Recipes for Post-training Quantization of Deep Neural Networks

Ashutosh Mishra, Christoffer Löffler and Axel Plinge
Fraunhofer Institute for Integrated Circuits IIS, Erlangen/Nuremberg, Germany

Abstract

Given the presence of deep neural networks (DNNs) in all kinds of applications, the question of optimized deployment is becoming increasingly important. One important step is the automated size reduction of the model footprint. Of all the methods emerging, post-training quantization is one of the simplest to apply. Without needing long processing or access to the training set, a straightforward reduction of the memory footprint by an order of magnitude can be achieved. A difficult question is which quantization methodology to use and how to optimize different parts of the model with respect to different bit width. We present an in-depth analysis on different types of networks for audio, computer vision, medical and hand-held manufacturing tools use cases; Each is compressed with fixed and adaptive quantization and fixed and variable bit width for the individual tensors.

1 Introduction

Artificial neural networks are becoming the de facto solution for many tasks, but are often too large to run on restricted hardware such as smartphones. By reducing the memory footprint of the model, the energy consumption is directly reduced [13]. At the same time, for distributed model updates the models can be distributed faster as the amount of data to be transferred is reduced.

The discipline of "deep compression" [3, 10] focuses on optimizing existing networks by structural or implementation changes. Most known techniques may be categorized in four principles: The first, pruning, is the automated removal of less significant weights, filters or neurons. This typically requires retraining to be effective [3]. Second, low-rank approximation allows to directly remove the redundancies within the tensors via singular value decomposition (SVD) [18]. Third, quantization can be used to reduce the bit width of the weights themselves [3]. Finally, entropy coding can be employed to further reduce the memory footprint in bytes on disc [17].

There are very diverse quantization methods, which we might subdivide by the actual numeric quantization and the scope of weights considered. Numerically, there are uniform and adaptive schemes. In uniform quantization, the dynamic range is directly mapped to integer values. Different mappings are possible [9]. For reducing to fewer bits, it is prudent to use quantization aware training in order not to lose performance [15]. This requires access to the training data and significant processing time. Adaptive quantization is based on the actual distribution of the weight values. It is implemented as vector quantization methods on the weights. During inference, recovering the scalar weights simply reduces to a lookup operation of the cluster index from a codebook. Regarding the scope, one can apply these methods either globally or per layer/tensor. If there are multiple quantizations, the question of bit width selection arises. Optimal bit widths for individual tensors can preserve the performance of the model with a reduced model size [14]. The selection of these optimal quantization levels leads to a huge optimization search space. Meng et al. introduced a greedy approach to automatically find a good bit assignment for the individual tensors [8].

In this paper, we investigate quantization-based methods, compare several quantization strategies, and evaluate the results. They aim to reduce the precision of model parameters while still maintain-

6th Workshop on Energy Efficient Machine Learning and Cognitive Computing (EMC2 2020), Online (San Jose, California).

ing the model accuracy on the target task. We propose an extension of the greedy method by Meng et al. [8]. We add a pre-calibration step to achieve better or similar bit assignments faster. On four different network structures for very diverse use cases, we compare these two with using a global or tensorwise quantization. We employ both uniform and adaptive quantization for all of them. To the best of our knowledge, this is the first comparative analysis of all these different variants.

2 Quantization Methods

The task is to search an alternate lower bit representation of weight tensors. Given pre-trained and batch-norm folded [6] coefficients of a neural network in full precision, our aim is to find an alternate, reduced bit representation of these weight coefficients. We do not address the question of quantizing the biases, but restrict ourselves to filter coefficients from convolutional layers and weights of fully connected layers. In this paper, we compare the following methods and scopes of quantization.

In the scope of our experiments, we use uniform and adaptive quantization. In uniform quantization **UQ-**, we can apply symmetric quantization by using the maximum absolute value as scaler [9]. In adaptive quantization **AQ-**, encoding of the weights is done by using a clustering algorithm on the collected scalar weights. For our work, we use $K = 2^n$ as the codebook size for n bits with k-means++ initialization.

One can apply the weight quantization to the whole model at once, looking at all weights as one big set of numbers. We evaluate this method by using a constant step size based on all the weights of the model or a single codebook and refer it as **-G** throughout this paper. On the contrary, using tensorwise step sizes or codebooks with the same fixed number of bits is denoted by **-TF**.

One approach for assigning different bit widths per tensor is the greedy approach from [8]: A fixed global bit width is set for all the tensors. The tensors are processed in order of decreasing size. For a tensor in current scope, the current bit width is reduced by one before the model is evaluated on a calibration set. If the model performance drop is within a predefined threshold, the bit width reduction for the current tensor is repeated until the performance drop threshold is violated, upon which the previous best bit width is set for the current tensor. This is designated **-TA**.

We propose to improve the performance of this approach in terms of convergence speed and greedy bit assignments by computing an optimal global bit width for the previous method as a preprocessing task. We assign a starting global bit width which is gradually reduced until the model performance drop is beyond 0.3%. The previous best bit width is now used as the fixed global bit width for the above state-of-the-art greedy method. The benefit of adding the preprocessing task is two-fold, first it reduces the overall search-space for the method and therefore reduces the total number of evaluations needed for convergence. Second, it produces bit-assignments which perform either better or comparable to the other approaches in our experiments. It is referred to as **-T2**.

3 Models and Datasets

Most studies on compression are evaluated only in a single domain task, mostly image or audio classification. In this paper, we compare the methods on four networks with different architectures trained with data from diverse application domains.

VGG-like for Acoustic Event Classification We use a model for the general-purpose audio tagging task of the 2018 edition of the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge and its dataset [2]. The model is a VGG-13 like network [5] built in Keras; It consists of 10 convolutional layers with 3×3 filters and increasing channel counts followed by a fully connected layer with softmax activation.

Resnet for Image Classification For evaluating the quantization methodologies on image domain task, we use ImageNet ILSVRC2012 dataset [11]. We use a pretrained Resnet18 network [4] from the Pytorch model zoo.

UNet for MRI Brain Segmentation As another orthogonal domain to benchmark our approach is the medical domain. Therefore, we use a pre-trained U-Net implemented in Pytorch and a corresponding brain segmentation dataset [1].

FCN for Hand-Tool Activity Recognition The system in [7] detects activities performed in manual labor with hand-held tools, in order to enhance quality assurance for various industrial domains,

Model	**UQ-TA**	**UQ-T2**	**AQ-TA**	**AQ-T2**
VGG13-like	102	**26**	64.8±6.14	**47.6± 3.82**
Resnet18	139	**55**	80.6±7.53	**74.6±14.22**
Auto-ML FCN	172	**34**	77.8±6.83	**31.8±10.40**
UNet	174	**63**	119.0±4.84	**76.4±14.92**

Table 1: Comparison of number of optimization steps w.r.t to performance drop of 1.0% using **-TA** and **-T2** for both **UQ** and **AQ** quantization. For **AQ**, mean and standard deviation over 5 runs is reported.

e.g., assembly. An embedded sensor module (intertial, magnetic and audio sensors) attached to a tool classifies activities into three classes. The authors use Auto-ML extensively. Their model is a fully convolutional network (FCN) [16] with sequential convolutional blocks (1D convolutions with 16 filters, dropout, batch norm, ReLU activation), distinct for each sensor type, followed by convolutional blocks with global average pooling over concatenated inputs, and finally a soft-max.

4 Evaluation

We choose to quantize only the convolutional and fully connected layers in the scope of our work. No explicit fine-tuning after applying the methods is done to avoid the need of having ground truth data for further training to recover any loss of accuracy. Following this scheme, the biases are intentionally not quantized due to their divergent behavior with respect to any quantization attempts.

We run each of our uniform quantization methods for bits ranging from 1 to 16. For adaptive quantization the range is 1 to 8 bits resulting in cluster sizes ranging from 2 to 256. We use the mini-batch k-means algorithm for clustering the weights with a maximum of 100 iterations and a batch size of five million weights in the global and one million weights in the tensorwise case [12]. As the initialization of the centroids leads to an inherent randomness in the clustering algorithm, we use results averaged over 5 runs of each adaptive quantization method and plotted with the mean and standard deviation of the runs.

For automatic search, we start searching with 16 bits and 8 bits for all tensors in the uniform and adaptive case, respectively. For selecting the number of bits, we use performance drop thresholds ($\%$) as [0.1, 0.5, 1.0, 3.0, 5.0, 7.0, 10.0, 15.0, 20.0] to cover extreme cases as well. Performance of models is reported in terms of Top-1 accuracy for the three classification use cases and Dice Similarity Coefficient (DSC) score for the brain segmentation task.

Our results are reported relative to the performance of the baseline model using full precision 32 bit floating-point. In the following figures, the vertical axis is set to the fraction of the original full precision model performance (Top-1 accuracy or DSC score) which is achieved with the compressed model. For the compression ratio, we divide the number of bytes required to represent the compressed model (including codebooks and step sizes) by the original size of all tensors (in byte) as 32 bit float. All metrics are displayed as a percentage.

In Figure 1 on the following page, uniform and adaptive quantization results are shown for all four use cases. We can see that in all cases, the adaptive quantization (**AQ-**) achieves higher compression than the uniform (**UQ-**) equivalent. Also, the global (**-G**) variants are outperformed by the tensorwise (**-TF**). In the tool tracking use case, the global quantization leads to the collapse of the weights of one tensor to a single value, which severely degrades the performance. The tensorwise allocation of different bit widths is performing better than the fixed ones in almost all cases. Only in the acoustic case they are similar, because greedy approach also results in an approximately constant bit width for all tensors. Our proposed method (**-T2**) is performing better or similar to the greedy approach (**-TA**) from [8] in some cases but is never outperformed by it.

As shown in Table 1, our method requires much fewer evaluations for convergence to a solution in terms of optimization steps. It is consistently faster than the greedy approach (**-TA**) from [8] by a significant margin.

5 Conclusions

In this paper, we investigated different strategies for post-training quantization of DNNs. Models such as Resnet18, Unet, VGG13-like network and an Auto-ML based fully convolutional network

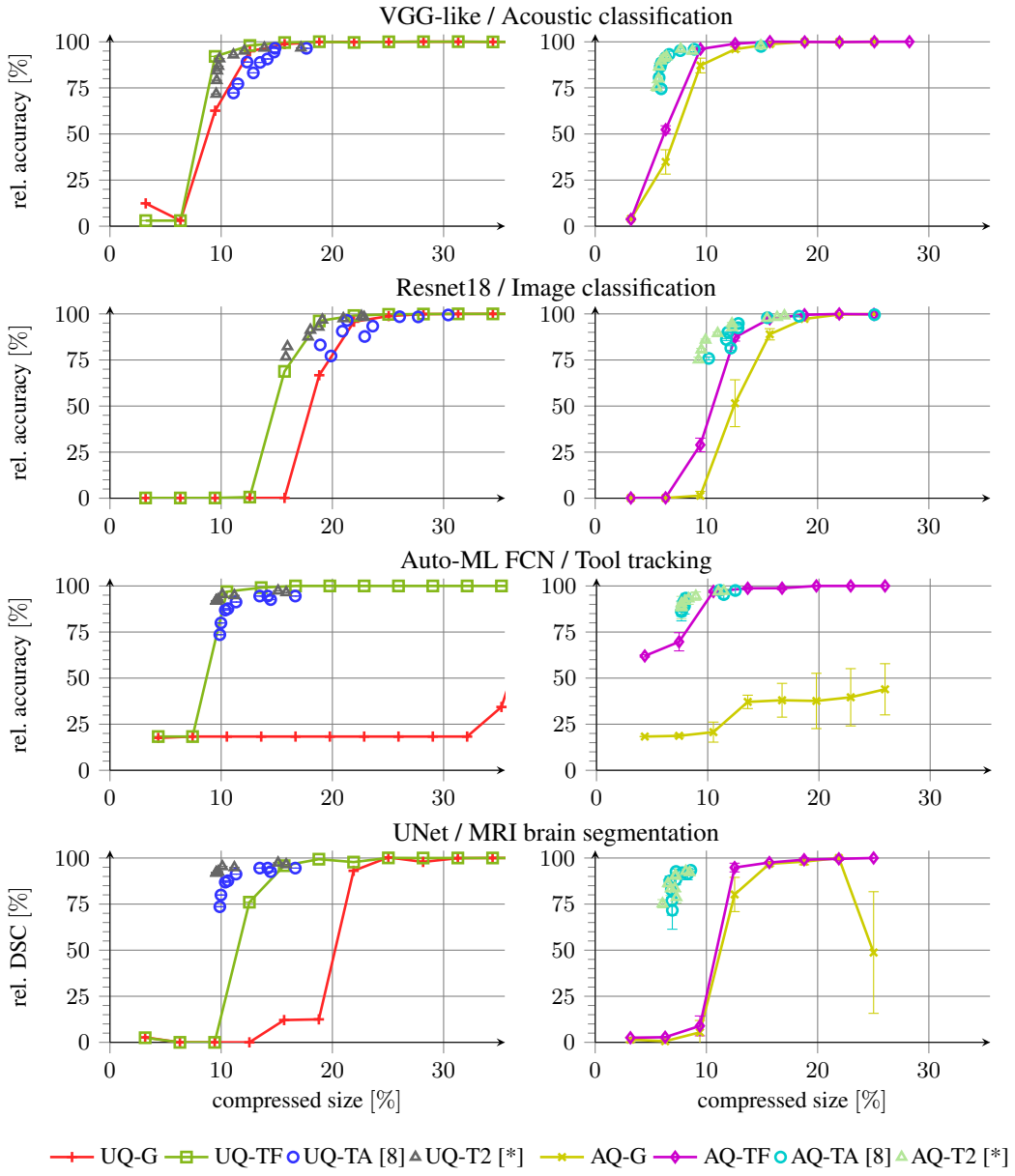

Figure 1: Compression results for both **UQ** = uniform (left) **AQ** = adaptive (right) quantization. Plots for using the global approach **-G** or the same number of bits per tensor **-TF**. The greedy approach for adaptive number of bits per tensor **-TA** and the proposed two-step approach **-T2**.

were used. Overall, the adaptive quantization achieves better compression ratios than uniform method. Also, it is beneficial to use tensorwise quantization in all cases.

An individual bit width per tensor could be shown to be superior in most cases, and performs similar to using the same bit width for all tensors in the remaining ones. Even though the assignment is done in a greedy fashion, this shows great potential. The proposed algorithm with a two-step selection of number of bits is faster than the single-step method as it requires fewer evaluations on the calibration set. At the same time, it is pushing the pareto front of performance over compression in many cases and performing similarly for the others.

This work was supported by the Bavarian Ministry for Economic Affairs, Infrastructure, Transport and Technology through the Center for Analytics-Data-Applications (ADA-Center) within the framework of "BAYERN DIGITAL II".

References

[1] M. Buda, A. Saha, and M. A. Mazurowski. Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm. *Computers in Biology and Medicine*, 109:218–225, Jun 2019.

[2] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra. General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline. In *DCASE Workshop*, Surrey, UK, 2018.

[3] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding. *CoRR*, abs/1510.00149, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[5] T. Iqbal, Q. Kong, M. D. Plumbley, and W. Wang. General-purpose audio tagging from noisy labels using convolutional neural networks. In *Detection and Classification of Acoustic Scenes and Events 2018 Workshop*, page 212–216, Woking, UK, 2018.

[6] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CoRR*, abs/1712.05877, 2017.

[7] C. Loeffler, C. Nickel, C. Sobel, D. Dzibel, J. Braat, B. Gruhler, P. Woller, N. Witt, and C. Mutschler. Automated quality assurance for hand-held tools via embedded classification and AutoML. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD)*, Ghent, Belgium, 2020.

[8] L. Meng, N. Suda, Y. Wang, and D. Loh. Neural network optimizations for on-device AI. In *Embedded World Conference*, Nuremberg, Germany, 2020.

[9] P. Nayak, D. Zhang, and S. Chai. Bit efficient quantization for deep neural networks. In *EMC2 – NeurIPS Workshop*, Vancouver, Canada, 2019.

[10] A. Plinge and A. Mishra. Getting AI in your pocket with deep compression. In *Embedded World Conference*, Nuremberg, Germany, 2020.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[12] D. Sculley. Web-scale k-means clustering. In *19th International Conference on World Wide Web*, pages 1177–1178, 2010.

[13] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, Dec. 2017.

[14] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. HAQ: hardware-aware automated quantization with mixed precision. *CoRR*, 1811.08886, Nov. 2018.

[15] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7675–7684. Curran Associates, Inc., 2018.

[16] Z. Wang, W. Yan, and T. Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *International Joint Conference on Neural Networks*, page 1578–1585, Anchorage, AK, 2017.

[17] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marban, T. Marinc, D. Neumann, A. Osman, D. Marpe, H. Schwarz, T. Wiegand, and W. Samek. DeepCABAC: Context-adaptive binary arithmetic coding for deep neural network compression. In *The International Conference on Machine Learning (ICML)*, May 2019.

[18] J. Xue, J. Li, and Y. Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, January 2013.