An Improving Method for Performance of DNN using 2-bit Quantization

Sang Hyeok Kim and Jae Heung Lee*

Department of Computer Engineering, Hanbat National University, 125, Dongseo-daero, KS015, Korea.
*Corresponding author. Tel.: +82-010-3404-6954; Email address: jhlee@hanbat.ac.kr

Article History:Received:11 november 2020; Accepted: 27 December 2020; Published online: 05 April 2021

Abstract : Background/Objectives: Recently, interest in AI(Artificial Intelligence) has increased, and many studies are being conducted to enable AI to be used in embedded and mobile environments. Among them, quantization is one of the methods to reduce the size of the model, and most quantization of less than 8 bits cannot be implemented without additional hardware such as FPGA. With this in mind, in this paper, we propose two new algorithms that can implement 2bit quantization in software.

Methods/Statistical analysis: In this paper, we propose a packing operation that quantizes a weight consisting of 32-bit real values into 2 bits, stores four 2-bit quantization weights in one 8-bit memory, and a Masking Matrix Multiplication function that performs the calculation of the packed weight and input values. These functions operate in parallel in the GPU memory.

Findings: The quantization model using the above function showed about 16 times more memory saving and 4 times faster when comparing the operation with the existing 32bit model. Nevertheless, the DNN model showed an error of around 1% in learning using MNIST and HandWritten data, and the CNN model showed an error of around 1% in learning using EEG (Electroencephalograpy) data.

Improvements/Applications: The function used in this study is focused on the domain of DNN, and although extended to CNN, quantization could be performed only in the FC (Fully Connected) part. To apply to the convolution layer, an additional function is required, and it is necessary to check whether the difference in accuracy is small even in a more complex data set in the future.

Keywords: AI, Embedded, Convolution Neural Network, Quantization, Deep Neural Network.

1. Introduction

As the hardware advances, the field of AI has also undergone remarkable development. Various companies and research institutes have conducted various learning using high-performance PCs, and the learning has shown surprising results in succession. However, in recent years, research to learn in an environment where resources such as mobile[1] and embedded are limited rather than a high-performance PC environment are increasing. With the creation of chips for AI computation and the release of accelerators using USB, the era of AI can be applied to mobile phones is coming. In such an environment, the existing 32bit model and matrix multiplication operation cannot be used because it requires a lot of resources. Therefore, processing such as reducing the number of bits [2-4] or reducing the operation [5] through quantization is necessary. The current software environment does not support quantization of less than 8 bits, and additional hardware such as FPGA must be used for this[6-10].

In this paper, we implemented a packing operation in which the weight of the model is quantized to 2 bits or less in a software environment, and four weights are stored in one 8-bit memory to reduce overhead. In addition, masking matrix multiplication operation is implemented for the operation of packed data and input values, and the operation for packing and masking matrix multiplication operation is as follows.

***Corresponding author:** Jae Heung Lee
Department of Computer Engineering, Hanbat National University, 125, Dongseo-daero, KS015, Korea.
Email address: jhlee@hanbat.ac.kr

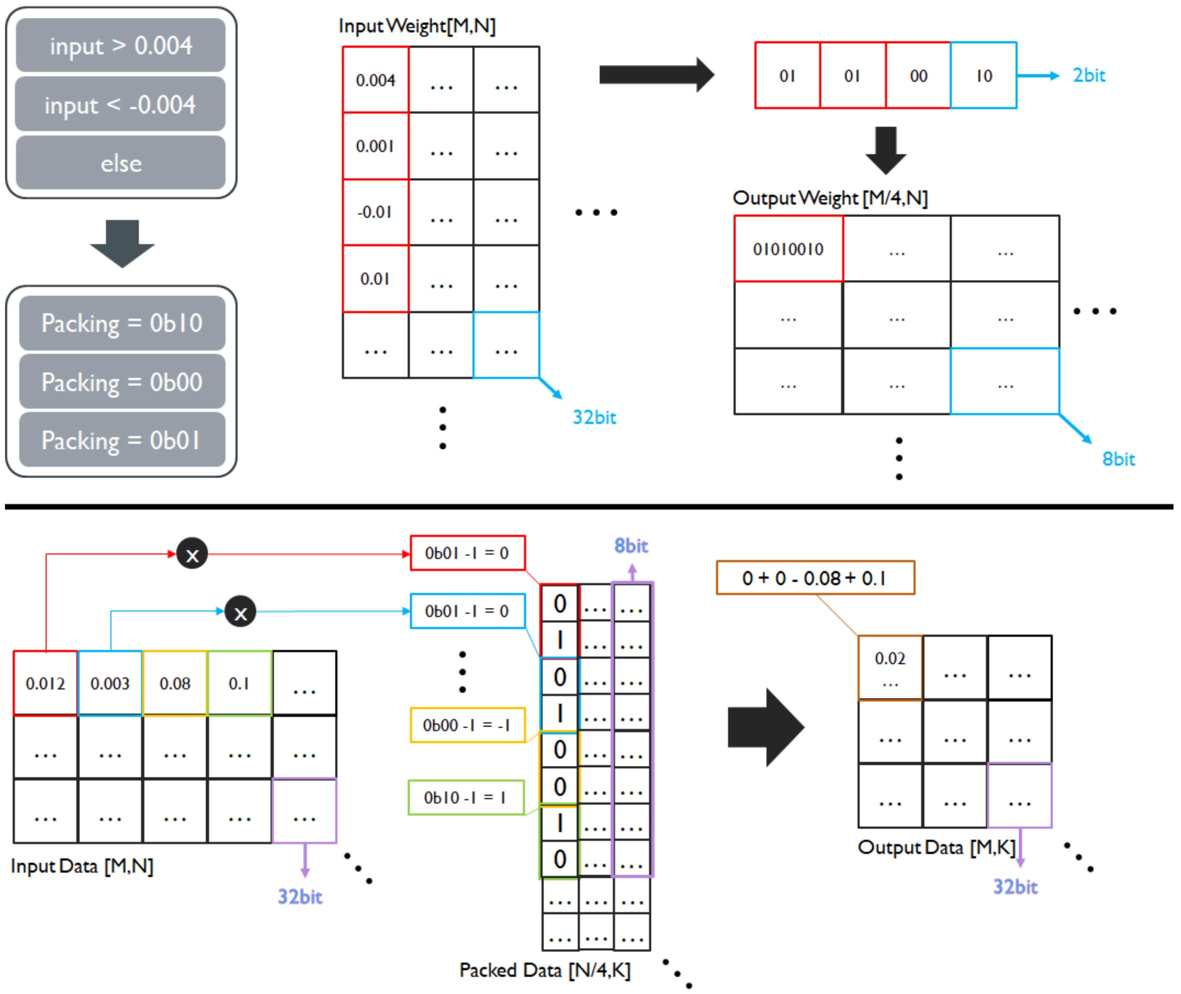

Figure 1 Packing Function (Top) Masking Matrix Multiplication Function (Bottom) operation

2. Materials and methods

2.1 Packing Function

Packing operation quantizes 32-bit data into 2 bits, and stores them in a big-endian format in an 8-bit memory through bit operation. The quantization method took a threshold comparison method, and the threshold value was 0.004 in this paper. The reason for using 0.004 is to find a region that equally divides it into 3 equal parts because the standard deviation of the normal distribution of random values is set to 0.01 and the mean value is 0. If the weight value is less than -0.004, quantization is performed with -1, when it is greater than 0.004, quantization is performed with 1, and otherwise, quantization is performed with 0. As mentioned above, 4 input values are divided into 2 bits and stored in one 8-bit memory, so if the input value has the size of [M, N], the output value has the size of [M/4, N]. The detailed operation for this is shown in the figure below.

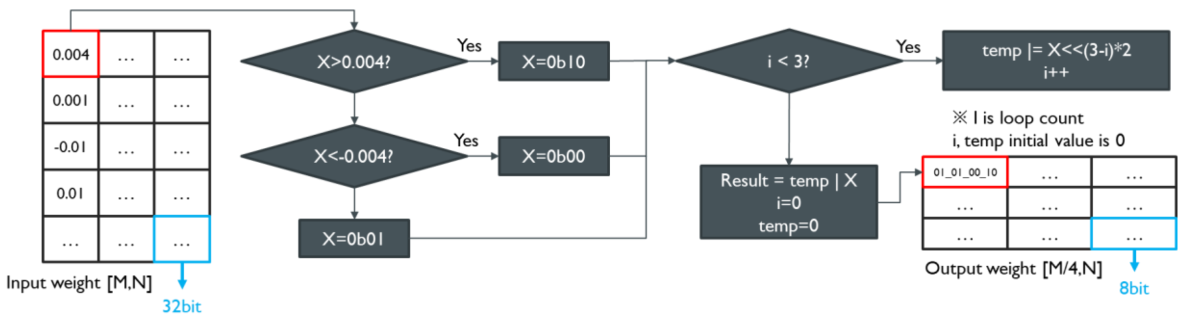

Figure 2 Packing Function Detail Algorithm

2.2 Masking Matrix Multiplication Function

The Masking Matrix Multiplication operation is a function created to solve the calculation of the above packed data and input values. Since the packed data contains 4 weight values in an 8-bit memory, it is necessary to connect 4 weights and 4 input values to calculate them. The process was carried out collectively through parallel operation, and the function for selecting the required weight value from the 8-bit memory was solved using bit operation. The detailed algorithm for this is as follows.

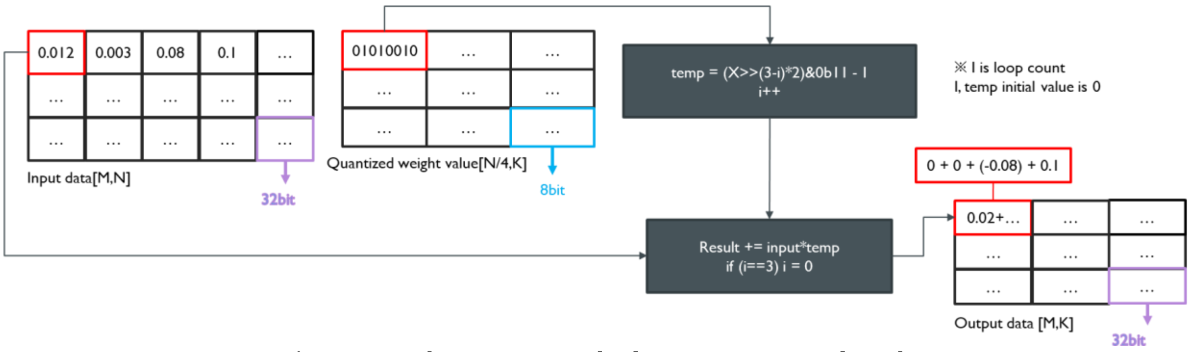

Figure 3 Masking Matrix Multiplication Function Algorithm

3. Results and Discussion

To test the results of this paper, two models were created, DNN and CNN, and the DNN was tested by learning the positive and negative emotions of humans using the MNIST and HandWritten data sets, and the CNN using the EEG data set. First, the environments where DNN was tested are desktop and Linux, and the results of this environment configuration and time are shown in the table below.

Table1 Desktop Environment

Type	Contents
OS	Windows 10 Pro 64 bit
CPU	Intel i7-6700K @ 4.00GHz
GPU	NVIDIA GeForce GTX 1060 3GB
RAM	32GB

Table2 Embedded Devices Environment

Type	Contents
OS	Ubuntu 18.04 LTS
CPU	Quard-core Arm Cortex-A57
GPU	128 NVIDIA CUDA core Maxwell
RAM	4GB 64bit LPDDR4

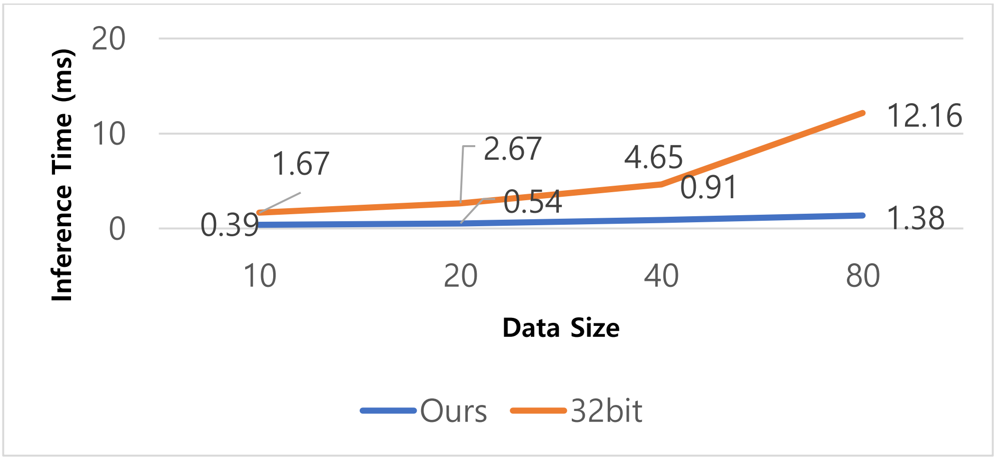

Figure 4 Time measurement in a desktop environment

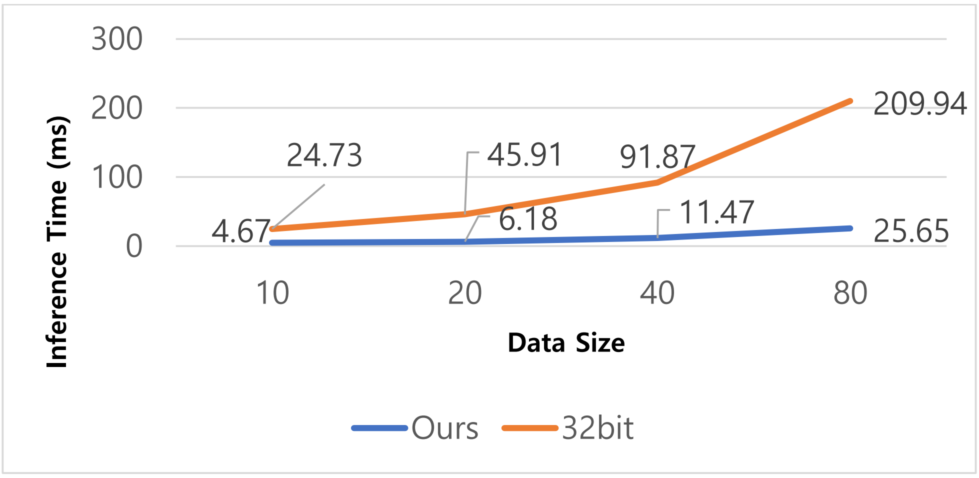

Figure 5 Time measurement in an embedded environment

In the case of CNN, the FC layer portion was quantized and learned, and the input value was used as the result of flattening after convolution. Since the convolution area calculation is required, the model was performed only in the desktop environment. The configuration of the model is shown in the table below.

Table3 CNN model configuration

Type	Filters	Size/Stride	Bit
Conv	32	5*5 / 1	32
Max Pool		3*3 / 1	
Conv	32	5*5 / 1	32
Max Pool		3*3 / 1	
Conv	64	3*3 / 1	32
Conv	64	3*3 / 1	32
Conv	64	3*3 / 1	32
Flatten			
FC		128 * 256	2
FC		64 * 2	2

The table comparing the overall memory occupancy and accuracy average of the above test is as follows.

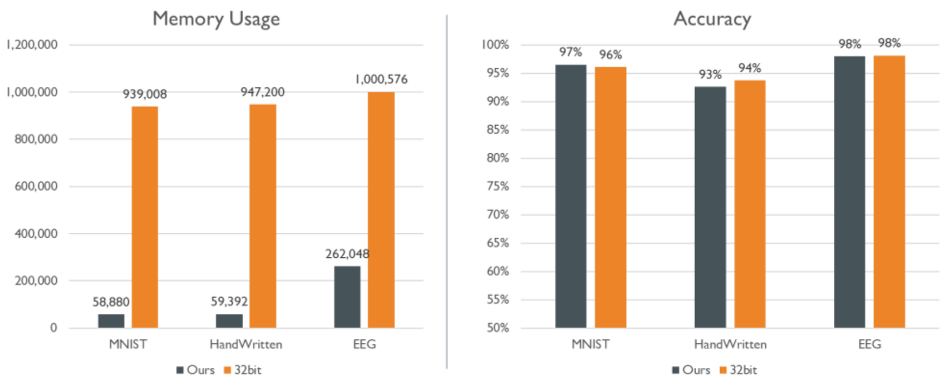

Figure 5 Memory Usage & Accuarcy

4. Conclusions

In this paper, a method of compressing and calculating a 32-bit model into 2 bits was proposed, and the operation was extended and applied to the FC layer of DNN and CNN and tested. The overall result is as follows. It is believed that if the quantization is carried out in the convolution layer in the future and the operation is improved in accordance with the learning of more complex data, better results will be achieved.

Table3 Overall Result

	MNIST (Ours / 32bit)	**HandWritten (Ours / 32bit)**
Model	DNN / DNN	DNN / DNN
Memory Usage	57.5MB / 917MB	58MB / 925MB
Accuarcy	96.5% / 96.1%	92.6% / 93.7%
Desktop Inference Time	0.39ms/1.67ms	0.4ms / 1.65ms
Embedded Inference Time	4.67ms / 24.73ms	4.82ms / 24.93ms
CNN	**Parameter / Memory Usage**	**EEG (%)**
2 bit	151,456 / 262,048	98.0 %
32 bit	250,144 / 1,000,576	98.1 %

5. References

1. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, “Mobilenets: Efficient convolutional neural networks for mobile vision applications”, arXiv preprint arxiv:1704.04861
2. Mi Sun Park, Xiaofan Xu, Cormac Brick, “SQuantizer: Simultaneous Learning for Both Sparse and Low-precision Neural Network” arxiv:1812.08301
1. Alexios Balatsoukas-Stimming, Oscar Castaneda, Sven Jacobsson, Giuseppe Durisi, Christopyh Studer, “Neural-Network Optimized 1-bit Precoding for Massive MU-MIMO”, arxiv:1903.03718
2. Matthieu Courbariaux, Yoshua Bengio, Jean-Pierre David, “Binary Connect: Training Deep Neural Networks with binary weights during propagations”, arxiv:1511.00363
3. Song Han, Huizi Mao, William J. Dally, “Deep Compression: Compressing Deep Neural Networks With Pruning, Trained Quantization And Huffman Coding”, arxiv 1510.00149
4. Hande Alemdar, Vincent Leroy, Adrien Prost-Boucle, Fredderic Petrot, “Ternary Neural Networks for Resource-Efficeint AI Applications”, arxiv:1609.00222
5. Md Zahangir Alom, Adam T Moody, Naoya Maruyama, Brian C Van Essen, Tarek M. Taha, “Effective Quantization Approaches for Recurrent Neural Networks”, arxiv:1802.02615
6. Jungwook Choi, Swagath Venkataramani, Vijayalakshmi Srinivasan, Kallash Gopalakrishnan, Zhuo Wang, Pierce Chuang, “Accurate And Efficeint 2-bit Quantized Neural Networks”, sysml.cc/doc/2019/168.
7. Bin Yang,n Lin Yang, Xiaochun Li, Wenhan Zhang, Hua Zhou, Yongxiong Ren, Yinbo Shi, “2-bit Model Compression of Deep Convolutional Neural Network on ASIC Enging for Image Retrieval”, arxiv:1905.03362
8. Jungwook Choi, Pierce I-Jen Chuang, Zhuo Wang, Swagath Venkataramani, Vijayalakshmi Srinivasan, Kailash Gopalakrishnan, “Bridging the Accuracy Gap for 2-bit Quantized Neural Networks(QNN), arxiv:1807.06964
9. Mallick, P. K., Ryu, S. H., Satapathy, S. K., Mishra, S., Nguyen, G. N., & Tiwari, P. (2019). Brain MRI image classification for cancer detection using deep wavelet autoencoder-based deep neural network. IEEE Access, 7, 46278-46287.
10. Satapathy, S. K., Mishra, S., Sundeep, R. S., Teja, U. S. R., Mallick, P. K., Shruti, M., & Shravya, K. (2019). Deep learning based image recognition for vehicle number information. International Journal of Innovative Technology and Exploring Engineering, 8, 52-55.