2017 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2017, Cancun, Mexico, December 4-7, 2017

Consideration on Quantization Functions in Quantized Neural Networks

Takumi Kadokura†, Hidehiro Nakano† and Arata Miyauchi†

†Faculty of Knowledge Engineering, Tokyo City University
1–28–1 Tamazutsumi, Setagaya-ku, Tokyo 158-8557, Japan
Email: g1681511@tcu.ac.jp, hnakano@tcu.ac.jp, miyauchi@ic.cs.tcu.ac.jp

Abstract— Quantized Neural Network (QNN) is a kind of neural networks in which its weights and activations are quantized. Since QNN can reduce computational quantity and energy consumption by quantization, it is expected to be used on embedded devices. This paper investigates quantization functions used for quantizing gradients in QNN. By performing the numerical experiments, the performances of some quantization functions are compared. We then show that there exists a quantization function which can keep high performance with low quantization bit rate.

1. Introduction

Microprocessors have long been embedded in various devices around us. Of late years, IoT (Internet of Things), the idea of connecting such devices to the Internet and allowing them to autonomously manage by exchanging information, became widely known. Apart from that, along with improvement of processor's capability, the methodology of machine learning using multi-layer neural networks called deep learning has been proposed and applied research such as speech recognition [1], scene recognition [2], and machine translation [3] has developed markedly. The development of these applied research is covered as the development of "Artificial Intelligence (AI)" by media, and the 3rd AI boom has come.

These two technological developments are often used in combination, and there are applications in which IoT senses the data and AI interprets the data for controlling other devices. However, multi-layer neural networks used with deep learning require a large number of product-sum operations of floating point number for forward (and backward) propagation. Therefore, in current common hardware, the use of multi-layer neural networks is very inefficient. In many application areas of multi-layer neural networks, in order to accelerate forward and backward propagation, GPUs (Graphics Processing Units) which are capable of performing these product-sum operations at high speed are used [4]. Nevertheless, for IoT devices which require low power consumption and low cost, it is not realistic to use GPUs with high power consumption and large heat generation.

In attempting to improve the efficiency of multi-layer neural networks toward such devices, a general technique is to compress trained networks [5]. In this paper, however, we focus on Quantized Neural Network (QNN) which reduces product-sum operations by quantizing parameters [6]. By using QNNs, it is expected to reduce the amount of calculation and accelerate the computation in forward propagation. Moreover, it is possible to quantize gradients which was the study subject in Binarized Neural Network (BNN) [7], the conventional method of QNN, and it can lead to reducing the amount of calculation in backward propagation. However, because there seem to be some unclear points in the description of the quantization function of QNN [6], the actual function used for experiments is unknown. Accordingly, we consider some quantization functions to quantize gradients, and find the function which can keep high performance with low quantization bit rate. We also find the relation between error rates on the test set and quantization errors. We show the results of numerical experiments.

2. Binarized Neural Network (BNN) [7]

BNN is a kind of neural networks in which its weights and activations (inputs of activation functions) are binarized to 1 bit. In BNN, while its activations are always binarized, its weights are only binarized at the time of forward propagation. The binarization is defined as follows:

$$\text{binarize}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{otherwise.} \end{cases} \tag{1}$$

While the stochastic binarization using random numbers is defined in [7], in this paper, we use the deterministic binarization of Eq. (1).

2.1. Gradients of Binarization Function

From the definition, the derivative of the binarization function binarize(x) is 0 almost everywhere. Therefore, the binarization function is not compatible with Stochastic Gradient Descent (SGD). Accordingly, by using the similar technique of straight-through estimator [8], we ignore the effect of binarization on gradients. However, taking the saturation effect of binarization into consideration, the gradients are set to 0 outside $[-1, +1]$. The derivative of the binarization function is defined in this manner as follows:

$$\frac{d}{dx}\text{binarize}(x) = \begin{cases} 1 & \text{if } -1 \leq x \leq +1 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

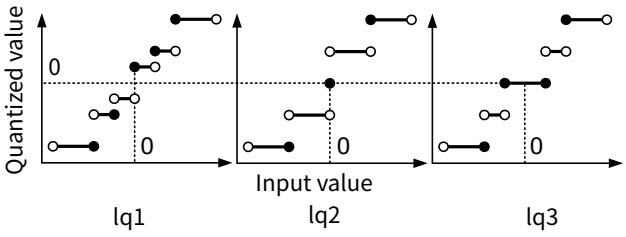

Figure 1: 3 quantization functions to be considered

2.2. Binarization of Weights

Above process is applied to all binarization. For binarization of weights, in addition, the real-valued weights are clipped to $[-1, +1]$ at the time of updating. This is to ensure that real-valued weights do not become too large despite having no influence on forward propagation.

3. Quantized Neural Network (QNN) [6]

QNN is the enhanced method of BNN, which replaces binarization of BNN with quantization to multiple bits. As a result, QNN can be applied to recurrent neural networks [6] such as Long Short Term Memory [9] [10] and Gated Recurrent Unit [11]. Also, to accelerate backpropagation, QNN quantizes gradients of activations.

However, because there seem to be some unclear points in the description of the quantization function which is supposed to be obtained by extending the binarization function of BNN to logarithmic quantization similar to the LogQuant function [12], the actual quantization function used for numerical experiments is unknown.

4. Consideration of Quantization Functions

In this section, we consider quantization functions for quantizing gradients. First, we propose 3 functions based on the equation of logarithmic quantization in [12]. Note that

$$\mathrm{lqs}(x, w, s) = \mathrm{sign}(x) 2^{\mathrm{clip}(\mathrm{round}(\log_2|x|), -2^{w-1}+s, 0)},$$

$$\mathrm{sign}(x) = \begin{cases} +1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{otherwise,} \end{cases}$$

$$\mathrm{clip}(x, l, u) = \begin{cases} u & \text{if } x > u \\ x & \text{if } l \le x \le u \\ l & \text{otherwise,} \end{cases}$$

$$\mathrm{round}(x) = \mathrm{sign}(x)\lfloor |x| + 0.5 \rfloor.$$

- A function obtained by simply extending binarization of BNN to logarithmic quantization:

$$\mathrm{lq1}(x, w) = \begin{cases} \mathrm{lqs}(x, w, 1) & \text{if } x \neq 0 \\ 2^{-2^{w-1}+1} & \text{otherwise.} \end{cases} \quad (3)$$

- A function obtained by adding a representation of 0 to lq1:

$$\mathrm{lq2}(x, w) = \begin{cases} \mathrm{lqs}(x, w, 2) & \text{if } x \neq 0 \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

- A function obtained by setting the wider range for 0 compared with lq2:

$$\mathrm{lq3}(x, w) = \begin{cases} \mathrm{lqs}(x, w, 2) & \text{if } |x| > 2^{-2^{w-1}+1.5} \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

Here, the parameter w is the number of quantization bits. The 3 quantization functions are illustrated in Fig. 1.

Furthermore, we consider quantization of gradients using the optimal quantization as a criterion of performance comparison. In order to perform optimal quantization on the gradients of activations, it is needed to obtain quantization representative values. We use the following method:

1. We train a BNN with the settings shown in Table 1 for 1000 epochs.
2. At this time, by randomly sampling one gradient of activation in each hidden layer, 50,000,000 pieces of gradient information (large gradient information) are obtained for each hidden layer by the end of training.
3. From the large gradient information, we randomly extract 500,000 pieces of gradient information (small gradient information) for each hidden layer with no duplication.
4. From the small gradient information, we search the quantization representative values that minimize the quantization error using k-means++ [14] with $k = 2^w$.

5. Experiments and Consideration

We apply the proposed 3 quantization functions and the optimal quantization to quantization of gradients of QNN and measure the performance using a benchmark.

As a benchmark, we use MNIST [13] which is a handwritten digit recognition data set. Each image of MNIST consists of 28×28 pixels, and each pixel is a gray scale of 256 gradations. MNIST has 10 classes of 0–9, and contains 50,000 samples for training, 10,000 samples for validation (data set for preliminary measuring classification rate during training), and 10,000 samples for testing, totaling 70,000 samples. In this paper, since the benchmark is a classification problem, we use the classification error rate as an index of performance. In addition, the number of quantization bits for quantizing gradients is varied to 4, 5, 6, and we train the network for 1000 epochs. Further, we compute quantization errors for each quantization function for the large gradient information obtained by the above method.

Table 1: Experimental parameters

Size of mini-batch		100
Adam [15]	Learning rate	Exponentially decaying from 3.0E−3 to 3.0E−7
	β_1	0.9
	β_2	0.999
	ϵ	1.0E−08
Batch Normalization [16]	α	0.1
	ϵ	1.0E−04
Number of neurons		$784 = 28 \times 28, 4096, 4096, 4096, 10$
Probability of Dropout [17]	Input layer	0.2
	Hidden layers	0.5
Activation function		binarize$(\cdot)$
Weight learning rate scaling factor		$1/\sqrt{1.5/(\text{Number of input neurons + Number of output neurons})}$

Experimental parameters are shown in Table 1. We test the proposed 3 quantization functions on 2 different loss functions: cross entropy loss function and squared hinge loss function. Note that we realize quantization by replacing the input value with the nearest quantization representative value.

5.1. Cross Entropy Loss Function

Experimental results for cross entropy loss function are shown in Fig. 2 (upper). lq2 has similar quantization errors as lq1, but its performance has improved. As compared with lq1, lq2 can represent 0, but the quantization around 0 is rough. However, the roughness is unlikely to directly contributes to performance improvement. On the other hand, in gradients of activations, we obtained the result that the frequency of 0 is an order of magnitude higher from the preliminary experiment, and Dropout [17] uses gradients of 0. The ability to represent 0 is important in quantization of gradients, and therefore performance has improved.

lq3 is better for both quantization errors and performance than lq2. lq2 is slightly better than lq3 in 5 bit, but the difference is less than 1%, which is the effect of random numbers.

lq3 is a comparable performance to the the optimal quantization. Since its quantization errors are lower than other functions, this is because quantization with less information loss can be achieved. However, since lq3 is inferior in performance and quantization errors to the optimal quantization in 4 bit, we should consider the range quantized to 0.

5.2. Squared Hinge Loss Function

Experimental results for squared hinge loss function are shown in Fig. 2 (lower). Like the case of cross entropy, the quantization functions that can handle 0 (lq2, lq3) are good performance. However, unlike the case of cross entropy, lq2 with the narrow range quantized to 0 is better. Therefore, the range quantized to 0 in the quantization for better performance may depend on the loss function. Besides, there are some cases where the optimal quantization shows worse performances than the proposed 3 quantization functions. This suggests that indexes other than the quantization error are necessary to optimize the quantization of gradients in squared hinge loss.

6. Conclusion

In this paper, we consider the quantization function which realizes appropriate quantization of gradients in QNN. We show that lq3 is appropriate if the loss function is cross entropy, and lq2 is if it is squared hinge. We found that in both cases the function that can quantize to 0 is high performance. Future works include detailed search of the range quantized to 0, investigation of the contribution of the quantization error to the performance in the squared hinge loss, and search for the optimal quantization in different parameters.

References

[1] D. Amodei, et al., "Deep speech 2 : end-to-end speech recognition in English and Mandarin," Proc. Machine Learning Research, vol.48, pp.173–182, New York, USA, June, 2016.

[2] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, Canada, Dec., 2014.

[3] Q. V. Le and M. Schuster, "Research Blog: A Neural Network for Machine Translation, at Production Scale," Google Inc., https://research.googleblog.com/2016/09/a-neural-network-for-machine.html, Sept., 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Resid-

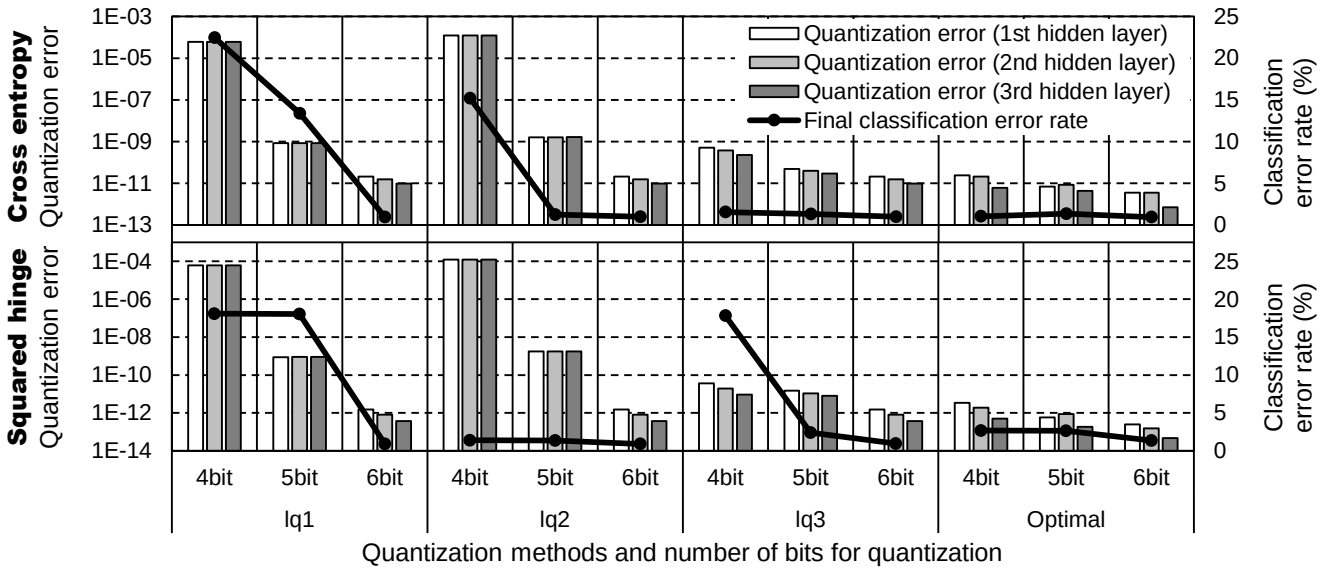

Figure 2: Experimental results

ual Learning for Image Recognition," ArXiv e-prints 1512.03385, Dec., 2015.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding," ArXiv e-prints 1510.00149, Oct., 2015.

[6] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: training neural networks with low precision weights and activations," ArXiv e-prints 1609.07061, Sept., 2016.

[7] 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 e-prints 1602.02830, Feb., 2016.

[8] Y. Bengio, "Estimating or propagating gradients through stochastic neurons," Technical Report, University of Montreal, ArXiv e-prints 1305.2982, May, 2013.

[9] S. Hochreiter and J. Schmidhuber, "Long short term memory," Technical Report FKI-207-95, Technische Universität München, Aug., 1995.

[10] S. Hochreiter and J. Schmidhuber, "Long short term memory," Neural Computation, vol.9, no.8, pp.1735–1780, Dec., 1997. DOI:10.1162/neco.1997.9.8.1735

[11] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," ArXiv e-prints 1412.3555, Dec., 2014.

[12] Daisuke Miyashita, Edward H Lee, and Boris Murmann, "Convolutional neural networks using logarithmic data representation," ArXiv e-prints 1603.01025, Mar., 2016.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol.86, no.11, pp.2278–2324, Nov., 1998.

[14] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding," Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp.1027–1035, New Orleans, USA, Jan., 2007.

[15] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," International Conference on Learning Representations, San Diego, USA, May, 2015.

[16] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," Proc. 32nd International Conference on Machine Learning, vol.37, pp.448–456, Lille, France, Jul., 2015.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural network from overfitting," Journal of Machine Learning Research, vol.15, no.1, pp.1929-1958, Jan., 2014.