mathematics

MDPI

Article

Neuron-by-Neuron Quantization for Efficient Low-Bit QNN Training

Artem Sher [1,2,*], Anton Trusov [1,2,3], Elena Limonova [2,3], Dmitry Nikolaev [2,4] and Vladimir V. Arlazarov [2,3]

[1] Phystech School of Applied Mathematics and Informatics, Moscow Institute of Physics and Technology, 141701 Moscow, Russia; trusov.av@smartengines.com
[2] Smart Engines Service LLC, 117312 Moscow, Russia; limonova@smartengines.com (E.L.)
[3] Department of Mathematical Software for Computer Science, Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, 119333 Moscow, Russia
[4] Vision Systems Laboratory, Institute for Information Transmission Problems (Kharkevich Institute) of Russian Academy of Sciences, 127051 Moscow, Russia
* Correspondence: sher.av@phystech.edu

Abstract: Quantized neural networks (QNNs) are widely used to achieve computationally efficient solutions to recognition problems. Overall, eight-bit QNNs have almost the same accuracy as full-precision networks, but working several times faster. However, the networks with lower quantization levels demonstrate inferior accuracy in comparison to their classical analogs. To solve this issue, a number of quantization-aware training (QAT) approaches were proposed. In this paper, we study QAT approaches for two- to eight-bit linear quantization schemes and propose a new combined QAT approach: neuron-by-neuron quantization with straight-through estimator (STE) gradient forwarding. It is suitable for quantizations with two- to eight-bit widths and eliminates significant accuracy drops during training, which results in better accuracy of the final QNN. We experimentally evaluate our approach on CIFAR-10 and ImageNet classification and show that it is comparable to other approaches for four to eight bits and outperforms some of them for two to three bits while being easier to implement. For example, the proposed approach to three-bit quantization of the CIFAR-10 dataset results in 73.2% accuracy, while baseline direct and layer-by-layer result in 71.4% and 67.2% accuracy, respectively. The results for two-bit quantization for ResNet18 on the ImageNet dataset are 63.69% for our approach and 61.55% for the direct baseline.

Keywords: quantized neural network; low-bit quantization; layer-by-layer, neuron-by-neuron training

MSC: 68T20

Citation: Sher, A.; Trusov, A.; Limonova, E.; Nikolaev, D.; Arlazarov, V.V. Neuron-by-Neuron Quantization for Efficient Low-Bit QNN Training. *Mathematics* **2023**, *11*, 2112. https://doi.org/10.3390/math11092112

Academic Editor: Georgios Tsekouras

Received: 16 March 2023
Revised: 17 April 2023
Accepted: 24 April 2023
Published: 29 April 2023

1. Introduction

Convolutional neural networks (CNNs) are widely used in modern computer vision problems such as pattern recognition [1], semantic segmentation [2], and many others [3–6]. To achieve outstanding results in terms of accuracy, modern deep CNNs require billions of floating-point parameters. This causes the large consumption of memory (for example, about 500MB for VGG-16 [7]) and computational resources. However, many devices are not efficient enough for the inference of large CNNs, for instance, Systems on a Chip (SoCs) and Internet of Things (IoT) systems with their Application-Specific Integrated Circuits (ASICs), Field-Programmable Gate Arrays (FPGAs), or low-power Central Processing Units (CPUs) [8]. Quantized neural networks (QNN) have been developed to resolve this issue [9,10].

QNN is a neural network in which floating-point values are replaced with fixed-point values. This transformation is called quantization. Either weights or weights and activations of the neural network (NN) can be quantized. Eight-bit quantization is the state-of-art standard that is implemented in popular machine learning frameworks such

as Pytorch [11] and Keras [12]. Such QNNs are used efficiently in CPUs, but can still be too resource-demanding for FPGAs and other low-power devices. Therefore, four-bit and lower quantization is of great interest. Unfortunately, there is a problem with QNNs: even though training on low-bit QNNs (such as binary, ternary, and others [13,14]) is possible, they suffer from significant accuracy loss in comparison to full-precision models [9].

There are two different approaches to QNN training: quantization-aware training (QAT) and post-training quantization (PTQ) [15]. Both of them include the pre-training of a full-precision model. PTQ quantizes the weights of the model and calibrates the parameters of the activations. In QAT, additional training of the NN takes place during the quantization process. Despite the higher computational cost, QAT results in higher-accuracy QNN rather than PTQ [15]. There is a problem with QAT approaches: the quantization operation is non-differentiable, so the backpropagation of the gradients through QNN is impossible. This problem can be solved with the use of the Straight-Through Estimator (STE) method [16], which approximates quantization with a differentiable function.

The simplest way to apply QAT is the "direct" method: it takes the pre-trained full-precision model and all training data, quantizes NN, and retrains the model to raise the accuracy using STE [16]. This is not an optimal solution, since, at low-bit width, the NN loses too much information after quantization and cannot recover from this loss during retraining [17]. The incremental quantization approach (INQ) [18] greedily quantizes groups of weights and retrains the model. For example, a widely used strategy is to quantize layer-by-layer [19,20]. This method does not require STE, and it is therefore more mathematically correct. However, it is greedy, and the resulting QNN may not be optimal.

There are even more variations in QAT methods. For example, in [21,22], the authors do not quantize the first and the last layers to low-bit, but they should be quantized for end-to-end integer inference [23]. In INQ [18], authors do not quantize activations, but in Hessian AWare Quantization (HAWQ) [21] and Quantization Interval Learning (QIL) [22], they do. Efficient batch normalization fusion for quantized architectures is proposed in [21]. Other authors use quantization to the powers of two [18,21] for better hardware implementation. Some of the methods operate only eight-bit quantization [23], which is rather well-investigated and widely used in industry. Quantizations lower than four bits are not as popular and require other QAT approaches. Non-uniform quantization [22,24] is also a field of research, but demonstrates hindered deployment on hardware and increased latency.

In this paper, we analyze different QAT methods and experimentally evaluate them for NNs with two- eight-bit weights and activations. Based on the results of this study, we analyze the efficiency of these methods and propose a novel combined method that allows for obtaining QNNs with higher accuracies and works for arbitrary bit width. We consider the "direct" training method and a layer-by-layer approach as baselines with toy LeNet models and regular ResNets on CIFAR-10 [25] and ImageNet [26], respectively. The paper is organized as follows. In Section 2, we consider related works, and Section 3 describes a linear quantization scheme for QNNs. Then, we give our particular research methodology and consider a combined approach in Section 4. In Section 5, we demonstrate experimental evaluations and show their results in Section 6. The results are discussed in Section 7. Finally, Section 8 concludes the paper. The list of all abbreviations used in this work can be found on page 16.

2. Related Works

There are several ways to improve the trade-off between the NN model's accuracy and efficiency, such as developing new NN architectures [27], pruning weights [28], knowledge distillation [29,30], and quantization.

The quantization uses low-bit precision values for weights or activations [10,23,24,31–34]. The benefit is the reduction in model size and the possibility of integer-only inference. Despite the increase in efficiency, quantization leads to accuracy degradation, especially for extremely low-bit weights and activations. To address this, some works propose non-uniform quantization [24], channel-wise quantization [35], or apply post-training

quantization [36]. In [21,37], the authors investigate mixed-precision quantization, in which some layers are kept at high precision and the rest are quantized for a lower bit-width.

The learnable quantization intervals proposed in [22] achieve state-of-the-art accuracy, but this algorithm is more complicated to implement and may cause some extra computational overhead. Moreover, the first and the last layers are not quantized, which is critical for integer-only inference in hardware.

Smooth quantization of the whole network with fine-tuning was introduced in INQ [18]. On each iteration, the authors proposed quantizing half of the remaining weights in each layer. Quantization is carried out by powers of two. One of the possible quantization streams is as follows: half of all layer's weights are quantized on the first iteration, a quarter of all weights on the second iteration, and the remaining quarter on the third.

HAWQ [21] proposes an integer-only quantization method with a uniform four-bit, eight-bit, or mixed four/eight-bit scheme. The choice of mixing for each layer is determined by solving an integer linear programming system with constraints on any combination of the total model size, bits operations per second, and latency. By applying dyadic arithmetic for FP32 scaling factors, this work also emphasizes that the inference does not resort to the 32-bit floating-point (FP32) casting. Thus, it allows for integer-only operations and bit-shifting for convolutions and skip connections of ResNets. In addition, the authors propose more efficient batch-normalization fusion than in [23]. The first and the last layers are always eight-bit quantized.

3. QNN Model with Linear Quantization

In this section, we describe QNN implementation based on the linear quantization scheme proposed in [23]. Some authors [22,24,38] have explored non-uniform quantization, which may achieve higher accuracy for a fixed bit-width. However, such approaches typically suffer from difficult deployment on hardware; therefore, in this work, we consider only uniform quantization.

3.1. Quantization Scheme

Linear quantization scheme maps the real value r to its quantized approximation q as follows:

$$q_{S,O}(r) = \left[\frac{r}{S} + O\right], \tag{1}$$

where S is a *scale*, O is an *offset*, and $[\cdot]$ stands for rounding to the nearest integer. The *scale* is a real-valued parameter and the *offset* is an integer parameter of quantization. This scheme is the base method used in all the further approaches.

3.2. Optimal Quantization Parameters Search

CNN quantization consists of quantizations of its weights and activations, which are arrays of real values. Let us consider several ways of obtaining the quantization parameters. In the gemmlowp library [39], the authors compute *scale* and *offset* as:

$$S = \frac{\max(0, r_{\max}) - \min(0, r_{\min})}{b - 1}, \tag{2}$$

$$O = -\frac{\min(0, r_{\min})}{S}, \tag{3}$$

where r_{max} is a maximum value in the array, r_{min} is a minimum in the array and b is the number of quantization levels (2^n for n-bit quantization).

In that approach, the real value 0 is exactly represented by O and is necessary for efficient computations and zero-padding. However, this method is hardly applicable for low-bit quantization. In that case, only a few quantized values would represent the major

part of array values (depending on its distribution). To overcome this issue, one may obtain quantization parameters by solving the minimum mean squared error (MMSE) problem:

$$S, O = \text{argmin}(\mathbb{E}[r - q_{S,O}(r)]^2) \tag{4}$$

This problem may be solved either analytically, if there are some assumptions about the r distribution (for CNN activations) [40], or directly, if the array is relatively small (for CNN weights) [41].

3.3. Batch Normalization Fusion

Batch normalization (BN) is often used in modern neural network architectures [42]. Given that the feature map x BN applies, the following operation is provided channel-wise:

$$\text{BN}(x) = \gamma \frac{x - \mu(x)}{\sqrt{\text{Var}(x) + \epsilon}} + \beta, \tag{5}$$

where $\mu(x)$ is the running mean, $\text{Var}(x)$ is the running variance, γ and β are trainable parameters, and ϵ is a small value that prevents division by zero. During inference, all of these parameters are fixed. Therefore, the BN operation can be fused into the previous convolutional layer, as it occurred [21]. We can combine BN and convolution into one operator as follows:

$$\text{ConvBN}(x) = \gamma \frac{W * x + b - \mu(x)}{\sqrt{\text{Var}(x) + \epsilon}} + \beta = W' * x + b', \tag{6}$$

where W and b are the weight and the bias parameters of the convolutional layer, respectively, W' and b' are the new weight and bias parameters, respectively, and $(*)$ denotes the convolution operator.

BN fusion (6) allows for integer-only computation in QNNs. To that end, fusion should be applied after several iterations of training. In this way, we preserve the benefits of BN, while gaining the performance of integer-only inference. In [21], the authors have proved the advantages of this approach.

4. QAT Research

In our research, we consistently combine and experimentally investigate methods proposed in various QAT works quantization weights and activations of NN for one to eight bits. Based on the results of this study, we draw conclusions about the efficiency of these methods and propose a combined method that allows QNN to be obtained with higher quality, working stably for arbitrary bits. Further, we use the following quantization model.

4.1. QNN Model

For the QNN model training, we propose our module, which is used instead of a convolutional layer (see Figure 1). The module consists of input quantization, weights quantization, convolution, and dequantization submodules.

The input quantization collects the histogram and then uses it to obtain optimum *scale_in* and *offset_in* and quantizes the input values by the corresponding scheme (1).

The weights quantization obtains optimum *scale_w* and *offset_w* and quantizes weights using the corresponding scheme.

The convolution calculates the convolution in quantized values.

The dequantization applies the reverse operation from (1) to obtain floating-point values, in particular, by subtracting the offset from the quantized value and multiplying the result by scale. This step may be unnecessary during the inference if the activation functions are piecewise linear [23], but in this paper, we are mainly concerned with

the training stage of the QNN, and we use the proposed module structure for better compatibility with the Pytorch pipeline.

The submodules change their behavior during different stages of the quantization process. In the beginning, quantization and dequantization submodules simply pass the floating-point values as they are. Then, the input quantization submodule constructs the histogram to obtain the optimum scale and offset. These parameters (*scale_in* and *offset_in*) are saved for further calculations (see Figure 1b). After that, we solve MMSE (4), resulting in *scale_w* and *offset_w* for weights and save these values as parameters as well.

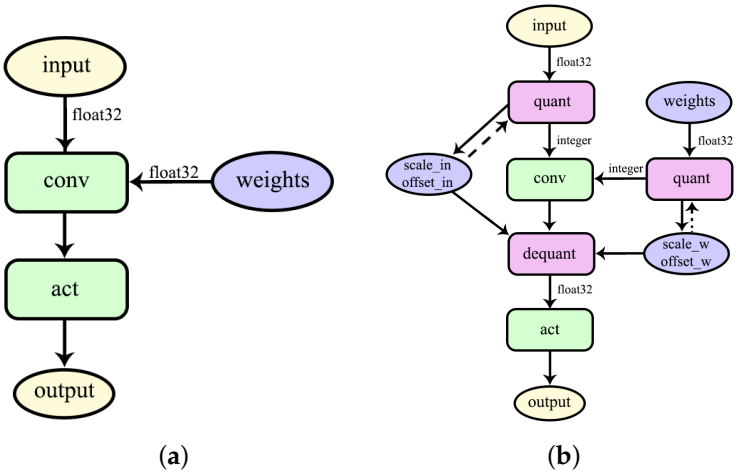


Figure 1. (**a**) Floating-point model. (**b**) Quantized model.

Thus, our module has two pairs of *scales* and *offsets* and floating-point weights as its parameters during training. During the inference, one can use integer weights directly without a corresponding quantization submodule. It is important to note that convolutional and activation layers can be connected without dequantization operation [23,43].

4.2. Optimal Quantization Parameters Solving

Having a pre-trained CNN and its training set, we obtain a quantization scheme for a convolutional layer as follows:

- For weights: we directly solve the MMSE problem (4), as did the authors of [41]. From the works [40,41], we can observe that MSE function $\mathbb{E}[r - q_{S,O}(r)]^2$ has one local minimum because the weight distribution in CNN is approximately normal. Therefore, we use a ternary search to solve Equation (4).
- For activations: one could solve the MMSE problem (4) for activations on the whole training set. However, the computational cost of this method is too high. Therefore, the authors of [41] used a training subset, but even in this way, the array size is too large. To overcome this issue, we approximate the activation distribution with a histogram and solve the MMSE problem (4) with a ternary search.

To use the ternary search, we consider each *offset* value from 0 to $b - 1$ and assume the following bounds δ_{min} and δ_{max} for the search:

$$\begin{aligned} \delta_{\max} &= \frac{\max(0, r_{\max}) - \min(0, r_{\min})}{b-1} \\ \delta_{\min} &= \frac{\max(0, \mu + \sigma) - \min(0, \mu - \sigma)}{b-1}, \end{aligned} \tag{7}$$

where μ is the mean of the values in the array and σ is their standard deviation. The numerator of $\delta_{\max}$ is a max difference between the border values of the distribution, that is, the length of the interval. Thus, $\delta_{\max}$ is the upper boundary for a ternary search, because if one increases this boundary, the quantization interval will include values that are not

contained in the distribution. Analogically, $\delta_{\min}$ is the lower boundary for ternary search, since its decreasing will lead to a state where a considerable number of values lay outside the quantization interval. For the ternary search, we multiply $\delta_{\max}$ by a factor of 1.1 and $\delta_{\min}$ by a factor of 0.9, and we thus increase the search range to avoid boundary effects. The ternary search provides us with an optimal *scale* and MSE for a given *offset*. Then, we choose the *scale* and *offset* with the minimum MSE.

4.3. Baseline Methods

The base idea for training QNN with QAT is a direct approach: quantize all weights on all input data layers and then retrain the network using the STE approximation for the backpropagation of gradients. A drawback of such an approach is the strong perturbation of the features caused by an insignificant change in the parameters of the primary layers of the network. This effect is amplified with an increase in a network's depth. It leads to high volatility during fine-tuning and lower quality than desirable, which is especially noticeable in the case of low-bit quantization.

To prevent the degradation of accuracy, one can use a layer-by-layer approach [19,20]. The procedure is described in Algorithm 1.

Algorithm 1 Layer-by-layer quantization

Input: Model to quantize M
Output: Quantized model M_Q

$M_Q = M$
for `each layer` $L_i \in M_Q$ **do**
    Quantize the input data of L_i
    Fine-tune the model
    Quantize weights $w_i \in L_i$
    Freeze w_i.
end for
Fine-tune model M_Q after last freezing (if there still are layers to retrain)

The layer-by-layer approach allows one to *smooth* the degradation of accuracy. Moreover, this procedure does not require STE approximation for fine-tuning.

Based on direct and layer-by-layer quantization, we improve these approaches. We consider some toy LeNet-like models on the CIFAR-10 dataset and regular ResNets on Imagenet with the BN fusion (see Section 3.3).

4.4. Proposed Combined Approach

The main idea for our research was that minimizing drastic accuracy drops during quantization allows better results to be achieved, because such drastic drops lead to strong perturbation of parameters and significant changes in the optimization problem, giving poor results. To overcome this drawback, we propose split quantization of the entire network in smaller steps to make less perturbation of parameters at once.

The first improvement is to use progressive quantization with smaller weight groups—a filter-by-filter approach. It consists of the quantization of the weights of individual groups with freezing such filters as shown in Algorithm 2. Following this approach allows us to achieve a smoother quantization for each layer.

Afterward, we complete research with the neuron-by-neuron method. With this approach, we quantize not only the part of the weights, but the input activations too. Moreover, the essence of this method is as follows: activations are quantized together with the corresponding weights, due to the preservation of the freezing mask between iterations, which is defined in Algorithm 3.

Algorithm 2 Filter-by-filter quantization

Input: Model to quantize M
Output: Quantized model M_Q

$M_Q = M$
for `each layer` $L_i \in M_Q$ **do**
  Quantize the input data of L_i
  Fine-tune the model
  while $w_i \in L_i$ is not freezed completely **do**
    Quantize some small part of w_i
    Freeze this part w_i
    Fine-tune the model
  end while
end for
Fine-tuning model M_Q after last freezing (if there still are layers to retrain)

Analogically for weights, we do this for activations, as the input of a layer connected to its corresponding weights, and this allows changing values only in some small part of the network.

Algorithm 3 Neuron-by-neuron quantization

Input: Model to quantize M
Output: Quantized model M_Q

$M_Q = M$
for `each layer` $L_i \in M_Q$ **do**
  if L_i is `first layer` **then**
    while input of L_i is not quantized completely **do**
      Quantize n channels of the input data of L_i and save the quantization mask
      Fine-tune the model
    end while
  else
    while $w_{i-1} \in L_{i-1}$ is not quantized completely **do**
      Quantize part of w_{i-1} accordingly with the quantization mask
      Create new mask for input data of L_i accordingly with channel of the L_{i-1}
      Fine-tune the model
    end while
    Freeze w_{i-1}
  end if
end for
Fine-tuning model M_Q after last freezing (if there still are layers to retrain)

The last idea is built on the possibility of fine-tuning already quantized values because it helps us to adapt old quantized parameters to new ones.

The final improvement consists of using the STE method [16] for gradient forwarding to the previous layer during neuron-by-neuron freezing.

5. Experimental Evaluation of QAT Approaches

5.1. Experimental Setup

5.1.1. CIFAR-10

We carried out a series of experiments on the CIFAR-10 [25] dataset with three CNN models: A, B, and C. The architecture of the models is described in Table 1. In Table 1, conv*n*-*m* denotes convolution with kernel size $n \times n$ and m filters, and st=x denotes stride x. In our convolutions, we do not use bias, padding, or dilation. After each convolution, we use the ReLU activation function.

Table 1. CNN models architecture.

A	B	C
4 layers with weights 31 k params	5 layers with weights 70 k params	6 layers with weights 235 k params
input 32 × 32 RGB image		
conv5–12 st = 2 conv3–36 st = 2 conv3–72 st = 2	conv5–12 st = 2 conv3–36 st = 2 conv3–60 st = 1 conv3–80 st = 1	conv5–20 st = 2 conv3–40 st = 1 conv3–60 st = 1 conv3–100 st = 2 conv3–160 st = 1
fully-connected-10		
soft-max		

The CIFAR-10 dataset consists of 60,000 32 × 32 color images in 10 classes, with 6000 images per class. There are 50,000 images in the training set and 10,000 images in the testing set.

We trained the model with floating-point weights (float pre-trained in the legend of Figures 2–6) for the first 1500 epochs with a batch size of 2000 and with a decreasing learning rate from 3×10^{-3} to 5×10^{-4}, and for the next 3000 epochs, it was trained with a batch size of 1000 and decreasing learning rate from 3×10^{-4} to 6×10^{-5}. For each architecture A–C, we trained three models (*baselines*) and averaged the results.

For all quantization approaches, the batch size and number of epochs during fine-tuning were set to 500. For each of the 3 *baselines* and each quantization approach, we trained 2 models for averaging, resulting in 6 models per QAT method.

In our experiments, the number of quantization bits was related to both the weights and activations. We used a 10,000-column histogram to approximate the input distribution (see Section 4.1).

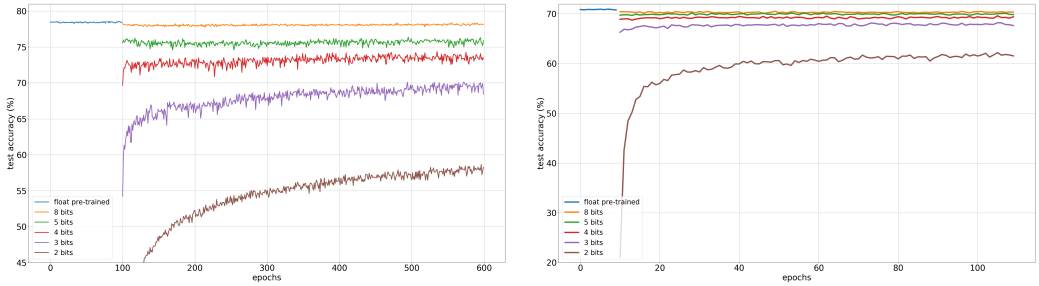
5.1.2. ImageNet

The ImageNet dataset consists of 14,197,122 color images in 1000 classes with a size of at least 256 × 256 for each one. There are 50,000 images in the validation set and 100,000 images in the testing set.

We trained original architecture ResNet baselines on the ImageNet dataset for 100 epochs with original hyper-parameters [44]; specifically: weight decay 0.0001, momentum 0.9, and a learning rate starting from 0.1 and decreasing for 0.1 every 25 epochs. The batch size was 256 during training baselines and all experiments. All experiments were trained for 100 epochs. The number of quantization bits was related to both weights and activations. We also used a 10,000-column histogram to approximate the input distribution (see Section 4.1).

5.2. Direct Training

The test accuracy during direct training for considered models is shown in Figure 2a,b. We can see that direct quantization changes the NN dramatically, which results in a decrease in recognition accuracy, especially with a low bit-width (for example, accuracy for 2-bit quantization is slightly below 60%, while full-precision accuracy is about 80% for CIFAR-10). Despite that, such a simple approach results in good enough accuracy for 5–8-bits quantization. Final test accuracy is shown in Section 6.

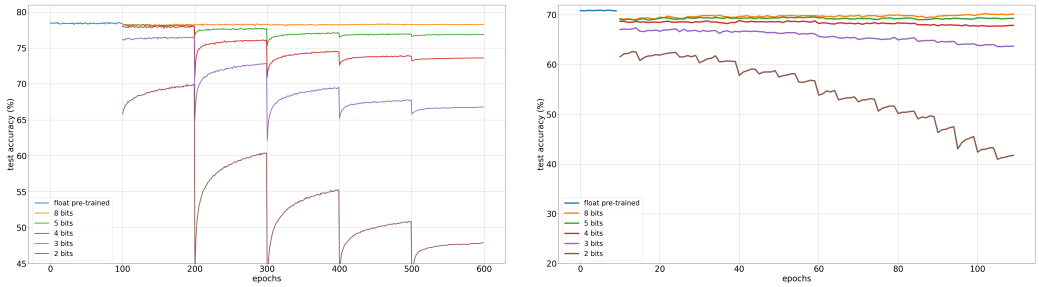


(**a**) (**b**)

Figure 2. Direct training, test accuracy: (**a**) CIFAR-10, Model B. (**b**) ImageNet, Resnet-18. The first, the last, and the skip connection layers use 8-bit (W8A8) quantization.

5.3. Layer-by-Layer Training

The test accuracy during layer-by-layer training for considered models is shown in Figure 3a,b. There are noticeable drops in the accuracy after the quantizations of each layer. The experiments presented in Section 6 show that this approach gives a slightly higher recognition accuracy of the model with 4–8 bits compared to direct training, while the accuracy for 3- and 2-bit quantization is significantly lower. This is reasonable because the layer-by-layer method does not use STE approximation during training. The final results are presented in Section 6.



(**a**) (**b**)

Figure 3. Layered training, test accuracy: (**a**) CIFAR-10, Model B. (**b**) ImageNet, Resnet-18. The first, the last, and skip connection layers are quantized the same way as the other layers.

5.4. Filter-by-Filter Training

5.4.1. Freezing by a Quarter of the Filters

The first improvement is that we do not quantize all weights in the layer at once. We divide the process into four iterations. In this way, each layer is "frozen" in quarters, and between those four "freezings", the model is retrained. We consider six different strategies to choose the order in which filters are "frozen":

Seq: filters are chosen sequentially: in the order they are presented in the NN;

Rand: filters are chosen randomly in the same amount;

Var: filters are chosen according to the maximum standard deviation relative to the whole layer;

Nvar: similar to the previous strategy with minimum standard deviation;

MSE: filters are chosen according to the maximum MSE between quantized and floating-point values;

NMSE: similar to the previous with minimum MSE.

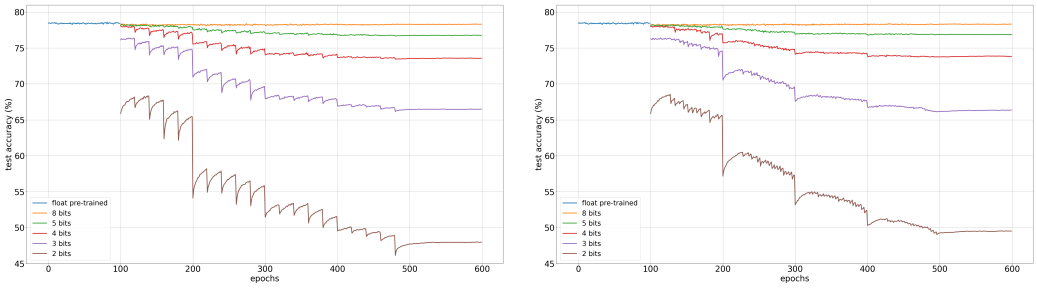
In Table 2, we present a comparison of different strategies of choosing the order in which the filters are "frozen" in a filter-by-filter approach. From the table, we can clearly

see that there is no significant difference between these strategies. Therefore, one can choose any. Note that the results presented in Table 2 differs from those in Section 6, as the measurements in Section 6 for the CIFAR dataset were averaged over six runs to improve the stability for the final strategy comparison.

Test accuracy during training is shown in Figure 4a. We can see that quantization of the weights inside one layer results in *smoother* training. However, there are still significant drops in quantization input stages (epochs 100, 200, 300, and 400).

Table 2. Comparison of strategies with a filter-by-filter approach for CIFAR-10 in terms of test accuracy, %. W*X* stands for *X* bits weight quantization and A*X* stands for *X* bits activation quantization.

Models	Float Train	Bits	Seq	Rand	Var	Nvar	MSE	NMSE
A	71.70	W8A8	71.36	71.39	71.21	71.33	71.23	**71.40**
		W5A5	**69.55**	69.37	69.36	69.52	69.02	68.90
		W4A4	65.66	**66.31**	65.11	66.24	65.88	66.03
		W3A3	58.84	59.33	**59.67**	59.65	58.83	58.54
		W2A2	44.26	44.23	43.24	42.80	**44.29**	42.29
B	78.59	W8A8	78.60	78.56	**78.69**	78.42	78.49	78.64
		W5A5	77.08	77.40	77.08	**77.54**	76.77	77.27
		W4A4	73.98	**74.34**	73.27	73.68	74.03	73.77
		W3A3	67.02	**67.70**	65.61	67.55	66.64	66.92
		W2A2	49.72	49.97	49.57	**51.04**	49.14	49.92
C	81.62	W8A8	**81.73**	81.37	81.54	81.49	81.53	81.54
		W5A5	79.88	79.82	**80.33**	79.53	80.17	80.13
		W4A4	77.29	77.23	77.38	**77.64**	77.64	77.64
		W3A3	70.62	70.63	70.37	70.69	**71.16**	70.48
		W2A2	48.38	50.65	49.78	**51.26**	50.34	51.00



(**a**) (**b**)

Figure 4. Filter-by-filter training CIFAR-10, Model B, test accuracy: (**a**) Quarters. (**b**) Smooth.

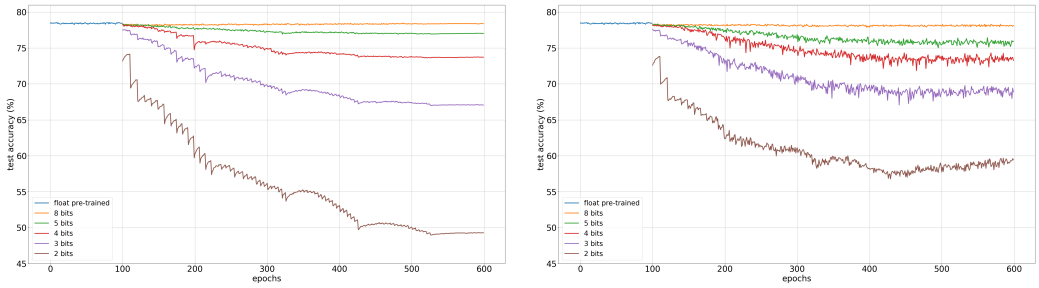
5.4.2. Smooth Filter Freezing

This method improves on the previous one by dividing filters into smaller sets. The number of "frozen" filters on each iteration of the "freezing" layer is small, so this process is smoother, and the resulting accuracy is slightly higher, which is demonstrated in Table 3 in Section 6. The strategies for choosing the order in which filters are "frozen" remain the same. While the quantizing single layer became *smoother*, drops in the input quantization remain big (see Figure 4).

5.5. Neuron-by-Neuron Training

In the previous subsection, we considered the gradual quantization of weights, but the input was quantized as in the layer-by-layer approach. This resulted in noticeable drops in accuracy when the input was quantized. Therefore, the next improvement we made was "freezing" both filters of one convolutional layer and a corresponding input of the next layer. This means that we froze one neuron (linear operation followed by a

non-linear activation) at a time. The test accuracy during training for such an approach is shown in Figure 5a. This demonstrates that the neuron-by-neuron training had slightly better accuracy than the previous one; however, it is still comparable with the layer-by-layer approach. However, it removes the subsidence of accuracy after input quantization, compared with a filter-by-filter method. The final averaged accuracy is shown in Section 6.



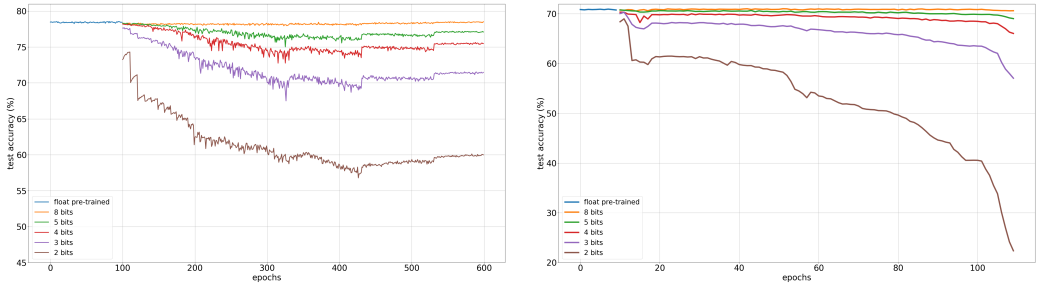
(**a**) (**b**)

Figure 5. Neuron-by-neuron training CIFAR-10, Model B, test accuracy: (**a**) without STE; (**b**) with gradient forwarding on all networks.

5.6. Neuron-by-Neuron with Gradient Forwarding

To improve the results of the neuron-by-neuron approach, we proposed using gradient forwarding. In the first version, we forwarded the gradient to all the quantized layers using STE the same way as the direct training, but quantized the weights and activations consecutively and smoothly as in the neuron-by-neuron approach. As expected, that led to a very "noisy" training process (see Figure 5b), since a small change at the beginning of the network seriously affected the rest of the network. However, the accuracy for two- and three-bit quantization dramatically increased in comparison to training without STE.

We considered the gradient forwarding only on the previous layer to the quantized one in the following version. As a layer-by-layer approach, this method is greedy and has low "noise" (see Figure 6a), but allows the network to adapt the weights of the layer depending on the quantization of the next one. This method shows notable results in terms of accuracy, as presented in in Section 6.



(**a**) (**b**)

Figure 6. Neuron-by-neuron training with gradient forwarding on the previous layer only, test accuracy: (**a**) CIFAR-10, Model B. (**b**) ImageNet, Resnet-18. The first, the last, and the skip connection layers are quantized the same way as the other layers.

6. Results

Finally, to provide a reliable comparison for the considered approaches, we present an averaged accuracy on the test set. The CIFAR-10 results are shown in Table 3. We use the following notation for the considered QAT methods: LbL—layer-by-layer; FbF—filter-by-filter with four iterations of quantization per layer; FbF-s—filter-by-filter with smooth quantization (many iterations) of each layer; NbN—neuron-by-neuron; NbN ste—neuron-by-neuron without freezing the quantized layers (they are trained using the STE method);

NbN ste-1l—neuron-by-neuron with STE gradient approximation applied only to the layer previous to the currently quantized one during training.

Table 3. Comparison between different approaches in terms of averaged (over six models) test accuracy on CIFAR-10, %. The highest mean accuracy colored in lime. Accuracies close to the highest, taking into account the measurement mean error, colored in green. W*X* stands for *X* bits weight quantization and A*X* stands for *X* bits activation quantization.

Models	Float Train	Bits	Direct	LbL	FbF (rand)	FbF (var)	FbF (nvar)	FbF-s (rand)	FbF-s (var)	FbF-s (nvar)	NbN	NbN ste	NbN ste-1l	Mean Error
A	72.3	W8A8	71.3	71.7	71.7	71.7	71.6	71.7	71.7	71.6	71.7	71.4	71.8	±0.3
		W5A5	68.9	69.7	69.7	69.8	69.7	69.7	69.7	69.5	69.6	69.5	70.4	±0.4
		W4A4	66.5	66.2	66.0	65.5	66.3	66.0	65.6	66.2	66.5	66.8	68.2	±0.4
		W3A3	62.0	58.4	58.8	58.9	59.1	59.1	58.8	59.0	59.3	61.8	63.9	±0.4
		W2A2	46.8	43.2	41.8	42.6	41.6	42.3	42.4	41.6	43.0	52.0	52.4	±0.8
B	78.7	W8A8	78.2	78.3	78.3	78.4	78.3	78.3	78.4	78.4	78.4	78.1	78.4	±0.2
		W5A5	75.9	76.9	76.8	76.6	76.9	76.9	76.8	77.1	77.0	76.5	77.1	±0.2
		W4A4	73.5	73.6	73.6	73.7	73.2	73.8	73.7	73.2	73.7	74.5	75.6	±0.2
		W3A3	68.5	66.8	66.5	66.3	67.2	66.4	66.5	67.0	67.1	70.1	71.3	±0.4
		W2A2	58.2	47.9	48.0	48.6	48.0	49.5	48.8	48.8	49.3	59.6	60.1	±0.6
C	81.2	W8A8	80.2	80.4	80.4	80.5	80.1	80.5	80.5	80.5	80.4	80.3	80.5	±0.5
		W5A5	77.9	78.6	78.6	78.8	78.2	78.6	78.7	78.7	79.0	78.3	78.8	±0.6
		W4A4	75.4	75.5	75.4	75.3	74.9	75.5	75.3	75.6	75.7	75.5	77.2	±0.7
		W3A3	71.4	67.2	67.1	66.6	67.5	66.9	66.5	68.0	68.3	70.8	73.2	±1.1
		W2A2	57.4	46.9	47.2	46.4	47.5	47.7	47.5	47.7	48.2	59.6	60.8	±1.1

As we can see, NbN ste-1l shows the best results on all models and all bit widths. The filter-by-filter approach turns out to be similar in quality to the layer-by-layer approach. We assume that it is caused by dramatic changes in NN after quantization of the whole input of a convolutional layer. We also notice that propagation of the gradient using STE greatly improves the results of the neuron-by-neuron method. Without STE, this method demonstrates lower accuracy than the direct method on two- and three-bit QNNs. Restrictions on the number of layers to which the gradient is propagated further improves the neuron-by-neuron approach. This is likely connected to the decrease in the "noise" during training (see Figures 5b and 6a).

In Figures 4a and 6a , we noticed that during the training process, some approaches to QNNs did not reach the limit of potential accuracy before the next quantization occurs. That was caused by our restriction on the number of training epochs, which was kept the same in all approaches. Unlimited fine-tuning till loss convergence may be a subject of future research.

Next, we will consider ImageNet recognition, since the results on CIFAR-10 are not representative. Firstly, we study the influence on accuracy freezing delay, a count of the layers that are not yet frozen and can be retrained through STE during subsequent quantization of layers. This number varies from 0, in the case we immediately freeze the layer after its quantization and quantized values are not updated with STE, to the number of layers in NN, which is similar to direct quantization. As shown in Table 4, it is important to notice that the absence of a delay negatively affects the final accuracy. In addition, we observed more noisy graph training for the delay in whole-NN layers. As a result, we can conclude that a delay of one to two layers is optimal. It allows us to reduce computational overhead and volatility of STE without a noticable loss in accuracy.

We also note that quantization of the first and the last layers greatly decreases the quality of the NN, as can be seen from Figure 6b. This confirms previous works' [10,45] conclusions about its significant impact on accuracy.

Table 4. Influence of the delay in freezing layers on Resnet18 on ImageNet dataset, %. WX stands for X bits weight quantization and AX stands for X bits activation quantization.

Precision	Direct	No Delay	Delay 1	Delay 2	Delay 15
W4A4	69.42	68.80	69.18	69.22	68.82
W3A3	67.66	67.00	67.18	67.63	67.58
W2A2	61.55	61.73	63.00	63.69	63.07

As a result, our final experiments on the ImageNet dataset were conducted with eight-bit quantization of the first convolutional, the last fully-connected layer, and the skip connection layers to keep the entire integer reference. The results on the ImageNet dataset with a comparison of the other methods are shown in Tables 5 and 6. We can see that the baseline direct method is adequate up to four-bit quantization and has a minimal loss of quality for three bits. Given that result, there is no need for complex methods for four-bit quantization without quantizing (or eight-bit quantizing) the first and last layers. Our NbN approach is better than the direct approach for two bits and slightly loses to other works for two- and three-bit quantization, but it is easy to understand and implement. It is also important to notice that two-bit NN with the direct method extremely suffers from quantization (about 21% accuracy without fine-tuning, while the full-precision model has 71%). Consequently, it still does not converge after 100 epochs (see Figure 2b); meanwhile, our NbN can converge faster or can achieve better accuracy at the same time.

We present the dependence of the accuracy on the used bit-width in Figure 7.

Table 5. Comparison between different approaches in terms of test Top-1 accuracy on Resnet18 on ImageNet dataset, %. W*X* stands for *X* bits weight quantization, and A*X* stands for *X* bits activation quantization.

Precision	Direct	NbN ste-2l	INQ [18] (A32 All)	PACT [46]	LQ-Nets [24]	HAWQ-V3 [21]
FP32	70.78	70.78	68.27	70.20	70.30	71.47
W8A8	70.46	70.11	-	-	-	71.56
W5A5	69.86	69.50	68.98	69.80	-	-
W4A4	69.42	69.22	68.89	69.20	69.30	68.45
W3A3	67.66	67.63	68.08	68.10	68.20	-
W2A2	61.55	63.69	66.02	64.40	64.90	-

Table 6. Comparison between different approaches in terms of test Top-5 accuracy on Resnet18 on ImageNet dataset, %. W*X* stands for *X* bits weight quantization and A*X* stands for *X* bits activation quantization.

Precision	Direct	NbN ste-2l	INQ [18] (A32 All)	PACT [46]	LQ-Nets [24]
FP32	89.54	89.54	88.69	89.60	89.50
W8A8	89.31	89.02	-	-	-
W5A5	88.76	88.42	89.10	89.30	-
W4A4	88.54	88.32	89.01	89.00	88.80
W3A3	87.70	87.74	88.36	88.20	87.90
W2A2	83.51	85.03	87.13	85.60	85.90

Overall,

- We confirmed that the quantization of the first and the last layers dramatically drops the accuracy and should be avoided;
- Using progressive quantization (FbF or NbN) methods, there is no significant difference between freezing strategies, and one can therefore choose neurons to freeze randomly;

- For eight-bit quantization, there is no need for complex approaches—direct linear quantization is good enough—and there is no need for extra fine-tuning;
- For quantization with four bits and higher, direct linear quantization achieves state-of-the-art results with little fine-tuning;
- For three-bit quantization, NbN ste and direct approaches show comparable accuracy;
- For two-bit quantization, NbN ste is better than the baseline and comparable to other approaches, but easier to implement.

Our research will be potentially helpful for practitioners in ultra-low-bit quantization due to its simplicity in deployment and better accuracy than the baseline approaches.

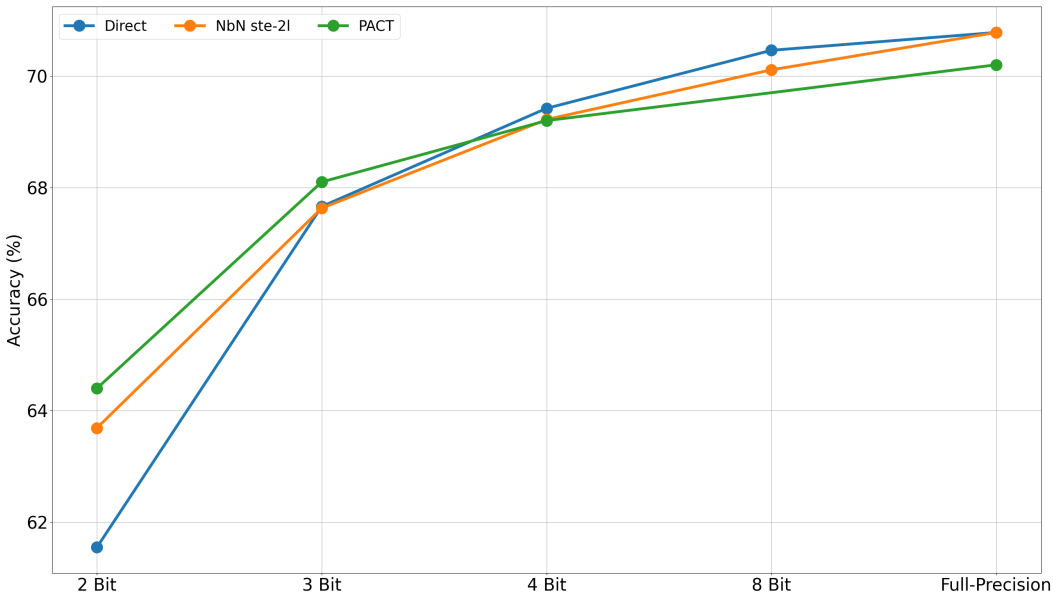


Figure 7. Trade-off between accuracy and bit-width on ImageNet, test top-1 accuracy.

7. Discussion

The efficient practical application of QNNs on mobile and embedded devices requires integer-only inference. To achieve it, there are several factors to consider: (1) The first and the last layers should be kept in eight bits to prevent accuracy degradation; (2) It can be useful to keep skip connections in eight bits also; (3) The batch-norm layer should be quantized through fusion with convolution, as was mentioned in [21].

We demonstrate that for eight-bit quantization, it is sufficient to apply a linear quantization scheme directly even without retraining in the considered tasks. This quantization is indeed used in many areas and is applied in practice without problems. For four-bit quantization, fine-tuning of the model takes is required. However, training quickly reaches convergence and achieves results comparable with other more complex methods [21,46]. Consequently, quantization approaches for two- and three-bit schemes are more interesting for research.

Intuitively, it seems that splitting the weights into small subgroups during progressive quantization should give a better result than the direct method. We experimentally confirm this statement, provided there are enough iterations for fine-tuning with the considered weight groups and the optimal selection of hyper-parameters. Despite that improvement, accuracy increases less than can be expected and still loses to a full precision model. The choice of optimal hyper-parameters can also be tricky. We conducted a study of the number of layers to freeze during fine-tuning and showed that a delay of one to two layers is optimal.

Note that weight freezing does not affect convergence, on the condition that it lasts to the end of the fine-tuning, as freezing itself does not change the values of the parameters. In addition, our method is a QAT one. QAT approaches successfully converge, as

demonstrated in [9]. Additionally, it is important to note that we do not quantize gradients during training.

Limitations of the study. The proposed neuron-by-neuron quantization NbN belongs to QAT methods, so it is necessary to have access to the source dataset for fine-tuning. This means that it is not applicable when only a trained neural network is available. We also did not consider non-linear quantization schemes, because of their algorithmic and possibly computational complexity, and the obtained results are not applicable in those cases.

8. Conclusions

In this paper, we propose several new approaches for training quantized CNNs. We consider a well-known layer-by-layer approach and point out two major drawbacks: (1) The significant perturbation of parameters during layers' quantization, which leads to a drop in accuracy, and (2) the absence of quantized layer fine-tuning. On the contrary, direct training using STE suffers from high volatility caused by updating discrete parameters in deep NNs.

We propose a method that balances the above methods and avoids their disadvantages. It allows for smooth network quantization using the neuron-by-neuron approach and propagates gradients via several (but not all) quantized layers. Thus, quantized parameters are fine-tuned without causing high volatility.

We experimentally confirm that our neuron-by-neuron method outperforms the direct approach using Resnet18 for two-bit quantization on the ImageNet dataset and is superior to the direct and layer-by-layer approaches using LeNet-like networks trained for image recognition on the CIFAR-10.

Author Contributions: Conceptualization, E.L., A.T. and A.S.; methodology, A.T. and A.S.; validations, A.S.; supervision, E.L., D.N. and V.V.A.; writing—original draft, A.S.; writing—review and editing, E.L. and A.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Grant from the Ministry of Science and Higher Education of the Russian Federation (internal number 00600/2020/51896) under Grant 075-15-2022-319.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

QNN	Quantized neural networks
QAT	Quantization-aware training
STE	Straight through estimator
NbN	Neuron-by-neuron
CNN	Convolutional neural networks
VGG-16	Visual Geometry Group Very Deep Convolutional Networks
SoC	System on a chip
IoT	Internet of things
ASIC	Application-specific integrated circuit
FPGA	Field-programmable gate array
CPU	Central processing unit
PTQ	Post-training quantization
NN	Neural network

INQ	Incremental quantization approach
HAWQ	Hessian AWare Quantization
QIL	Quantization Interval Learning
FP32	32-bit floating-point
MMSE	Minimum mean squared error
BN	Batch normalization

References

1. Janiszewski, I.M.; Arlazarov, V.V.; Slugin, D.G. Achieving Statistical Dependence of the CNN Response on the Input Data Distortion for OCR Problem. *ITiVS* **2019**, *44*, 94–101. [CrossRef]
2. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
3. Agafonova, J.D.; Gaidel, A.V.; Zelter, P.M.; Kapishnikov, A.V. Efficiency of machine learning algorithms and convolutional neural network for detection of pathological changes in MR images of the brain. *Comput. Opt.* **2020**, *44*, 266–273. [CrossRef]
4. Sheshkus, A.; Limonova, E.; Nikolaev, D.; Krivtsov, V. Combining Convolutional Neural Networks and Hough Transform for Classification of Images Containing Lines. In Proceedings of the ICMV 2016, Nice, France, 18–20 November 2016; Volume 10341, pp. 1–5. [CrossRef]
5. Chernyshova, Y.S.; Chirvonaya, A.N.; Sheshkus, A.V. Localization of characters horizontal bounds in text line images with fully convolutional network. In Proceedings of the ICMV 2019, Amsterdam, The Netherlands, 16–18 November 2019 ; Volume 11433, pp. 1–8. [CrossRef]
6. Jin, X.B.; Yang, N.X.; Wang, X.Y.; Bai, Y.T.; Su, T.L.; Kong, J.L. Deep Hybrid Model Based on EMD with Classification by Frequency Characteristics for Long-Term Air Quality Prediction. *Mathematics* **2020**, *8*, 214. [CrossRef]
7. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
8. Motamedi, M.; Fong, D.; Ghiasi, S. Machine intelligence on resource-constrained IoT devices: The case of thread granularity optimization for CNN inference. *ACM Trans. Embed. Comput. Syst. (TECS)* **2017**, *16*, 1–19. [CrossRef]
9. Gholami, A.; Kim, S.; Dong, Z.; Yao, Z.; Mahoney, M.W.; Keutzer, K. A survey of quantization methods for efficient neural network inference. *arXiv* **2021**, arXiv:2103.13630.
10. Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **2017**, *18*, 6869–6898.
11. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: New York, NY, USA, 2019; pp. 8024–8035.
12. Chollet, F. Keras. 2015. Available online: https://keras.io (accessed on 27 April 2022)
13. Qin, H.; Gong, R.; Liu, X.; Bai, X.; Song, J.; Sebe, N. Binary neural networks: A survey. *Pattern Recognit.* **2020**, *105*, 107281. [CrossRef]
14. Alemdar, H.; Leroy, V.; Prost-Boucle, A.; Pétrot, F. Ternary neural networks for resource-efficient AI applications. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, Alaska, 14–19 May 2017; pp. 2547–2554.
15. Nagel, M.; Fournarakis, M.; Amjad, R.A.; Bondarenko, Y.; van Baalen, M.; Blankevoort, T. A white paper on neural network quantization. *arXiv* **2021**, arXiv:2106.08295.
16. Bengio, Y.; Léonard, N.; Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv* **2013**, arXiv:1308.3432.
17. Guo, Y. A survey on methods and theories of quantized neural networks. *arXiv* **2018**, arXiv:1808.04752.
18. Zhou, A.; Yao, A.; Guo, Y.; Xu, L.; Chen, Y. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv* **2017**, arXiv:1702.03044.
19. Ilin, D.; Limonova, E.; Arlazarov, V.; Nikolaev, D. Fast Integer Approximations In Convolutional Neural Networks Using Layer-By-Layer Training. In Proceedings of the ICMV 2016, Nice, France, 18–20 November 2016; Volume 10341, pp. 1–5. [CrossRef]
20. Shin, S.; Boo, Y.; Sung, W. Fixed-point optimization of deep neural networks with adaptive step size retraining. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 1203–1207.
21. Yao, Z.; Dong, Z.; Zheng, Z.; Gholami, A.; Yu, J.; Tan, E.; Wang, L.; Huang, Q.; Wang, Y.; Mahoney, M.; et al. Hawq-v3: Dyadic neural network quantization. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 6–12 June 2021; pp. 11875–11886.
22. Jung, S.; Son, C.; Lee, S.; Son, J.; Han, J.J.; Kwak, Y.; Hwang, S.J.; Choi, C. Learning to quantize deep networks by optimizing quantization intervals with task loss. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4350–4359.

23. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2704–2713.
24. Zhang, D.; Yang, J.; Ye, D.; Hua, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 365–382.
25. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images 2009. Available online: http://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 27 April 2022)
26. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
27. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
28. Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; Kautz, J. Pruning convolutional neural networks for resource efficient inference. *arXiv* **2016**, arXiv:1611.06440.
29. Yin, H.; Molchanov, P.; Alvarez, J.M.; Li, Z.; Mallya, A.; Hoiem, D.; Jha, N.K.; Kautz, J. Dreaming to distill: Data-free knowledge transfer via deepinversion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 8715–8724.
30. Polino, A.; Pascanu, R.; Alistarh, D. Model compression via distillation and quantization. *arXiv* **2018**, arXiv:1802.05668.
31. Asanović, K.; Morgan, N. Experimental determination of precision requirements for back-propagation training of artificial neural networks. In Proceedings of the Second International Conference Microelectronics for Neural Networks, Citeseer, Munich, Germany, 1991; pp. 9–16.
32. Rastegari, M.; Ordonez, V.; Redmon, J.; Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016 ; Springer: Berlin/Heidelberg, Germany, 2016; pp. 525–542.
33. Dong, Z.; Yao, Z.; Gholami, A.; Mahoney, M.W.; Keutzer, K. Hawq: Hessian aware quantization of neural networks with mixed-precision. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 293–302.
34. Novac, P.E.; Boukli Hacene, G.; Pegatoquet, A.; Miramond, B.; Gripon, V. Quantization and Deployment of Deep Neural Networks on Microcontrollers. *Sensors* **2021**, *21*, 2984. [CrossRef] [PubMed]
35. Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv* **2018**, arXiv:1806.08342.
36. Hubara, I.; Nahshan, Y.; Hanani, Y.; Banner, R.; Soudry, D. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv* **2020**, arXiv:2006.10518.
37. Shen, S.; Dong, Z.; Ye, J.; Ma, L.; Yao, Z.; Gholami, A.; Mahoney, M.W.; Keutzer, K. Q-bert: Hessian based ultra low precision quantization of bert. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 8815–8821.
38. Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; Han, S. Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 8612–8620.
39. Jacob, B.; Warden, P. Gemmlowp: A Small Self-Contained Low-Precision GEMM Library, 2017. Available online: https://github.com/google/gemmlowp (accessed on 27 April 2022)
40. Banner, R.; Nahshan, Y.; Soudry, D. Post Training 4-bit Quantization of Convolutional Networks for Rapid-Deployment. *Adv. Neural Inf. Process. Syst.* **2019**. Available online: https://papers.nips.cc/paper_files/paper/2019 (accessed on 17 April 2023).
41. Choukroun, Y.; Kravchik, E.; Yang, F.; Kisilev, P. Low-bit quantization of neural networks for efficient inference. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Republic of Korea, 27–28 October 2019; pp. 3009–3018.
42. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 7–9 July 2015; pp. 448–456.
43. Trusov, A.V.; Limonova, E.E.; Slugin, D.G.; Nikolaev, D.P.; Arlazarov, V.V. Fast Implementation of 4-bit Convolutional Neural Networks for Mobile Devices. In Proceedings of the ICPR 2020, Milan, Italy, 10–15 January 2021; pp. 9897–9903. [CrossRef]
44. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
45. Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv* **2016**, arXiv:1606.06160.
46. Choi, J.; Wang, Z.; Venkataramani, S.; Chuang, P.I.J.; Srinivasan, V.; Gopalakrishnan, K. Pact: Parameterized clipping activation for quantized neural networks. *arXiv* **2018**, arXiv:1805.06085.