Low-precision CNN Model Quantization based on Optimal Scaling Factor Estimation

Yilun Zhou[2], Li Chen[1*], Rong Xie[1,2], Li Song[1,2], Wenjun Zhang[1,2]
[1]*Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Shanghai, China*
[2]*Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, Shanghai, China*
{dell3253, hilichen, xierong, song_li, zhangwenjun}@sjtu.edu.cn

***Abstract*—With the development of convolutional neural networks (CNNs), researchers have acquired satisfactory performances on computer vision tasks such as image classification and semantic segmentation. However, CNNs contain millons of weight parameters and demand huge storage and computation resources, making it hard to deploy on constrained hardwares. To compress the model size and accelerate the deployment speed, many model compression methods have been proposed, including quantization methods that aim to reduce network redundancy by decreasing the representational bits for weights (and activations). However, for low-bit quantization, the inevitable quantization error will lead to significant accuracy degradation and gradient mismatch. In this paper, we propose Scale Estimation Quantization (SEQ). To reduce the quantization error, we analyze the variance of error derived by the quantization process. By exploiting the distributions of network values, we reduce quantization error and estimate the optimal scale parameters for our proposed quantization function. Further more, to deal with gradient mismatch problem in backward propagation, we propose backward approximation. We apply our algorithm on image classification tasks. Our method achieves a close performance to full-precision counterparts on VGG-Small and AlexNet with 1-bit weights and 2-bit activations.**

***Index Terms*—convolutional neural networks, model compression, model quantization, image classification**

I. Introduction

Convolutional Neural Networks (CNNs) have proved to be reliable tools in many computer vision tasks, such as image classification, object detection and semantic segmentation. In general, CNNs involve millions of neural connections. The huge amounts of parameters garantee the capacities of processing high dimentional data and acheiving state-of-the-art performances. In the meantime, the general trend to make deeper and more complicated networks is making deep learning more expensive and difficult to apply. The demands of abundant storage and computation resource prevent these methods from high-efficiency hardware deployment, especially on mobile or calculation restrained devices.

In recent years, researchers have noticed this issue and the inference speed of networks is taken into consideration. Aiming at reducing the redundancy in networks and models, model compression and quantization methods have been proposed [6]. These methods achieves promising results and show the existence of redundancies in current CNN models.

*Corresponding author

Most quantization literatures take into consideration the accuracy loss compared to the full precision model. Since full precision values are replaced with lower bits, quantization error is inevitable among all these methods, leading to huge accuracy degradation. Many means such as scaling and sparsification have been proposed to alleviate this problem. SYQ [4] uses scalar matrix to expand the volumn of the codebook. Clip-Q [12] and TSQ [13] use sparsity pruning to keep a high quantization resolution. Additionally, most works rely on finetuning after quantization to restore accuracy. Since quantization function is piecewise and has zero derivative almost everywhere, Straight Through Estimator (STE) is broadly adpoted to approximate the backward propagation. However the gradient mismatch problem still exists and more specified backward estimators have been proposed in [2] and [4].

Acceleration effect of the quantized model is weighed as well. For existing full-precision networks, both weights and activations in the network should be quantized to achieve an actural acceleration effect. Some existing works like [12] and [6] focus on reducing the on-device storage of the model and only quantize the weights that will be saved on devices. When inferencing, the weights must be restored to floating point numbers to match the full-precision activaitons. Though achieving a outstanding compression rate, this kind of quantization methods does not improve deployment speed. In comparison, methods quantizing both weights and activations have a relatively low compression rate. [9] shows that 8-bit integer quantization can achieve $4\times$ reduction of model size and improved inference efficiency on existing hardwares with a slight accuracy loss. Other literatures like [14], [4] and [2] propose weight-activation quantization methods with lower bits. SYQ [4] introduces scalar matrix with $K \times K$ or K scales to expand the capacity of quantized convolution layers, where K denotes the kernel size. However, quantization methods with multiple scales need costomized hardwares to accelerate the deployment speed. On existing deep learning devices, it may achieve less acceleration effect. HWGQ [2] utilizes batch normalization to constraint layer distributions to approximately normal distributions of zero mean and unit variance and applies Lloyd's algorithm to generate the best parameters for all layers. However, this method requires a batch norm layer before a quantized layer. Hence it is not suitable for all networks. BinaryConnect [3], TTQ [15] and Xnor-Net [11] quantize weights and activations to binary or

978-1-7281-2150-5/19/$31.00 ©2019 IEEE

ternary values to enable bitwise operations. But with extremely low bitwidth, the model performances are not garanteed.

To address the above issues, we propose our quantization algorithm Scale Estimation Quantization (SEQ). Our work includes:

- We propose a concise quantization strategy that is implementable on current deep learning hardwares. The method allows efficient fixed point operations and only an extra scaling procedure is needed. Our forward quantization function only has a scale parameter to decide.
- We analyze the quantization error derived by the quantization procedure. Based on the assumption that the distributions of network parameters and values are close to normal distributions, we utilize the property of normal distributions to estimate the scaling factors.
- Based on Straight Through Estimator, we provide a backward approximation method to deal with the gradient mismatch issue caused by quantization.

We apply our strategy to image classification tasks. The result shows the flexibility of our quantization algorithm. The rest of the paper is organized as follows. In section II, we will introduce and describe our proposed SEQ in details, including forward inferencing strategy, scaling factor estimation and backward gradient propagating strategy. Coresponding experiments will be conducted and analyzed in section III and the conclusion is drawn in section IV.

II. Quantization Method

In this section, we propose a flexible quantization strategy which can be used for efficient model quantization. The algorithm is stated in three parts, forward approximation, scaling factor estimation and backward approximation. Forward approximation provides a uniform quantization function to quantize weights and activations of convolution layers in training and inferencing. Backward approximation helps to address the gradient mismatch problem and is only used in training steps. The scaling factor estimation section proposes an algorithm to estimate the optimal scale parameter for our quantization function by exploiting the distributions of network values. The algorithm is based on the basic assumption that the distributions of activations and weights of CNNs are close to normal distributions.

A. Forward Approximation

In CNNs, full-precision values x are approximated by a quantization function Q to get low-precision values q.

$$q = Q(x) \tag{1}$$

Similar to [9], we introduce uniform quantization function to enable the ability of acceleration by scaling.

$$Q(x) = \frac{clip(round(x \times s))}{s} \tag{2}$$

s is a scaling factor larger than 0 and $round$ function maps a value to its nearest integer. With s and $round$ we can transform a floating point value into an integer, but the quantized values are unbounded. So a $clip$ function is added to restrain the range of integers. Then the quantized values are reverted to the original range by dividing s. The $clip$ function is stated as

$$clip(x) = \begin{cases} T_1, & \text{if } x > T_1 \\ x, & \text{if } T_2 \le x \le T_1 \\ T_2, & \text{if } x < T_2 \end{cases} \tag{3}$$

In the above equation, T_1 is the upper bound threshold and T_2 is the lower bound threshold. Generally, we regard the values to have both positive and negative numbers. So for n-bit quantization that quantized values are represented by at most n bits, we set $T_1 = 2^{n-1} - 1$ and $T_2 = 1 - 2^{n-1}$. If the former layer of the quantized layer is a ReLU layer, the activations must be nonnegative. Then $T_1 = 2^n - 1$ and $T_2 = 0$. Since large values are very rare, the error introduced by clipping is acceptable with a proper scale s. The figure of our quantization function is shown in Fig. 1.

Value $clip(round(x \times s))$ in (2) is the intermediate integer to be stored or transmitted. With these integers, we can apply low-precision arithmetic operations. For example, floating-point matrix multiplications can be represented by efficient fixed-point matrix multiplications and only an extra scaling procedure is needed afterwards. Thus the capacity of simultaneously compressing and accelerating the model is garanteed. Given a hyper parameter n which determines the compression rate, the only parameter to be determined for each layer in our quantization function is the scale s. This is to be discussed in the next section.

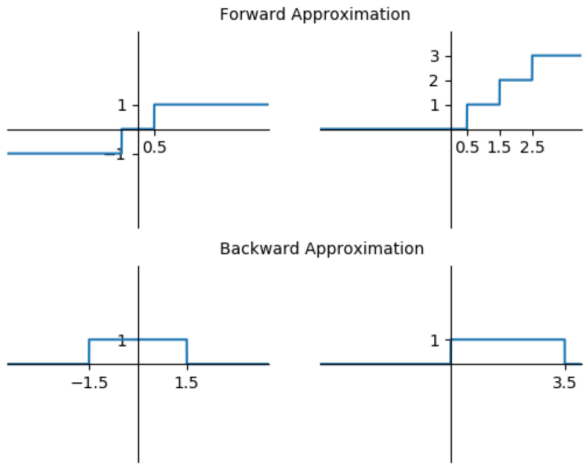

Fig. 1. Our forward approximation (top) and backward approximation (bottom). In the example $s = 1$ and $n = 2$. Approximation with a former ReLU layer is on the right.

B. scaling factor estimation

We estiamte the best scale parameters for all layers by minimizing the total quantization error of the network. For our quantization function Q, quantization error E_Q includes rounding error and clipping error, where rounding error E_R is

generated by $round$ function and clipping error E_C is derived by $clip$ function in (2). So our goal is

$$\begin{aligned} \min \quad E_Q &= E_R + E_C \\ &= \sum_{x \in X} (E_r(x) + E_c(x)) \end{aligned}$$

Given an input x, the corresponding rounding error is stated as

$$E_r(x) = \left| x - \frac{round(x \times s)}{s} \right| \tag{4}$$

Since $|x \times s - round(x \times s)| \leq 0.5$, the rounding error can be bounded by $\frac{1}{2s}$. With the increase of s, rounding error decreases. Identally, the clipping error of an given input x is

$$E_c(x) = \left| x - \frac{clip(x \times s)}{s} \right| \tag{5}$$

According to (3), clipping error $E_c(x)$ is not zero only when $x > T_1$ or $x < T_2$. So this error is generated by part of the inputs, but it will grow exponentially and its upper bound is not limited. With the increase of s, more values is clipped and clipping error increases.

We then analyze the variation tendency of these two kinds of errors with the variation of scale s. For a very small scale, denoted s_1, we definitly have $clip(x \times s_1) = x \times s_1$. Thus the clipping error E_C is nearly zero. However, the rounding error E_R can be relatively large. For a very large scale, denoted s_2. We can garantee rounding error E_R will be small, while more inputs will be out of the boundary of $clip$ function compared to s_1. So the clipping error can be huge.

To establish a stable quantization system, we expect to choose a proper scale s to minimize the total quantization error, which is the sum of rounding error and clipping error. But the distributions of X is unknown so we cannot infer the optimal solution. As an alternative way we study the general distributions of weights and activations to estimate the target scales.

The distributions of activations and weights of convolutional neural networks has long been studies. It has been shown in [8] that convolution operations tend to generate distributions close to normal distributions. HWGQ [2] and TSQ [13] follow this thinking and achieve great quantization progress. They restrain the layer distributions to approximative standard normal distributions and generate optimal quantization parameters. To go further, we deal with more general parameter distributions of arbitrary means and variances. A normal distribution can be determined by its mean μ and variance σ. When $\mu = 0$, the following equation holds for any σ

$$P(-\sigma \leq x \leq \sigma) \approx 0.683 \tag{6}$$

Additionally, given a multiplication factor α, if we replace σ with $\alpha\sigma$, probability $P(-\alpha\sigma \leq x \leq \alpha\sigma)$ is still a constant whatever value σ takes. Based on the above analysis, we associate clipping rate with the variance of the distribution and we can adjust the clipping rate for all layers with a single, alterable parameter α. Given the assumption that CNN layer distributions are all normal distributions, the quantization error of each layer should be minimized with the same clipping rate. Hence the total quantization error of the network is minimized. In fact $\alpha\sigma$ denotes a clipping threshold. So we have

$$\alpha\sigma \geq max(Q(x)) \tag{7}$$

Utilizing (2) and (3), we get

$$\alpha\sigma \geq max(Q(x)) \tag{8}$$

$$\geq max(\frac{clip(round(x \times s))}{s}) \tag{9}$$

$$\geq \frac{T_1}{s} \tag{10}$$

Eq. (10) shows the lower bound of scale parameter s, and the optimal scale s^* is achieved when taking the equal sign.

$$s^* = \frac{T_1}{\alpha\sigma} \tag{11}$$

In the above equation, $T_1 = 2^{n-1} - 1$ or $2^n - 1$ where n is the bitwidth. σ differs among differnet layers and is calculated by the fill-precision parameters in advance. The only parameter to be searched is α, which is equal among the whole network. So the searching space can be traversed by one single loop.

C. Backward Approximation

Since the forward quantization function of our proposed method contains a step-wise constant function $round$ in (2), the derivative is either zero or is not derivable everywhere, causing a serious problem of gradient vanishing. Hence a customed gradient function such as piecewise linear function is needed during backward propagating to enhance the capibility of convergence. In the literature [1], this problem is settled by Straight Through Estimator (STE), which is defined as

$$\frac{\partial Q}{\partial X} = 1 \tag{12}$$

Where X is the activation or weight and Q is the quantization function. Hence the gradient of output Z is

$$\frac{\partial Z}{\partial X} = \frac{\partial Z}{\partial Q} \tag{13}$$

For values quantized to the upper and lower bound of $clip$ function, the quantization error $|Q(x) - x|$ is definitely smaller than $\frac{1}{2s}$, where s is the quantization scale. But for values quantized to the upper and lower bound, the clipping error is unbounded so the quantization error can be much larger than this limitation, since our clipping threshold $\alpha\sigma$ is smaller than $max(x)$. To address the possible large-value gradient mismatch, we propose an alternative backward approximation, which only transmits gradients when $|Q(x) - x| \leq \frac{1}{2s}$. The clipped STE is stated in the following format

$$\frac{\partial Q}{\partial X} = \begin{cases} 1, & \text{if } -\alpha\sigma - \frac{1}{2s} \leq x \leq \alpha\sigma + \frac{1}{2s} \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

Fig. 1 illustrates our backward approximation function. With clipped STE, the network is finetuned more efficiently and converges with fewer iterations. The whole procedure of

our proposed algorithm is shown in **Algorithm. 1**. In our experiments, the while loop takes no more than 10 iterations to reach the optimum if Δ_α is set to 1.0.

Algorithm 1: Training procedure of our quantization method

Input: Bitwidth n, Alpha step Δ_α
Output: The best performance p^* and the corresponding alpha α^*

Train the full precision network
Feed some training data into the network and get layer distributions
Calculate the variances $\sigma_1, \sigma_2, \cdots$ of all layers
$p_{max} = 0$
$\alpha = \Delta_\alpha$
while True **do**
    Calculate the scales $s_1, s_2, \cdots$ with n and α using (11).
    Quantize the network with $s_1, s_2, \cdots$ and finetune it. Use (2) for forward propagation and (14) for backward propagation
    Evaluate the quantized network and get the performance p
    if $p_{max} > p$ **then**
        break
    end if
    $p_{max} = p$
    $\alpha = \alpha + \Delta_\alpha$
end while
return p_{max}, α

III. Experiments

A. VGG-Small On CIFAR-10

We conduct experiments on CIFAR-10 dataset [10] and compare with other state-of-the-art quantization methods using VGG-Small network. Our network structure and training policies follow [2]. The strategies quantizing activations and weights are different. We quantize activations to 2 bits and weights to 1 bit. For weights, we adopt weight binarization with scaling in [2] because our method does not support 1-bit quantization. For activations, we use our proposed SEQ to quantize activations to 2 bits. First we train a full-precision VGG-Small. The float network achieve the same accuracy as the original model. Then, we randomly choose 100 images from CIFAE-10 training set and feed them into the network. Then we compute the means and variances of activations of all layers using our SEQ method. If the layer follows a ReLU layer, we compute the values before ReLU. Finally we utilize **Algorithm. 1** to find the optimal parameter α. The accuracy loss under different α is shown in Tab. I. The result illustrates that our algorithm can reach the optimum in a few iterations.

The comparison result with other state-of-the-art quantization methods on CIFAR-10 are shown in Tab. II. The results indicate that our quantization method SEQ has no accuracy degradation compared to the full-precision VGG-Small network and outperforms other quantization methods. Our loss is only 0.28% higher than TSQ, which quantizes weights to ternary values.

TABLE I
Accuracy loss on CIFAR-10 with different α, with 2-bit activations and 1-bit weights.

alpha	1.0	2.0	3.0	4.0	5.0	6.0
Loss	7.90%	6.80%	6.79%	6.91%	7.27%	8.09%

TABLE II
Comparison with state-of-the-art quantization methods on CIFAR-10 dataset.

Method	**Activation**	**Weight**	**Loss**
VGG-Small	Full	Full	6.82%
TWN [5]	Full	Ternary	7.44%
BinaryConnect [3]	Full	Binary	8.27%
BNN [7]	Binary	Binary	10.15%
HWGQ [2]	2-bit	Binary	7.49%
TSQ [13]	2-bit	Ternary	6.51%
SEQ	2-bit	Binary	6.79%

B. AlexNet On ImageNet

We also conduct experiments on more complicated network and dataset. Tab. III shows the classification accuracy of various quantization methods on AlexNet. Compared with other low-bit quantization methods, the accuracy of our method SEQ is closer to the original full-precision model. The result shows that SEQ performs well given various datasets and network structures.

TABLE III
Comparison with state-of-the-art quantization methods on ImageNet dataset.

Method	**Activation**	**Weight**	**Top-1**	**Top-5**
AlexNet	Full	Full	58.5%	81.5%
DOREFA [14]	2-bit	Binary	47.7%	-
HWGQ [2]	2-bit	Binary	52.7%	76.3%
SEQ	2-bit	Binary	53.8%	77.4%

IV. Conclusion

In this paper, we propose a new model quantization method SEQ. We introduce the quatization function with a scale paramter to determine for each layer. Based on the idea that activations and weights of CNN layers have similar distributions as normal distributions, we utilize the variances to estimate the optimal scale parameter for each layer. Additionally, improvement on STE helps to deal the gradient mismatch problem. Our proposed method shows its flexibility on various networks and datasets and reaches state-of-the-art performances. In the meantime, by utilizing intermediate integers of quantization, both compression and acceleration abililties are garanteed. These results suggest that SEQ can be a valid quantization method in practical applications.

ACKNOWLEDGMENT

This work was supported in part by Natural Science Foundation of Shanghai (18ZR1418100), National Natural Science Foundation of China (61771306, 61671296), the Shanghai Key Laboratory of Digital Media Processing and Transmissions (STCSM18DZ2270700)

REFERENCES

[1] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[2] Z. Cai, X. He, J. Sun, and N. Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. *arXiv preprint arXiv:1702.00953*, 2017.

[3] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[4] J. Faraone, N. Fraser, M. Blott, and P. H. Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4300–4309, 2018.

[5] L. Fengfu, Z. Bo, and L. Bin. Ternary weight networks. In *NIPS Workshop on EMDNN*, volume 118, page 119, 2016.

[6] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[7] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[8] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[9] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[10] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[11] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[12] F. Tung and G. Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7873–7882, 2018.

[13] P. Wang, Q. Hu, Y. Zhang, C. Zhang, Y. Liu, J. Cheng, et al. Two-step quantization for low-bit neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4376–4384, 2018.

[14] 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. *arXiv preprint arXiv:1606.06160*, 2016.

[15] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.