A novel quantization method combined with knowledge distillation for deep neural networks

Zhou Hu, Chunxiao Fan* and Guangming Song

School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230009, Anhui, China

*Corresponding author. Email: fanchunxiao@hfut.edu.cn

Abstract. The massive parameters and intensive computations in neural networks always limit the deployment on embedded devices with poor storage and computing power. To solve this problem, a novel quantization algorithm combined with Knowledge Distillation (KD) is proposed to reduce the model size and speed up the inference of deep models. The proposed method consists of two phases, KD-Training and Quantization-Retraining. KD-Training attempts to train a compact student model with pre-quantized weights by the proposed pre-quantized constraint loss. In Quantization-Retraining, the pre-quantized weights are quantized to $2^n$ and the first and last layers of the network are retained to make up for the accuracy loss caused by quantization. Experiments on the CIFAR-10 dataset show that the proposed method can obtain a low-precision (2-5bit) quantized student model with a compact structure, and the test accuracy even exceeds its full-precision (32bit) reference as the improvement of the generalization ability. It can get higher performance compared with that obtained by other quantization methods. Also, since the quantized weights are constrained in $\{\pm 2^n\}$, it is suitable for the acceleration of the network calculation in hardware.

1. Introduction

Deep neural networks (DNN) have achieved state-of-the-art performance in various computer vision tasks, such as the object detection, image classification and semantic segmentation. But these neural network models always need considerable storage and expensive computing because of massive learnable parameters. For instance, VGG-16[1] has a model size of about 528MB (15.25 million parameters) and perform about 15.5 billion floating-point operations to achieve a classification task. Therefore, it remains a challenge to deploy superior models on the embedded devices or smart phones with poor computation and limited storage.

To address the computational and memory costs of neural network during the training and inference phase, many compression methods have been proposed. Some works adopt to quantize the parameters in neural networks to make storage and computation more efficient. Binary Neural Networks (BNN)[2] propose a quantization scheme constraining weights to $\{-1, +1\}$ (1bit) and achieve state-of-the-art performance for small datasets such as MNIST, SVHN and Cifar10. Compared with the full-precision (32bit) network, these two methods can achieve up to $32\times$ model compression. Although tremendous progress has been made on the quantization, these methods still suffer from a non-negligible accuracy loss, especially under scenarios of using extremely low bit-width values to quantize the weights.

The researchers also proposed some compression algorithms based on reduced model structures. Knowledge Distillation[3] tries to reduce model size through transferring knowledge. By introducing a
complex and high-precision teacher network, knowledge is transferred to a student network model. Because of the additional supervision information, the student network with the reduced structure can obtain higher accuracy. However, all the methods mentioned above focus on obtaining a compact network with fewer parameters and less calculation compared with the original network, but there are still lots of floating-point operations, which can still be the bottlenecks of hardware applications.

In this paper, we propose a novel quantification method combined with KD for deep neural networks, which can get a compact network with fewer parameters to address the challenges mentioned above. The proposed method combines the advantages of KD and quantized parameters. KD can reduce the number of parameters by obtaining a compact structure and improve network accuracy. Besides, the quantized parameters in the network are constrained in \{\pm 2^n\}, which can be efficiently achieved by shifters in hardware to speed up the inference of models.

2. Proposed method

In this section, the details of the proposed method are presented. As shown in Fig. 1, the proposed algorithm is divided into two phases: KD-Training and Quantization-Retraining. In KD-Training, the pre-trained teacher model \( T_F \) distills knowledge into the student model \( S_F \) with a compact structure. The proposed pre-quantized constraint loss constrains the weights of the student model to pre-quantized values represented by approximated \( 2^n \). During Quantization-Retraining, the pre-quantized weights in \( S_{PQ} \) are quantized to low bit weights represented by \{\pm 2^n\}. Also, the weights in the first and last layers are retrained to compensate for the accuracy loss caused by quantization.

2.1. Phase 1: KD-Training

In this operation, the obtained pre-quantized model \( S_{PQ} \) with a compact structure is trained from the full-precision teacher model \( T_F \). The weights in different layers of \( S_{PQ} \) are constrained to near \{\pm 2^n\}, which can reduce the performance loss in the following quantization.

To obtain \( S_{PQ} \), a pre-quantized constraint loss for limiting weights of one layer in approximated \{\pm 2^n\} is proposed as (1):

\[
L_{QC} = \sum_i \sum_{j} f^{QC}(\sum |W_{i,j}|) \quad (1)
\]

where \( W_{i,j} \) is an element of the weight matrix of \( f^{th} \) layer. \( L \) is the total number of layers in the network. The formula of calculating \( f^{QC}(x) \) is as follows:

\[
f^{QC}(x) = \tan^2((\log_2(x)) \times \pi) \quad (2)
\]

It is obvious that \( f^{QC}(x) \) can get a minimum value at \{x = \pm 2^n\}. Therefore, \( L_{QC} \) can constrain \( x \) to \{\pm 2^n \pm \varepsilon \} (where \( \varepsilon \) is a very small value). As the quantized bit-width of weights is finite, \(|W_{i,j}| \) should be constrained in a certain range. Thus, the constraint range is calculated according to the following formula:
The pre-quantized weights of $S_l$ are represented as $S_{W_l}$ weights of pre-trained $S_l$ using the following formula:

The quantized formula of weights can be expressed as:

$$f^R(x,n_1,n_2) = \begin{cases} 
2^{n_1} \cdot \frac{1}{2^\delta} , & x < 2^{n_1} \cdot \frac{1}{2^\delta} \\
2^{n_1} \cdot \frac{1}{2^\delta} & 2^{n_1} \cdot \frac{1}{2^\delta} \leq x \leq 2^{n_1} \cdot \frac{1}{2^\delta} \\
x & x > 2^{n_1} \cdot \frac{1}{2^\delta} 
\end{cases} (3)$$

where $f^R(x,n_1,n_2)$ limits $|W_l(i,j)|$ to $[2^{n_1} \cdot \frac{1}{2^\delta} , 2^{n_1} \cdot \frac{1}{2^\delta}]$ ($n_1$ and $n_2$ are two integers, and they satisfy $n_2 \leq n_1$, $\delta$ is a very small value). The detailed value of $n_1, n_2$ can be calculated with bit-width and the $l$th layer weights of pre-trained $S_l$ before KD-Training. They are used to determine the constraint range of $W_l(i,j)$.

The formula of calculating $n_1$ is as follows:

$$n_1 = \text{floor}(\log_2(4 \max(\text{abs}(W_l))/3)) (4)$$

floor(·) is a round-down operation. max(·) can get the largest elements of its input. abs(·) is an absolute operation on the input. To represent the $b$-bit quantized weights, $n_2$ can be easily calculated by the following formula:

$$n_2 = n_1 \cdot 2^{b^{-1}} + 1 (5)$$

Since the gradient of $|W_l(i,j)|$ outside $[2^{n_1} \cdot \frac{1}{2^\delta} , 2^{n_1} \cdot \frac{1}{2^\delta}]$ is zero everywhere, we adopt the STE proposed by [2] to solve the problem that the gradient cannot be updated correctly during the back propagation.

The total pre-quantized constraint loss $L_{QC}$ is defined as follows:

$$L_{QC} = \sum_{l=1}^{L-1} L_{QC} (6)$$

With (6), the total loss can be a formula as (7), which combines KD and pre-quantization operation:

$$L_{total} = L_{KD} + \lambda L_{QC} (7)$$

$L_{KD}$ is the common loss function for KD-Training, which is used to guarantee the accuracy of $S_{PQ}$. In addition, the component $L_{QC}$ is used to constrain the weights to approximated $2^n$. $\lambda$ is a hyperparameter for adjusting ratio of two loss functions.

### 2.2. Phase 2: Quantization-retraining

The pre-quantized weights of $S_{PQ}$ are quantized to $\{\pm 2^n\}$ in the Quantization-Retraining phase. The full-precision weights of each layer in $S_r$ are represented as $W_{PQ}$. After phase 1, the $2^{th}$ to $(L-1)^{th}$ layers are pre-quantized values $W_1^{PQ}$. Then, first and the last layers of $S_{PQ}$ are retrained in phase 2. Finally, all weights of intermediate layers are quantized to $\{\pm 2^n\}$ (the $h$th layer quantized weight is indicated by $W_h^{Q}$).

During the Quantization-Retraining, $W_1^{PQ}$ is quantized to $Q=\{\pm 2^{1}, \pm 2^{2}, ..., \pm 2^{1}, \pm 2^{1}\}$. The quantized formula of weights can be expressed as:

$$W_1^{Q}(i,j) = \text{sign}(W_1^{PQ}(i,j)) \cdot f^Q(|W_1^{PQ}(i,j)|, n_1, n_2) \quad (8)$$

where sign(·) in (8) can obtain the sign of $W_1^{PQ}(i,j)$ and restore the sign of the elements of quantized weights. $n_1$ and $n_2$ are obtained using (4), (6) mentioned above. $f^Q(x,n_1,n_2)$ ensures that all parameters of $|W_1^{PQ}(i,j)|$ can be constrained to $2^n$. $f^Q(x,n_1,n_2)$ is defined as the following formula:

$$f^Q(x,n_1,n_2) = \begin{cases} 
\beta & 2^{n_1} \leq x \leq 2^{n_2} \\
\beta \frac{x}{2^{n_2}} & \text{otherwise} \\
\end{cases} \quad (9)$$
where $\alpha$ and $\beta$ are two adjacent values from sorted $Q$. We sequentially use the adjacent $\alpha$ and $\beta$ in $Q$ to determine the quantized interval and quantize the pre-quantized weights by $f^Q(x, n, n')$.

Considering that the first convolution layer and the last fully connected layer of the quantized network have a greater impact on the network accuracy, the weights of the first and last layers remain full-precision and are retrained to compensate for the accuracy loss caused by quantization.

### 3. Experiments and discussions

In this section, we will perform extensive experiments on CIFAR-10 and CIFAR-100 datasets to evaluate the proposed method. CIFAR-10 and CIFAR-100 consist of 10 and 100 classes respectively and both contain 50K training images and 10K testing images. For the training images, the same preprocessing method for two datasets are adopted, including padding 4 pixels on each side of the image, randomly cropping the padding image to 32×32, and then randomly flipping it horizontally. To demonstrate the robustness of the experiment, different bit-width quantization experiments are performed on the Resnet[4].

In the experiments, Resnet-32 is employed as teacher models for CIFAR10 and Resnet-20 is chosen as the student network as a compact structure.

The experimental results of different quantization methods on the CIFAR-10 dataset are summarized in Table 1.

**Table 1.** CIFAR-10 top-1 accuracy(%) results of Full precision, INQ and the proposed method.

| Model      | Params (M) | Full precision (32bit) | INQ (5bit) | INQ (4bit) | INQ (3bit) | Ours (5bit) | Ours (4bit) | Ours (3bit) |
|------------|------------|------------------------|------------|------------|------------|-------------|-------------|-------------|
| Resnet-32  | 1.85       | 92.65                  | 92.25      | 92.30      | 91.43      | -           | -           | -           |
| Resnet-20  | 1.07       | 92.13                  | 91.64      | 91.68      | 91.03      | 92.41       | 92.53       | 92.75       |

In Table 1, it can be found that the test accuracy of the quantized Resnet-20 obtained by the proposed method exceeds all quantized models with different bit in INQ[5]. The test accuracy of the proposed method is improved by 0.77%, 0.85% and 1.72% respectively compared with the 3-bit, 4-bit and 5-bit quantized Resnet-20 of INQ. Also, the quantized Resnet-20 with the proposed method can achieve about 42% parameter reduction while maintaining higher test accuracy. It is obviously that the quantized Resnet-20 obtained by the proposed method has fewer parameters and higher accuracy than the quantized Resnet-32 of INQ. Compared with the full-precision Resnet-32 and Resnet-20, the accuracy of the quantized ResNet-20 obtained by the proposed method at different quantized bit-width has negligible loss or even improvement, owing to the improved generalization ability.

### 4. Conclusion

In this paper, we propose a novel quantization algorithm combined with Knowledge Distillation for the compression and acceleration of deep neural networks. The algorithm includes two phases, KD-Training and Quantization-Retraining. In the first phase, we use the proposed pre-quantized constraint loss function combined with KD-Training to train a pre-quantized model with high precision, compact structure and pre-quantized weights. In the second phase, we quantize the pre-quantized weights by $\{\pm 2^n\}$ and retrain the first and last layers to make up for the accuracy loss caused by quantization. The experimental results on the benchmark dataset CIFAR-10 show that the proposed method can improve the performance and reduce parameters compared with other quantization methods.

**Acknowledgement**

This work was supported in part by the National Natural Science Foundation of China (No. 61802105, 61701154, 61702154, 61632007, 61976076), and Natural Science Foundation of Anhui Province (No. 1908085QF265 and 1808085QF185).
References

[1] K. Simonyan, A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations* (2015).

[2] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research.* 18, 6869–6898 (2017).

[3] G. Hinton, O. Vinyals, J. Dean. Distilling the knowledge in a neural network. Advances in neural information processing systems, 55 (2014).

[4] K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016).

[5] Zhou, A. Yao, Y. Guo, L. Xu, Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *International Conference on Learning Representations* (2017).