ISSN Cetak 2745-925X
ISSN Online 2722-273X

JESSI Volume 03 Nomor 1 May 2022

POST TRAINING QUANTIZATION IN LENET-5 ALGORITHM FOR EFFICIENT INFERENCE

Dary Mochamad Rifqie[1], Dewi Fatmarani Surianto[2], Nurul Mukhlisah Abdal3, Wahyu Hidayat M[4], Hartini Ramli[5]

[1]darifqie@gmail.com, [2]dewifatmaranis@gmail.com, [3]nm.abdal@unm.ac.id,
[4]wahyuh38@gmail.com, [5]hartiniramli023@gmail.com
[12345]Universitas Negeri Makassar

Received : 28 Feb 22
Accepted : 23 May 22
Published : 27 May 22

Abstract

Abstract: *While deep neural networks have become more powerful, there is an increasing desire to implement them in the real world. However, the energy consumption and accuracy of neural networks is large because of the size and complexity, making them difficult to implement in embedded devices. Neural network quantization has recently come to meet this demand to decrease the size and complexity of neural networks by reducing the precision of the weight and activaton. With smaller networks, it is possible to run neural networks at the edge. This paper studies about quantization that have been developed in the last few decade. In this study, we implement quantization in lenet-5 algorithm, that is the first convolutional neural network algorithm ever, and evaluated in MNIST and Fashion-MNIST datasets.*

Keywords: Post Training Quantization, Deep Neural Network, Inference, lenet-5

Abstrak

Ketika model jaringan saraf tiruan menjadi lebih baik , keinginan untuk mengimplementasikannya di dunia nyata semakin meningkat. Namun, konsumsi energi dan akurasi jaringan saraf tiruan sangat besar karena ukuran dan kompleksitasnya, sehingga sulit untuk diimplementasikan pada *embedded devices*. Kuantisasi jaringan saraf ini adalah sebuah teknik untuk dapat memecahkan masalah seperti mengurangi ukuran dan kompleksitas jaringan saraf tiruan dengan mengurangi ketepatan parameter dan aktivasi. Dengan jaringan yang lebih kecil, dimungkinkan untuk menjalankan jaringan saraf di lokasi yang diinginkan. Artikel ini mengkaji tentang kuantisasi yang telah berkembang dalam beberapa dekade terakhir. Dalam penelitian ini, kami mengimplementasikan kuantisasi dalam algoritma lenet-5, yang merupakan algoritma jaringan saraf convolutional pertama yang pernah ada, dan dievaluasi dalam dataset MNIST dan Fashion-MNIST.

Kata kunci: *Post Training Quantization*, Jaringan Saraf Tiruan, *Inference, lenet-5*

ISSN Cetak 2745-925X
ISSN Online 2722-273X

JESSI Volume 03 Nomor 1 May 2022

1. Introduction

Deep Neural Network (DNN) have shown an unprecedented success in machine learning task. This algorithm have achieved great results in many fields, which make it the most frequently used for machine learning applications. However, DNN is not efficient, especially for edge devices, because of the massive numbers of multiply-accumulate (MAC) operations required to calculate the weight parameter [1].

There is an increasing used of DNN for applications in embedded devices. This devices typically have low computing capabilities, and also are limited in energy consumption and memory [2]. Enabling DNN inference in resource-constrained device is important so that DNN can solve tasks like object detection, speech recognition, and image classification at the edge [3]. Therefore, in order to achieve that, there is a need for new techniques implemented in DNN so that it can achieve real-time latency, lower power consumption, and high accuracy [4].

These purposes is difficult to achieve because DNN are designed to reach high accuracy by having billions of parameters. In addition, these parameters are represented in 32-bit floating point numbers due to their high precision, meaning it needs expensive floating point operations to run [5]. The large sizes and computation requirements in DNN results major obstacles to obtain efficient and fast inference.

The complexity of DNN is problems that prevent an effective inference. In order to handle these issues, some approaches have been designed, such as designing a new DNN model, pruning, knowledge distillation, and quantization. These approaches have same purposes of reducing the size and complexity of DNN, and keeping the model in high accuracy [6].

This paper focuses on DNN quantization, converting high precision value in weight and activations (32-bit floating point) to a lower precision number (16 bit and 8 bit integer). Moreover, we used per-layer quantization in the model to quantize the filter in the model.

This study evaluates DNN quantization in lenet-5 algorithm. Lenet-5 is the first convolutional neural network algorithm and is proposed by Yann Lecun et al [7]. The proposed quantization method is evaluated with two different datasets (MNIST and Fashion-MNIST).

2. Methodology

2.1 Quantization and Dequantization

A quantization maps a floating point number $r \in [\alpha,\beta]$ to a b-bit integer number $X(r) \in [\alpha q, \beta q]$ by the formula of :

$$X(r) = \text{Round}\,(r/S) + Z$$

where S is a floating point scaling factor which specify step size quantization operation. Zero-point (Z) is an integer, that ensure zero is quantized with no error. This is necessary to make sure that operations like zero padding do not cause quantization error [8]. The round() function converts the scaled r to an integer through rounding. $X(r)$ is often referred a uniform quantization because all of r input values are scaled by the same value S, meaning the distance between quantized values is equal. It can be applied non-uniform spaces between quantized values, which is called as non-uniform quantization. Nevertheless, this topic is not discussed in this paper. In the study of quantizing DNN, r can be a weight or activation [8].

Recovering real values r from the quantized values $X(r)$ is possible through an operation that is often called dequantization:

$$r = S\,(X(r) + Z)$$

The recovered real values r will not be same as r because of the rounding operation[8].

2.2 Uniform Asymmetric Quantization

The Important point in uniform quantization is the choice of the scaling factor S. This essentially divides a range of real values r. The formula of finding a scaling parameter is :

$$S = (\beta - \alpha) / (2b - 1)$$

where $[\alpha,\beta]$ is the clipping range, a range that we are clipping the real values, and b is the quantization bit width. Therefore, if the scaling parameter want to be defined, we must first determined the clipping range, which often referred to calibration process. There are three different kind calibration processes, that is min/max calibration, KL divergence and percentile process. In this paper, we use the easiest way to implement, that is min/max calibration. If the value of clipping range parameter is not same, $-\alpha \neq \beta$,

ISSN Cetak 2745-925X
ISSN Online 2722-273X

JESSI Volume 03 Nomor 1 May 2022

then this process of quantization is called asymmetric quantization.

2.3 Uniform symmetric Quantization

A simple version of the quantization is the symmetric quantizer, where the zero-point is restricted to 0, and the value of clipping range is same, which is $-\alpha = \beta$. With the symmetric quantization, the conversion is simplified to:

$$X(r) = \text{Round}\,(r/S)$$

2.4 Granularity of Quantization

In convolutional neural network (CNN), the activation is convolved with many filters. Each of these filters can have a different range of values. There are 2 ways of quantizing filter in CNN model[9]. First, quantizer that have the same quantization parameter (scaling factor and zero point) is specified for an entire layer, often defined as per-layer quantization. Second, quantizater that has a different scale and zero-point for each convolutional kernel is assigned, often referred as per-channel quantization. Accuracy can be improved by adapting the per-channel quantization to each filter within the layer [9]. For instance, the weight in tensor is 4 dimensional and is a set of 3 dimensional filter, each responsible for producing one output feature map. Per-channel quantization is not recommended for activations as this would complicate the inner product computations at the core of MAC and convolutions operations[9].

3. Result and Discussion

3.1 Post Training Quantization

Fine tuning the parameters in the DNN after quantization is often necessary. This can be done by retraining the model, that is called Quantization Aware Training (QAT), or without retraining, a process that is called as Post Training Quantization (PTQ). In this paper, PTQ techniques are implemented due to the reasons that it simpler to use and allow for quantization with limited data. We do an experiment with different quantization schemes for weight only quantization and for both activations and weights.

3.2 Weight Quantization

A simple method is to only decrease the precision of the weights of the network to 8-bits or 16 bits from 32-bit float. When only the weights are quantized, it does not require calibration data. This method is useful if we only wants to decrease the model size for storage and transmissions, and does not mind the cost of computation in floating point numbers.

3.2 Weight and Activation Quantization

We can quantize a floating point number to a lower bit precision by calculating all of the quantizer parameters (activation and weight). When activations need to be quantized, we requires calibration data and needs to calculate the dynamic ranges in activations for the clipping range. Usually, around 100 mini-batches are sufficient to estimates the ranges of the activation to converge.

4. Experiment

For evaluating the DNN model with different quantization schemes, we study lenet-5 networks and evaluate the accuracy in the top-1 classification. Note that all results are obtained using schemes of weight only quantization and both weight and activations quantization, as depicted in figure 1. Moreover, we use per-layer quantization in filter of convolutional operator due to the simplicity of the implementation.

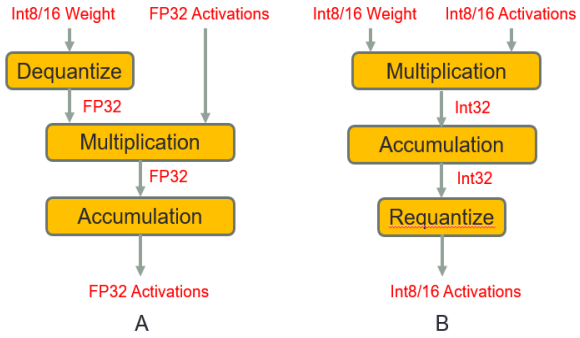

Figure 1. Scheme of quantization method. (A) This method just use quantized the weight, (B) and the other method using quantization method for both activations and weight.

We first do quantization for the weights only and leave the activations unquantized. From table 2, we compare the accuracy of the model where the weight is represented in 8-bit integer, 16-bit integer weight, and 32-bit floating point. In addition, we assign symmetric quantizer and use min/max for the clipping range. As depicted in figure 2, for example, we implement the min/max clipping range in the filter of convolution in layer 2.

ISSN Cetak 2745-925X
ISSN Online 2722-273X

JESSI Volume 03 Nomor 1 May 2022

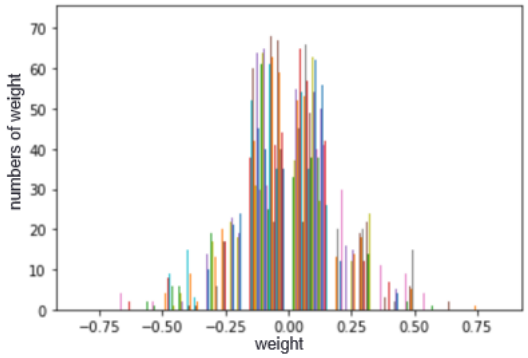

Figure 2. Histogram of convolutional filter in layer 2

In the table 1 below, we see that the model's accuracy in quantized 16-bit and 8-bit parameter have the same accuracy with the model that have 32-bit floating point weight. This accuracy match to real model (FP32) for both MNIST and Fashion-MNIST datasets.

Table 1. Accuracy of model in quantized weight only

Datasets	Accuracy		
	FP32	INT16	INT8
MNIST	98,33%	98,33%	98,33%
Fashion-MNIST	84,88%	84,88%	84,88%

The next step, we quantize weights and activations to 16-bits and 8-bits, with per-layer quantization scheme. For weights we consider use symmetric quantizers at granularities of a layer, and in activations we use asymmetric quatizer for the clipping range. The result of this method is showed in table 2 below.

Table 2. Accuracy of model in quantized weight and activation.

Datasets	Accuracy		
	FP32	INT16	INT8
MNIST	98,33%	98,33%	98,32%
Fashion-MNIST	84,88%	84,88%	84,86%

In the table above, the model's accuracy in quantized 16-bit have the same accuracy with the model that have 32-bit floating point weight. However, the quantized 8-bit model have a slightly decrease in accuracy by 0,01% in MNIST data set, and 0,02% in Fashion-MNIST datasets.

5. Conclusion

A Post-Training quantization in lenet-5 algorithm has been tested in MNIST and Fashion-MNIST datasets. We variate the bit width when converting the model into the quantized model (16-bit and 8-bit) and also use per-layer quantization in filter of convolutional layer. We also do an experiment for parameter that we want to quantize, in this case we quantize weight only, and both weight and activations. The result of this study shows that there is no decrease in accuracy of weight only quantization. This implies that quantized weight only has been successfully decrease the size of the parameter up to 4 times, and has advantages for storing and transmission. Moreover, the result of quantization in both weight and activations shows there is small decrease in accuracy of 8-bit quantization, while in 16-bit quantization shows no decrease compared to 32-bit floating number models. This gives advantages not only in storage and transmissions of the model, but also in computation of the models.

Reference

[1] R. Banner, "Scalable methods for 8-bit training of neural networks," in *Advances in neural information processing systems*, vol. 31, 2018.

[2] M. Denil, "Predicting parameters in deep learning," in *Advances in neural information processing systems 26*, 2013.

[3] B. Jacob, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2018.

[4] T. Liang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[5] O. Weng, "Neural Network Quantization for Efficient Inference:A Survey." 2021.

[6] M. Nagel, "A white paper on neural network quantization." 2021.

[7] Y. LeCun, "LeNet-5, convolutional neural networks." p. 14, lecun. com/exdb/lenet .5 2015. [Online]. Available: http://yann.

[8] Y. LeCun, "LeNet-5, convolutional neural networks." p. 14, lecun. com/exdb/lenet .5 2015. [Online]. Available: http://yann.

[9] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization." 2018.