Instant Quantization of Neural Networks using Monte Carlo Methods

Gonçalo Mordido[*][†]
Hasso Plattner Institute
Potsdam, Germany
goncalo.mordido@hpi.de

Matthijs Van Keirsbilck[*]
NVIDIA
Berlin, Germany
matthijsv@nvidia.com

Alexander Keller
NVIDIA
Berlin, Germany
akeller@nvidia.com

Abstract

Low bit-width integer weights and activations are very important for efficient inference, especially with respect to lower power consumption. We propose Monte Carlo methods to quantize the weights and activations of pre-trained neural networks without any re-training. By performing importance sampling, we obtain quantized low bit-width integer values from full-precision weights and activations. The precision, sparsity, and complexity are easily configurable by the amount of sampling performed. Our approach, called Monte Carlo Quantization (MCQ), is linear in both time and space, with the resulting quantized, sparse networks showing minimal accuracy loss when compared to the original full-precision networks. Our method either outperforms or achieves competitive results on CIFAR-10 and ImageNet when compared to previous methods that do require additional training.

1 Introduction

Developing novel ways of increasing the efficiency of neural networks is of great importance due to their widespread usage in today's variety of applications. Reducing the network's footprint enables local processing on personal devices without the need for cloud services. In addition, such methods allow for reducing power consumption - also in data centers. Very compact models can be fully stored and executed on-chip in specialized hardware like for example ASICs or FPGAs. This reduces latency, increases inference speed, improves privacy concerns, and limits bandwidth cost.

The precision of the network weights and activations can be lowered while making use of sparsity, allowing for efficient hardware implementations that reduce the cost or eliminate most floating point operations. Quantization methods usually require re-training of the quantized model to achieve competitive results. This leads to an additional cost and complexity. The proposed method, Monte Carlo Quantization (MCQ), aims to avoid retraining by approximating the full-precision weight and activation distributions using importance sampling. The resulting quantized networks achieve accuracies close to the full-precision counterparts without the need for additional training. Most importantly, the complexity of the resulting quantized networks is proportional to the number of samples taken, with MCQ being linear in both time and space in the number of weights and activations.

First, we normalize the weights and activations of a given layer to treat them as probability distributions. Then, we randomly sample from the corresponding cumulative distributions and count the number of hits for every weight and activation. Finally, we approximate the weights and activations by their integer count values, by performing a discrete approximation of the original continuous values. Since the quality of this approximation relies entirely on random sampling, the accuracy of the quantized model is directly dependant on the amount of sampling performed. Thus, accuracy may be traded for higher sparsity and speed by adjusting the number of samples.

[*]Equal contribution.
[†]Work done during a research internship at NVIDIA.

2 Related Work

Low precision and sparsity allow for efficient hardware implementations that eliminate most floating point operations and skip operations on zero values. Sparsification through iterative pruning is known to work well [3], but comes with the increase of the training cost. Another approach is to modify the training procedure so the network learns to deal with low-precision weights and/or activations.

BinaryConnect [1] proposes training networks with binary weights, while XNOR-Net [10] and BNN [4] extend this binarization to both weights and activations. TWNs [6] proposes to quantize using ternary weights instead to increase the model expressiveness. Similarly, TTQ [16] uses ternary weights with a positive and negative scaling learned during training. LR-Net [11] uses both binary and ternary weights by using stochastic parameterization while INQ [14] constrains weights to powers of two and zero. FGQ [8] divides weights in groups and uses different scaling factors to minimize the element-wise distance between full and low precision weights. Recently, [13] proposed outlier channel splitting, which complements our approach by reducing the large bit-width required by bigger weights and activations after quantization.

Similarly, quantization techniques can also be applied in the backward pass to quantize gradients and increase training speed as well [15, 2, 1]. In particular, RQ [7] propose a differentiable quantization procedure to allow for gradient-based optimization using discrete values and Wu et al. [12] recently proposed to discretize weights, activations, gradients, and errors both at training and inference time.

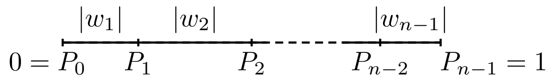
3 Neural Networks and Monte Carlo Methods

Many state-of-the-art networks use ReLU as their activation function, which has interesting properties such as scale-invariance. This enables a scaling factor to be propagated through all network layers without affecting the network's original output. This principle can be used to normalize network values, such as weights and activations, without affecting the network output. Given weights $w_{l-1,i,j}$ connecting the i-th neuron in layer $l-1$ to the j-th neuron in layer l, where $i \in [0, N_{l-1}-1]$ and $j \in [0, N_l - 1]$, with N_{l-1} and N_l being the number of neurons of layer $l-1$ and l, respectively. Let $a_{l,j}$ be the j-th activation in the l-th layer and f a positive number $\mathbb{R}^+$:

$$a_{l,j} = max\left\{0, \sum_{i=0}^{N_{l-1}-1} w_{l-1,i,j}a_{l-1,i} + b_{l,j}\right\} = f \cdot max\left\{0, \frac{\sum_{i=0}^{N_{l-1}-1} w_{l-1,i,j}a_{l-1,i} + b_{l,j}}{f}\right\}.$$

With f as the L1-Norm, these values can be treated as probabilities, enabling the simulation of discrete probability densities by constructing a probability density function (PDF) and then sampling from the corresponding cumulative density function (CDF). The number of references of a weight/activation is then the quantized approximation of its continuous value. The following discussion is within the weight quantization scope, however, the same process is applied for activations at inference time.

Without loss of generality, given n weights, assuming $\sum_{k=0}^{n-1} |w_k| = \|w\|_1 = 1$ and defining a partition of the unit interval by $P_m := \sum_{k=1}^{m} |w_k|$, we have the following partitions:



(1)

Then, given N uniformly distributed samples $x_i \in [0,1)$, we approximate the weight distribution with $\sum_{j=0}^{n-1} w_j a_j \approx \frac{1}{N} \sum_{i=0}^{N-1} \underbrace{\text{sign}(w_{j_i})}_{\in\{-1,0,1\}} \times a_{j_i}$, where j_i is uniquely determined by $P_{j_i - 1} \leq x_i < P_{j_i}$.

We use jittered equidistant (stratified) sampling, to ensure uniform distribution. Given a random variable $\xi \in [0,1)$, we generate N samples $x_i \in [0,1)$ such that $x_i = \dfrac{i+\xi}{N}$, where $i \in \{0, \dots, N-1\}$. This also reduces the cost of the sampling process from $\mathcal{O}(NlogN)$ to $\mathcal{O}(N)$, as searching for the value corresponding to a sample no longer requires a binary search. Sorting values before creating the PDF groups smaller values together and, due to uniform sampling, they are sampled less often. This produces higher sparsity and a better approximation of the larger values. Due to the additional cost of $\mathcal{O}(NlogN)$, we only applied sorting to our CIFAR-10 experiments, but not to ImageNet.

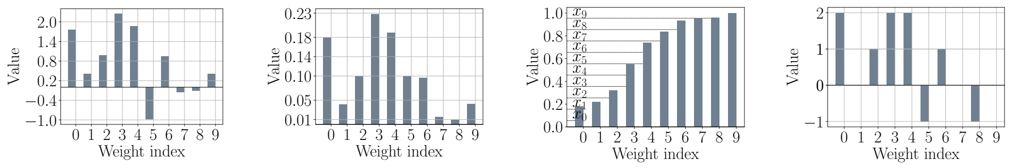
4 Monte Carlo Quantization (MCQ)

Our approach builds on the aforementioned principles of network normalization and quantization using Monte Carlo methods to quantize the weights and activations of pre-trained full-precision neural networks. While we mainly focus on the procedure for weight quantization throughout this Section, which is performed offline, activations are also quantized online in a similar manner during inference time. Our method, called Monte Carlo Quantization (MCQ), may be summarized by the following steps, which are executed layer by layer (also illustrated in Figure 1):

(1) Create a PDF for all $N_{l,w}$ weights of layer l such that $\sum_{i=0}^{N_{l,w}-1} |w_{l,i}| = 1$.

(2) Perform importance sampling on the weights based on their magnitude by sampling from the corresponding CDF and counting the number of hits per weight.

(3) Replace each weight with its quantized integer value, *i.e.* its hit count while maintaining its original sign, to obtain a low bit-width, integer weight representation.



(a) full precision weights (b) PDF (c) sampling on CDF (d) integer weights

Figure 1: Starting from full-precision weights (a), we create a PDF of the absolute values (b) and sample with jittered equidistant samples from the corresponding CDF (c). The sampling method produces quantized integer network weights based on the number of hits per weight (d). Note that weights 1, 7, and 9 were not hit, creating sparsity in the quantized network.

Performing normalization neuron-wise, as in Section 3, may result in an inferior approximation when the number of weights is small due to sampling errors, *e.g.* first convolutional layers. We propose to normalize all neurons simultaneously in a layer-wise manner by using the 1-norm of all weights of a layer l as the scaling factor f. Thus, each normalized weight can be seen as a probability with respect to all connections between layer $l-1$ and layer l, instead of a single neuron. This creates an interesting connection between MC-sampled networks, random forests and conditional computation networks [5]. An additional advantage is that samples can be redistributed from low-importance to high-importance neurons, resulting in an increased level of sparsity and better global approximation.

As introduced in Section 3, we generate ternary samples (hit positive weight, hit negative weight, or no hit), and count such hits during sampling. Note that even though the individual samples are ternary, the final quantized values may not be since each value can be sampled multiple times. For jittered sampling, we use one random offset per layer, with a number of samples $N = K \cdot N_{values}$, where $K \in \mathbb{R}^+$ is a user-specified parameter to control the number of samples and N_{values} is the number of weights or activations of a given layer. By varying K, computational cost can be traded off for better approximation, with generally lower accuracy loss. In our experiments, K is set the same for all network layers.

5 Experiments

The proposed method was evaluated on CIFAR-10 and ImageNet using a variety of models. The quantization level is indicated by the average number of bits used to store weights and activations on each layer, with 1 extra bit being used to represent the sign on weights. Comparison results are presented in the accuracy difference, Δ, between the quantized and the baseline models reported in each of the respective works to ensure a fair comparison. An extended version of this report with additional experiments on NLP tasks is available in [9].

5.1 CIFAR-10

The baseline VGG-7 model was trained for 300 epochs with Adam, initial learning rate of 0.1 reduced by factor 10 at epochs 150 and 225, batch size of 128, and weight decay of 0.0005.Table 1 shows results of quantization with MCQ using $K = 1.0$. We observe that MCQ achieves minimal accuracy loss, outperforming or being competitive with existing methods that do require additional training. Figure 2 shows the effects of varying the sampling amount, *i.e.* changing K, on the accuracy, percentage of used weights and activations, and number of bits for weights and activations. We observe a rapid accuracy increase, with good accuracy even with high sparsity and low bit-width.

Table 1: Comparison results on CIFAR-10. Quantizing only weights can outperform the baseline. Quantizing both weights and activations only slightly degrades accuracy. Quantizing the first layer results in additional accuracy loss.

METHOD	VGG-7
FULL PRECISION (32W-32A)	91.23
Δ MCQ (QUANTIZED W)	-0.48 / +0.04[a] (6.1W-32A)
Δ MCQ (QUANTIZED A)	-0.12[a] (32W-5.68A)
Δ MCQ (QUANTIZED W + A)	-0.58 / -0.13[a] (6.1W-5.6A)
Δ TWNS (2W-32A) [6]	-0.06
Δ BC (1W-32A) [1]	+0.74
Δ BNN (1W-1A) [4]	+0.49[a]
Δ BWN (1W-32A) [10]	-0.36 / +0.76[a]
Δ XNOR-NET (1W-1A) [10]	+0.47[a]
Δ RQ (8W-8A)) [7]	+0.25
Δ LR-NET (2W-32A) [11]	-0.11[b]

[a]Not quantizing the first layer.
[b]Not quantizing the last layer.

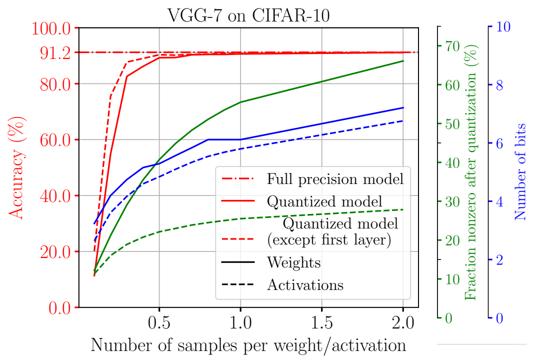


Figure 2: Effects of using different sampling amounts. The quantized models start to reach close to full-precision accuracy at ≈ 40% the weights and ≈ 20% of the activations using ≈ 5 bits. Not quantizing the first layer allows for better accuracy at higher sparsity and lower bit-width.

5.2 ImageNet

We further evaluated our approach on ImageNet using AlexNet, ResNet-18, and ResNet-50 with the pre-trained weights provided by Pytorch's model zoo. Table 2 shows MCQ's results with $K = 5.0$ on the different models, quantizing both weights and activations. MCQ quantized models achieve close to full-precision accuracy. Figure 3 shows the effects on the quantized model of varying K.

Table 2: Comparison results on ImageNet. When quantizing weights, accuracy drops less than 1% in all tested models, whereas quantizing activations generally leads to a lower drop. Quantizing both weights and activations leads to an additional accuracy loss of 0.6% in the worst case, *i.e.* ResNet-50. We note that FGQ makes use of higher precision (8w-8a) on their first layer.

METHOD	ALEXNET	RESNET-18	RESNET-50
FULL PRECISION (32W-32A)	56.52	69.76	76.13
Δ MCQ (QUANTIZED W)	-0.99 / -0.68[a] (8.00W-32A)	-0.72 / -0.63[a] (8.00W-32A)	-0.73 / -0.20[a] (8.28W-32A)
Δ MCQ (QUANTIZED A)	+0.02[a] (32W-8.36A)	-0.58[a] (32W-7.36A)	-0.76[a] (32W-7.45A)
Δ MCQ (QUANTIZED W + A)	-1.05 (7.88W-8.46A) / -0.75[a] (8.00W-7.2A)	-1.13 (8.00W-7.35A) /-1.03[a] (8.00W-7.36A)	-1.64 (8.26W-7.43A) / -1.21[a] (8.28W-7.45A)
Δ FGQ (2W-8A) [8]	-7.79	-	-4.29
Δ TTQ (2W-32A) [16]	+0.3[a,b]	-3.0[a,b]	-
Δ TWNS (2W-32A) [6]	-2.7[a,b]	-4.3[a,b]	-
Δ BWN (1W-32A) [10]	+0.2	-8.5[a,b]	-
Δ XNOR-NET (1W-1A) [10]	-12.4	-18.1[a,b]	-
Δ DOREFA (1W-32A) [15]	-3.3[a,b]	-	-
Δ INQ (5W-32A) [14]	-0.15	-0.71	-1.59
Δ RQ (8W-8A) [7]	-	+0.43	-
Δ LR-NET (2W-32A) [11]	-	-6.07[a]	-

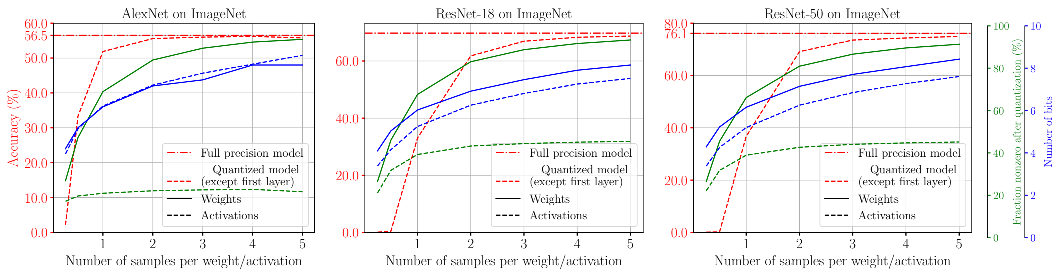


Figure 3: Effects of using different sampling amounts. All quantized models reach close to baseline accuracy at ≈ $K = 3$, with more samples required compared to CIFAR-10.

6 Discussion

The experimental results show low accuracy loss on CIFAR-10 and ImageNet for sparse, quantized models compared to the full-precision counterparts. MCQ either outperforms or is competitive with existing quantization methods that require additional training of the quantized network. A limitation of MCQ is that it often requires a higher number of bits to represent the quantized values than retraining methods. Nevertheless, this sampling-based approach mathematically leads to an arbitrarily good approximation of the full-precision values and high accuracy without retraining. Moreover, the trade-off between accuracy, sparsity, and bit-width can be controlled by adjusting the amount of sampling, while the complexity of the quantized network can be controlled by varying the number of samples. Ultimately, the use of sparse integer weights and activations throughout the network can enable efficient hardware implementations.

References

[1] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[2] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1737–1746, Lille, France, 07–09 Jul 2015. PMLR.

[3] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[4] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[5] Yani Ioannou, Duncan P. Robertson, Darko Zikic, Peter Kontschieder, Jamie Shotton, Matthew Brown, and Antonio Criminisi. Decision forests, convolutional networks and the models in-between. *CoRR*, abs/1603.01250, 2016. URL `http://arxiv.org/abs/1603.01250`.

[6] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[7] Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. Relaxed quantization for discretized neural networks. *arXiv preprint arXiv:1810.01875*, 2018.

[8] Naveen Mellempudi, Abhisek Kundu, Dheevatsa Mudigere, Dipankar Das, Bharat Kaul, and Pradeep Dubey. Ternary neural networks with fine-grained quantization. *arXiv preprint arXiv:1705.01462*, 2017.

[9] Gonçalo Mordido, Matthijs Van Keirsbilck, and Alexander Keller. Instant quantization of neural networks using monte carlo methods. *arXiv preprint arXiv:1905.12253*, 2019.

[10] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[11] Oran Shayer, Dan Levi, and Ethan Fetaya. Learning discrete weights using the local reparameterization trick. *arXiv preprint arXiv:1710.07739*, 2017.

[12] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. *CoRR*, abs/1802.04680, 2018. URL `http://arxiv.org/abs/1802.04680`.

[13] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *International Conference on Machine Learning*, pages 7543–7552, 2019.

[14] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[15] 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.

[16] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.