# Post-training quantization of neural network through correlation maximization

Maria Pushkareva
*Center of Optical Neural Technologies, Scientific Research Institute of System Analisys RAS*
Moscow, Russia
pushkareva.mariia@yandex.ru

Iakov Karandashev
*Center of Optical Neural Technologies, Scientific Research Institute of System Analisys RAS*
Moscow, Russia
karandashev@niisi.ras.ru

***Abstract*—In this paper, we propose a method for quantizing the weights of neural networks by maximizing the correlations between the initial and quantized weights, taking into account the distribution of the weight density in each layer. Quantization is performed after the neural network training without further post-training. We tested the algorithm using the ImageNet dataset for VGG-16, ResNet-50, and Xception neural networks [2]. In the case of ResNet-50 and Xception neural networks, 4-5 bits of memory are required for the weights of a single layer to obtain acceptable Top-5 accuracy, for VGG-16, 3-4 bits are sufficient to store the weights of a single layer.**

***Keywords*—weights quantization, post-learning, linear quantization, exponential quantization**

## I. INTRODUCTION

The majority of neural networks, which we use when solving image recognition problems, have many parameters that have to be stored. Consequently, a substantial memory capacity is necessary and this requirement limits such neural networks applicability. For example, the storage capacity of the neural networks VGG-16 and ResNet152V2 are 528 [3] and 232 [4] MBs, respectively. The quantization and reduction of the number of weights are basic approaches allowing us to decrease the memory size necessary to store the neural network weights.

The quantization is a reduction in the variety of the different values of the weights in the layer. The most popular quantization methods are application of the fixed-point formats in place of the floating-point formats [9], binarization [10], ternarization [11], use of a logarithmic scale [12] and so on. Usually, one can reduce the number of the weights with the aid of such methods as pruning algorithms [13], sharing weights (including application of the convolution operation) [14], tensor expansions [15] and so on.

In the present paper, we explore the quantization process for trained neural networks. The number $B$ of bits per weight defines the number of different values of the weights; and it, consequently, is equal to $2^B$. We perform the quantization process independently for each layer. In the given layer we split the whole range of weights from the minimal to the maximal value into $2^B$ intervals. Then the weights belonging to one interval we replace by a single value. In what follows we examine a question related the optimal choice of the interval boundaries as well as the values with which we have to replace the weights.

The authors of paper [7] discussed the optimal quantization problem for the Hopfield neural network. They showed that when maximizing correlations between the initial and quantized values of the weights it was possible to minimize the errors of the quantized neural network. We believe that this result is correct and in what follows, we choose the interval boundaries and the quantized values of the weights inside of the intervals proceeding from a maximal correlation principle.

Frequently to get a sufficient accuracy of the neural network processing we have to combine the quantization and the neural network post-training. This procedure requires substantial resources [5, 6]. In the present paper, we perform the quantization after the neural network training without the following post-training. This method allows us to reduce the quantization costs substantially.

## II. ESTIMATE OF CORRELATION AND ITS GRADIENT

For each layer, let us quantize the weights inside the interval $[w_{\min}, w_{\max}]$, where $w_{\min}$ is the minimal and $w_{\max}$ is the maximal value of the weights in the given layer, respectively. (We normalized all the weights, so that $\tilde{w} = (w - \overline{w}) / \sigma_w$). Let $B$ be the number of bits necessary to store the weights of one layer. Then $n = 2^B$ is the number of the quantized weight values, as $x_i$ we define the boundaries of the intervals where the weight values are constant. Consequently,

$$w_{\min} = x_0 < x_1 < \ldots < x_{n-1} < x_n = w_{\max} . \quad (1)$$

Let $w$ be the input weights, and $y_i$ the quantized value inside the interval $(x_i, x_{i+1})$. It is not evident how we have to choose the interval boundaries $x_i$ as well as the values $x_i$ inside each interval. In the present paper we suppose that the stronger correlation between the input and the quantized values the less the error of the quantized neural network performance compared with the initial neural network:

$$\rho(w, y) = \frac{\overline{wy}}{\sigma_w \sigma_y} \to \max . \quad (2)$$

where $\overline{wy}$ is a covariation between the initial and the quantized values; $\sigma_w$ and $\sigma_y$ are standard deviations of the values of the input weight and their quantized values, respectively. For simplicity we suppose that inside a layer the distribution of the weights is symmetric and the averaged value of the weights is equal to zero. As we show in what follows this assumption is nearly always carried out in the case of large deep neural networks.

We can estimate the covariation $\overline{wy}$ between the input and the quantized values as

$$\overline{wy} = \int_{-\infty}^{\infty} wy(w)\,p(w)\,dw \;. \tag{3}$$

Where $p(w)$ is the density of the weight distribution inside the layer. Since $y$ is a constant inside the interval $(x_i, x_{i+1})$, the last equation takes the form

$$\overline{wy} = \sum_{i=0}^{n-1} y_i \int_{x_i}^{x_{i+1}} wp(w)\,dw \;. \tag{4}$$

Similarly, the variance of the quantized values is

$$\sigma_y^2 = \int_{-\infty}^{\infty} y^2 p(w)\,dw = \sum_{i=0}^{n-1} y_i^2 \int_{x_i}^{x_{i+1}} p(w)\,dw \;. \tag{5}$$

If we introduce additional notations

$$c_i = \int_{x_i}^{x_{i+1}} wp(w)\,dw \text{ and } p_i = \int_{x_i}^{x_{i+1}} p(w)\,dw \;. \tag{6}$$

we can simplify Eqs. (4-5) significantly:

$$\overline{wy} = \sum_{i=0}^{n-1} y_i c_i \qquad \sigma_y^2 = \sum_{i=0}^{n-1} y_i^2 p_i \;. \tag{7}$$

To maximize the correlation (see Eq. 2) we have two sets of parameters. They are the boundaries of the intervals $x_i$ and the quantized values $y_i$ inside the intervals. For some time, we forget about splitting into intervals and suppose that we know the boundaries $x_i$. Then after the optimization procedure with regard to $y_i$ we obtain:

$$y_i = c_i / p_i \;. \tag{8}$$

When we substitute this value into Eq. (2), account for the formulas (4-5), and have in mind that $\sigma_w = const$ it is independent of either $x_i$ or $y_i$, we obtain the following optimization problem:

$$\rho(w,y) \square \sum_{i=0}^{n-1} c_i^2 / p_i \to \max \;. \tag{9}$$

We differentiate this expression with respect to $x_i$ and obtain an expression for the gradient $\nabla\rho_i$:

$$\nabla\rho_i = \frac{\partial\rho}{\partial x_i} = \frac{p(x_i)(y_{i+1} - y_i)(y_{i+1} + y_i - 2x_i)}{2\sigma_w} \;. \tag{10}$$

## III. Description of Quantization Procedure

The obtained expression (10) for the gradient $\nabla\rho_i$ allowed us to implement a quick algorithm for correlation maximization based on the gradient descent algorithm. In the course of the algorithm running it adjusts the boundaries of the intervals $x_i$, which we use as the optimization parameters. For the given values of $x_i$ we define the quantized weights $y_i$ with the aid of Eq. (8).

As the density function $p(w)$ we use its kernel estimation calculated taking into account 10,000 random weights from the layer. When the number of the weights is less than 10,000, all the weights have to be taken into account.

The integral formulas Eq. (6) we replaced by numerical estimates for $c_i$ and $p_i$ calculated using the real weights in the layer:

$$\begin{aligned} c_i &= \sum_w wI(x_i < w < x_{i+1}) / N_w \\ p_i &= \sum_w I(x_i < w < x_{i+1}) / N_w \end{aligned} \tag{11}$$

Here $N_w$ is the number of the weights in the layer, $I(x_i < w < x_{i+1})$ is an indicator function that accounts for the weights belonging to the interval $(x_i, x_{i+1})$ only.

To initialize the gradient ascent it is necessary to choose an initial partition that is an initial set $[w_{\min}, x_1, x_2, ..., x_{n-1}, w_{\max}]$. In our simulations, we used the linear and exponential partitions from [8] as the initial sets and examined quantization of the pre-trained neural networks ResNet-50, Xception, and VGG-16. We employed the programming language Python and the framework Keras [2].

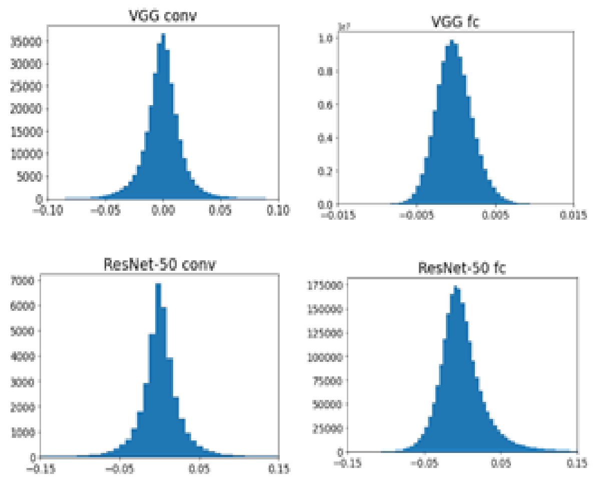

Fig. 1. Examples of weight histograms in convolution and fully connected layers for VGG-16 and ResNet-50.

As a result of minimization we obtained the optimal boundaries of the intervals $[w_{\min}, x_1, x_2, ..., x_{n-1}, w_{\max}]$ as well as the corresponding set of the quantized weights $[y_0, y_1, ..., y_{n-1}]$. Then we used the quantized weights in place of the input weights without post-training of the neural networks. The Python code is given in Appendix B.

## IV. Results

In Fig. 1, we show weight histograms for convolution and fully connected layers of the VGG-16 and ResNet-50 neural networks. As we mentioned before, the weight distributions in the layers of deep neural networks are nearly symmetric with respect to zero and their average values are close to zero.

TABLE I. CORRELATIONS AVERAGED OVER ALL LAYERS FOR RESNET-50 NEURAL NETWORK AFTER LINEAR (LINEAR) AND EXPONENTIAL (EXPONENTIAL) QUANTIZATION AND SUBSEQUENT CORRELATION MAXIMIZATION

| Average correlation for ResNet-50 | | | | |
|---|---|---|---|---|
| | Lin | Max | Exp | Max |
| 3 bit | 0.8169 | 0.9376 | 0.9461 | 0.9659 |
| 4 bit | 0.8473 | 0.9726 | 0.9822 | 0.9898 |
| 5 bit | 0.9233 | 0.9890 | 0.9945 | 0.9972 |
| 6 bit | 0.9739 | 0.9957 | 0.9983 | 0.9990 |
| 7 bit | 0.9925 | 0.9980 | 0.9995 | 0.9993 |

TABLE II. CORRELATIONS AVERAGED OVER ALL LAYERS FOR XCEPTION NEURAL NETWORK AFTER LINEAR (LINEAR) AND EXPONENTIAL (EXPONENTIAL) QUANTIZATION AND SUBSEQUENT CORRELATION MAXIMIZATION

| Average correlation for Xception | | | | |
|---|---|---|---|---|
| | Lin | Max | Exp | Max |
| 3 bit | 0.8630 | 0.9552 | 0.9566 | 0.9735 |
| 4 bit | 0.9072 | 0.9814 | 0.9855 | 0.9919 |
| 5 bit | 0.9611 | 0.9926 | 0.9955 | 0.9972 |
| 6 bit | 0.9880 | 0.9960 | 0.9986 | 0.9984 |
| 7 bit | 0.9967 | 0.9972 | 0.9996 | 0.9986 |

TABLE III. CORRELATIONS AVERAGED OVER ALL LAYERS FOR VGG-16 NEURAL NETWORK AFTER LINEAR (LINEAR) AND EXPONENTIAL (EXPONENTIAL) QUANTIZATION AND SUBSEQUENT CORRELATION MAXIMIZAION

| Average correlation for VGG-16 | | | | |
|---|---|---|---|---|
| | Lin | Max | Exp | Max |
| 3 bit | 0.8325 | 0.9342 | 0.9464 | 0.9669 |
| 4 bit | 0.8469 | 0.9659 | 0.9816 | 0.9899 |
| 5 bit | 0.8968 | 0.9862 | 0.9943 | 0.9973 |
| 6 bit | 0.9579 | 0.9943 | 0.9983 | 0.9992 |
| 7 bit | 0.9872 | 0.9980 | 0.9995 | 0.9995 |

TABLE IVa. TOP-1 ACCURACY FOR RESNET50 NEURAL NETWORK; EXP DENOTES EXPONENTIAL QUANTIZATION, OPT STANDS FOR OPTIMAL QUANTIZATION, MAX DENOTES THE BEST OF TWO WAYS OF QUANTIZATION, AND FAIL OPT IS FRACTION OF NEURAL NETWORK LAYERS WHERE OPTIMIZATION LEADS TO WORTH CORRELATION THAN EXPONENTIAL QUANTIZATION

| | **Res-Net-50** | | | |
|---|---|---|---|---|
| | %fail | exp | opt | max |
| 3 bit | 0% | 0.03 | 0.02 | 0 |
| 4 bit | 0% | 0.71 | 0.82 | 0.79 |
| 5 bit | 0% | 0.91 | 0.93 | 0.93 |
| 6 bit | 7% | 0.94 | 0.94 | 0.94 |
| 7 bit | 26% | 0.94 | 0.94 | 0.94 |
| 32 bit | **0.94** | | | |

In Tables 1, 2, and 3, we present the values of the average correlations of prior weights and weights after quantization for the ResNet-50, Xception, and VGG-16 neural networks. We used the linear and exponential quantization, implemented the correlation maximization algorithm, and after that calculated the values of the average correlations.

Tables 1 – 3 show that when the number of bits is small (up to 5 bits, i.e. when the number of intervals is less or equal to 32) the maximization algorithm does increase the correlation averaged over the layers. The linear quantization with the subsequent maximization leads to the average correlation growth in all the examined cases. The exponential quantization with the subsequent maximization provides the growth of the average correlation only when the number of intervals is equal to 8, 16 or 32. When the number of bits is larger (that is when the number of intervals is 64 or 128), the maximization algorithm fails since for some layers its results are worth comparing with the results of the exponential quantization.

TABLE IVb. TOP-1 ACCURACY FOR VGG-16 NEURAL NETWORK; EXP DENOTES EXPONENTIAL QUANTIZATION, OPT STANDS FOR OPTIMAL QUANTIZATION, MAX DENOTES THE BEST OF TWO WAYS OF QUANTIZATION, AND FAIL OPT IS FRACTION OF NEURAL NETWORK LAYERS WHERE OPTIMIZATION LEADS TO WORTH CORRELATION THAN EXPONENTIAL QUANTIZATION

| | **VGG-16** | | | |
|---|---|---|---|---|
| | %fail | exp | opt | max |
| 3 bit | 0% | 0.69 | 0.73 | 0.76 |
| 4 bit | 0% | 0.87 | 0.91 | 0.91 |
| 5 bit | 0% | 0.89 | 0.9 | 0.93 |
| 6 bit | 0% | 0.9 | 0.93 | 0.93 |
| 7 bit | 6% | 0.9 | 0.94 | 0.94 |
| 32 bit | **0.94** | | | |

TABLE IVc. TOP-1 ACCURACY FOR XCEPTION NEURAL NETWORK; EXP DENOTES EXPONENTIAL QUANTIZATION, OPT STANDS FOR OPTIMAL QUANTIZATION, MAX DENOTES THE BEST OF TWO WAYS OF QUANTIZATION, AND FAIL OPT IS FRACTION OF NEURAL NETWORK LAYERS WHERE OPTIMIZATION LEADS TO WORTH CORRELATION THAN EXPONENTIAL QUANTIZATION

| | **Xception** | | | |
|---|---|---|---|---|
| | %fail | exp | opt | max |
| 3 bit | 0% | 0 | 0.02 | 0.01 |
| 4 bit | 0% | 0.43 | 0.65 | 0.61 |
| 5 bit | 10% | 0.89 | 0.86 | 0.89 |
| 6 bit | 15% | 0.89 | 0.9 | 0.9 |
| 7 bit | 34% | 0.92 | 0.86 | 0.92 |
| 32 bit | **0.92** | | | |

May be the maximization algorithm does not always run correctly because the speed of the gradient ascent $lr$ has to be chosen more accurately. This was the reason why for each layer we also used an algorithm allowing us to select a quantization with the maximal correlation. When after our optimization the correlation decreased, we used the initial exponential quantization. Such an algorithm for quantizing a neural network, we called the “best”.

In Tables 4a-c are Top-5 accuracies for the ResNet-50, VGG-16 and Xception neural networks quantized using the exponential scale (exp), by the algorithm for maximizing the correlation with the prior exponential splitting (opt), and with the aid of the algorithm holding the exponential distribution in the layers where the correlation maximization failed (max). The column %fail max shows the percentage of layers for which $\rho_{\max corr} < \rho_{\exp}$. The examined neural networks confirmed our hypothesis that the larger the correlation between the input and quantized values the better the accuracy of the quantized neural network. This is true only if the above-mentioned correlation is larger on each layer of the neural network while an increase of the correlation averaged over all the layers does not guarantee an increase of the accuracy.

In Tables 4a-c, we also compare the Top-5 accuracies of the neural networks quantized with the aid of our best algorithm (max) with Top-5 accuracies of the initial neural networks (32 bit). For the neural networks ResNet-50 and Xception the Top-5 accuracy drop was 20-30% and less than 3.5% when for storing weights we reserved 4 bit and 5 bit, respectively.

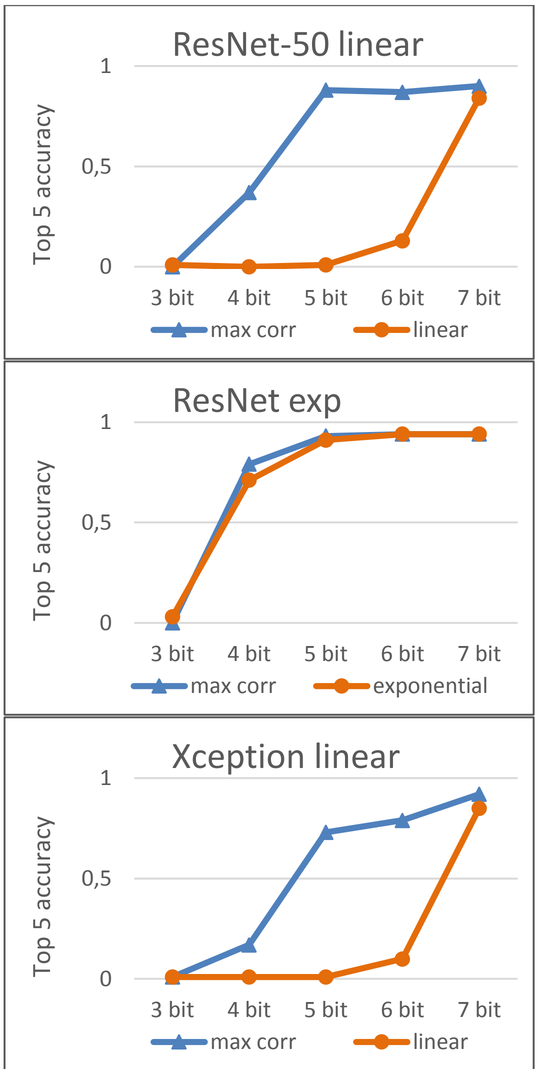

Fig. 2. Top-5 accuracies for ResNet-50, Xception and VGG-16: linear quantization and quantization with correlation maximization with prior linear splitting (max corr).

In the case of the VGG-16 neural network, the Top-5 accuracy drop was about 20% when we reserved 3 bits per layer and it was less than 3% when the number of the reserved bits was larger.

In Figure 3a-b, for the Xception, ResNet-50 and VGG-16 neural networks we show the dependences of the Top-5 accuracies on the number of bits reserved for storage of each weight obtained when we initialized them by linear and exponential splitting. The model accuracy increases when we choose the quantization with the maximal correlation between the quantized and input weights. The figures for Top-1 accuracies are given in Appendix A.

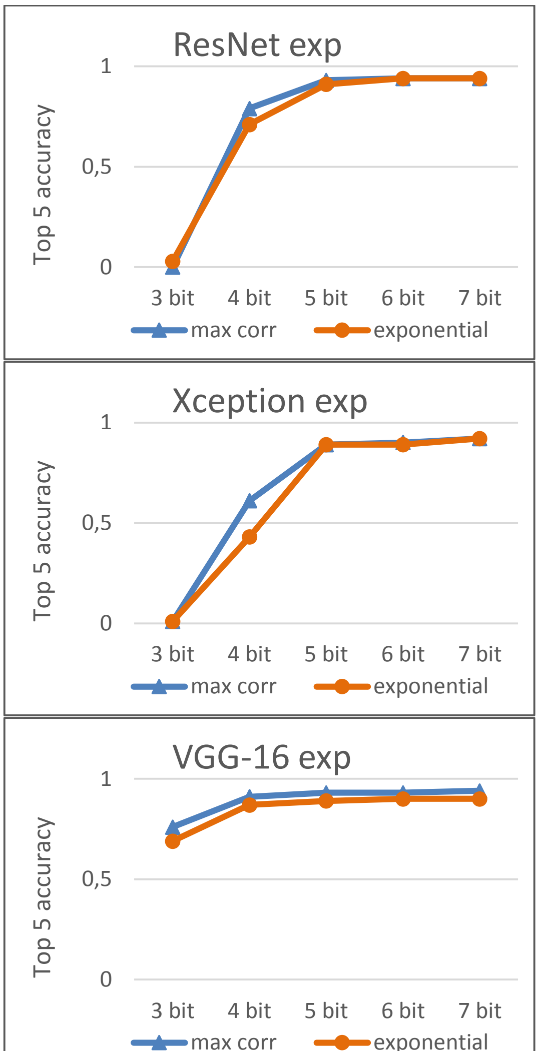

Fig. 3. Top-5 accuracies for ResNet-50, Xception and VGG-16: exponential quantization and quantization with correlation maximization with prior exponential splitting (max corr).

## V. Conclusions

We developed the algorithm for neural network quantization based on maximization of correlations between the quantized and the input weights. When using this algorithm no post-training of the neural network is necessary. Reserving 5 bits per layer, we succeeded in quantization of the VGG-16 neural network that leaded to 1% of the Top-5 accuracy drop only. Under such compression, the required memory necessary to store weights is approximately 6 times less than in the case of the full precision float (32 bits). For comparison, in paper [5] the VGG-16 neural network was compressed about 2.5 times by quantization and full algorithm compression is around 49 times, however their neural network required a post-training for which a substantial computing power was necessary. When comparing with the results of paper [8] we see that for the ResNet-50, VGG-16 and Xception neural networks at the same compression our Top-1 and Top-5 accuracies are better. Our compression allows to use only 3-4 bits to achieve more than 0.6 Top-5 accuracy for different architecture without re-training.

## Acknowledgment

The work financially supported by State Program of SRISA RAS No. 0065-2019-0003 (AAA-A19-119011590090-2).

## References

[1] ImageNet – huge image dataset [Online]. URL: http://www.image-net.org.

[2] Models for image classification with weights trained on ImageNet [Online]. URL: https://keras.io/applications/.

[3] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv Preprint: 1409.1556.

[4] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," arXiv Preprint: 1512.03385.

[5] S. Han, H. Mao and W.J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," CoRR, ArXiv Preprint: 1510.00149. 2, 2015.

[6] Sh. 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: 1606.06160.

[7] B.V. Kryzhanovsky, M.V. Kryzhanovsky and M.Yu. Malsagov, "Discretization of a matrix in quadratic functional binary optimization," Doklady Mathematics, vol. 83, pp. 413-417, 2011. DOI: 10.1134/S1064562411030197.

[8] M.Yu. Malsagov, E.M. Khayrov, M.M. Pushkareva and I.M. Karandashev, "Exponential discretization of weights of neural network connections in pre-trained neural networks," preprint, 2020.

[9] M. Courbariaux, Y. Bengio and J. David, "Training deep neural networks with low precision multiplications," arXiv Preprint: 1412.7024.

[10] M. Courbariaux, Y. Bengio, J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," Conference on Neural Information Processing Systems, arXiv:1511.00363.

[11] Zh. Lin, M. Courbariaux, R. Memisevic and Y. Bengio, "Neural networks with few multiplications," Proceedings of the International Conference on Learning Representations, arXiv:1510.03009.

[12] E.H. Lee, D. Miyashita, E. Chai, B. Murmann and S.S. Wong, "LogNet: Energy-efficient neural networks using logarithmic computation," Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2017.

[13] S. Han, J. Pool, J. Tran and W. Dally, "Learning both Weights and Connections for Efficient Neural Networks," arXiv: 1506.02626, 2015.

[14] W. Chen, J. Wilson, S. Tyree and K. Weinberger, "Compressing Neural Networks with the Hashing Trick. Compressing Neural Networks with the Hashing Trick," arXiv: 1504.04788, 2015.

[15] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets and V. Lempitsky, "Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition," 3rd International Conference on Learning Representations, ICLR San Diego, CA, USA, Conference Track Proceedings, 2015.

## Appendix A: Top-1 accuracies

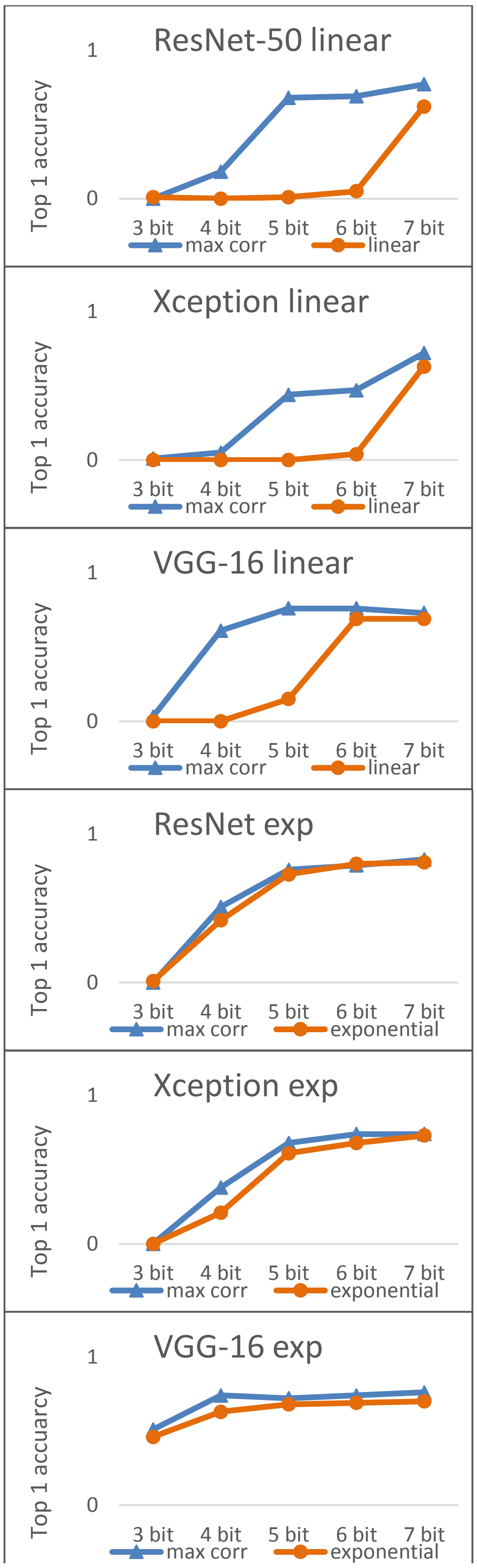

Fig. 4. Top-1 accuracies for ResNet-50, Xception and VGG-16: linear/ exponential (lin/exp) quantization and quantization with correlation maximization with prior linear/exponential splitting (max corr).

## Appendix B: Algorithm

```
# accessory functions
def f(x, func, kde, X, x_min, x_max):
    y, px, cov, p = func(x, kde, X, x_min, x_max)
    return cov

def grad(x, func, kde, X, x_min, x_max, alpha=10):
    y, px, cov, p = func(x, kde, X, x_min, x_max)
    step = alpha * px  * (y[1:] - y[:-1]) * (y[1:] + y[:-
1] - 2 * x) / 2
    return step

def cov_kde(x0, kde, X, x_min, x_max):
    '''
    calculate distribution function, quantized values
and covariation on the set x0
    X – weights in this layer,
    x_min and x_max – minimal and maximal weight
value in the layer
    x0 current set (variable values only)
    '''
    p = np.zeros(len(x0) + 1)
    C = np.zeros(len(x0) + 1)
    y = np.zeros(len(x0) + 1)
    x_ext = sorted(np.append(x0, [x_min, x_max]))
    for i in range(len(x_ext)-1):
        mask = np.logical_and(x_ext[i] < X, X <=
x_ext[i + 1])
        p[i] = len(X[mask])
        C[i] = np.sum(X[mask])
        if p[i] == 0:
            C[i] = 0
            p[i] = 1
    y = C / p
    px = kde.evaluate(x0)
    cov = np.linalg.norm(C / np.sqrt(p)) #/ sigma_kde
    return y, px, cov, p

def results(kde, w, x0, x_min, x_max, func, bits,
kde_std, ans_case='CG'):
    '''
    correlation maximization procedure for initial set
x0 (only variable values),
    w – layer,
    kde – kernel density estimation on random sample
from weights,
    x_min and x_max – minimal and maximal weight
values
    '''
    n_d = 2 ** bits
    fx = lambda x: -f(x, func, kde, w, x_min, x_max)
    gradx = lambda x: -grad(x, func, kde, w, x_min,
x_max, alpha)
    tol_curr = 1e-4
    alpha = 10
    ans = minimize(fun=fx, x0=x0, jac=gradx,
method='CG', tol=tol_curr
    solutions = ans['x']
    correlations = -ans['fun']
    gradients = np.linalg.norm(gradx(ans['x'])) / alpha
/ n_d
    return solutions, correlations, gradients
```