Quantization noise in low bit quantization and iterative adaptation to quantization noise in quantizable neural networks

D Chudakov[1], A Goncharenko[1], S Alyamkin[1] and A Densidov[1,]

[1] Novosibirsk State University, Novosibirsk, Russia

E-mail: {d.chudakov, a.goncharenko, a.denisov}@expasoft.tech, s.alyamkin@expasoft.com

Abstract. Quantization is one of the most popular and widely used methods of speeding up a neural network. At the moment, the standard is 8-bit uniform quantization. Nevertheless, the use of uniform low-bit quantization (4- and 6-bit quantization) has significant advantages in speed and resource requirements for inference. We present our quantization algorithm that offers advantages when using uniform low-bit quantization. It is faster than quantization-aware training from scratch and more accurate than methods aimed only at selecting thresholds and reducing noise from quantization. We also investigated quantization noise in neural networks for low-bit quantization and concluded that quantization noise is not always a good metric for quantization quality.

1. Introduction

Neural networks are widely used in modern audio-, video- software solutions including image classification, object detection, and segmentation. Neural networks are extremely demanding on computing resources not only at the stage of training but also at the stage of inferencing. In order to use neural networks in large-scale software solutions, it is necessary to apply different approaches for network compression and reducing latency. Post-training quantization is one of the major approaches for neural network speed and size. The main advantages of quantization over other methods of neural network acceleration are:

1. Low-bit integer calculations are much faster than analogous calculations in single-precision floating-point numbers.

2. Quantization significantly reduces the amount of memory needed to store weights of the neural network model. In the case of 8-bit quantization the size of the neural network is reduced by 4 times in comparison with 32-bit

3. The low bit precision digit of the neural network weights allows putting more data in the CPU cache, which allows reducing the frequency of access to the main memory. That leads to a decrease in power consumption and inference acceleration.

4. The implementation of floating-point calculations at the hardware level is more difficult than the implementation of integer calculations, and therefore low-cost microcontrollers often support only integer calculations.

2. Related Work

2.1. Quantization scheme.

First of all, let us consider the quantization scheme proposed in [1]. The quantization scheme describes the process of translation of the original neural network into a quantized one. This scheme uses asymmetric linear quantization. That is, the conversion of quantized integer values of weights and activations into floating-point numbers is affine:

$$r = S(q - Z) \tag{1}$$

Where, r is the original floating-point number, S and Z are the scaling and shift coefficients, q are quantized integers. S is represented as a real number, and Z is represented as an integer, this allows the quantized zero to be exactly matched to the real one. In modern neural network computation methods, the convolution procedure is represented as matrix multiplication, using the image representation as a vector. Thus in quantized form, the convolution operation can be represented as:

$$S_3\left(q_3^{(i,k)} - Z_3\right) = \sum_{j=1}^{N} S_1\left(q_1^{(i,j)} - Z_1\right) S_2\left(q_2^{(j,k)} - Z_2\right) \tag{2}$$

Where, i,j,k are numbers of rows and columns in the matrix representation of the linear convolution operator. Z_l, S_l, q_l are shift factor, scaling factor and quantized weight value. The lower index $l = 1, 2, 3$ corresponds to the first multiplier, the second multiplier, and the result of the convolution operation. By opening the brackets, it is easy to obtain:

$$q_3^{(i,k)} = Z_3 + M \sum_{j=1}^{N} \left(q_1^{(i,j)} - Z_1\right)\left(q_2^{(j,k)} - Z_2\right) \tag{3}$$

$$M := \frac{S_1 S_2}{S_3} \tag{4}$$

$$M = 2^{-n} M_0 \tag{5}$$

Where M is the re-quantization coefficient. Experimentally the authors of the scheme found that the coefficient is in the range [0, 1]. It means that we can represent it as an integer 32-bit number and fixed shift, this type of calculation can be effectively implemented on the processor. After the multiplication of 8-bit numbers, calculation results are added to the 32-bit container, so subsequent multiplications by re-quantization coefficients and shift additions are done with 32-bit integers.

2.2. Calculation of quantization thresholds.

The described quantization scheme does not fix the way of obtaining scaling and shift coefficients. The correct selection of those coefficients is the key to preserve the initial neural network quality. The naive minimax approach suggests taking the maximum and minimum values of weights and activations as thresholds.

$$\text{clamp}(r; a, b) := \min(\max(r, a), b) \tag{6}$$

$$s(a, b, n) := \frac{b - a}{n - 1} \tag{7}$$

$$q(r; a, b, n) := \left[\frac{\text{clamp}(r; a, b) - a}{s(a, b, n)}\right] \tag{8}$$

Where, n is number of intervals (2^b where b bit capacity) of quantization, $a = min(W)$ is left threshold of quantization, $b = max(W)$ right threshold of quantization. W weights the

neural network, r is the initial real number. The naive approach does not take into account the probabilistic nature of the weights and possible outliers, which may cause a significant drop in accuracy. In order to reduce the effect of outliers in a single filter of the convolutional layer on the entire layer, it was proposed [11] to use vector quantization for the weights.

Nvidia's TenortRT library [13] proposes a time-consuming iterative method to find suitable quantization thresholds based on Kullback-Leibler divergence [8], which requires collecting activation statistics for all layers. Intel - Artificial Intelligence Product Group [4] proposed an analytical method for selecting quantization thresholds by assuming that the weights have a Laplace distribution and minimizing the discretization error of the weights (9). Here X is the original distribution of weights, and $Q(X)$ is the discretized distribution of weights, α is the quantization threshold, Δ is the interval size, q is the discretized value of weights, and M is the number of intervals:

$$QN_1 = \int_{-\infty}^{-\alpha} f(x) \cdot (x+\alpha)^2 dx \tag{9}$$

$$QN_2 = \sum_{i=0}^{2^M-1} \int_{-\alpha+i\Delta}^{-\alpha+(i+1)\Delta} f(x) \cdot (x-q_i)^2 \, dx \tag{10}$$

$$QN_3 = \int_{\alpha}^{\infty} f(x) \cdot (x-\alpha)^2 dx \tag{11}$$

$$E\left[(X-Q(X))^2\right] = QN_1 + QN_2 + QN_3 \tag{12}$$

There was proposed a method [3] to adjust quantization thresholds during training procedure with gradient descent by approximating the gradient through undifferentiated functions with a constant equal to 1.

2.3. Quantization aware training

In the scheme proposed in [1] it is also proposed a way of training a neural network taking into account quantization, using differentiable real approximation of the quantized network. Weights and activations are stored in single-precision floating-point numbers, and all calculations are performed in single-precision floating-point numbers. Special nodes are placed before each operation to simulate the quantized model by discretizing the weights. These nodes perform quantization and dequantization of received input, thus obtaining discretized real numbers in the same value range and approximating noise from quantization effect. The gradients for rounding operations are approximated by a constant equal to one. The advantage of this method is high accuracy as compared to quantization using the minimax approach. The disadvantage of this approach is that it leads to an increase in learning time.

3. Method description

The proposed neural network quantization method [2] improves the FAT [3] method. As a result of the threshold adjusting training process, the accuracy increases because of quantization noise reduction.

$$QN = E[(X-Q(X))^2] \tag{13}$$

Here we will call the quantization noise QN which is the expectation of the standard deviation of the original distribution X (weights or activations) from the discretized value $Q(X)$. Since we have a finite and relatively small number of intervals, it is not possible to truncate this noise to zero. Choosing a small number of intervals results in a significant increase in noise. (e.g. 6-bit or 4-bit quantization). Also, relatively small noise can strongly influence quantization accuracy in the case of small neural networks (e.g. EfficientNet [6]). The proposed algorithm, in addition to reducing quantization noise, adapts the neural network to deal with the resulting noise. This

Table 1. Results of quantization by different approaches. The obtained accuracy drawdown is given in brackets. Quantization parameters are bit depth and type (vector or scalar)

Architecture	Quantization parameters	MinMax, %	FAT, %	IQNA, %
ResNet50	8, Scalar	73.9 (-1.5)	73.8 (-1.6)	72.0 (- 3.4)
ResNet50	4, Vector	1.0 (-74.4)	59.8 (-15.5)	66.4 (- 9.0)
MobileNetV2	8, Vector	71.4 (-0.5)	71.5 (-0.4)	71.0 (-0.9)
EfficientNet-B0	8, Scalar	4.6 (- 72.7)	74.8 (- 2.2)	75.9 (-1.1)
EfficientNet-B0	8, Vector	72.0 (-5.0)	76.5 (- 0.5)	76.0 (-1.0)
EfficientNet-B0	6, Vector	0.1 (-76.9)	1.1 (-75.9)	75.6 (-1.4)
EfficientNet-B0	4, Vector	0.1 (-76.9)	0.3 (-76.7)	62.3 (-14.7)
MobileNetV2	4, Vector	0.2 (-71.7)	0.3 (-71.6)	62.3 (-9.6)

is achieved through iterative adaptation to the quantization noise. The algorithm consists of four stages:

1. The algorithm begins to iteratively quantize the network from the first layer to the last
2. For the selected layer the initial quantization thresholds are calculated using the minimax approach.
3. Using FAT [3] method, thresholds are optimized for this layer.
4. The quantized part of the neural network is frozen and the non-quantized part of the neural network is trained using distillation [12] to minimize the error for a given problem. The distillation approach is to minimize the deviation of the output of the "student" neural network from the output of the "teacher" neural network. The advantage of this approach is that there is no requirement for data to be labeled. The distillation process optimizes the Euclidean distance between the outputs of the quantized and original neural networks.

4. Experiments and Results

To conduct experiments, we have developed our framework that allows us to accurately simulate a quantized neural network. In this way, we could measure the precision of a quantized neural network. The disadvantage of this approach is that we cannot measure the real gain in inference speed and, for example, power consumption. The issue of real acceleration of neural networks during quantization is studied in detail in the article [1]

4.1. Measurement on a Imagenet task

A number of experiments were conducted for different neural network architectures trained for classification on the Imagenet dataset *ImageNet* [5]. Algorithms were compared: Naive MinMax, Fast Adjustable Thresholds (FAT), Iterative Quantization Noise Adaptation (IQNA)

From the acquired results it is clear (Table 1) that the minimax method leads to a significant drop in accuracy, since it completely ignores the probabilistic nature of the weights and activations of the neural network, for scalar quantization resulting outliers can reduce the accuracy to zero. The FAT method avoids significant accuracy degradation for 8-bit quantization, but for quantization of 6 or 4 bits, the network accuracy drops to zero. For the iterative adaptation method, an Adam [7] optimizer with a learning rate of $5 * 10^{-5}$ and parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ was used in the pre-training phase of the regular network. In this case, the choice of parameters for training is not so important, they should be selected based on the parameters used for training a regular non-quantized neural network on a given task. We took the parameters carefully selected in the article [3]. At each iteration, a random part of the training data of

Table 2. Quantization-induced noise was also measured using the MobileNetV2 neural network with vector 4-bit quantization as an example.

Convolutional layer number	MinMax	FAT	IQNA
3	15.3	27.74	3.76
4	1.5	2.33	0.268
5	0.56	0.7	0.07
6	30.93	20.9	2.29

Table 3. Quantization-induced noise for ResNet50 neural network with vector 4-bit quantization.

Convolutional layer number	MinMax	FAT	IQNA	FAT without WA
7	0.037	0.227	0.418	0.031
8	0.121	0.334	0.478	0.096
9	0.056	0.136	0.274	0.025
10	0.156	0.222	0.466	0.076

100000 objects was taken and training took place during one epoch. The running time of the algorithm takes about 12 hours for MobleNetV2 on Nvidia GeForce 1080ti.

The method showed to be effective in cases of quantization of networks with a low number of bits when it is not possible to reduce the noise from quantization sufficiently. Thus it was possible to quantize a MobileNetV2 [9] and EfficientNet-B0 [6] network at 4 bits, avoiding a significant drop in accuracy ($< 20\%$). In this case, for neural networks with a large number of parameters and at 8-bit quantization the method of iterative adaptation can give results worse than the basic FAT method, as seen in the example of ResNet-50 [10].

4.2. Quantization-induced noise measurements

From the obtained measurements it is clear that the method of iterative adaptation to noise quantization significantly reduces noise in comparison to the naive minimax approach and FAT method (Table 2), in some cases by an order of magnitude. Figure 1 shows how the method of iterative adaptation to quantization noise allows us to use a small number of intervals more efficiently and to approximate the original activation distribution more accurately.

Although we see a decrease in quantization noise, the interlayer distillation method was ineffective and did not improve accuracy. Possible reasons are related to the fact that the IQNA method, as well as the FAT method, does not always reduce the noise from quantization. Using the example of the ResNet50 neural network quantized using 4-bit vector quantization (Table 3), we show that the FAT and IQNA methods increase the quantization noise, while simultaneously increasing the accuracy of the neural network. The FAT without WA (Without weight adjustment) method, in which the stage of fine adjustment of the weights was skipped, yielded a reduction in quantization noise compared to the MinMax method, but the final accuracy after this algorithm was only 10.7% (compared to 59.8% for the regular method FAT). Figure 2 shows how the method of iterative adaptation to quantization noise also help us to approximate the original activation distribution more accurately.

5. Conclusion

We proposed our uniform quantization algorithm and showed its efficiency when using low-bit quantization. This algorithm has shown promising results when using 4- and 6-bit quantization. Our algorithm is not tied to a specific quantization scheme and can be used with other even non-

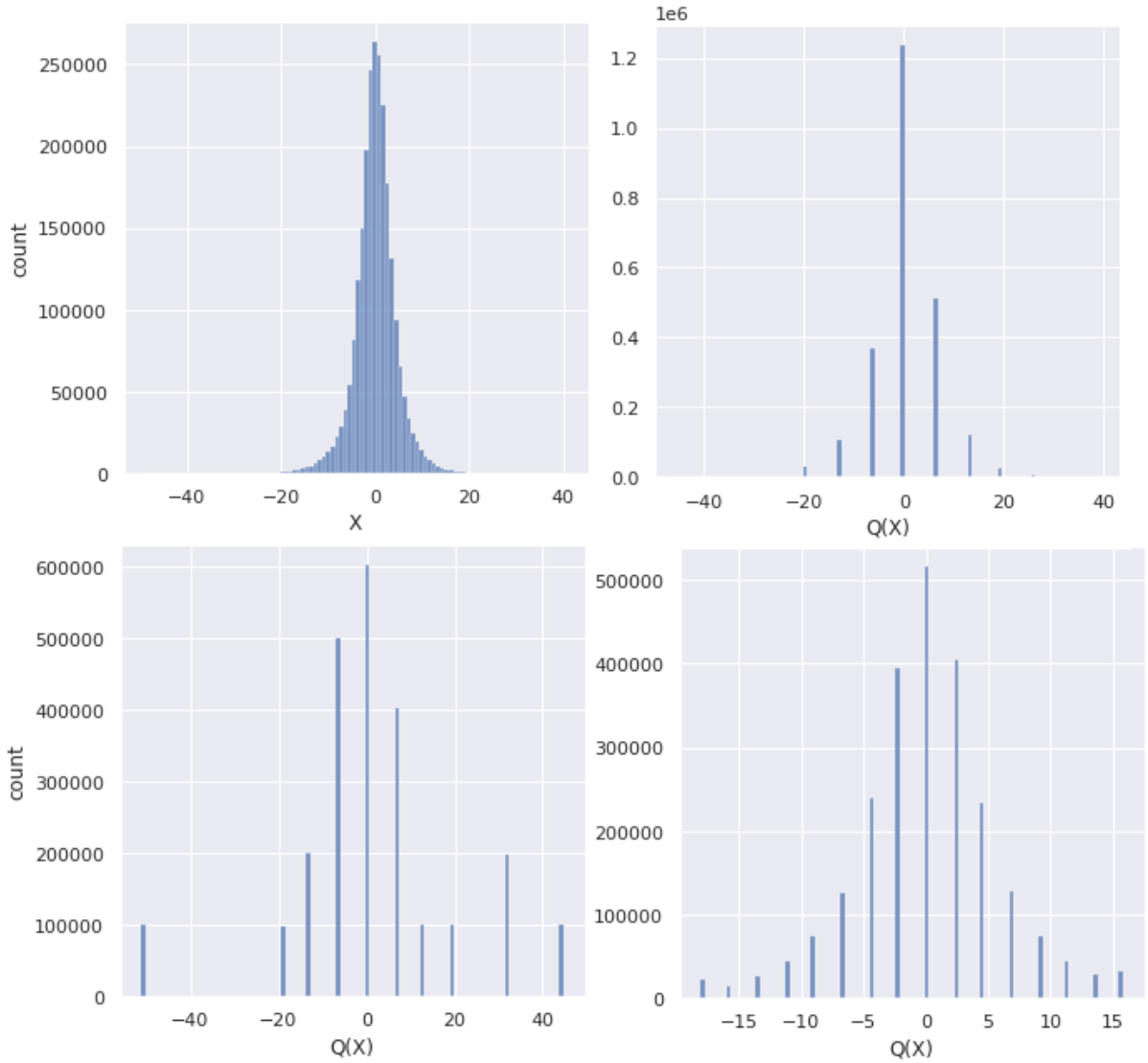

Figure 1. Histograms of activation after 9-th convolutional layer in MobileNetV2 after applying different quantization methods. (from left to right, from top to bottom: Original non-quantized, MinMax, FAT, IQNA

uniform quantization methods. However, due to instability, the algorithm can lead to a drop in precision when quantizing at 8 bits, when standard methods give good precision. Also, using the example of several quantization algorithms, we studied the noise caused by quantization, and also showed by example that the noise from quantization does not always reduce, and algorithms aimed to reduce this noise may not be effective enough.

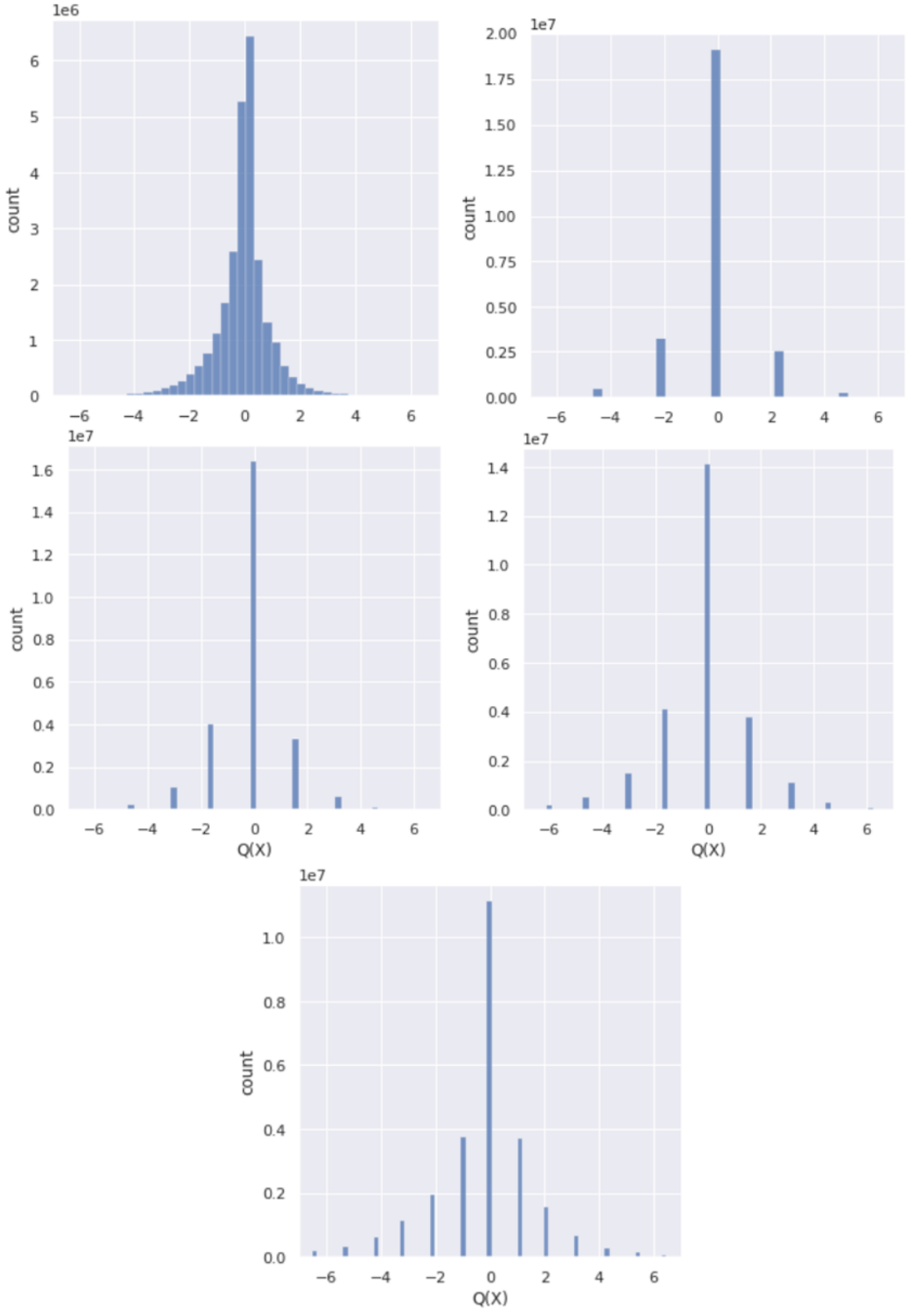

Figure 2. Histograms of activation after 8-th convolutional layer in ResNet50 after applying different quantization methods. (from left to right, from top to bottom: Original non-quantized, MinMax, FAT without WA, FAT, IQNA)

References

[1] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2704-2713).

[2] Chudakov, D., Alyamkin, S., Goncharenko, A., Denisov, A. (2021, June). Iterative Adaptation to Quantization Noise. *In International Work-Conference on Artificial Neural Networks* (pp. 303-310). Springer, Cham.

[3] Goncharenko, A., Denisov, A., Alyamkin, S., Terentev, E. (2019). Fast adjustable threshold for uniform neural network quantization. *International Journal of Computer and Information Engineering*, **13(9)**, 495-499.

[4] Banner, R., Nahshan, Y., Hoffer, E., Soudry, D. (2018). Post-training 4-bit quantization of convolution networks for rapid-deployment. *arXiv preprint arXiv:1810.05723.*

[5] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. *In 2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255).

[6] Tan, M., Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. *In International Conference on Machine Learning* (pp. 6105-6114).

[7] Kingma, D. P., Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[8] Kullback, S. (1959). Information theory and statistics. john riley and sons. Inc. New York.

[9] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).

[10] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[11] Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342.*

[12] Hinton, G., Vinyals, O., Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531.*

[13] S. Migacz., *GPU Technology Conference (2017).* (pp. 10-30)