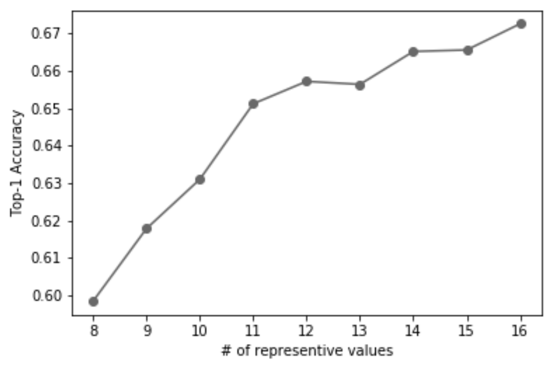

Fig. 10. Changes in the accuracy for the number of representative values

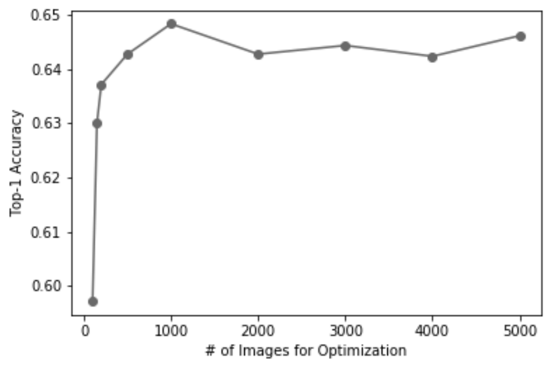

Fig. 11. The number of images for Optimization (16 representative values)

different set of 5000 images. The results are shown in Figs. 12 and 13. AlexNet and VGG 16 show the similar tendency: accuracy becomes significantly worse as N dropped below 16 (4 bits), while it reaches FP32-equivalent. We compared the inference accuracy of proposed method with other two methods for compressing already-trained models [10], [12]. Different from our work, these prior works reduced bit widths of activation as well as those of weights. Miyashita et al reduced the weight of the convolutional layer to 5 bits, the weights of the FC layer to 4 bits, and the activation to 4 bits. The drop of the top-5 accuracy was 1.7% in AlexNet and 0.5% in VGG16. We succeeded in reducing the weight of all layers to 4 bits with decreasing top-5 accuracy by 1.84% in AlexNet and 1.5% in VGG16 although we still used FP32 for the activations. Kamiya et al reduced the bit width to 6 bits for both weight and activation, and they succeed

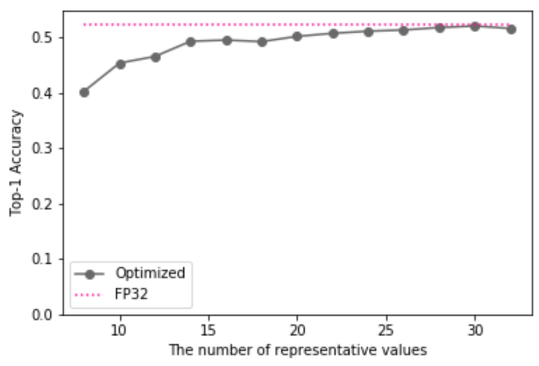

Fig. 12. Top-1 accuracy of AlexNet using various number of representative values

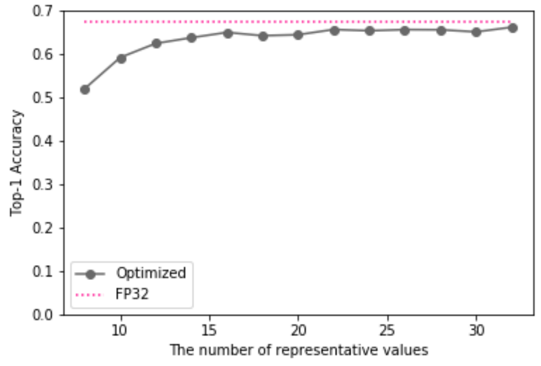

Fig. 13. Top-1 accuracy of VGG16 using various number of representative values

TABLE I
INFERENCE ACCURACY(TOP-1/TOP-5) OF ALEXNET AND VGG16 ON IMAGENET USING VARIOUS MODEL QUANTIZATION METHODS

	Model	Decrease of Accuracy (Top-1 / Top-5)	Bit widths Weights	Bit widths Activation	Acceleration
Miyashita et al [10]	AlexNet	- / 6.2%	Conv 5bit(Log2 Format) + FC 4bit	4bit	-
		- / 1.7%	Conv 5bit(Log$\sqrt{2}$ Format) + FC 4bit	4bit	-
	VGG16	- / 6.1%	Conv 5bit(Log2 Format) + FC 4bit	4bit	-
		- / 0.5%	Conv 5bit(Log$\sqrt{2}$ Format) + FC 4bit	4bit	-
Kamiya et al [12]	AlexNet	1.7% / 1.2%	6bit	6bit	1.79
	VGG16	- / 2.16%	6bit	6bit	2.07
Proposed	AlexNet	0.34% / 0.54%	5bit	- (32bit)	1.0
		2.88% / 1.84%	4bit	- (32bit)	1.0
	VGG16	1.36% / 1.26%	5bit	- (32bit)	1.0
		2.54% / 1.50%	4bit	- (32bit)	1.0

in accelerating the calculation 1.79 times in AlexNet and 2.07 times in VGG16. In our work, the required amount of computation has not been changed because calculations are performed in FP32. Calculation with the reduced bit width in a dedicated hardware is our future work.

V. Conclusion

In this paper, we introduced VBQ for optimizing the quantization method in the weights during CNN inference. First, we used a genetic algorithm for the optimization method and demonstrated the feasibility of this approach. We discovered that the optimal quantization is different for each network and layer. Next, we deduced a heuristic formula for the distribution of the representative values, with which the effort required to optimize VBQ was greatly reduced: we optimized the parameters of the formula using a simulated annealing and showed that higher accuracy can be obtained than in the previous researches. Although the bit width of the VBQ representation was taken to be the same for all layers, we noticed that the minimum bit width to reproduce the FP32 results may differ from layer to layer. Optimization of the bit width is our future work. Our future work also includes quantization of activations, complexity analysis of the VBQ approach, and hardware acceleration of the VBQ representation.

Acknowledgment

The authors would like to thank Dr. Jason Haga for his invaluable feedback. This paper is based on results obtained from a project by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] T. Akiba, S. Suzuki, and K. Fukuda, “Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes,” *ArXiv:1711.04325*, 11 2017. [Online]. Available: https://arxiv.org/abs/1711.04325

[2] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, “ImageNet Training in Minutes,” in *Proceedings of the 47th International Conference on Parallel Processing - ICPP 2018*. New York, New York, USA: ACM Press, 2018, pp. 1–10. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3225058.3225069

[3] H. Mikami, H. Suganuma, P. U-chupala, Y. Tanaka, and Y. Kageyama, “ImageNet/ResNet-50 Training in 224 Seconds,” *ArXiv:1811.05233*, 11 2018. [Online]. Available: http://arxiv.org/abs/1811.05233

[4] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 6 2016, pp. 770–778. [Online]. Available: http://ieeexplore.ieee.org/document/7780459/

[5] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, “Deep learning with limited numerical precision,” *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, pp. 1737–1746, 2015. [Online]. Available: https://dl.acm.org/citation.cfm?id=3045303

[6] M. Courbariaux, Y. Bengio, and J.-P. David, “BinaryConnect: Training Deep Neural Networks with binary weights during propagations,” *ArXiv:1511.00363*, 11 2015. [Online]. Available: http://arxiv.org/abs/1511.00363

[7] Rastegari Mohammad, , V. Ordonez, and Redmon Joseph, and and Farhadi Ali, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," in *Computer Vision ECCV 2016*, Leibe Bastian, , J. Matas, and Sebe Nicu, and and Welling Max, Eds. Cham: Springer International Publishing, 2016, pp. 525–542. [Online]. Available: https://arxiv.org/abs/1603.05279

[8] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights," *ArXiv:1702.03044*, 2 2017. [Online]. Available: http://arxiv.org/abs/1702.03044

[9] S.-i. O'uchi, H. Fuketa, T. Ikegami, W. Nogami, T. Matsukawa, T. Kudoh, and R. Takano, "Image-Classifier Deep Convolutional Neural Network Training by 9-bit Dedicated Hardware to Realize Validation Accuracy and Energy Efficiency Superior to the Half Precision Floating Point Format," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 5 2018, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/8350953/

[10] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional Neural Networks using Logarithmic Data Representation," *ArXiv:1603.01025*, 3 2016. [Online]. Available: http://arxiv.org/abs/1603.01025

[11] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," pp. 2849–2858, 2016. [Online]. Available: https://dl.acm.org/citation.cfm?id=3045690

[12] R. Kamiya, T. Yamashita, M. Ambai, I. Sato, Y. Yamauchi, and H. Fujiyoshi, "Binary-decomposed DCNN for accelerating computation and compressing model without retraining," in *Workshop on International Conference on Computer Vision*, 2017. [Online]. Available: https://arxiv.org/abs/1709.04731

[13] Z. Cheng, D. Soudry, Z. Mao, and Z. Lan, "Training Binary Multilayer Neural Networks for Image Classification using Expectation Backpropagation," *ArXiv:1503.03562*, 3 2015. [Online]. Available: http://arxiv.org/abs/1503.03562

[14] M. Kim and P. Smaragdis, "Bitwise Neural Networks," *ArXiv:1601.06071*, 1 2016. [Online]. Available: http://arxiv.org/abs/1601.06071

[15] D. Soudry, I. Hubara, and R. Meir, "Expectation Backpropagation: parameter-free training of multilayer neural networks with continuous or discrete weights," pp. 963–971, 2014. [Online]. Available: https://dl.acm.org/citation.cfm?id=2968934

[16] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," *ArXiv:1602.02830*, 2 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[17] P. Merolla, R. Appuswamy, J. Arthur, S. K. Esser, and D. Modha, "Deep neural networks are robust to weight binarization and other non-linear distortions," *ArXiv:1606.01981*, 6 2016. [Online]. Available: http://arxiv.org/abs/1606.01981

[18] Y. Zhou, S.-M. Moosavi-Dezfooli, N.-M. Cheung, and P. Frossard, "Adaptive Quantization for Deep Neural Network," *Proceedings of AAAI*, 2018. [Online]. Available: http://arxiv.org/abs/1712.01048

[19] S. Khoram and J. Li, "Adaptive Quantization of Neural Networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=SyOK1Sg0W

[20] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *ArXiv:1503.02531*, 3 2015. [Online]. Available: http://arxiv.org/abs/1503.02531

[21] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," *ArXiv:1802.05668*, 2 2018. [Online]. Available: http://arxiv.org/abs/1802.05668

[22] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing Neural Networks with the Hashing Trick," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, pp. 2285–2294. [Online]. Available: http://arxiv.org/abs/1504.04788

[23] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," *International Conference on Learning Representations (ICLR)*, 2016.

[24] J. H. Holland, "Genetic Algorithms," pp. 66–73, 1992. [Online]. Available: https://www.jstor.org/stable/24939139

[25] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing." *Science (New York, N.Y.)*, vol. 220, no. 4598, pp. 671–80, 5 1983. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17813860

[26] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51, 1 1985. [Online]. Available: http://link.springer.com/10.1007/BF00940812

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[28] Michael Guerzhoy and Davi Frossard, "AlexNet implementation + weights in TensorFlow," 2016. [Online]. Available: https://www.cs.toronto.edu/ guerzhoy/tf_alexnet/