International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:13, No:9, 2019

Fast Adjustable Threshold for Uniform Neural Network Quantization

Alexander Goncharenko, Andrey Denisov, Sergey Alyamkin, Evgeny Terentev

***Abstract*—**The neural network quantization is highly desired procedure to perform before running neural networks on mobile devices. Quantization without fine-tuning leads to accuracy drop of the model, whereas commonly used training with quantization is done on the full set of the labeled data and therefore is both time- and resource-consuming. Real life applications require simplification and acceleration of quantization procedure that will maintain accuracy of full-precision neural network, especially for modern mobile neural network architectures like Mobilenet-v1, MobileNet-v2 and MNAS.

Here we present a method to significantly optimize training with quantization procedure by introducing the trained scale factors for discretization thresholds that are separate for each filter. Using the proposed technique, we quantize the modern mobile architectures of neural networks with the set of train data of only $\sim$ 10% of the total ImageNet 2012 sample. Such reduction of train dataset size and small number of trainable parameters allow to fine-tune the network for several hours while maintaining the high accuracy of quantized model (accuracy drop was less than 0.5%). Ready-for-use models and code are available in the GitHub repository.

***Keywords*—**Distillation, machine learning, neural networks, quantization.

I. Introduction

MOBILE neural network architectures [1]–[3] allow running AI solutions on mobile devices due to the small size of models, low memory consumption, and high processing speed while providing a relatively high level of accuracy in image recognition tasks. In spite of their high computational efficiency, these networks continuously undergo further optimization to meet the requirements of edge devices. One of the promising optimization directions is to use quantization to int8, which is natively supported by mobile processors, either with or without training. Both methods have certain advantages and disadvantages.

Quantization of the neural network without training is a fast process as in this case a pre-trained model is used. However, the accuracy of the resultant network is particularly low compared to the one typically obtained in commonly used mobile architectures of neural networks [4]. On the other hand, quantization with training is a resource-intensive task which results in low applicability of this approach.

Current article suggests a method which allows speeding up the procedure of training with quantization and at the same time preserves a high accuracy of results for 8-bit discretization.

A. Goncharenko, A. Denisov and S. Alyamkin are with Expasoft LLC, Novosibirsk, 630090 Russia (e-mail: a.goncharenko@expasoft.ru, a.denisov@expasoft.ru, s.alyamkin@expasoft.ru, https://expasoft.com/).

A. Goncharenko is with the Information Technology Department and A. Denisov is with the Department of Physics of Novosibirsk State University, Novosibirsk, 630090 Russia (e-mail: https://english.nsu.ru/).

E. Terentev is with Microtech, Moscow, Russia (e-mail: et@microtech.ai, https://microtech.ai).

II. Related Work

In general case the procedure of neural network quantization implies discretization of weights and input values of each layer. Mapping from the space of float32 values to the space of signed integer values with n significant digits is defined by the following formulae:

$$S_w = \frac{2^n - 1}{T_w} \tag{1}$$

$$T_w = max|W| \tag{2}$$

$$W_{int} = \lfloor S_w \cdot W \rceil \tag{3}$$

$$\begin{aligned} W_q &= clip(W_{int}, -(2^{n-1} - 1), 2^{n-1} - 1) = \\ &= min(max(W_{int}, -(2^{n-1} - 1)), 2^{n-1} - 1) \end{aligned} \tag{4}$$

Here $\lfloor \rceil$ is rounding to the nearest integer number, W weights of some layer of neural network, T quantization threshold, max calculates the maximum value across all axes of the tensor. Input values can be quantized both to signed and unsigned integer numbers depending on the activation function on the previous layer.

$$S_i = \frac{2^n - 1}{T_i} \tag{5}$$

$$T_i = max|I| \tag{6}$$

$$I_{int} = \lfloor S_i \cdot I \rceil \tag{7}$$

$$I_q^{signed} = clip(I_{int}, -(2^{n-1} - 1), 2^{n-1} - 1) \tag{8}$$

$$I_q^{unsigned} = clip(I_{int}, 0, 2^n - 1) \tag{9}$$

After all inputs and weights of the neural network are quantized, the procedure of convolution is performed in a usual way. It is necessary to mention that the result of operation must be in higher bit capacity than operands. For example, in [5] authors use a scheme where weights and activations are quantized to 8-bits while accumulators are 32-bit values.

Potentially quantization threshold can be calculated on the fly, which, however, can significantly slow down the

processing speed on a device with low system resources. It is one of the reasons why quantization thresholds are usually calculated beforehand in calibration procedure. A set of data is provided to the network input to find desired thresholds (in the example above - the maximum absolute value) of each layer. Calibration dataset contains the most typical data for the certain network and this data does not have to be labeled according to procedure described above.

A. Quantization with Knowledge Distillation

Knowledge distillation method was proposed by G. Hinton [6] as an approach to neural network quality improvement. Its main idea is training of neural networks with the help of pre-trained network. In [7], [8] this method was successfully used in the following form: a full-precision model was used as a model-teacher, and quantized neural network - as a model-student. Such paradigm of learning gives not only a higher quality of the quantized network inference, but also allows reducing the bit capacity of quantized data while keeping an acceptable level of accuracy.

B. Quantization without Fine-Tuning

Some frameworks allow using the quantization of neural networks without fine-tuning. The most known examples are TensorRT [9], Tensorflow [10] and Distiller framework from Nervana Systems [11]. However, in the last two models calculation of quantization coefficients is done on the fly, which can potentially slow down the operation speed of neural networks on mobile devices. In addition, to the best of our knowledge, TensorRT framework does not support quantization of neural networks with the architectures like MobileNet.

C. Quantization with Training/Fine-Tuning

One of the main focus points of research publications over the last years is the development of methods that allow to minimize the accuracy drop after neural network quantization. The first results in this field were obtained in [12]–[15]. The authors used the Straight Through Estimator (STE) [16] for training the weights of neural networks into 2 or 3 bit integer representation. Nevertheless, such networks had substantially lower accuracy than their full-precision analogs.

The most recent achievements in this field are presented in [17], [18] where the quality of trained models is almost the same as for original architectures. Moreover, in [18] the authors emphasize the importance of the quantized networks ensembling which can potentially be used for binary quantized networks. In [5] authors report the whole framework for modification of network architecture allowing further launch of learned quantized models on mobile devices.

In [19] the authors use the procedure of threshold training which is similar to the method suggested in our work. However, the reported approach has substantial shortcomings that prevent its usage for fast conversion of pre-trained neural network on mobile devices. First of all there is a requirement to train the threshold on the full ImageNet dataset [20], and second of all there are no examples demonstrating the accuracy of networks which are considered to be the standards for mobile platforms.

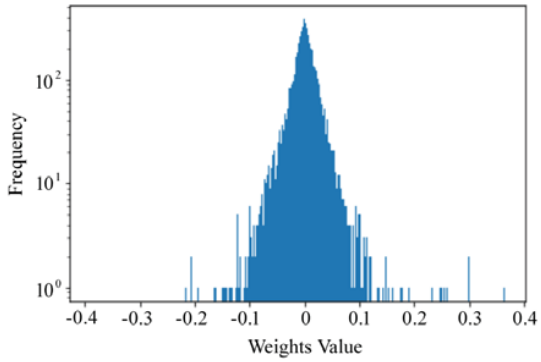

Fig. 1 Distribution of weights of ResNet-50 neural network before the quantization procedure

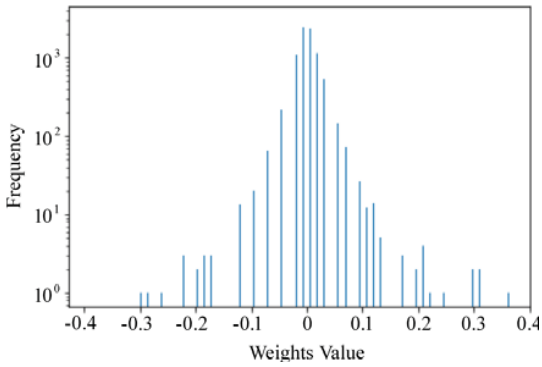

Fig. 2 Distribution of weights of ResNet-50 neural network after the quantization procedure (the number of values appeared in bins near zero increased significantly)

In current paper we propose a novel approach to set the quantization threshold with fast fine-tuning procedure on a small set of unlabeled data that allows to overcome the main drawbacks of known methods. We demonstrate performance of our approach on modern mobile neural network architectures (MobileNet-v2, MNAS).

III. Method Description

Under certain conditions (see Figs. 1 and 2) the processed model can significantly degrade during the quantization process. The presence of outliers for weights distribution shown in Fig. 1 forces to choose a high value for thresholds that leads to accuracy degradation of quantized model.

Outliers can appear due to several reasons, namely specific features of calibration dataset such as class imbalance or non-typical input data. They also can be a natural feature of the neural network, that are, for example, weight outliers formed during training or reaction of some neurons on features with the maximum value.

Overall it is impossible to avoid outliers completely because they are closely associated with the fundamental features of neural networks. However, there is a chance to find a trade-off between the value of threshold and distortion of other

values during quantization, and thus get a better quality of the quantized neural network.

A. Quantization with Threshold Fine-Tuning

1) Differentiable Quantization Threshold: In [13], [15], [16] it is shown that the Straight Through Estimator (STE) can be used to define a derivative of a function which is non-differentiable in the usual sense ($round$, $sign$, $clip$, etc). Therefore, the value which is an argument of this function becomes differentiable and can be trained with the method of steepest descent, also called the gradient descent method. Such variable is a quantization threshold and its training can directly lead to the optimal quality of the quantized network. This approach can be further optimized through some modifications as described below.

2) Batch Normalization Folding: Batch normalization (BN) layers play an important role in training of neural networks because they speed up train procedure convergence [21]. Before making quantization of neural network weights, we suggest to perform batch normalization folding with the network weights similar to method described in [5]. As a result we obtain the new weights calculated by the following formulae:

$$W_{fold} = \frac{\gamma W}{\sqrt{\sigma^2 + \varepsilon}} \tag{10}$$

$$b_{fold} = \beta - \frac{\gamma \mu}{\sqrt{\sigma^2 + \varepsilon}} \tag{11}$$

We apply quantization to weights which were fused with the BN layers because it simplifies discretization and speeds up the neural network inference. Further in this article the folded weights will be implied (unless specified otherwise).

3) Threshold Scale: All network parameters except quantization thresholds are fixed. The initial value of thresholds for activations is the value calculated during calibration. For weights it is the maximum absolute value. Quantization threshold T is calculated as

$$T = clip(\alpha, min_\alpha, max_\alpha) \cdot T_{max} \tag{12}$$

where α is a trained parameter which takes values from min_α to max_α with saturation. The typical values of these parameters are found empirically, which are equal to 0.5 and 1.0 correspondingly. Introducing the scale factor simplifies the network training since the update of thresholds is done with different learning rates for different layers of neural network as they can have various orders of values. For example, values on the intermediate layers of VGG network may increase up to 7 times in comparison with the values on the first layers.

Therefore the quantization procedure can be formalized as follows:

$$T_{adj} = clip(\alpha, 0.5, 1) \cdot T_i \tag{13}$$

$$S_I = \frac{2^n - 1}{T_{adj}} \tag{14}$$

$$I_q = \lfloor I \cdot S_I \rceil \tag{15}$$

The similar procedure is performed for weights. The current quantization scheme has two non-differentiable functions, namely $round$ and $clip$. Derivatives of these functions can be defined as:

$$I_q = \lfloor I \rceil \tag{16}$$

$$\frac{dI_q}{dI} = 1 \tag{17}$$

$$X_c = clip(X, a, b) \tag{18}$$

$$\frac{dX_c}{dX} = \begin{cases} 1, if X \in [a, b] \\ 0, otherwise \end{cases} \tag{19}$$

Bias quantization is performed similar to [5]:

$$b_q = clip(\lfloor S_i \cdot S_w \cdot b \rceil, -(2^{31} - 1), 2^{31} - 1) \tag{20}$$

4) Training of Asymmetric Thresholds: Quantization with symmetric thresholds described in the previous sections is easy to implement on certain devices, however it uses an available spectrum of integer values inefficiently which significantly decreases the accuracy of quantized models. Authors in [5] effectively implemented quantization with asymmetric thresholds for mobile devices, so it was decided to adapt the described above training procedure for asymmetric thresholds.

For asymmetric thresholds there are left (T_l) and right (T_r) range limits. However, for quantization procedure it is more convenient to use other two values: left limit and width, and train these parameters. If the left limit is equal to 0, then scaling of this value has no effect. That is why a shift for the left limit is introduced. It is calculated as:

$$R = T_r - T_l \tag{21}$$

$$T_{adj} = T_l + clip(\alpha_T, min_{\alpha_T}, max_{\alpha_T}) \cdot R \tag{22}$$

The coefficients min_{α_T}, max_{α_T} are set empirically. They are equal to -0.2 and 0.4 in the case of signed variables, and to 0 and 0.4 in the case of unsigned. Range width is selected in a similar way. The values of min_{α_R}, max_{α_R} are also empiric and equal to 0.5 and 1.

$$R_{adj} = clip(\alpha_R, min_{\alpha_R}, max_{\alpha_R}) \cdot R \tag{23}$$

5) Vector Quantization: Sometimes due to high range of weight values it is possible to perform the discretization procedure more softly, using different thresholds for different filters of the convolutional layer. Therefore, instead of a single quantization factor for the whole convolutional layer (scalar quantization) there is a group of factors (vector quantization). This procedure does not complicate the realization on devices, however it allows increasing the accuracy of the quantized model significantly. Considerable improvement of accuracy

is observed for models with the architecture using the Depth-wise separable convolutions. The most known networks of this type are MobileNet-v1 [1] and MobileNet-v2 [2].

B. Training on the Unlabeled Data

Most articles related to neural network quantization use the labeled dataset for training discretization thresholds or directly the network weights. In the proposed approach it is recommended to discard initial labels of train data which significantly speeds up transition from a trained non-quantized network to a quantized one as it reduces the requirements to the train dataset. We also suggest to optimize root-mean-square error (RMSE) between outputs of quantized and original networks before applying the softmax function, while leaving the parameters of the original network unchanged.

Suggested above technique can be considered as a special type of quantization with distillation [7] where all components related to the labeled data are absent.

The total loss function L is calculated by the following formula:

$$\begin{aligned} &L(x; W_T, W_A) = \\ &= \alpha H(y, z^T) + \beta H(y, z^A) + \gamma H(z^T, z^A) \end{aligned} \tag{24}$$

In our case α and β are equal to 0, and

$$H(z^T, z^A) = \sqrt{\sum_{i=1}^{N} \frac{(z_i^T - z_i^A)^2}{N}} \tag{25}$$

where:

- z^T is the output of non-quantized neural network,
- z^A is the output of quantized neural network,
- N is batch size,
- y is the label of x example.

IV. Experiments and Results

A. Experiments Description

1) Researched Architectures: The procedure of quantization for architectures with high redundancy is practically irrelevant because such neural networks are hardly applicable for mobile devices. Current work is focused on experiments on the architectures which are actually considered to be a standard for mobile devices (MobileNet-v2 [2]), as well as on more recent ones (MNasNet [3]). All architectures are tested using 224 x 224 spatial resolution.

2) Training Procedure: As it is mentioned above in Section III-B ("Training on the unlabeled data"), we use RMSE between the original and quantized networks as a loss function. Adam optimizer [22] is used for training, and cosine annealing with the reset of optimizer parameters - for learning rate. Training is carried out on approximately 10% part of ImageNet dataset [20]. Testing is done on the validation set. 100 images from the training set are used as calibration data. Training takes 6-8 epochs depending on the network.

TABLE I
Quantization in the 8-Bit Scalar Mode

Architecture	Symmetric thresholds, %	Asymmetric thresholds, %	Original accuracy, %
MobileNet v2	8.1	19.86	71.55
MNas-1.0	72.42	73.46	74.34
MNas-1.3	74.92	75.30	75.79

TABLE II
Quantization in the 8-Bit Vector Mode

Architecture	Symmetric thresholds, %	Asymmetric thresholds, %	Original accuracy, %
MobileNet v2	71.11	71.39	71.55
MNas-1.0	73.96	74.25	74.34
MNas-1.3	75.56	75.72	75.79

B. Results

The quality of network quantization is represented in the Tables I and II.

Experimental results show that the scalar quantization of MobileNet-v2 has very poor accuracy. A possible reason of such quality degradation is the usage of ReLU6 activation function in the full-precision network. Negative influence of this function on the process of network quantization is mentioned in [23]. In case of using vector procedure of thresholds calculation, the accuracy of quantized MobileNet-v2 network and other researched neural networks is almost the same as the original one.

For implementation the Tensorflow framework [10] is chosen because it is rather flexible and convenient for further porting to mobile devices. Pre-trained networks are taken from Tensorflow repository [24]. To verify the results, the program code and quantized scalar models in the .lite format, ready to run on mobile phones are presented in the GitHub repository [25].

V. Conclusion

This paper demonstrates the methodology of neural network quantization with fine-tuning. Quantized networks obtained with the help of our method demonstrate a high accuracy that is proved experimentally. Our work shows that setting a quantization threshold as multiplication of the maximum threshold value and trained scaling factor, and also training on a small set of unlabeled data allow using the described method of quantization for fast conversion of pre-trained models to mobile devices.

References

[1] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[2] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018)*.

[3] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," *arXiv preprint arXiv:1807.11626*, 2018.

[4] J. H. Lee, S. Ha, S. Choi, W. Lee, and S. Lee, "Quantization for rapid deployment of deep neural networks," *arXiv preprint arXiv:1810.05488*, 2018.

[5] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic only inference," in *Conference on Computer Vision and Pattern Recognition (CVPR 2018)*.
[6] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[7] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," *arXiv preprint arXiv:1711.05852*, 2017.
[8] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr, "Wrpn: Wide reduced-precision networks," *arXiv preprint arXiv:1709.01134*, 2017.
[9] https://developer.nvidia.com/tensorrt, NVIDIA TensorRT platform, 2018.
[10] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Largescale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.
[11] https://github.com/NervanaSystems/distiller.
[12] M. Courbariaux, Y. Bengio, and J. David, "Training deep neural networks with low precision multiplications," in *International Conference on Learning Representations (ICLR 2015)*.
[13] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems (NIPS 2016)*, pp. 41074115.
[14] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision (ECCV 2016)*, Springer, pp. 525542.
[15] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.
[16] Y. Bengio, N. Leonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.
[17] M. D. McDonnell, "Training wide residual networks for deployment using a single bit for each weight," in *International Conference on Learning Representations (ICLR 2018)*.
[18] S. Zhu, X. Dong, and H. Su, "Binary ensemble neural network: More bits per network or more networks per bit?" *arXiv preprint arXiv:1806.07550*, 2018.
[19] C. Baskin, N. Liss, Y. Chai, E. Zheltonozhskii, E. Schwartz, R. Giryes, A. Mendelson, and A. M. Bronstein, "Nice: Noise injection and clamping estimation for neural network quantization," *arXiv preprint arXiv:1810.00162*, 2018.
[20] 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," *arXiv preprint arXiv:1409.0575*, 2014.
[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning (ICML 2015)*.
[22] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR 2015)*.
[23] T. Sheng, C. Feng, S. Zhuo, X. Zhang, L. Shen, and M. Aleksic, "A quantization-friendly separable convolution for mobilenets," *arXiv preprint arXiv:1803.08607*, 2018.
[24] https://github.com/tensorflow/tensorflow/blob/61c6c84964b4aec80aeace187aab8cb2c3e55a72/tensorflow/lite/g3doc/models.md, Image classification (Quantized Models).
[25] https://github.com/agoncharenko1992/FAT-fast-adjustable-threshold.