Shift Quantization

Damb Matei, Markis Iustin, Kovacs Attila

January 2023

Abstract

This paper presents a comprehensive study of shift quantization for Convolutional Neural Networks and shows that it can be a promising technique for reducing the size of CNNs while preserving their accuracy.

Convolutional Neural Networks (CNNs) have become a popular and effective method for image and video analysis, natural language processing and many other tasks. Hovewer, the high memory and computational requirements of CNNs are a major challenge, especially for mobile devices, embedded systems and self-driving cars. One approach to address this problem is to reduce the precision of weights and activations of a CNN. Shift Quantization is a technique that aims to reduce the precision of the weights and activations by shifting their values by a fixed offset before quantization. This allows for a smaller number of bits to represent each value, resulting in a reduction of the memory and computational requirements for CNNs on various datasets and architectures.

We will investigate the performance of Shift-Quantized CNNs using the VGG-7 architecture on the MNIST dataset. We use a simple VGG-7 based CNN as our baseline and compare it to the results obtained with the Incremental Network Quantization (INQ) and the Alternating Direction Method of Multipliers (ADMM) on the same dataset and architecture.

Our results show that shift quantization can significantly reduce the size of a CNN without sacrificing too much accuracy.

This paper is organized as follows:in Section 1, there is a brief introduction, in Section 2, we will review the existing literaure survey. In Section 3, we will describe the principle of VGG-7 architecture and the INQ method's implememnation and in Section 4 the experimental results are presented. Finally, in Section 5 we will discuss the results and in Section 6 there is a brief conclusion.

1 Introduction

Neural networks are a powerful tool for solving complex problems in various fields. Convolutional Neural Networks are a specific type of neural network that are particularly well-suited for image and video analysis tasks. They are based on the idea of convolutional layers that extract features from images, and pooling layers that reduce the dimensionality of the features. This architecture allows CNNs to be highly effective in tasks such as image classification, object detection and image segmentation.

Shift Quantization is a technique used to reduce the size of a convolutional neural network (CNN), while trying to preserve its accuracy. It is a specific type of quantization that reduces the number of bits used to represent the weights and activations of a CNN by shifting the values to a new range by multiplying them with a scaling factor(fixed offset), allowing a smaller number of bits to represent each value. The scaling factor is determined by the desired number of bits and the range of the values. Simply, it can be taken as $2^{number-of-bits-1}/original - range$. The scaled values have to be rounded. The rounding step can be done using different rounding methods such as round to the nearest, round towards zero, etc. The choice of rounding method can affect the final accuracy of the quantized model.

The main advantage of Shift Quantization is that it can be applied to a wide range of neural network architectures. The smaller number of bits allows to reduce the memory and computation requirements of CNNs, making it useful for deploying deep learning models on resource-constrained devices such as self-driving cars, embedded systems and IoT devices.This technique can be used to significantly reduce the size of a CNN without sacrificing too much accuracy.

Despite its advantages, shift quantization also poses several challenges and limitations, such as noise, distortion and computational complexity. The goal is to find the optimal trade-off between model size and accuracy, investigating the performance of two methods: Incremental Shift Quantization (INQ) and the Alternating Direction Method of Multipliers (ADMM) (not done).

This paper will begin by reviewing the existing literature on shift quantization, and then present experimental results and analysis on a VGG-7 based CNN on the MNIST dataset, comparing the baseline with two shift quantization methods: INQ and ADMM (ADMM not done).

2 Literature survey

The reduction in size of convolutional neural networks and computational power needed to run them is an important topic and many papers approached the subject. The CNN achieved great success in computer vision tasks as image classification or object detection. Deep CNN's tend to have many model parameters that can have quite a significant impact, demanding such a high computational power and a generous storage space, especially where those are limited due to various reasons. To address the matter,a diverse range of solutions were explored, including Shift Quantization and optimization with ADMM [1]. The latter one, consisting in the representation the network's weights by a very small number of bits and modelling the problem as a discretely constrained optimization problem. Continuous parameters are decoupled from the discrete constraints of the network and cast the original hard problems into several sub-problems. These sub-problems are solved using extragradinent and iterative quantization algorithms.

The Incremental Network Quantization is a novel method, targeting to efficiently convert any pre-trained full-precision convolutional neural network model into a low-precision version whose weights are constrained to be either powers of two or zero[3]. The methods that already exist are suffering from accuracy loss, this method having the possibility to solve this matter. Three idependent operations are intrduced,weight partition, group-wise quantization and re-training.A well proven measure is employed to divide the weights in each layer of a pre-trained CNN model into two disjoint groups. The weights in the first group are responsible to form a low-precision base, thus they are quantized by a variable-length encoding method. The weights in the other group are responsible to compensate for the accuracy loss from the quantization, thus they are the ones to be re-trained. On the other hand, these three operations are repeated on the latest re-trained group

in an iterative manner until all the weights are converted into low-precision ones, acting as an incremental network quantization and accuracy enhancement procedure.

Other techniques may be: Neural architecture search, Low-rank factorization, Knowledge distillation, Hardware-specific optimization. In the future an optimal method may be obtained using a combination of these.

3 Implementation

To prof the concept of neural network quantization a classification problem was selected, more specific, the chosen task is to classify handwritten digits. To accomplish that the MNIST dataset was used to train and test the neural network. A VGG-7 architecture featuring four convolutional layers and three fully connected ones was used for our model. The diagram of our architecture can be seen in figure 1. Specific to the VGG architecture the convolution is done using 3x3 filters with same padding and the pooling is done with a 2x2 window with stride 2 and without padding. The function of choice for activation was ReLU and as for the down-sampling operation max pooling was used.

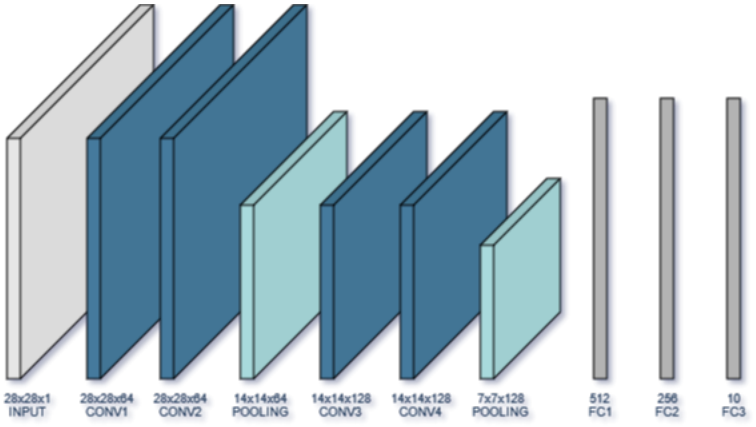

Figure 1: VGG-7 architecture design

3.1 Baseline

Using the VGG-7 [2] architecture a handwritten digit classifier was developed in floating point precision to serve as a baseline to compare the results of network quantization. To train the baseline we chose stochastic gradient descent for the optimization algorithm using a learning rate of $1e-2$. After training we obtained a model with 98.45% accuracy on testing data.

3.2 Incremental Network Quantization

In order to achieve faster computational times and compress a floating point precision neural network with high performance while also keeping the loss in accuracy negligible we used the Incremental Network Quantization technique [3].

This method consists in three inter-dependent steps done in a iterative fashion. Namely group-wise partition where a number of weight is selected on a pruning-inspired measure, group-wise quantization where the partitioned weights are quantized in either power of two values or zero and then, after quantization, the selected group of weights the retraining can begin to adjust the possible errors introduced by the quantization operation.

Given a weight matrix W_l, where l represents the layer index, we want each of its entry to be part of the set $P_l = \{\pm 2^{n_2}; \ldots; \pm 2^{n_1}; 0\}$. Where n_1 and n_2, with the property $n_1 \geq n_2$ bound the interval. The n_1 is found by using the equation (1) which basically finds the biggest exponent needed to quantize the weights.

$$n_1 = \left\lfloor \log_2 \left(\frac{4}{3} max\{|W_l|\} \right) \right\rfloor \tag{1}$$

Considering the fact that a bit is needed to represent the zero value and by setting b as the bit width as initial condition we can obtain n_2 as follows:

$$n_2 = n_1 + 1 - 2^{b-1} \tag{2}$$

Finally to quantize every entry of a weight matrix W_l the equation (3) is used. The equation translates into finding the nearest power of two approximate of a specific weight.

$$\hat{W}_l(i,j) = \begin{cases} sign\{W_l(i,j)\} * 2^n; & n_1 \geq n \geq n_2 \\ 0, & \text{otherwise} \end{cases} \qquad \text{where } n = \left\lfloor \log_2 \left(\frac{4}{3} |W_l(i,j)| \right) \right\rfloor \tag{3}$$

To quantize the whole neural network those steps need to be done in a iterative fashion. In this example the group-wise partition size is divided in 4 steps: $\sigma = \{50\%, 75\%, 85.5\%, 100\%\}$ as was done in the original INQ paper [3]. To implement the method a tensor T_l was used to keep track of the blocked weights as the partition changed, having zero on the position corresponding to a non-quantized weight and ones for the blocked weights. Moreover to make sure that the quantized weights are not updating while training, using the mask tensor T_l, the corresponding gradients are set to zero so that after the optimizer takes a step just the non-quantized weights should be updated. To gain a better view of the algorithm see it's pseudo-code illustrated below:

Algorithm 1 Incremental network quantization

Require: W - neural network weights; $\sigma = \{\sigma_0, \sigma_1, \ldots, \sigma_N\}$ - partition sizes; b - bit width
Ensure: $\hat{W}$ - quantized weights

```
T ← zeros_like(W)
for i = 0 : N − 1 do
    W ← quantize_weights(W, T, σ_i, b)        ▷ group-wise partitioning and quantization
    for each epoch do
        W ← W − lr * ∂L/∂W * (ones_like(T) − T)        ▷ updating just non-quantized weights
    end for
end for
W ← quantize_weights(W, T, 1, b)
```

4 Experiments

The goal is to achieve a compressed size neural network that can run faster using shift while keeping the performance. To test those assumptions we trained a handwritten digit classifier using the MNIST dataset in floating point precision to serve as a baseline and then the obtained model was quantized so that we can measure those metrics by comparison.

A critical metric that needs to be verified is the change in accuracy that appears after quantization so the baseline went through the quantization operation multiple time while changing the bit width to see how low is possible to go and also mentain accuracy.

To check how much the network can save memory we determined the space needed to store our floating precision network and compare it with the memory needed to save a quantized neural network with different bit-width values.

It is wisely to check the reliability of the proposed method of quantization. A one time happy result can be a lucky try, we don't have a guarantee that the accuracy will always be with negligible loss. To test the reliability of the method multiple tries were done with the same bit-width to see how much can the performance vary.

5 Results

5.1 Influence of bit-width variation on accuracy

To see how variant is the accuracy of the quantized model using different bit-widths we considered taking test on a range from 8 to 3 bits. The results are illustrated in table 1.

Bit-width	Accuracy [%]	Difference [%]
(Full precision) float32	98.45	-
8	98.77	+0.32
7	98.74	+0.29
6	98.66	+0.21
5	98.70	+0.25
4	98.63	+0.18
3	97.76	-0.69

Table 1: Bit-width accuracy variation

We can see that the proposed method of quantization behaves invariant of bit-width on this dataset. It is worth to note that even with very low precision quantization the effect of retraining corrects and adapts the network to the quantization changes and thus keeping the accuracy variation negligible.

5.2 Compression

Another important matter is to check how mach memory can be saved through quantization. The network architecture on this dataset gives a total of 3603520 trainable weights. Table 2 shows the compression that can be achieved with various bit-width values.

Bit-width	Size [MB]
(Full precision) float32	14.4
5	2.25
4	1.9
3	1.4

Table 2: Size compression

The results show that the size can be reduced about ten times and so being easier to integrate on a mobile or limited hardware device.

5.3 Reliability

To eliminate the thought of a quantization operation as being a lucky try we applied quantization multiple times for a bit-width of 3 on the same fully precision neural network baseline that had initially 98.45% accuracy. The results obtained are around a mean of $\mu = 97.76\%$ with a standard deviation of $\sigma = 0.006\%$.

6 Conclusion

In conclusion, we implemented our version of INQ and demonstrated that the solutions indeed works. After quantization, the size of the networks was compressed but the accuracy was not affected ,we even obtained better results that the floating point precision baseline for a bit-width ranging from 8 to 4. For the bit-width of 3 the loss in accuracy is negligible. The full results are presented below.

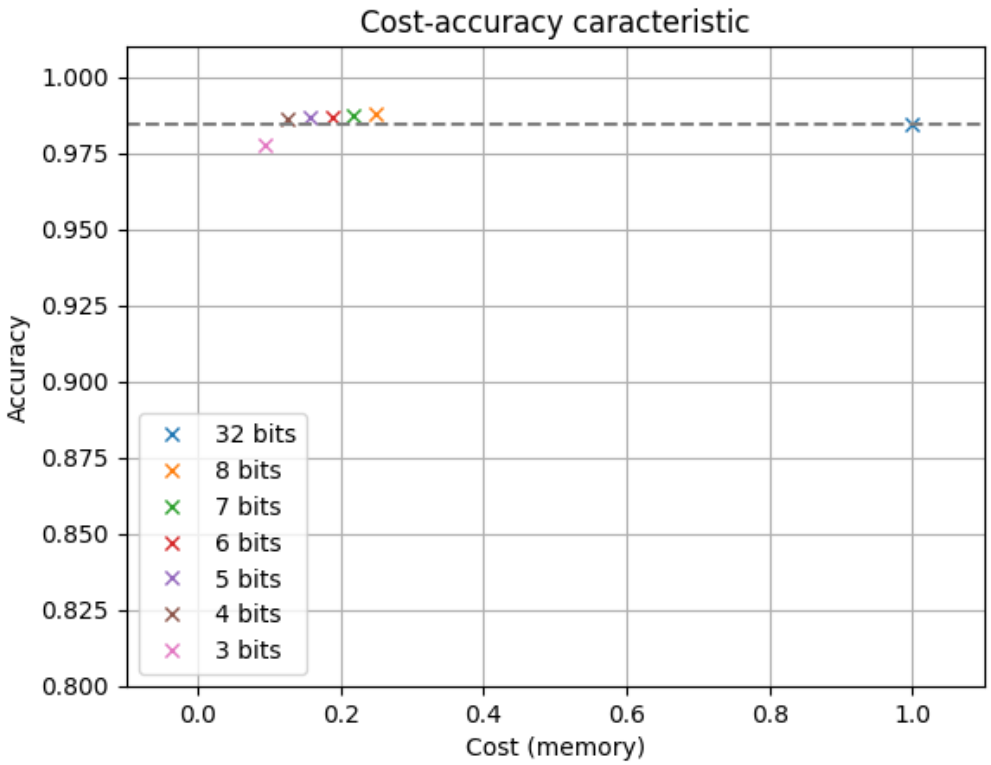

Figure 2: Accuracy and Cost Graph

References

[1] Cong Leng et al. *Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM*. 2017.

[2] Fengfu Li et al. *Ternary Weight Networks*. 2016.

[3] Aojun Zhou et al. *Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights*. 2017.