

Fast Adjustable Threshold For Uniform Neural Network Quantization

Alexander Goncharenko^{*1,2}, Andrey Denisov^{*1,2}, Sergey Alyamkin¹, Evgeny Terentev³
{a.goncharenko, a.denisov, s.alyamkin}@expasoft.ru, et@microtech.ai

¹Expasoft LLC, ²Novosibirsk State University, ³Microtech

Abstract

Neural network quantization procedure is the necessary step for porting of neural networks to mobile devices. Quantization allows accelerating the inference, reducing memory consumption and model size. It can be performed without fine-tuning using calibration procedure (calculation of parameters necessary for quantization), or it is possible to train the network with quantization from scratch. Training with quantization from scratch on the labeled data is rather long and resource-consuming procedure. Quantization of network without fine-tuning leads to accuracy drop because of outliers which appear during the calibration. In this article we suggest to simplify the quantization procedure significantly by introducing the trained scale factors for quantization thresholds. It allows speeding up the process of quantization with fine-tuning up to 8 epochs as well as reducing the requirements to the set of train images. By our knowledge, the proposed method allowed us to get the first public available quantized version of MNAS without significant accuracy reduction - 74.8% vs 75.3% for original full-precision network. Model and code are ready for use and available at:

https://github.com/agoncharenko1992/FAT-fast_adjustable_threshold.

1. Introduction

Mobile neural network architectures [19, 20, 21] 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 are further optimized to launch them on edge devices. It can be done using quantization to int8 which is natively supported by the mobile processor either with or without training. Both methods have certain advantages and disadvantages.

* - main contribution

Quantization of the neural network without training is a fast process as in this case a pre-trained model is used. However, its accuracy is much worse compared to the original one for mobile architectures of neural networks [33]. Second approach to quantization - quantization with training has its own disadvantage: it requires too much effort resulting 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.

2. Related works

a. General description of uniform quantization methodology

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 formulas:

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

$$W_{int} = \text{round}(S_w W)$$

$$W_q = \text{clip}(W_{int}, -(2^{n-1} - 1), 2^{n-1} - 1) \text{ where } T_w = \max(|W|)$$

Here *round* is rounding to the nearest integer number, 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.

$$S_i = \frac{(2^n - 1)}{T_i}$$

$$I_{int} = \text{round}(S_i I)$$

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

$$\text{where } T_i = \max(|I|)$$

After all inputs and weights of the neural network have been quantized, the procedure of convolution is performed in a usual way. It is necessary to mention that the resultant of operation must be in higher bit capacity than operands. For example, in the paper [1] 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, but it can significantly slow down the processing speed on a device with the low system resources. It is one of the reasons why quantization thresholds are usually calculated beforehand. A set of calibration 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 may be unlabeled according to procedure described above.

b. Quantization with knowledge distillation

Knowledge distillation method was proposed by G. Hinton [4] as an approach to neural network quality improvement. Its main idea is training of neural networks with the help of pre-trained network. In papers [8, 9] 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 and at the same time having an acceptable level of accuracy.

c. Quantization without fine-tuning

Some frameworks allow using the quantization of neural networks without fine-tuning. The most popular are TensorRT [2], quantization from Tensorflow framework [17] and distiller framework from Nervana Systems [18]. However, in the last two models calculation of quantization coefficients is made on the fly, and it can slow down the speed of neural networks on mobile devices. TensorRT framework does not support quantization of neural networks with the architectures like MobileNet.

d. Quantization with training / fine-tuning

One of the main points of research papers during the last several years is to develop methods that allow reducing the accuracy drop after neural network quantization. The first results in this field were received by authors of [10, 11, 12, 13]. They used the State Through Estimator (STE) [3] for training the weights of neural networks into 2 or 3 bit integer representation. However, such networks had much lower accuracy than their full-precision analogs.

The last achievements in this field are presented in papers [14, 15] where the quality of trained models is almost the same as for original architectures. Moreover, the authors of [14] raise an interesting question about the quantized networks ensembling which may have a practical use for binary quantized networks. In paper [1] authors represent the whole framework for modification of network architecture allowing further launch of learned quantized models on mobile devices.

Authors of [16] use the procedure of threshold training which is similar to the method suggested in our work. However, there are two shortcomings preventing the usage of this approach for fast conversion of pre-trained neural network on mobile devices. The first one is a requirement to train the threshold on the full ImageNet dataset [5], and the second one is the absence of examples demonstrating the accuracy of networks which are considered to be the standards for mobile platforms.

So the key features of current work are the following.

1. A novel approach to neural network quantization without weights modification and with the trained quantization threshold is suggested. Current approach was introduced independently of the paper [16] within the work for MicroTech company [32] and used on the LPIRC-2018 contest [31].
2. A novel approach to set the quantization threshold is proposed to simplify the training procedure. It is calculated as multiplication of scaling factor (its gradient is estimated by State Through Estimator [3]) and initial threshold value.
3. A training procedure can be done on a small set of unlabeled images.
4. Proposed method is verified on mobile architectures of neural networks, and the drop of accuracy during quantization does not exceed 0.1%.

All mentioned above points out that suggested method is an effective and fast procedure of porting of pre-trained neural networks to mobile devices with a small reduction of accuracy.

3. Method description

During the quantization process there can arise a problem that leads to significant degradation of processed model. It is illustrated on the Figure 1.

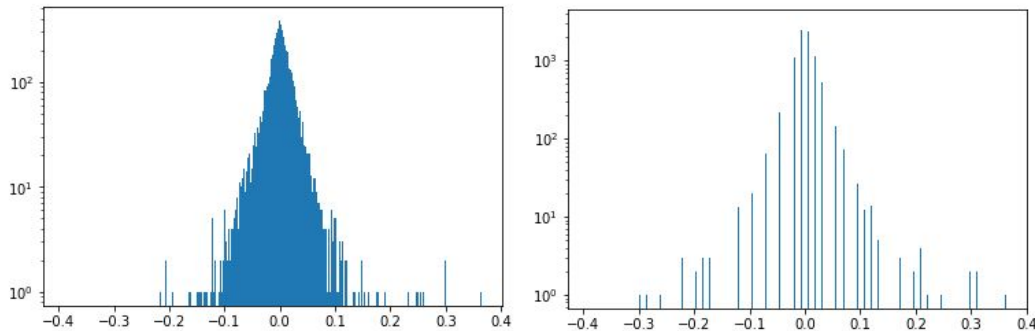


Figure 1. Distribution of weights of ResNet-50 neural network before the quantization procedure (on the left) and after it (on the right). The number of values appeared in bins near zero increased significantly.

The presence of outliers for weights distribution shown on Figure 1 forces to choose a high value for thresholds that leads to accuracy degradation for quantized model. Outliers can appear due to many reasons: peculiarities of calibration dataset such as classes disbalance or non-typical input data. They also can be a natural feature of neural network, for example, weight outliers formed during training, or reaction of some neurons on features with the maximum value.

On the whole it is impossible to get rid of 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

Differentiable quantization threshold. In papers [3, 6, 7] it is shown that the State 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). So 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 quantization threshold; its training can directly lead to the optimal quality of the quantized network. To get better results, this approach can be used with some modifications.

Batch normalization folding. Batch normalization (BN) layers play important role in training of neural networks because it speeds up their convergence [18]. Before making quantization of neural network weights, we suggest to perform batch normalization folding with the network weights similar to [1]. As a result we obtain the new weights calculated by the following formulas.

$$W_{fold} = \frac{\gamma W}{\sqrt{\sigma^2 + \epsilon}}$$

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

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 otherwise specified).

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 $T = clip(\alpha, min_\alpha, max_\alpha) T_{max}$, 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; they are equal to 0.5 and 1.0 correspondingly. Introducing the scale factor simplifies the network training because the update of thresholds is made with the 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.

So the quantization procedure looks as follows:

$$T_{adj} = clip(\alpha, 0.5, 1.) T_i$$

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

$$I_q = round(I * \frac{2^n - 1}{T_{adj}})$$

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

$$I_q = round(I), \frac{dI_q}{dI} = 1$$

$$X_c = clip(X, a, b), \frac{dX_c}{dX} = \begin{cases} 1, & \text{if } X \in [a, b] \\ 0, & \text{otherwise} \end{cases}$$

Bias quantization is made similar to [1]:

$$b_q = \text{clip}(\text{round}(S_i S_w b), -(2^{31} - 1), 2^{31} - 1)$$

Training of asymmetric thresholds. Quantization with symmetric thresholds described in the previous sections is simply implemented on the certain devices, however it uses an available spectrum of integer values inefficiently which significantly degrades the accuracy of quantized models. Authors of [1] 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:

$$R = T_r - T_l$$

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

The coefficients \min_{α_T} , \max_{α_T} are set empirically. They are equal to -0.2 and 0.4 in case of signed variables, and to 0 and 0.4 in 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.

$$R_{adj} = \text{clip}(\alpha_R, \min_{\alpha_R}, \max_{\alpha_R})R$$

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. So instead of one 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 popular networks of this type are MobileNet-v1 [19] and MobileNet-v2 [20].

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 refuse from initial labels of train data that 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, leaving the parameters of the original network unchanged.

4. Experiments and Results

a. Experiments description

Researched architectures. The procedure of quantization for architectures with the high redundancy is unreasonable because such neural networks are poorly 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 [20]), as well as on the relatively new ones (MNasNet [21]). All architectures were tested using 224 x 224 spatial resolution.

Training procedure. As it was mentioned above in the section “Training on the unlabeled data”, there was used RMSE between the original and quantized networks as a loss function. Adam optimizer [22] was used for training, and cosine annealing with the reset of optimizer parameters - for learning rate. Training was carried out on approximately 10% part of ImageNet dataset [5]. Testing was made on the validation set. 100 images from the training set were used as calibration data. Training took 6-8 epochs depending on the network.

b. Results

The quality of network quantization is represented in the tables below.

Architecture	Symmetric thresholds, %	Asymmetric thresholds, %	Original accuracy, %
MobileNet v2	8.1	19.86	71.99
MNas-1.0	71.87	72.78	73.79
MNas-1.3	74.58	74.8	75.30

Table 1. Quantization in the 8-bit scalar mode.

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 was mentioned in [30]. 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.

Architecture	Symmetric thresholds, %	Asymmetric thresholds, %	Original accuracy, %
MobileNet v2	71.8	71.98	71.99
MNas-1.0	73.63	73.76	73.79
MNas-1.3	75.19	75.23	75.30

Table 2. Quantization in the 8-bit vector mode.

For implementation the Tensorflow framework [17] was chosen because it is rather flexible and convenient for further porting to mobile devices. Pre-trained networks were taken from slim and tensorflow repositories [28, 29]. 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 repository specified in the abstract.

5. Conclusion

This paper demonstrates the methodology of neural network quantization with fine-tuning. Quantized networks have a high accuracy proved experimentally. Setting a quantization threshold as multiplication of the maximum threshold value and trained scaling factor, as well as 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. 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. CVPR, 2018.
2. NVIDIA, 2018. TensorRT: <https://developer.nvidia.com/tensorrt>.
3. Yoshua Bengio, Nicholas Leonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. CoRR, abs/1308.3432, 2013. <http://arxiv.org/abs/1308.3432>.
4. G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.
5. 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:1409.0575, 2014.
6. Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In Advances in neural information processing systems, pp. 4107–4115, 201.

7. 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.
8. Asit Mishra and Debbie Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. International Conference on Learning Representations, 2018. <https://openreview.net/forum?id=Blae1lZRb>.
9. Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. Wrpn: Wide reduced-precision networks. International Conference on Learning Representations, 2018. <https://openreview.net/forum?id=BlZvaeAZ>.
10. M. Courbariaux, Y. Bengio, and J. David. Training deep neural networks with low precision multiplications. International Conference on Learning Representations (ICLR), 2015.
11. Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. European Conference on Computer Vision, pp. 525–542. Springer, 2016.
12. Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In Advances in neural information processing systems, pp. 4107–4115, 2016.
13. 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.
14. Shilin Zhu, Xin Dong, and Hao Su. Binary Ensemble Neural Network: More Bits per Network or More Networks per Bit? arXiv preprint arXiv:1806.07550, 2018.
15. M. D. McDonnell. Training wide residual networks for deployment using a single bit for each weight. International Conference on Learning Representations (ICLR), 2018.
16. NICE: Noise injection and clamping estimation for neural network quantization, <https://arxiv.org/abs/1810.00162>.
17. 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: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
18. S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. ICML, 2015.
19. 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.
20. M. Sandler, A.G. Howard, M. Zhu, A. Zhmoginov, L.C. Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

21. Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. arXiv preprint arXiv:1807.11626, 2018.
22. D. Kingma and J. Ba. Adam. A method for stochastic optimization. ICLR, 2015.
23. K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. ECCV, 2016.
24. S. Woo, J. Park, J.-Y. Lee, and I. Kweon. Cbam: Convolutional block attention module. ECCV, 2018.
25. J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. ICCV, 2017.
26. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in Neural Information Processing Systems, pages 6000–6010, 2017.
27. B. Milde, A. Köhn. Open Source Automatic Speech Recognition for German. arXiv preprint arXiv:1807.10311, 2018.
28. <https://github.com/tensorflow/models/tree/master/research/slim/nets/mobilenet>
29. <https://github.com/tensorflow/tensorflow/blob/master/tensorflow/lite/g3doc/models.md#image-classification-quantized-models>
30. 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.
31. S. Alyamkin et al. Low-Power Image Recognition Challenge. arXiv preprint arXiv:1810.01732, 2018.
32. Microtech.ai, Evgeny Terentev.
33. J. H. Lee et al. Quantization for Rapid Deployment of Deep Neural Networks. arXiv preprint arXiv:1810.05488, 2018.