Efficient Computation of Quantized Neural Networks by $\{-1, +1\}$ Encoding Decomposition

Anonymous Author(s)
Affiliation
Address
email

Abstract

Deep neural networks require extensive computing resources, and can not be efficiently applied to embedded devices such as mobile phones, which seriously limits their applicability. To address this problem, we propose a novel encoding scheme by using $\{-1, +1\}$ to decompose quantized neural networks (QNNs) into multi-branch binary networks, which can be efficiently implemented by bitwise operations (*xnor* and *bitcount*) to achieve model compression, computational acceleration and resource saving. Our method can achieve at most $\sim 59\times$ speedup and $\sim 32\times$ memory saving over its full-precision counterparts. Therefore, users can easily achieve different encoding precisions arbitrarily according to their requirements and hardware resources. Our mechanism is very suitable for the use of FPGA and ASIC in terms of data storage and computation, which provides a feasible idea for smart chips. We validate the effectiveness of our method on both large-scale image classification (e.g., ImageNet) and object detection tasks.

1 Introduction

Deep Neural Networks (DNNs) have been successfully applied in many fields, especially in image classification, object detection and natural language processing. Because of numerous parameters and complex model architectures, huge storage space and considerable power consumption are needed. However, for mobile phones and embedded platforms, whose resources are limited, it's hard to achieve satisfactory performance for industrial applications. With the rapid development of DNNs, more and more computing resources are needed, and the requirements for hardware are becoming higher and higher.

In order to improve the energy efficiency of hardware, achieve model compression or computational acceleration, many solutions have been proposed, such as network sparse and pruning [7, 22, 12, 25, 21], low-rank approximation [5, 11, 23], architecture design [20, 10, 8, 19, 16], model quantization [1, 13, 3, 9, 18, 14], and so on. [17, 6, 28] constrain their weights to $\{-1, +1\}$ or $\{-1, 0, 1\}$ and achieve limited acceleration by using simple accumulation instead of complicated multiplication-accumulations. In particular, [2, 18, 4, 27, 24] quantize activation values and weights to bits and use bitwise logic operations to achieve extreme acceleration ratio in inference process but they are suffering from significant performance degradation. However, most models are proposed for fixed precision, and can not extend to other precision models. They easily fall into local optimal solutions and face slow convergence speed in training process. In order to bridge the gap between low-bit and full-precision and be applied to many cases, we propose a novel encoding scheme of using $\{-1, +1\}$ to easily decompose trained QNNs into multi-branch binary networks. Therefore, the inference process can be efficiently implemented by bitwise operations to achieve model compression, computational acceleration and resource saving.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

2 Model Decomposition

As the basic computation in most neural network layers, matrix multiplication costs lots of resources and also is the most time consuming operation. Modern computers store and process data in binary format, thus non-negative integers can be directly encoded by {0, 1}. We propose a novel decomposition method to accelerate matrix multiplication as follows: Let $x = [\mathrm{x}^1, \mathrm{x}^2, ..., \mathrm{x}^N]^T$ and $w = [\mathrm{w}^1, \mathrm{w}^2, ..., \mathrm{w}^N]^T$ be two vectors of non-negative integers, where $\mathrm{x}^i, \mathrm{w}^i \in \{0, 1, 2, ...\}$ for $i = 1, 2, ..., N$. The dot product of those two vectors can be represented as follows:

$$x^T \cdot w = [\mathrm{x}^1, \mathrm{x}^2, ..., \mathrm{x}^N][\mathrm{w}^1, \mathrm{w}^2, ..., \mathrm{w}^N]^T = \sum_{n=1}^{N} \mathrm{x}^n \cdot \mathrm{w}^n. \tag{1}$$

All of the above operations consist of N multiplications and $(N-1)$ additions. Based on the above encoding scheme, the vector x can be encoded to binary form using M bits, i.e.,

$$x = [\overbrace{\mathrm{x}_M^1 \mathrm{x}_{M-1}^1 ... \mathrm{x}_1^1}, \overbrace{\mathrm{x}_M^2 \mathrm{x}_{M-1}^2 ... \mathrm{x}_1^2}, ..., \overbrace{\mathrm{x}_M^N \mathrm{x}_{M-1}^N ... \mathrm{x}_1^N}]^T. \tag{2}$$

Then we convert the right-hand side of (2) into the following form:

$$\begin{bmatrix} \mathrm{x}_M^1 & \mathrm{x}_M^2 & \cdots & \mathrm{x}_M^N \\ \mathrm{x}_{M-1}^1 & \mathrm{x}_{M-1}^2 & \cdots & \mathrm{x}_{M-1}^N \\ \vdots & \vdots & \cdots & \vdots \\ \mathrm{x}_1^1 & \mathrm{x}_1^2 & \cdots & \mathrm{x}_1^N \end{bmatrix} = \begin{bmatrix} x_M \\ x_{M-1} \\ \vdots \\ x_1 \end{bmatrix}, \tag{3}$$

where $\mathrm{x}^j = \sum_{m=1}^{M} 2^{m-1} \cdot \mathrm{x}_m^j, \ \mathrm{x}_m^j \in \{0, 1\}, \ \ x_i = [\mathrm{x}_i^1, \mathrm{x}_i^2, ..., \mathrm{x}_i^N]$.

In such an encoding scheme, the number of represented states is not greater than 2^M. In addition, we encode another vector w with K-bit numbers in the same way. Therefore, the dot product of the two vectors can be computed as follows:

$$x^T \cdot w = \sum_{n=1}^{N} \mathrm{x}^n \cdot \mathrm{w}^n = \sum_{n=1}^{N} \left(\sum_{m=1}^{M} 2^{m-1} \cdot \mathrm{x}_m^n \right) \cdot \left(\sum_{k=1}^{K} 2^{k-1} \cdot \mathrm{w}_k^n \right) \tag{4}$$

$$= \sum_{m=1}^{M} \sum_{k=1}^{K} 2^{m+k-2} \cdot x_m \cdot w_k^T. \tag{5}$$

From the above formulas, the dot product is decomposed into $M \times K$ sub-operations, in which each element is 0 or 1. Because of the restriction of encoding and without using the sign bit, the above representation can only be used to encode non-negative integers. However, it's impossible to limit the weights and the values of the activation functions to non-negative integers. In order to encode both positive and negative integers, we propose a novel encoding scheme, which uses {-1, +1} as the basic elements rather than {0, 1}. Then we can use multiple bitwise operations (i.e., *xnor* and *bitcount*) to effectively achieve the above vector multiplications. Our operation mechanism can be suitable for all vector/matrix multiplications. Besides fully connected layers, our mechanism is also suitable for convolution and deconvolution layers in deep neural networks.

3 M-bit Encoding Functions

As an important part in neural networks, activation function can enhance the nonlinear characterization of the networks. In our proposed model decomposition method, encoding function plays a critical role and can encode input data to multi bits (-1 or +1). Those numbers represent the encoding of input data. Therefore, the dot product can be computed by the formula (6). Without other judgment and mapping calculation, we use trigonometric functions as the basic encoding functions. In the end, we use the sign function to hard divide to -1 or +1. The mathematical expression can be formulated as follows:

$$MBitEncoder(x) = \begin{cases} \varphi_M^m(x) : sign(-sin(\frac{2^M-1}{2^m}\pi \cdot x)), \ m \in \{1, 2, ..., M-1\}, \\ \varphi_M^M(x) : sign(sin(\frac{2^M-1}{2^M}\pi \cdot x)), \ \text{otherwise}, \end{cases} \tag{6}$$

where $\varphi_M^M(x)$ is the encoding function of the highest bit of MBitEncoder (i.e., $m = M$). The periodicity is obviously different from others because it needs to denote more states.

4 Experiments

In this section, we use the same network architecture described in [17, 2] for CIFAR-10 and choose ResNet-18 as the basic network for ImageNet. It is very hard to train on large-scale training sets (e.g., ImageNet), and thus parameter initialization is particularly important. In particular, the well-trained full-precision model parameters activated by *ReLU* can be directly used as initialization parameters for our 8-bit quantized network. After fine-tuning dozens of epochs, 8-bit quantized networks can be well-trained. Similarly, we use the 8-bit model parameters as the initialization parameters to train 7-bit quantized networks, and so on. We use the loss computed by quantized parameters to update full precision parameters described as the straight-through estimator [26]. Table 1 lists the performance (e.g., accuracy, speedup ratio, memory saving ratio) of our method and several typical models mentioned above. The accuracies were achieved after dozens of times fine-tuning. If continue training those networks, we can reach slightly better performance. We also use the trained ResNet-18 with the Single Shot MultiBox Detector (SSD) framework [15] to validate object detection tasks. We also use the trained model parameters in ImageNet classification to initialize SSD, and report the experimental results in Table 1 after dozens of times fine-tuning.

We analyze the theoretical performance of our encoding scheme. The theoretical speedup and model compression ratios are given in the following table. Thus, our method can obtain at most $\sim 59\times$ speedup and $\sim 32\times$ memory saving over its full-precision counterparts. It can achieve $\sim 59/MK\times$ speedup and $\sim 32/K\times$ memory saving by constraining activation values to M-bit and the values of weights to K-bit, where $M, K \in \{1, 2, ..., 8\}$. In fact, our method can provide 64 available encoding choices, and hence our encoded network with different encoding precisions has different calculation speed, memory requirements and experimental precisions. Here, we use 64-bit binary operation in one clock cycle. If those decompositions are implemented in the FPGA or ASIC platform, the speedup ratios can be much higher.

Table 1: Results of classification and object detection.

Method	CIFAR-10	ImageNet (Top-1)	ImageNet (Top-5)	VOC (mAP)	Speedup	MemorySave
BWN [17]	90.10%	60.80%	83.00%	-	∼2x	∼32x
BNN [2]	88.60%	42.20%	67.10%	-	∼64x	∼32x
TWN [6]	92.56%	61.80%	84.20%	-	∼2x	∼16x
XNOR-Net [18]	-	51.20%	73.20%	-	∼58x	∼32x
ABC-Net [14]	-	65.00%	85.90%	-	-	∼6.4x
Full-Precision	91.40%	68.60%	88.70%	0.6392	1x	1x
Encoded activations and weights						
M=K=1	90.39%	47.10%	71.70%	-	∼59.00x	∼32x
M=K=2	91.06%	56.30%	79.48%	-	∼14.75x	∼16x
M=K=3	91.27%	58.69%	81.84%	-	∼6.56x	∼10.7x
M=K=4	91.15%	59.57%	82.35%	-	∼3.69x	∼8x
M=K=5	90.92%	65.09%	86.42%	0.5423	∼2.36x	∼6.4x
M=K=6	91.01%	67.04%	87.69%	0.6131	∼1.64x	∼5.3x
M=K=7	90.20%	68.37%	88.47%	-	∼1.20x	∼4.6x
M=K=8	90.43%	68.63%	88.70%	0.6351	∼0.92x	∼4x

5 Conclusions

In this paper, we proposed a novel encoding scheme of using {-1, +1} to decompose QNNs into multi-branch binary networks, in which we used bitwise operations (*xnor* and *bitcount*) to achieve model compression, computational acceleration and resource saving. In particular, we can use the high-bit model parameters to initialize a low-bit model and achieve good results in various applications. Thus, users can easily achieve different encoding precisions arbitrarily according to their requirements (e.g., accuracy and speed) and hardware resources (e.g., memory). This special mechanism of data storage and calculation can yield great performance in FPGA and ASIC, and thus our mechanism is a feasible idea for smart chips. Future works will focus on improving the hardware implementation and chip technology, and exploring some ways to automatically select proper bits for various network architectures (e.g., VGG and ResNet).

References

[1] Matthieu Courbariaux, Yoshua Bengio, and Jean Pierre David. Training deep neural networks with low precision multiplications. *Computer Science*, 2014.

[2] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[3] Lin Darryl, Talathi Sachin, and Annapureddy Sreekanth. Fixed point quantization of deep convolutional networks. *Computer Science*, 2015.

[4] Lei Deng, Peng Jiao, Jing Pei, Zhenzhi Wu, and Guoqi Li. Gated xnor networks: deep neural networks with ternary weights and activations under a unified discretization framework. *arXiv preprint arXiv:1705.09283*, 2017.

[5] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, pages 1269–1277, 2014.

[6] Li Fengfu, Zhang Bo, and Liu Bin. Ternary weight networks. In *NIPS Workshop on EMDNN*, 2016.

[7] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, pages 164–171, 1993.

[8] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[9] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18:187–1, 2017.

[10] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[11] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[12] Michael James, Jack Lindsey, and Ilya Sharapov. Adaptive weight sparsity for training deep neural networks. In *ICLR*, 2018.

[13] Wu Jiaxiang, Cong Leng, Wang Yuhang, Hu Qinghao, and Cheng Jian. Quantized convolutional neural networks for mobile devices. In *CVPR*, pages 4820–4828, 2016.

[14] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *NIPS*, pages 345–353, 2017.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *ECCV*, pages 21–37. Springer, 2016.

[16] Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. ThiNet: Pruning cnn filters for a thinner net. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[17] Courbariaux Matthieu, Bengio Yoshua, and David Jean Pierre. BinaryConnect: training deep neural networks with binary weights during propagations. In *NIPS*, pages 3123–3131, 2015.

[18] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542. Springer, 2016.

[19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[20] Ioffe Sergey and Szegedy Christian. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[21] Han Song, Pool Jeff, Tran John, and Dally William J. Learning both weights and connections for efficient neural networks. In *NIPS*, pages 1135–1143, 2015.

[22] Srinivas Suraj and Babu R. Venkatesh. Data-free parameter pruning for deep neural networks. *Computer Science*, pages 2830–2838, 2015.

[23] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*, 2015.

[24] Diwen Wan, Fumin Shen, Li Liu, Fan Zhu, Jie Qin, Ling Shao, and Heng Tao Shen. TBN: Convolutional neural network with ternary inputs and binary weights. In *ECCV*, pages 315–332, 2018.

[25] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NIPS*, pages 2074–2082, 2016.

[26] Bengio Yoshua, Leonard Nicholas, and Courville Aaron. Estimating or propagating gradients through stochastic neurons for conditional computation. *Computer Science*, 2013.

[27] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[28] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *ICLR*, 2017.