C-017

Approximately Quantizing Algorithm for In-memory Machine Learning Classifier

セキ カテイ† 鶴 隆介† 山内 寛行†
Jiazhen Xi Ryusuke Tsuru Hiroyuki Yamauchi

1. Introduction

Nowadays, with computing models in higher complexity, there's increasing need of ultra-low-power MAC operation to control system power. To get over such challenge, direction of in-memory computing, is highlighted. However, conventional necessary 1-Bit training algorithm increases system power. This work proposed Approximate Quantizing, a simple 1-Bit quantization algorithm for linear model. The proposed method can reach the similar accuracy as original 64-Bit with MNIST database and with faster and lower computing complexity, compared to conventional work.

2. In-memory Machine Learning Classifier

Fig. 1 shows the concepts of the in-memory computation process and conventional compute-out-of-memory (out-memory) process. The goal of the process in Fig. 1 is to recognize images with a size of 128 ×128 pixels.

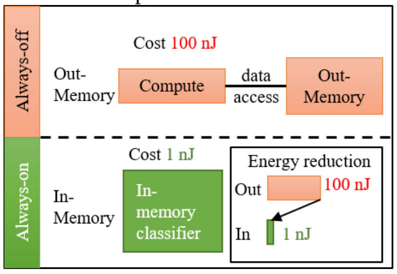

Fig. 1. Comparison between the out-memory process and in-memory process

The conventional out-memory process first serially accesses data from the memory and then performs a computation for recognition. During this process, the frequent data access makes it cost lots of energy, up to 100 nJ, to recognize one image. However, the computation of in-memory processes is performed within the memory (SRAM cells), which can avoid the frequent access and only costs 1 nJ per image inference [1].

2.1 Basic Structure of In-memory Classifier

In Fig.2, the left figure shows one column of cell array of standard 6T SRAM Each column of SRAM cell array is composed of WL in horizontal direction, BL/BLB pair in vertical direction and the BLs with equal number of WLs. The data, either +1 or -1, is stored in bit-cell by setting the state of transistors with different BL voltages through 'write' mode When an SRAM array which stores the model parameters is treated as a binary linear classifier, the classification process per decision is as following: The BL/BLB is pre-charged first. Then the same analogy voltages, which are from the DAC converting the input data to the analogy signals, are applied to all the WLs. Because of the low voltage range (0~400mV), the state of the bit-cells will not be changed. The responding bit-cell current will be applied to either BL or BLB according to the data stored in the bit-cell. Therefore, the voltage on the BL/BLB can be treated as a differential signal, which is equal with a multiplication between the input data and stored 1-Bit model parameter [2].

† Department of Computer Science and Engineering, Fukuoka Institute of Technology, Fukuoka, Japan

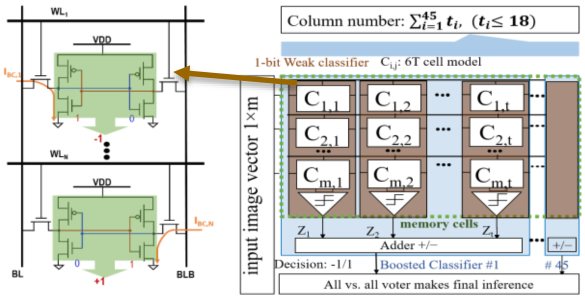

Fig. 2. Basic Structure

2.2 Quantized Linear Model

The conventional works [2-3] employed 'Constraint-Resolution Regression'(CRR) to solve the 1-Bit model parameters and applied a Boosting framework with 'one-vs-one' strategy on the multiclass classification task like MNIST [4], shown as right figure in Fig. 2. However, totally 45×18＝810 columns of binary linear models generate huge energy cost. What's more, with the number of classification target n increasing, the total number of columns needed is increasing with $O(n^2)$, which is a big challenge for the extendibility of the generalization of the system.

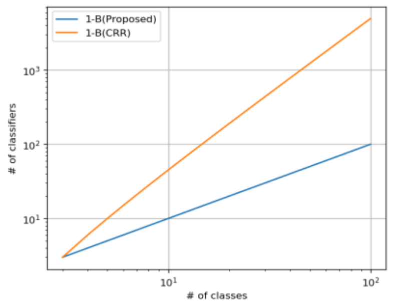

Fig. 3. Comparison of Complexity of Model

This work, however, considers to quantize a full precision model ('one-vs-all' strategy) with 64-Bit float parameters to 1-Bit using multiple columns of binary classifiers and a row-wise scaling vector, but with a much lower model complexity ($O(n)$).

2.3 Approximately quantizing algorithm for linear model

To consider the reason of loss of accuracy, it can be found that the values standing for the weights of the features quantized to be only 2 values after quantizing original 64-Bit parameters to 1-Bit, losing the ability to express the importance of the features. Thus, the performance is largely affected after quantization.

See the comparison of parameter distribution before/after 1-Bit direct quantization referring to Fig.4.

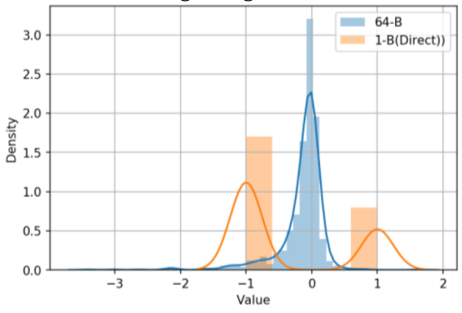

Fig. 4. Original and directly quantized parameters

Consider that the parameters of linear model stand for the importance of the features and the parameter with larger absolute value stands for a more important feature in some extents. To protect such characteristics, an array of scaling factors α can be introduced to store the maximum absolute value of each row of the model parameter matrix, ensuring the important weights be saved. Then reconstruct each column vector, standing for one binary classifier from original parameter matrix, to several new 1-Bit vectors horizontally to approximate the original binary classifier. The procedure of proposed algorithm is as following:
For any row of binary model's parameter with c elements

$$\boldsymbol{W} = (w_1, w_2, \dots w_c) \tag{2.1}$$

Consider reconstructing each element $w_i, i \in \{1, \dots, c\}$ to binary vector u_i with L 1-Bit elements approximately, where L stands for the approximate level. Then entire reconstructed binary row vector can be expressed as

$$\boldsymbol{U} = (u_1, u_2, \dots u_c)$$

$$where\ u_i = (u_{i,1}, u_{i,2} \dots u_{i,L}), u_{i,j} \in \{1, -1\}, i \in \{1, \dots, c\}, j \in \{1, \dots, L\} \tag{2.2}$$

Introduce the scaling factor $\alpha = \max(|w_1|, |w_2|, \dots |w_c|)/L$, the equivalent new approximate parameter row vector is

$$\boldsymbol{W}^* = (w_1^*, w_2^*, \dots w_c^*)$$

$$where\ w_i^* = (u_{i,1} + u_{i,2} + \dots + u_{i,L})\alpha, i \in \{1, \dots, c\} \tag{2.3}$$

As the binary vector is just in 1-Bit resolution and expresses the approximate parameter by the multiplication of the sum of binary elements and scaling factor α, consider introducing an integer array $\boldsymbol{N} = (n_1, n_2, \dots n_c), n_i \in \boldsymbol{N}^0$ which expresses the number of positive(one) elements. Since the total number is L, Approximate the original parameter w_i by w_i^* each of the approximate parameter w_i^* can be expressed as

$$w_i^* = (2n_i - L)\alpha \approx w_i, n_i \in \boldsymbol{N}^0 \tag{2.4}$$

Therefore, the target, which is quantization of the original parameter, is transferred as solving n_i given the approximate level L. Since each n_i is non-negative integer, with (2.4) we can reach final expression as following, where the brackets '[]' means rounding to integer

$$n_i = \left[\frac{L + \frac{w_i}{\alpha}}{2}\right] = \left[\frac{L + \frac{w_i}{\max(|w_1|, |w_2|, \dots |w_c|)/L}}{2}\right], i \in \{1, \dots, c\} \tag{2.5}$$

When w_i is the maximal element of the parameters, with (2.5), $n_i = L(or\ 0)$ can be achieved, meaning when the reconstructed binary vector is all ones(or zeros), the w_i can be obtained by (2.4) without loss. The approximate level L is a controllable parameter. With a large L, the precise approximation can be achieved on the cost of more binary row vectors needed. With a small L, the number of total columns can be reduced on a cost of more accuracy loss. Therefore, there's a trade-off to be considered that smaller L is preferred when the resources are limited and larger L is preferred when the accuracy is more important. The relation between final accuracy and approximate level L is as following Fig. 5, and the detailed result comparison for proposed 1-Bit approximately quantizing algorithm and boosted 1-Bit CRR [2-3] is as following table for dataset of down-sampling size 9×9:

Metrics	CRR[3]	Proposed (L=20)	Proposed (L=48)	64B
Accuracy%	93.11	90.75	91.09	90.98
# of columns	810	200	480	10 (64B)
Training time(sec)	2160	148	148	147
Complexity	N*(N-1)/2	N	N	N

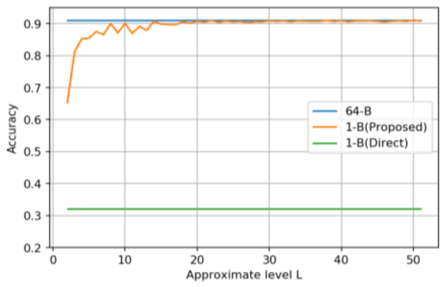

Fig. 5. Relation of accuracy and approximate level L

3. Conclusion

This work proposed Approximate Quantizing, a simple 1-Bit quantization algorithm for after trained linear model weight. The proposed method can reach the similar accuracy as original 64-Bit in MNIST database (~91%) and with faster (<7% of time) and lower computing complexity (<25% of columns needed), compared to conventional CRR algorithm.

Reference

[1] M. Kang, S. K. Gonugondla, M. S. Keel and N. R. Shanbhag, "An energy-efficient memory-based high- throughput vlsi architecture for convolutional networks", ICASSP, pp. 1037-1041. (2015).
[2] J. Zhang, Z. Wang, and N. Verma. "A machine-learning classifier implemented in a standard 6T SRAM array", VLSI-Circuits, pp. 1-2. (2016).
[3] Z. Wang and N. Verma, "A low-energy machine-learning classifier based on clocked comparators for direct inference on analog sensors," IEEE Trans. on Circuits and Systems I: Regular Papers, (2017).
[4] Y. LeCun and C. Cortes, "MNIST handwritten digit database," AT&T Labs [Online]. Available: http://yann.lecun. com/exdb/mnist, (2010).