Using Quantization-Aware Training Technique with Post-Training Fine-Tuning Quantization to Implement a MobileNet Hardware Accelerator

Ching-Che Chung, *Senior Member*
Department of Computer Science and Information Engineering
National Chung Cheng University
Chia-Yi, Taiwan
wildwolf@cs.ccu.edu.tw

Wei-Ting Chen
Department of Computer Science and Information Engineering
National Chung Cheng University
Chia-Yi, Taiwan

Ya-Ching Chang
Department of Computer Science and Information Engineering
National Chung Cheng University
Chia-Yi, Taiwan

***Abstract*— In recent years, the internet of things (IoT) has been developed near the public's life circle. At the edge device, for real-time data analysis of data, a lightweight deep learning neural network (DNN) is required. In this paper, the lightweight model MobileNet is used to design an energy efficiency hardware accelerator at the edge device. In the software framework (Tensorflow), the quantization-aware training technique with post-training fine-tuning quantization is applied to quantize the model to improve training convergence speed and parameter minimization. In hardware design considerations, fixed-point operations can reduce computational complexity and memory storage space as compared to floating-point operations, which directly affects the power consumption of the circuit. The proposed MobileNet hardware accelerator can achieve low power consumption and is suitable for the edge devices.**

Keywords*— *MobileNet, model parameter quantization, hardware accelerator

I. INTRODUCTION

Last few years, fast hardware, such as GPU are developed, the development of fast hardware makes the deep neural networks (DNNs) can be trained, and various network architectures are proposed for many applications. For small device applications, NasNet, ShuffleNet, MobileNet, and MobilenetV2 are created. Those models [1–3] reduce hardware resources, and these models can solve various recognition problems with mobile devices in daily life.

The number of parameters of the neural network model is directly related to the circuit complexity and memory space/bandwidth requirement, so MobileNet-v2 is used in this paper. The depthwise convolution layer and the global average pooling layer [1, 2] are beneficial to reduce parameters. Also, memory access times are reduced by adjusting the order of data access.

TensorFlow is useful in machine learning, and the TensorFlow platforms own many proven functions on various neural networks, which can significantly reduce development time. In this paper, the software implementation of TensorFlow and DoReFa framework is with Tensorpack library [4–6]. DoReFa [4] is an effect uniform quantization method for quantization-aware training, and with post-training quantization, the accuracy of the neural network model can be improved. The rest of this paper is organized as follows. The MobieNet with quantization-aware training and post-training quantization is presented in Section II. Section III describes the implementation of the proposed hardware accelerator. Section IV shows the experimental results. Finally, the conclusion is given in Section V.

This work was supported in part by the Ministry of Science and Technology of Taiwan under Grant MOST-108-2221-E-194-051- and was financially/partially supported by the Advanced Institute of Manufacturing with High-tech Innovations (AIM-HI) from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

II. MOBILENET QUANTIZATION

In this paper, the hardware accelerator for the convolution neural network is implemented to recognize ten different objects on CIFAR-10 dataset. The CIFAR-10 dataset is composed of 50,000 training images and 10,000 test images. Single image size is 32x32, and one class has 6,000 images. Also, the proposed network architecture is based on MobileNetV2, as shown in Fig. 1. The network architecture includes the zero-padding layers, a convolution layer of 3x3 kernel, the batch normalization layers, the activation layers, the bottleneck layers, the convolution layers of the 1x1 kernel, a global average pooling layer, and a fully connected layer. Particularly, the zero-padding layers, the depthwise convolution layers of the 3x3 kernel, the convolution layers of 1x1 and 3x3 kernel, the batch normalization layers, and the activation layers are included in the bottleneck layer.

Some efforts had been made in the training and inference processes. The first step is to apply a lightweight MobileNet architecture. Because the concept of depthwise convolution is used, the number of parameters can be less than using the

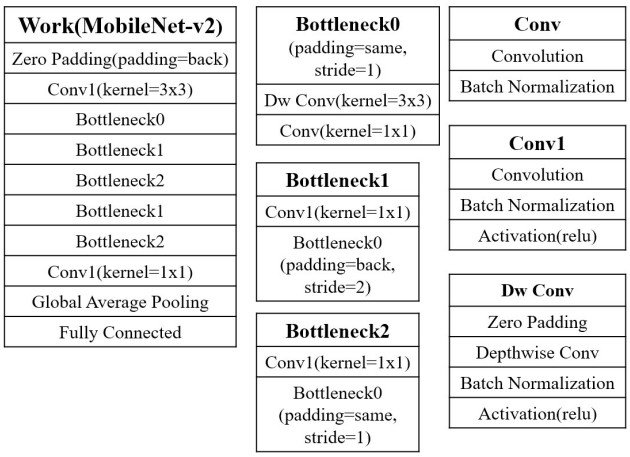

Fig. 1. Proposed network architecture based on MobileNet

978-1-7281-4999-8/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: National Chung Cheng University. Downloaded on September 03,2020 at 06:05:47 UTC from IEEE Xplore. Restrictions apply.

general convolution layer. In consideration of the memory usage and the power consumption of the hardware design, fine-tuning of the parameters can reduce the circuit complexity with acceptable loss to the classification accuracy. The second step is to train this model with uniform quantization method by DoReFa [4], and this quantization method is useful for reducing the bit-width of weights. After the training is finished, the classification accuracy will be verified during inference process. If the classification accuracy is low, network architecture will be modified until the accuracy is acceptable.

In the third step, after the training process, parameters will be stored in the format of TensorFlow, and then the Python code is written to extract the parameters of the model into a text file. In the extracted parameters, the four parameters of the batch normalization is reduced to two parameters by simplifying the calculation formula, and then, parameters associated with batch normalization are reduced by half. In the fourth step, parameters associated with batch normalization are grouped by the linear transform to further reduce the model size. To reduce the parameter quantity and accuracy loss, the number of groups will be determined by the set number of bits.

The fifth step is to restored the saved model and quantize with DoReFa again. Because the weights of the model will be limited to the quantization interval of -1 to 1, before the inference process weights should be returned to the same value interval. Otherwise, the entire model will not match. The sixth step is to perform the fixed-point processing of the weight values so that the accuracy of the inference process can be closer to the result of the hardware implementation. After the fixed-point processing is completed, the classification accuracy is verified. If the accuracy is low, the number of groups are increased in the clustering process in fourth step. Finally, the network architecture is implemented as a hardware accelerator.

The method of parameter quantization can be divided into two classes, which are quantization-aware training and post-training quantization. The quantization-aware training is to quantize the parameters while training. In the DoReFa quantization [4], weights, activations, and gradients are quantized. The memory requirement for the hardware is mainly considered in this paper, so only the weights are quantized. In the weight quantization formula, values pass a tanh function, and output values are between -1 to 1, as shown in Eq. 1.

$$r(x_i) = \tanh(x_i) \tag{1}$$

Subsequently, values are biased and normalized to remove the negative values, as shown in Eq. 2.

$$s(r_i) = \frac{r_i}{2\max(|r_i|)} + \frac{1}{2} \tag{2}$$

Finally, these values are quantized according to the bit width k using the uniform quantization method, as shown in Eq. 3.

$$t(s_i) = 2.\frac{1}{2^k - 1}\,\text{round}\;s((2^k - 1)s_i) - 1 \tag{3}$$

In the quantization-aware training, only the weights are quantized, and weights are quantized to the interval of -1 to 1. The parameters of the batch normalization are not quantized by the DoReFa method but are quantized in post-training quantization.

In this paper, two methods, K-means algorithm and linear transform are applied in post-training quantization for comparison. K-means algorithm uses the cluster center method to classify all values into different group. Then, the values grouped in the same group are replaced by the center value of the group. In this paper, the linear transform method is presented. First, the maximum and minimum values are found, and then all values are transferred to the interval of 0 to 1, as shown in Eq. 4.

$$p(x_i) = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)} \tag{4}$$

If the bit width is l, from 0 to 1, there are 2l subintervals. The values transferred into the same subintervals are averaged. Subsequently, all values in the same subinterval are replaced with the averaged value. Finally, the values are restored to the original interval by Eq. 5.

$$q(p_i) = p_i \times [\max(x_i) - \min(x_i)] + \min(x_i) \tag{5}$$

In this paper, K-means algorithm and linear transform are applied to quantize four parameters of the batch normalization layer, including mean, variance, beta, and gamma. These parameters are further reduced to two parameters, as shown in Eqs. 6, 7, and 8. The reduced operation requires a multiplication and an addition, which can reduce half of the memory space requirements and also reduce computing resources.

$$\mu^o \leftarrow -\left(\frac{1}{\sqrt{a_B^2}}\mu_B\gamma\right) + \beta \tag{6}$$

$$a^o \leftarrow \frac{1}{\sqrt{a_B^2}}\gamma \tag{7}$$

$$y_i \leftarrow a^o x_i + \mu^o \tag{8}$$

III. Proposed Hardware Accelerator

Fixed-point arithmetic operations are performed in the proposed hardware accelerator. The pixel values of the image of the CIFAR-10 dataset are divided by 256 and expressed in 1 signed bit and 8 decimal bits format. Weights of the network model are also expressed in 1 signed bit and 8 decimal bits format. The parameters of the batch normalization layer are expressed in 1 signed bit, 3 integer bits, and 8 decimal bits format. The decimal bits require 8 bits to avoid loss of the accuracy.

In the implementation of the proposed neural network, the main three layers are an input layer, hidden layers, and an output layer. The Input layer is shown in Fig. 2, the block memory (BRAM) contains the images RAM and output feature maps. Images RAM stores CIFAR-10 images, and the output feature map compute unit calculates the partial sum, and then the output feature maps are stored in the BRAM as the input to the next layer. In the Input layer, the zero-padding operation will be performed. I_data gives four pixel values of the image at each clock cycle, and then the

Authorized licensed use limited to: National Chung Cheng University. Downloaded on September 03,2020 at 06:05:47 UTC from IEEE Xplore. Restrictions apply.

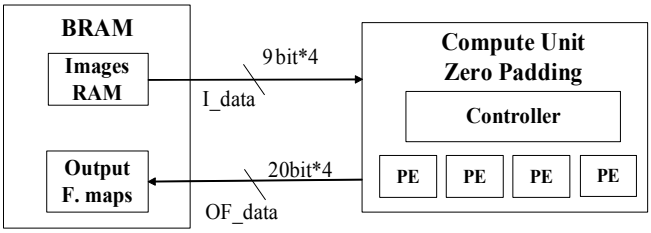

Fig. 2. Input layer of the proposed neural network

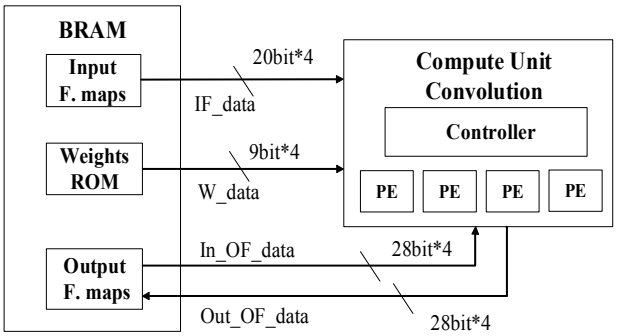

Fig. 3. Hidden layer of the proposed neural network

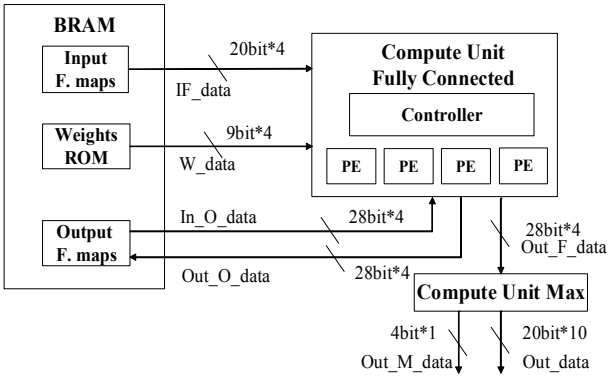

Fig. 4. The output layer of the proposed neural network

compute unit uses four PEs to process these data. Then, the OF_data which contains four data and are stored in the output feature map. Also, the value in the output feature map is expressed in 1 signed bit, 6 integer bit, and 13 decimal bits format. The number of integer bits are determined to avoid overflow. Also, the number of decimal bits are determined by the classification accuracy requirement and the accuracy should be close to the floating-point arithmetic operations.

In the hidden layer, for example, the convolution layer, the BRAM contains input feature maps, the weight ROM, and output feature maps. The input feature maps are the output feature maps of the upper layer. Weight ROM stores the weights of the trained neural network, and the output feature maps are calculated by the compute unit and then stored in the output feature maps for the next layer, as shown in Fig. 3.

In the hidden layer shown in Fig. 3, IF_data gives four values of the input feature map at each clock cycle, and W_data also gives four weights of the filter at each clock cycle. Then the compute unit processes convolution layer with four PEs. Subsequently, the convolution layer needs to store partial sum for the next calculation. Out_OF_data outputs four partial sum data to the output feature maps at each clock cycle. The compute unit inputs required partial sum data from In_OF_data for the convolution operation. Also, the value in the output feature map is expressed in 1 signed bit, 11 integer bit, and 16 decimal bits format. The number of integer bits are determined by partial sum accumulation to avoid overflow. Also, the number of decimal bits are determined by the classification accuracy requirement and the accuracy should be close to the floating-point arithmetic operations.

In the output layer shown in Fig. 4, the fully connected operation will be performed. The BRAM contains input feature maps, weight ROM, and output ROM. The output layer has the same behavior as the hidden layer in IF_data, W_data, In_OF_data, and Out_OF_data, and the choice of the bit size is also the same. Differently, in the output layer, the 28-bit value of 10 classes will output to the compute unit max. In compute unit max, the maximum value of the 10 classes will be found and output to Out_M_data, the 20-bit maximum value will also output to Out_data.

TABLE I. TOTAL PARAMETERS IN EACH LAYER.

Work(MobileNetV2)	Weight	Batch Normalization (2 parameters)
Zero Padding(back)	0	0
Conv1(3x3)	6,912	512
Bottleneck0	16,640	624
Bottleneck1	35,280	1,424
Bottleneck2	21,360	1,040
Bottleneck1	25,200	1,072
Bottleneck2	40,656	1,456
Conv1(1x1)	28,672	1,024
Global Average Pooling	0	0
Fully Connected	5,120	0
Sum. parameters	179,840	7,152
Total parameters	186,992	

TABLE II. ACCURACY IN DIFFERENT BIT-WIDTH IN WEIGHT QUANTIZATION

Bit-width / bit of fixed-point	DoReFa	K-means by layer	Linear transform by layer
	to fixed-point	*to DoReFa to fixed-point*	
8/8bit	0.8505	0.8605	0.8454
4/4bit	0.7388	0.5147	0.8236
4/8bit	0.8448	0.8189	0.826
2/2bit	0.1012	0.1	0.1489
2/4bit	0.6338	0.2536	0.6575
2/8bit	0.8095	0.6453	0.7478

TABLE III. ACCURACY ANALYSIS IN BATCH NORMALIZATION PARAMETERS QUANTIZATION

Bit-width : Batch normalization	DoReFa	Linear transform by layer
	to fixed-point	*to DoReFa to fixed-point*
Linear transform		
2	0.2144	0.2006
4	0.8085	0.8128
8	0.8452	0.8274
K-means		
2	0.4046	0.3843
4	0.8397	0.8287
Linear transform & K-means		
2	0.435	0.4537
4	0.8536	0.8418

Authorized licensed use limited to: National Chung Cheng University. Downloaded on September 03,2020 at 06:05:47 UTC from IEEE Xplore. Restrictions apply.

TABLE IV. PERFORMANCE COMPARISONS

	[7] ISSCC'16	[8] VLSI'17	[9] JSCC'18	[10]IBM TrueNorth	Our work
Technology	65nm	65nm	40nm	28nm	40nm
Algorithm	CNN	DNN	CNN/RNN	CNN	CNN
Architecture	AlexNet	N/A	AlexNet	TrueNorth	MobileNet
Dataset	MNIST/ CIFAR-10	MNIST	CIFAR-10	CIFAR-10	CIFAR-10
Parameters(k)	1071	1006	1071	N/A	187
Supply(V)	1.2	0.55~1.0	1.1	1.0	1.0
Frequency (MHz)	125	100~400	300	N/A	200
Precision (bits)	16-bit	Binary	4-bit	Binary	4-bit
Power(mW)	45	50-600	2083	204.4	36.75
Gate Count	3.2M	N/A	N/A	N/A	0.44M
Memory	on-chip 36KB	on-chip 0.8MB	on-chip 7.68MB off-chip 96MB	on-chip 51.2MB	ROM 0.42MB RAM 34.48MB
Accuracy (TOP1)	N/A	90.1%	76.7%1	83.41%	85.1%
					99.2%1

[1]Accuracy (TOP 5).

IV. EXPERIMENTAL RESULT

Table I shows the total number of parameters in each layer in the proposed neural network based on MobileNetV2, and the proposed design is to recognize ten different objects on CIFAR-10 dataset. The quantization-aware training and post-training quantization are mentioned in Section II. The bit width in weight quantization affects the classification accuracy of the proposed hardware accelerator. The higher bit width in weight quantization has relatively high accuracy than with lower bit width, as shown in Table II. Also the bit number for fixed-point number also affects the accuracy of the proposed hardware accelerator.

In Table II, "4/8bit" means the bit width for weight quantization is 4, and weight values are expressed in 8-bit fixed-point format. Therefore, with uniform quantization method by DoReFa [4], there are $16=(2^4)$kinds of weight values after training process, and 4 bits are required to save one weight value in memory. Subsequently, a lookup table with 2^4 entries is created, and in each entry, weight value is expressed in 8-bit format. "K-means by layer" and "Linear transform by layer" means that the K-means algorithm and linear transform method are applied for post-training weight quantization. Finally, 4/8bit configuration with DoReFa method is used in the proposed design.

In Table III, weight and batch normalization are quantized with different bit-width, and the classification accuracy is compared between the linear transform, K-means algorithm, and mixed quantization. Bit width for weight and batch normalization parameter quantization are at least 4 bits to minimize the loss of accuracy.

The proposed hardware accelerator is implemented in TSMC 40nm CMOS process. Memory compiler is used to generate the RAMs and ROMs for the proposed design. The power consumption of the proposed hardware accelerator with power-gating of unused memories is 20.46mW for logic gates, 1.41mW for ROMs, and 14.88mW for RAMs, and total power consumption is 36.75mW at 200MHz.

Table IV compares other neural network hardware accelerators for the same CIFAR-10 dataset (except [8]). The proposed network architecture is based on MobileNetV2, and the lightweight network can have much smaller parameters than the other architecture but still have sufficient accuracy. With fewer parameters, the memory space can be greatly reduced. Also, the classification accuracy of the proposed neural network accelerator has better than [9] and [10]. Finally, the energy efficiency of the proposed design is 10.65 (GOPS/W).

V. CONCLUSION

In this paper, the implementation of a hardware accelerator for CIFAR-10 dataset from training of the neural network to hardware design is presented. The memory requirement for model parameters is reduced using the quantization-aware training technique with post-training fine-tuning quantization method. The proposed hardware accelerator is implemented in TSMC 40nm CMOS process. The top-1 accuracy of the proposed design is 85.1%, and the top-5 accuracy is 99.2%. The proposed MobileNetV2 hardware accelerator can achieve low power consumption and is suitable for the edge devices.

REFERENCES

[1] A. G. Howard et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," ArXiv170404861 Cs, Apr. 2017,. [Online]. Available: http://arxiv.org/abs/1704.04861. [Accessed: Sep. 15, 2019].

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, pp. 4510–4520, Jun. 2018. doi: 10.1109/CVPR.2018.00474.

[3] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, pp. 6848–6856, Jun. 2018. doi: 10.1109/CVPR.2018.00716.

[4] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients," ArXiv160606160 Cs, Feb. 2018, [Online]. Available: http://arxiv.org/abs/1606.06160. [Accessed: Sep. 30, 2019].

[5] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," ArXiv160304467 Cs, Mar. 2016, [Online]. Available: http://arxiv.org/abs/1603.04467 [Accessed: Sep. 25, 2019].

Authorized licensed use limited to: National Chung Cheng University. Downloaded on September 03,2020 at 06:05:47 UTC from IEEE Xplore. Restrictions apply.

[6] W. Yuxin and Others, Tensorpack. Tensorpack, 2019.

[7] J. Sim, J.-S. Park, M. Kim, D. Bae, Y. Choi, and L.-S. Kim, "14.6 A 1.42TOPS/W deep convolutional neural network recognition processor for intelligent IoE systems," in 2016 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, USA, pp. 264–265, Jan. 2016. doi: 10.1109/ISSCC.2016.7418008.

[8] K. Ando et al., "BRein memory: A 13-layer 4.2 K neuron/0.8 M synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm CMOS," in 2017 Symposium on VLSI Circuits, Kyoto, Japan, pp. C24–C25, Jun. 2017. doi: 10.23919/VLSIC.2017.8008533.

[9] K. Ueyoshi et al., "QUEST: Multi-Purpose Log-Quantized DNN Inference Engine Stacked on 96-MB 3-D SRAM Using Inductive Coupling Technology in 40-nm CMOS," IEEE J. Solid-State Circuits, vol. 54, no. 1, pp. 186–196, Jan. 2019, doi: 10.1109/JSSC.2018.2871623.

[10] S. K. Esser et al., "Convolutional networks for fast, energy-efficient neuromorphic computing," in Proceedings of the National Academy of Sciences, vol. 113, pp. 11441–11446, Oct. 2016. doi: 10.1073/pnas.1604850113.

Authorized licensed use limited to: National Chung Cheng University. Downloaded on September 03,2020 at 06:05:47 UTC from IEEE Xplore. Restrictions apply.