Data-Free Network Compression via Parametric Non-uniform Mixed Precision Quantization

Vladimir Chikin
Huawei Noah's Ark Lab
vladimir.chikin@huawei.com

Mikhail Antiukh
HSE University
mikhail.antiukh@gmail.com

Abstract

Deep Neural Networks (DNNs) usually have a large number of parameters and consume a huge volume of storage space, which limits the application of DNNs on memory-constrained devices. Network quantization is an appealing way to compress DNNs. However, most of existing quantization methods require the training dataset and a fine-tuning procedure to preserve the quality of a full-precision model. These are unavailable for the confidential scenarios due to personal privacy and security problems. Focusing on this issue, we propose a novel data-free method for network compression called PNMQ, which employs the Parametric Non-uniform Mixed precision Quantization to generate a quantized network. During the compression stage, the optimal parametric non-uniform quantization grid is calculated directly for each layer to minimize the quantization error. User can directly specify the required compression ratio of a network, which is used by the PNMQ algorithm to select bitwidths of layers. This method does not require any model retraining or expensive calculations, which allows efficient implementations for network compression on edge devices. Extensive experiments have been conducted on various computer vision tasks and the results demonstrate that PNMQ achieves better performance than other state-of-the-art methods of network compression.

1. Introduction

DNNs have achieved impressive results in a large number of problems from image classification to various generation tasks, etc. But with an increasing number of problems to be solved the models grow in size significantly, architectures become more complex, the number of parameters is estimated in hundreds of millions. Unfortunately, in the real world, the use and storage of such models is difficult, especially on various peripheral and mobile devices for which computing and memory resources are a bottleneck. Since the demand for the use of neural networks in various mobile processes and applications is growing, there is a great need for matching neural models to the technical parameters of devices. Many researchers try to offer some approaches to solve this problem, such as making changes to existing model structures or applying various compression techniques to the original structure.

Popular approaches include techniques such as pruning, quantization, encoding methods, knowledge distillation and their various combinations. Now there are a lot of well-known methods and ready-made frameworks that can significantly reduce the model size, but to keep a high quality of the compressed model, many of these methods require a lot of training data and expensive calculations such as model retraining, which is a significant drawback. Methods that do not use expensive calculations for compression often produce a significant quality drop of the compressed models, since the use of uniform quantization without model training leads to a low quality of approximation from quantization for most of the weights.

In this paper, we propose a novel method of network compression called PNMQ, which does not use expensive calculations such as gradient descent training or clustering, and at the same time allows us to achieve better results compared to existing data-free and training-free compression methods, and even to some methods that use data, model training or other additional compression techniques. First of all, we propose a parametric family of non-uniform quantization grids that depends on a single scalar parameter and is therefore easily optimized. The use of the proposed non-uniform quantization grids significantly improves the quality of quantization approximation relative to the uniform quantization. We provide an optimization pipeline, which allows to tune these non-uniform grids for each model layer, both without using data and using a small amount of data. In addition, we demonstrate that different layers of models require different bitwidths for a high degree of quantization approximation, and propose a universal data-free algorithm for selecting the sufficient bitwidths of layers based on comparison of the quantization errors of different layers and bitwidths. This algorithm automatically selects the optimal bitwidths for different layers to achieve the required compression ratio that the user can specify directly, and this is a very convenient feature of the proposed method. PNMQ can be applied to any model, does not depend on the layer types and does not require changing the network structure. Moreover, PNMQ is compatible with most existing weight compression techniques such as weights pruning, transformations or lossless coding.

2. Related works

One of the fairly popular technique for reducing the number of model parameters is pruning [7, 14, 38, 40]. The idea of pruning is to zero out individual insignificant neurons in the neural network (*unstructured pruning*) or to reduce some subsets (kernels, channels) of the weight tensors (*structured pruning*). Knowledge distillation means training a completely new smaller model, which is trained on the outputs of the original model [1, 16]. This approach can show good compression results but it requires expensive training and implies a change in the structure of the original model. Quantization is a powerful tool for compressing models by using low-precision numbers. It can be uniform [9, 20, 30, 44] or non-uniform [5, 6, 18, 27, 46]. For uniform quantization, special function Q that maps the model parameters to a finite uniformly distributed set is usually used. A common example of a such function is the following:

$$Q(x) = \lfloor x/s \rceil, \tag{1}$$

where x is a tensor with float values, and s is the quantization scale. By $\lfloor \cdot \rceil$ we denote the round function by the set of integers from the segment $[-2^{n-1}, 2^{n-1} - 1]$, where n is the quantization bitwidth. With uniform quantization, we pay equal attention to the values over the entire quantization interval. In practice, however, model weights can be distributed unevenly, and non-uniform quantization can provide a better approximation of more important areas in weight tensors.

Non-uniform quantization Weights clustering is one of the ways of non-uniform quantization [3, 10, 23, 41, 49]. The main idea of weights clustering is to divide the weight tensor into several clusters, the number of which is determined by the required bitwidth. We need to store full-precision values of the cluster centers and integer labels of clustered weight components. One of the most popular methods for weights clustering is the Lloyd-Max quantizer [23, 26, 36, 41]. Despite the popularity of weights clustering approach, it has significant drawbacks – it requires expensive clustering procedure and storing a big set of float numbers, which negatively affects the compression ratio. In our work, we propose a different approach to non-uniform quantization, which solves the above-mentioned disadvantages.

Compression-aware training Model training during compression is a popular way to maintain the quality of a model with a high compression ratio [4, 25, 34, 41–43]. There are many works about quantization and pruning-aware training, in which the authors achieved sufficiently high compression ratios with a final quality close to the original. Compression-aware training methods have a few drawbacks. They require a lot of data and long training, which is often impossible in real problems. The problem with data can be solved by special methods of generating artificial data [11, 17, 39] for any compression approach, but such methods also require expensive model training. In our work, we have focused on compression approach without training and showed that our techniques can compete and even beat the methods with training.

Compression without training It is a more difficult and demanding task as it has a higher execution speed and can be effectively applied on mobile devices [15, 22, 47, 48]. One of the popular techniques is the application of special transformations to the weight tensors [22, 47] aimed at changing the distribution of weights to increase the compression ratio. Cross-layer equalization and quantization bias correction techniques from [29], which is one of our main direct competitors, are effective methods for preserving quality of the quantized model that do not require data and model retraining. One of the popular and effective methods is post-training quantization method AdaRound [28], which allows to adaptively adjust the round function for different weight components of model layers. Despite the fact that this method does not involve model training, it uses data and a gradient descent procedure to adjust a big set of its own special parameters, which is a computationally complex procedure. Batch normalization folding before model quantization [19] is also a useful step, since in this case we do not need to store the parameters of batch normalizations in a compressed model.

Lossless compression The final step in compression procedure can be the application of efficient lossless coding algorithms to the quantized model parameters. There are many well-known coding methods, such as entropy coding (Huffman) or universal coding (arithmetic) algorithms. A popular approach is to use bzip2 encoder [2, 22]. Method described in papers [31, 45] demonstrates a compression algorithm based on binary arithmetic coder CABAC, which achieves one of the best results among the compression methods without model training.

3. Method description

Our proposed compression method consists of several steps. The overall pipeline of the algorithm is shown in Fig. 1. PNMQ is based on a special type of non-uniform quantization of the model weights, as well as on a special al-

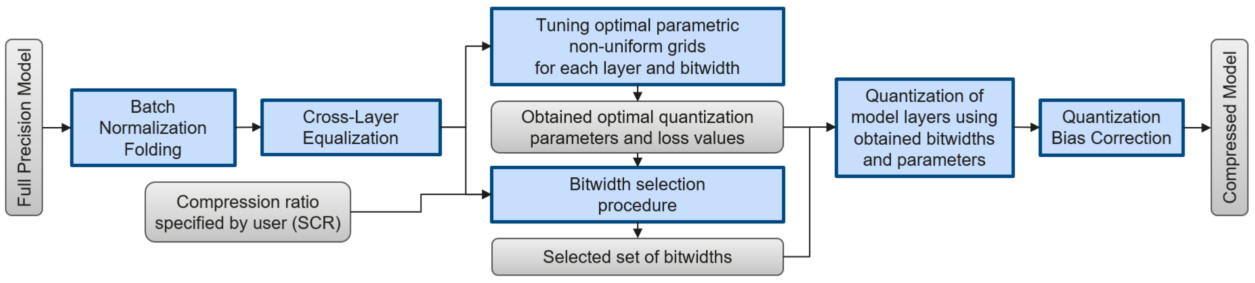
Full Precision Model
Batch Normalization Folding
Cross-Layer Equalization
Compression ratio specified by user (SCR)
Tuning optimal parametric non-uniform grids for each layer and bitwidth
Obtained optimal quantization parameters and loss values
Bitwidth selection procedure
Selected set of bitwidths
Quantization of model layers using obtained bitwidths and parameters
Quantization Bias Correction
Compressed Model

Figure 1. After batch normalization folding and cross-layer equalization, we iterate over the bitwidths for each model layer and solve a special optimization problem for obtaining values of non-uniform quantization parameters for each of them. Based on the achieved results and the compression ratio specified by the user (SCR), the bitwidths allowed for different layers are estimated. The method uses those bitwidths and the parameters calculated earlier for a non-uniform quantization of the model layers. Finally, we apply quantization bias correction.

gorithm for selecting the bitwidths of different layers. During the compression procedure, special optimal parametric non-uniform quantization grid is tuned for each layer. PNMQ supports two modes: *Data-Free* – in this case the data are not used for compression, and *Data-Aware* – in this case a small number of calibration samples are used for compression. A distinctive and very convenient feature of PNMQ is that the required compression ratio of model is a parameter of the method that a user can specify directly. We call this parameter the *Specified Compression Ratio* (*SCR*). SCR is used by the algorithm during the selection of bitwidths of different layers, resulting in the final compression ratio of the quantized model being almost equal to the specified value.

In addition to a non-uniform mixed precision quantization of the model weights, PNMQ uses preliminary batch normalization folding [19], and also cross-layer equalization and quantization bias correction techniques from [29]. No data are required for the use of these procedures. Finally, we propose applying lossless encoding to quantized weights to further increase the compression ratio. In this paper, we use traditional Huffman coding algorithm – a popular dictionary-based entropy coding algorithm, although any encoding algorithm can be used in combination with PNMQ.

3.1. Parametric non-uniform grids

In this paper, by *grid* we mean the set of points. Let W be the weight tensor of a layer, n – the quantization bitwidth, $\tau = 2^{n-1}$, and s is a quantization scale. We propose to use the following non-uniform grid for quantization:

$$G_p^n = \{x_i : x_i = -d\sum_{k=0}^{i} p^k, i \in [0, \tau-1]\} \cup \\ \cup \{0\} \cup \{y_i : y_i = d\sum_{k=0}^{i} p^k, i \in [0, \tau-2]\} \tag{2}$$

Parameter $p \in [1, 2]$ defines this grid, and parameter d is expressed in terms of p and τ:

$$d = \frac{\tau}{1 + p + p^2 + \cdots + p^{\tau-1}} \tag{3}$$

This grid contains 2^n elements, and due to this choice of d, the maximum absolute value of the numbers from this grid $|x_{\tau-1}|$ is equal to τ. Using $\lfloor\cdot\rceil_p^n$, we denote the rounding procedure for non-uniform grid G_p^n. We provide an effective way to implement rounding to such non-uniform grids in Appendix A. The values of the components of the tensor $\lfloor W/s\rceil_p^n$ belong to a discrete set, but are not integers. To store integer values, we use integer indices of these components in the proposed non-uniform grid G_p^n. We denote the tensor of these integer indices by I_p^n. In the case of n-bit quantization, the set of values of these indices is a set of numbers from 0 to $2^n - 1$, which in fact corresponds to the use of n-bit arithmetic when storing compressed weights of the model.

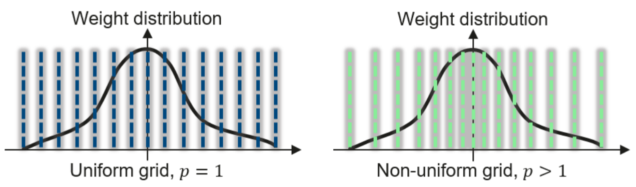

Figure 2. Example of the proposed parametric non-uniform grid in comparison with a uniform grid for $n = 4$.

For $p = 1$, the proposed grid is uniform, and as p increases, the distance between the adjacent numbers close to zero decreases, and increases between the adjacent numbers far from zero, see Fig. 2. The proposed technique of non-uniform quantization allows us to approximate small modulo weight values in more detail, which, as a rule, make up most of the weight tensors. Sometimes the distribution of the weights of some layers has a very low entropy, and for such layers the use of the proposed non-uniform quantization especially improves the average accuracy of the quantization approximation.

3.2. Setting the quantization parameters

For each model layer with weight tensor W and given bitwidth n, we tune the proposed parametric non-uniform quantization grid (parameter p) and the quantization scale (parameter s) by solving a special optimization problem. For data-free mode we propose to minimize the following loss function:

$$L_{DF}(W, s, p) = \left\| W - s \left\lfloor \frac{W}{s} \right\rceil_p^n \right\| \longrightarrow \min_{s,p}, \tag{4}$$

and for data-aware mode we propose to minimize the following loss function:

$$L_{DA}(W, X, s, p) = \left\| WX - s \left\lfloor \frac{W}{s} \right\rceil_p^n X \right\| \longrightarrow \min_{s,p}, \tag{5}$$

where X is some batch of data. For our data-aware method, we use significantly less data than during source model training. Parameter p is optimized on the segment $[1, 2]$, and parameter s is optimized on the segment $[0, \frac{\max|W|}{\tau}]$. We optimize the proposed loss functions with respect to only two parameters s and p. In this regard, to solve these optimization problems and find the optimal values of parameters s and p, we can use fast algorithms that use brute force and do not use gradient descent. Due to this, the deployment of PNMQ is possible on most mobile devices.

We have compared different types of norms that can be used in the proposed loss functions. Based on the experimental results, we propose to use L_4 norm as the most effective. See Appendix B for a detailed comparison of various norms.

3.3. Selection of the layer bitwidths

As a rule, in order to achieve good accuracy of quantization approximation, different layers require different quantization bitwidths. We propose a method for selecting different bitwidths for different layers, which takes this fact into account. This method is based on the use of a special function to estimate the accuracy of quantization approximation, same for different layers and bitwidths. This approach allows us to obtain a significantly higher compression ratio without reducing the quality compared to quantization of the entire model to a fixed bitwidth.

The minimum and maximum possible bitwidth values n_{min} and n_{max} are set by user. In our work, we use $n_{min} = 3$ and $n_{max} = 8$ as default values. For each layer and bitwidth value from segment $[n_{min}, n_{max}]$, the method solves the problem of minimizing the loss from Eq. (4) or Eq. (5) and preserves the achieved minimum value L_{opt} of this loss with the corresponding optimal parameters of non-uniform quantization s and p, using which this loss value is achieved. This is most time-consuming part of our compression algorithm. All further calculations require absolutely insignificant time.

The scheme of the proposed bitwidth selection procedure is shown in Fig. 3. The method collects all values of L_{opt} for different layers and bitwidths into a general list and sorts it in ascending order. Next, the method iterates through this list from the smallest to the largest L_{opt} value, and for each current value of L_{opt} considers it as a threshold value. For each threshold value, the method forms a set of layer bitwidths according to the following rule: for each layer, we select the minimum bitwidth, at which the value of L_{opt} is less than the current threshold L_{opt}. Since used loss functions from Eq. (4) and Eq. (5) are the norms of the quantization error, the value of L_{opt} monotonically decreases as the bitwidth value increases.

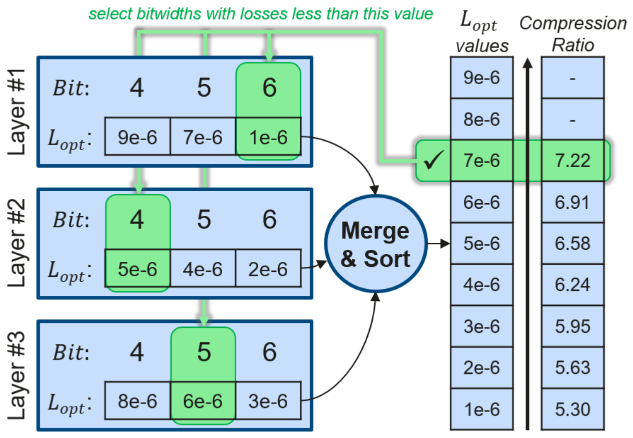

Figure 3. The scheme of the bitwidth selection procedure. Example for SCR equal to 7.

Based on the obtained set of layer bitwidths, we can estimate the current *Compression Ratio* (*CR*) of the model using the following formula:

$$CR = \frac{F \cdot 32}{\sum_{i \in Q} p_i \cdot b_i + B \cdot 32 + M}, \tag{6}$$

where F is the number of parameters in the full-precision model, Q – a set of quantized model layers, p_i – the size of the i-th layer, b_i – bitwidth of the i-th layer, B – the number of the full-precision parameters in the quantized model (biases and quantization parameters) and M – the size of metadata with the information about bitwidths of layers. In practice, the value of M turns out to be absolutely insignificant. As the current threshold value L_{opt} increases, the compression ratio from Eq. (6) of the model monotonically increases. When the SCR is reached, we stop and fix the current threshold value L_{opt} and the corresponding set of bitwidths for different layers. We use this set of bitwidths and corresponding stored optimal parameters s and p to quantize the model layers.

3.4. Decompression procedure

Decompression procedure is performed very quickly. Having the bitwidth parameter n and the parameter of the non-uniform quantization p for each layer, we can restore the used non-uniform quantization grid G_p^n. Having the scale factor s and the compressed model weights encoded by integer indices I_p^n of the points of grid G_p^n, we can calculate the approximate values of the initial weights of the model – model weights after decompression:

$$W = s \cdot \lfloor W/s \rceil_p^n = s \cdot G_p^n\left(I_p^n\right). \tag{7}$$

4. Experiments

In this section, we evaluate the effectiveness of our compression method. In this regard, we conducted experiments with several different widely-used neural networks for image classification, object recognition and image segmentation tasks. In all experiments, we quantize the weights of all model layers using symmetric per-tensor quantization [21], and also conduct batch normalization folding before quantization. It is clear that the use of additional techniques such as affine quantization or per-channel quantization [21] can improve the final quality of the compressed model. However, these techniques involve storing additional float parameters, which entails a certain reduction of the compression ratio. See the results of additional experiments with per-channel version of PNMQ in Appendix C.

Experimental setup In our experiments, we solve optimization problems from Eq. (4) and Eq. (5) using a naive brute force algorithm that selects optimal values of parameters s and p as a result of optimization on a uniform grid. Any other optimization method can be used, in particular, more efficient brute force algorithms. In all our experiments with data-aware version of our method we used 32 random samples for object recognition task and 160 random samples for image classification task from the training dataset. Since the results of the data-aware method depend on the used data, we provide the average quality metrics for several runs with different random sets of samples from the training set.

Metrics To evaluate compression methods, we use the following widely-used performance metrics:

- Top-1 and Top-5 accuracy of the compressed model for image classification tasks.
- Mean Average Precision (mAP) and mean Average Recall (mAR) of the compressed model for object detection (*Bbox*) and image segmentation (*Mask*) tasks.
- Compression ratio of the model after quantization, which is determined by formula from Eq. (6).
- Compression ratio for quantized models after applying Huffman coding algorithm to the quantized part of model, taking into account the codebook size. To calculate this coefficient, one need to replace the size of the quantized part of model with the size of the resulting Huffman code in formula from Eq. (6), and also add the codebook size to the denominator.

Most of the works with which we compare our results use a similar measurement scheme for the compression ratio. With the exception of a few papers, in [47] the authors calculate the compression as the ratio of the original size of the weight tensors to the quantized one, without taking into account the quantization parameters and other full-precision parameters. On the contrary, the authors of [22] take into account not only the quantization parameters, but also other metadata of the model, such as information about the structure, layers, etc., which may insignificantly reduce the compression ratio. Also, some works use other approaches instead of the Huffman algorithm, for example bzip2 [22] and CABAC [31, 45] coders.

4.1. Image classification

Ablation study Our proposed method consists of several independent techniques, and in this section we examine impact from them. For this, we conducted experiments with ResNet-50 [13] and MobileNet-v2 [37] models on the ImageNet dataset [8], see Tab. 1. As a baseline, we consider the DFQ method for weights quantization only. After folding of batch normalizations, we apply the cross-layer equalization, quantize the weights of all layers, and then apply the quantization bias correction. We do not apply bias absorbing because it is aimed at efficient quantization of activations. We perform the listed DFQ techniques in all experiments in this section. Since our method allows tuning of quantization scales using a special loss optimization, we compare our method with a version of DFQ, in which we apply a similar procedure for scale tuning, and also with a version of PNMQ, which uses uniform quantization (p = 1). In addition, we conducted experiments using Lloyd-Max quantizer for non-uniform quantization of models to fixed bitwidth. In our experiments, we used the implementation of Lloyd-Max quantization method from *scikit-learn* library [33]. We also implemented the combination of Lloyd-Max quantizer with our bitwidth selection method – in this case, the non-uniform grid tuning procedure is replaced by the weight clustering procedure using Lloyd-Max algorithm. We call

Table 1. Compression of ResNet-50 and MobileNet-v2 models on ImageNet. Investigation of the impact of the used techniques.

Model	Method	SCR	Top-1 acc.	Top-5 acc.	CR without Huffman coding	CR with Huffman coding
ResNet-50	Baseline DFQ	all to 5 bit	32.08%	55.96%	6.36	15.17
	DFQ with scale tuning	all to 5 bit	46.60%	71.69%	6.36	11.67
	DFQ + Lloyd-Max	all to 5 bit	73.77%	91.74%	6.37	6.48
	DFQ + Mix-Bit Lloyd-Max	6.36	75.56%	92.71%	6.43	6.59
	Data-Free PNMQ with $p = 1$	6.36	74.17%	91.96%	6.37	11.4
	Data-Free PNMQ	6.36	75.32%	92.68%	6.43	8.34
	Data-Aware PNMQ	6.36	75.50%	92.74%	6.46	7.8
MobileNet-v2	Baseline DFQ	all to 5 bit	55.31%	79.34%	6.23	8.81
	DFQ with scale tuning	all to 5 bit	64.14%	85.50%	6.23	7.98
	DFQ + Lloyd-Max	all to 5 bit	68.62%	88.25%	6.24	6.32
	DFQ + Mix-Bit Lloyd-Max	6.23	70.93%	89.91%	6.56	6.68
	Data-Free PNMQ with $p = 1$	6.23	69.73%	89.13%	6.32	7.83
	Data-Free PNMQ	6.23	70.35%	89.71%	6.32	7.15
	Data-Aware PNMQ	6.23	70.64%	89.74%	6.26	6.75

this approach *Mix-Bit Lloyd-Max*. That allows us to significantly improve the compression results relative to the traditional Lloyd-Max approach, however, the Mix-Bit Lloyd-Max method takes a very long time to run.

We set the value of SCR equal to the compression ratio in the case of baseline DFQ. As can be seen from Tab. 1, in the case of our methods, the final compression ratio (CR) of models is indeed almost equal to the given value of SCR. In turn, the compression ratio increases after the subsequent application of Huffman coding to quantized weights. From the presented results it can be seen that, as a rule, the compression ratio with the Huffman coding of the baseline uniform quantization is larger than in the case of our approach with non-uniform quantization. This can be explained by the fact that the non-uniform grid approximates area around zero more actively and allocates more values to this area; thus, quite fewer quantized values are zeroed out than during uniform quantization. Since Huffman's algorithm is entropy coding, it encodes the most frequent value (zero) using the smallest number of bits. Thus, a uniform grid will provide a greater degree of compression, since it converts a larger number of values of the weight tensor to zero.

We can also observe that our data-free approach has a better compression ratio than our data-aware method but smaller accuracy despite the equal number of float and quantized parameters. We explain this by the fact that the selection of parameters s and p in the data-aware version is performed by optimizing the loss function, which contains more information about the original problem leading to a higher quality. Similar observations were made in [28]. At the same time, this approach can better approximate the area around zero, which can lead to a slightly worse compression with the Huffman coding.

Comparison with the state-of-the-art In this section we compare our methods with state-of-the-art methods for neural network compression. We conducted experiments with ResNet-18, ResNet-50 [13] and MobileNet-v2 [37] models on the ImageNet dataset, see Tab. 2. We compare our results with the results of experiments aimed exclusively at the weight compression, without quantization of activations. Our direct competitors are the following methods: DFQ [29], OSC [48], Transform coding [22] and DeepCABAC [31], which do not use data, pruning and tuning of a large number of parameters by gradient descent. We demonstrate that PNMQ show better results not only among the main known methods of data-free compression and compression without training, but also can be comparable or even better than some compression-aware training methods and quantization methods with pruning, despite the fact that these approaches are not our direct competitors since they use more complex techniques that are poorly applicable in real tasks and on mobile devices. Method [25] has a slightly higher compression ratio, but lower accuracy of compressed model than PNMQ on ResNet-18 model. This approach uses a complex weight retraining during model binarization procedure. We also have better result in terms of compression ratio and accuracy than CLIP-Q method [43], which uses model retraining and weights pruning. Also note that the results of PNMQ are comparable with the results of the method from [47], which uses special transformations and pruning of the weight tensors, and with the results of AdaRound method [28], which uses data and the gradient descent procedure to tune a large number of parameters for adaptive rounding.

Table 2. Comparison with the state-of-the-art: results of different weight compression methods on ImageNet.

Model	Method	Data free	Pruning	Gradient descent for tuning	Top-1 acc.	Top-5 acc.	CR	CR with additional coding
ResNet-18	Full-precision model	-	-	-	69.76%	89.08%	1	-
	DFQ (our impl.)	✓	×	×	68.81%	88.52%	4.56	7.98
	Data-Free PNMQ	✓	×	×	69.13%	88.65%	6.61	8.82
	AdaRound [28]	×	×	✓	68.71%	-	7.97	-
	Lin *et al.* [25]	×	×	✓	68.30%	-	8.36	-
	row-KLT [47]	×	✓	×	68.50%	88.40%	8.2	-
	row-ELT [47]	×	✓	×	68.60%	88.50%	8.2	-
	Data-Aware PNMQ	×	×	×	69.21%	88.76%	6.13	7.4
ResNet-50	Full-precision model	-	-	-	76.13%	92.86%	1	-
	DFQ (our impl.)	✓	×	×	74.67%	92.18%	4.55	7.77
	OCS [48]	✓	×	×	66.2%	-	7.94	-
	Transform Coding [22]	✓	×	×	-	90.86%	-	9.7
	Transform Coding [22]	✓	×	×	-	91.86%	-	8.1
	DeepCABAC [31]	✓	×	×	74.99%	-	-	4.44
	Data-Free PNMQ	✓	×	×	73.61%	91.92%	7.94	10.88
	Data-Free PNMQ	✓	×	×	75.32%	92.68%	6.43	8.34
	AdaRound [28]	×	×	✓	75.23%	-	7.94	-
	row-KLT [47]	×	✓	×	74.80%	92.30%	7.8	-
	row-ELT [47]	×	✓	×	74.70%	92.40%	8.4	-
	Data-Aware PNMQ	×	×	×	73.95%	91.99%	7.95	9.9
	Data-Aware PNMQ	×	×	×	75.50%	92.74%	6.46	7.8
MobileNet-v2	Full-precision model	-	-	-	71.88%	90.29%	1	-
	DFQ (our impl.)	✓	×	×	68.48%	88.36%	5.22	6.87
	DeepCABAC [31]	✓	×	×	71.48%	-	-	3.5
	Data-Free PNMQ	✓	×	×	70.35%	89.71%	6.32	7.15
	Data-Free PNMQ	✓	×	×	71.62%	90.25%	4.61	5.01
	AdaRound [28]	×	×	✓	69.78%	-	7.72	-
	CLIP-Q [43]	×	✓	✓	70.30%	-	6.14	-
	Data-Aware PNMQ	×	×	×	70.64%	89.74%	6.23	6.75
	Data-Aware PNMQ	×	×	×	71.68%	90.20%	4.64	4.91

4.2. Object recognition and image segmentation

We conducted experiments with Faster R-CNN [35] for object recognition task and with Mask R-CNN [12] for object segmentation task on the COCO-2017 dataset. Full-precision models with ResNet-50 backbone trained on COCO-2017 dataset [24] are taken from TorchVision [32]. We provide quality metrics of the compressed models, see Tab. 3, and several examples of the output images produced by the compressed models, see Fig. 4 and Appendix D. Our results show that the quality of models compressed by DFQ is much worse than the quality of models compressed by PNMQ with a similar compression ratio. For example, after compressing the models using 6-bit DFQ method, the quality of compressed models significantly decreases, in particular, there are loss of objects, inaccurate masks and other artifacts, see Fig. 4. After compressing the models using PNMQ methods with SCR of 7.64, the quality of compressed models is very close to the quality of full-precision models.

5. Conclusion

In this paper we have proposed a novel method PNMQ for networks compression. Our approach belongs to techniques that do not require data and any kind of model retraining, and provides excellent results not only by the standards of data-free methods but also among the approaches that use data and complex calculations for compression. Thus, in our work, we solve the problem of compressing

Table 3. Comparison of baseline DFQ method and our compression methods for Faster R-CNN and Mask R-CNN models on COCO-2017.

Method	Bitwidth	SCR	Bbox mAP	Bbox mAR	Mask mAP	Mask mAR	CR without Huffman coding	CR with Huffman coding
Full-precision model	32	-	0.379	0.519	0.346	0.474	1	1
Baseline DFQ	4	-	0	0.001	0	0.001	7.64	20.48
	5	-	0.249	0.371	0.233	0.350	6.18	14.02
	6	-	0.350	0.486	0.322	0.449	5.18	9.77
Data-Free PNMQ	mix	9.5	0.332	0.468	0.32	0.446	9.53	14.11
	mix	7.64	0.369	0.504	0.341	0.466	7.65	10.46
	mix	6.18	0.377	0.516	0.345	0.473	6.18	7.97
Data-Aware PNMQ	mix	9.5	0.346	0.487	0.327	0.459	9.53	12.61
	mix	7.64	0.371	0.509	0.343	0.469	7.66	9.62
	mix	6.18	0.377	0.516	0.346	0.473	6.19	7.43

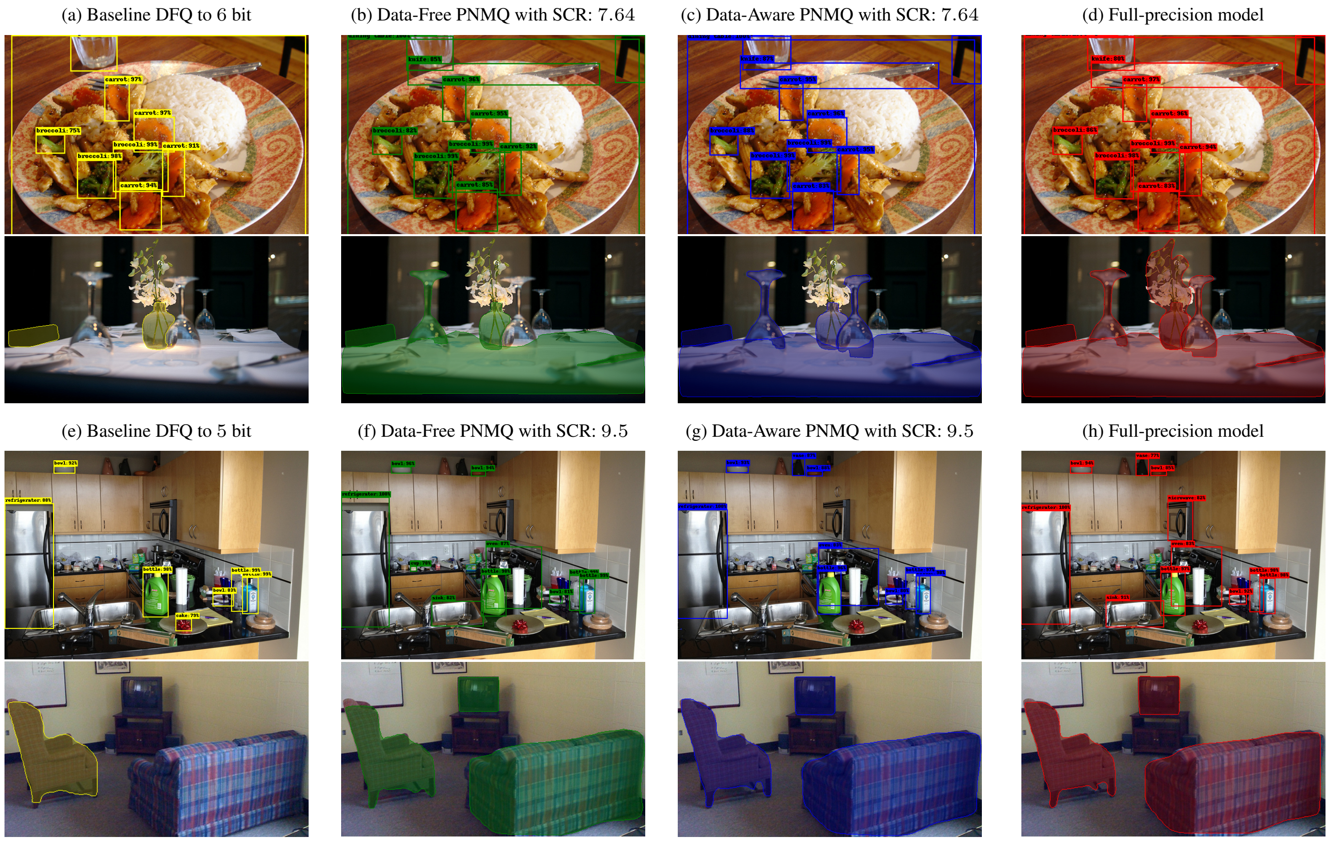
(a) Baseline DFQ to 6 bit
(b) Data-Free PNMQ with SCR: 7.64
(c) Data-Aware PNMQ with SCR: 7.64
(d) Full-precision model
(e) Baseline DFQ to 5 bit
(f) Data-Free PNMQ with SCR: 9.5
(g) Data-Aware PNMQ with SCR: 9.5
(h) Full-precision model

Figure 4. Comparison of baseline DFQ method and PNMQ methods for Faster R-CNN and Mask R-CNN models on COCO-2017.

models in real conditions when data and large computational resources may not be available for many reasons. PNMQ is quite simple to implement and apply, since it does not require any changes of the model and implies the optimization of very simple functionals. It can be easily compatible with various other compression techniques, such as pruning, special weight transformations, coding algorithms, and also with advanced quantization techniques, such as per-channel quantization or affine quantization, to further increase the compression ratio and the quality of the compressed models. We hope that our work will be able to inspire research community to further improve compression techniques, including by combining different ideas with our approach.

References

[1] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006. 2

[2] Zhuo Chen, Weisi Lin, Shiqi Wang, Lingyu Duan, and Alex C Kot. Intermediate deep feature compression: the next battlefield of intelligent sensing. *arXiv preprint arXiv:1809.06196*, 2018. 2

[3] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543*, 2016. 2

[4] Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Universal deep neural network compression. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):715–726, 2020. 2

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations, 2015. 2

[6] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016. 2

[7] Elliot J Crowley, Jack Turner, Amos Storkey, and Michael O'Boyle. A closer look at structured pruning for neural network compression. *arXiv preprint arXiv:1810.04622*, 2018. 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[9] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4852–4861, 2019. 2

[10] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014. 2

[11] Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2020. 2

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 7

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 5, 6

[14] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018. 2

[15] Yuhang He, Ziyu Pan, Lingxi Li, Yunxiao Shan, Dongpu Cao, and Long Chen. Real-time vehicle detection from short-range aerial image with compressed mobilenet. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8339–8345. IEEE, 2019. 2

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[17] Maxwell Horton, Yanzi Jin, Ali Farhadi, and Mohammad Rastegari. Layer-wise data-free cnn compression. *arXiv preprint arXiv:2011.09058*, 2020. 2

[18] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations, 2016. 2

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018. 2, 3

[20] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *CoRR*, abs/1712.05877, 2017. 2

[21] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 5

[22] Thorsten Laude, Yannick Richter, and Jörn Ostermann. Neural network compression using transform coding and clustering. *arXiv preprint arXiv:1805.07258*, 2018. 2, 5, 6, 7

[23] Clément Levrard. Quantization/clustering: when and why does k-means work? *Journal de la société française de statistique*, 159(1):1–26, 2018. 2

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 7

[25] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. *arXiv preprint arXiv:1711.11294*, 2017. 2, 6, 7

[26] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 2

[27] Daisuke Miyashita, Edward H Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016. 2

[28] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020. 2, 6, 7

[29] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019. 2, 3, 6

[30] Maxim Naumov, Utku Diril, Jongsoo Park, Benjamin Ray, Jedrzej Jablonski, and Andrew Tulloch. On periodic functions as regularizers for quantization of neural networks, 2018. 2

[31] David Neumann, Felix Sattler, Heiner Kirchhoffer, Simon Wiedemann, Karsten Müller, Heiko Schwarz, Thomas Wiegand, Detlev Marpe, and Wojciech Samek. Deepcabac: Plug & play compression of neural network weights and weight updates. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 21–25. IEEE, 2020. 2, 5, 6, 7

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 7

[33] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. 5

[34] Brandon Reagan, Udit Gupta, Bob Adolf, Michael Mitzenmacher, Alexander Rush, Gu-Yeon Wei, and David Brooks. Weightless: Lossy weight encoding for deep neural network compression. In *International Conference on Machine Learning*, pages 4324–4333. PMLR, 2018. 2

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. 7

[36] Michael Sabin and Robert Gray. Global convergence and empirical consistency of the generalized lloyd algorithm. *IEEE Transactions on information theory*, 32(2):148–155, 1986. 2

[37] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2018. 5, 6

[38] Jialiang Tang, Mingjin Liu, Ning Jiang, Huan Cai, Wenxin Yu, and Jinjia Zhou. Data-free network pruning for model compression. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2021. 2

[39] Jialiang Tang, Mingjin Liu, Ning Jiang, Huan Cai, Wenxin Yu, and Jinjia Zhou. Data-free network pruning for model compression. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2021. 2

[40] Enzo Tartaglione, Andrea Bragagnolo, Attilio Fiandrotti, and Marco Grangetto. Loss-based sensitivity regularization: towards deep sparse neural networks. *arXiv preprint arXiv:2011.09905*, 2020. 2

[41] Enzo Tartaglione, Stéphane Lathuilière, Attilio Fiandrotti, Marco Cagnazzo, and Marco Grangetto. Hemp: High-order entropy minimization for neural network compression. *Neurocomputing*, 461:244–253, 2021. 2

[42] Cheng-Hao Tu, Jia-Hong Lee, Yi-Ming Chan, and Chu-Song Chen. Pruning depthwise separable convolutions for mobilenet compression. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020. 2

[43] Frederick Tung and Greg Mori. Deep neural network compression by in-parallel pruning-quantization. *IEEE transactions on pattern analysis and machine intelligence*, 42(3):568–579, 2018. 2, 6, 7

[44] Stefan Uhlich, Lukas Mauch, Kazuki Yoshiyama, Fabien Cardinaux, Javier Alonso García, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Differentiable quantization of deep neural networks. *CoRR*, abs/1905.11452, 2019. 2

[45] Simon Wiedemann, Heiner Kirchhoffer, Stefan Matlage, Paul Haase, Arturo Marban, Talmaj Marinč, David Neumann, Tung Nguyen, Heiko Schwarz, Thomas Wiegand, et al. Deepcabac: A universal compression algorithm for deep neural networks. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):700–714, 2020. 2, 5

[46] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016. 2

[47] Sean Young, Zhe Wang, David Taubman, and Bernd Girod. Transform quantization for cnn compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2, 5, 6, 7

[48] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *International conference on machine learning*, pages 7543–7552. PMLR, 2019. 2, 6, 7

[49] Zihan Zhao, Yuncong Liu, Lu Chen, Qi Liu, Rao Ma, and Kai Yu. An investigation on different underlying quantization schemes for pre-trained language models. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 359–371. Springer, 2020. 2