Channel Balancing for Accurate Quantization of Winograd Convolutions

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

Vladimir Kryzhanovskiy
Huawei Noah's Ark Lab
kryzhanovskiy.vladimir@huawei.com

Abstract

It is well known that Winograd convolution algorithms speed up the widely used small-size convolutions. However, the problem of quantization of Winograd convolutions is challenging – while quantization of slower Winograd algorithms does not cause problems, quantization of faster Winograd algorithms often leads to a significant drop in the quality of models. We introduce a novel class of Winograd algorithms that balances the filter and input channels in the Winograd domain. Unlike traditional Winograd convolutions, the proposed convolution balances the ranges of input channels on the forward pass by scaling the input tensor using special balancing coefficients (the filter channels are balanced offline). As a result of balancing, the inputs and filters of the Winograd convolution are much easier to quantize. Thus, the proposed technique allows us to obtain models with quantized Winograd convolutions, the quality of which is significantly higher than the quality of models with traditional quantized Winograd convolutions. Moreover, we propose a special direct algorithm for calculating the balancing coefficients, which does not require additional model training. This algorithm makes it easy to obtain the post-training quantized balanced Winograd convolutions – one should just feed a few data samples to the model without training to calibrate special parameters. In addition, it is possible to initialize the balancing coefficients using this algorithm and further train them as trainable variables during Winograd quantization-aware training for greater quality improvement.

1. Introduction

Lightweight architectural designs of Convolutional Neural Networks (CNNs) together with quantization have paved the way for the deployment of demanding computer vision applications on mobile devices. Parallel to this, alternative formulations of the convolution operation, such as FFT or Winograd, have been adapted for the use in CNNs allowing further speedups. Winograd convolutions [28] (also known as Toom-Cook algorithm [7, 27]) are the fastest known algorithm for spatially small convolutions, but their use in a quantized context is often accompanied by a quality drop due to significant numeric errors. The significant speed and power consumption advantage of quantized Winograd convolutions motivates the research in this direction, see Tab. 1. As we can see, the quantized Winograd is $2.3\times$ $(50/21.5)$ faster than the float16 direct convolution. In practice, speedups up to $4\times$ (ARM CPUs) can be achieved [20].

Implementation	FP16, ms	INT8, ms
Direct conv	50	30
Winograd conv	25	21.5

Table 1. Inference time for ResNet-18 on $224 \times 224 \times 3$ input, BOLT inference framework [11], ARM CPU: Kirin 990.

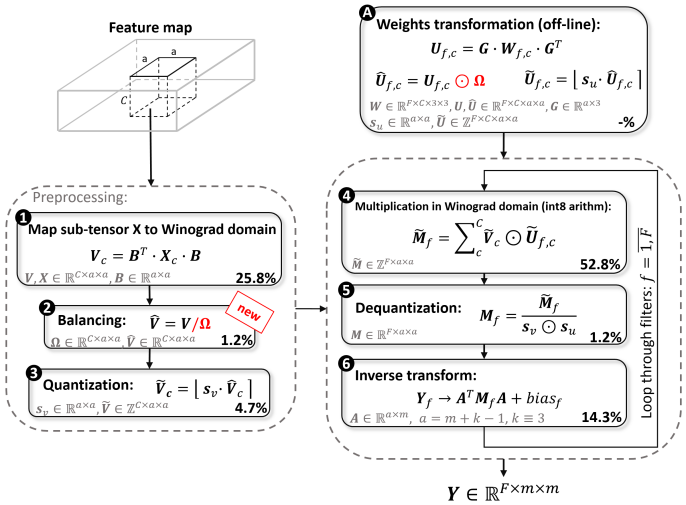
Winograd convolutions Lavin et al. [15] generalized the Winograd's minimal filtering algorithm [28] for filters with $k \times k$ kernels that are widely used in modern CNNs. This generalization is called Winograd 2D-convolution. Its main idea is based on the fact that the minimal filtering algorithm for computation of m outputs with k-tap FIR filter requires

$$\mu(F(m,k)) = m + k - 1 \tag{1}$$

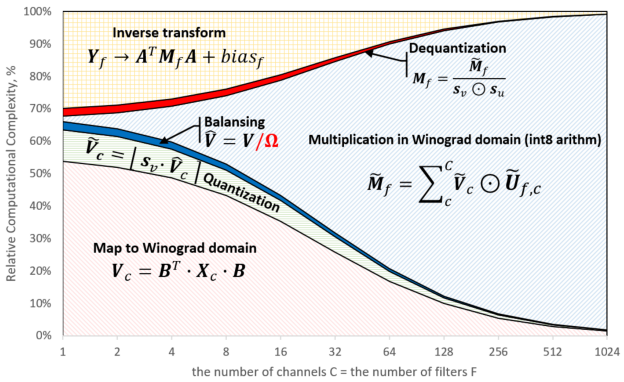
multiplications, instead of mk multiplications required for the direct 1D-convolution. Further in the paper we will use the short form $F(m,k)$ that denotes $F(m \times m, k \times k)$ – a 2D-convolution with kernel size $k \times k$ producing $m \times m$ outputs. The parameter m is called the *tile size*. In this paper, we consider the most popular case of Winograd convolutions – with 3×3 kernel and stride 1. Traditionally, the Winograd convolution algorithm is written in the following matrix form:

$$\mathbf{Y} = \mathbf{A}^T\Big((\mathbf{B}^T\mathbf{X}\mathbf{B}) \odot (\mathbf{G}\mathbf{W}\mathbf{G}^T)\Big)\mathbf{A}, \tag{2}$$

where the tensor indices are omitted for simplicity. Various versions of Winograd transformation matrices $\mathbf{A}, \mathbf{B}, \mathbf{G}$ are used for different tile sizes m. Our proposed method can be generalized to any m (including complex Winograd [22]). In this paper, we use traditional Winograd transformation matrices from [15] for F$(4,3)$ and F$(6,3)$ Winograd algorithms. The definitions of these matrices are presented in Appendix A.



(a) Balanced Quantized Winograd algorithm. The relative complexity is noted in the bottom right corners. We add a new cheap operation to the prepossessing stage – the channel balancing of Winograd-domain inputs and filters (highlighted in red). The weights transformation is done offline, therefore its complexity is not taken into account. $\lfloor \cdot \rceil$ is a quantization operator defined by Eq. (3).



(b) The dependence of the proportions of the algorithm stages complexity on the number of filters and channels ($C = F$). Multiplication in Winograd domain is the heaviest stage for large C and F. Hence, its speedup by quantization (integer arithmetic) is very relevant. The results of the theoretical evaluation of the balancing overhead, as well as the results of the inference time measurements for our implementation of the quantized Winograd convolutions (with and without balancing), can be found in Appendix C. The balancing overhead is small and shrinks fast with increasing C and F. The relative complexities on the left scheme (a) are written for $F = C = 32$.

Figure 1. The scheme of Balanced Quantized Winograd convolution and computational complexity proportions for the case of 3×3 kernel and stride 1.

Fig. 1a shows how the stages of the Winograd algorithm. Note that the standard Winograd convolution does not include the balancing and quantization/dequantization stages also shown in the figure. Initially, a sliding window of size $a \times a$ with stride m is moved over the feature map, and a sub-tensor $\mathbf{X}$ of the corresponding size (please, see inscriptions in the figure) is cut out from the feature map tensor, where $a = m + k - 1$. Then, the c-th tile $\mathbf{X_c} \in \mathbb{R}^{\mathbf{a}\times\mathbf{a}}$ is mapped to the Winogard domain using matrix $\mathbf{B}$: $\mathbf{X_c} \rightarrow \mathbf{V_c}$, $c = 1, 2, ..., C$. The same procedure is performed for the weights ($\mathbf{W} \rightarrow \mathbf{U}$), however, it is done off-line before the inference, hence, its computational complexity is not taken into account. After these steps, the mapped input $\mathbf{V}$ is convolved with the mapped weights $\mathbf{U}$. Finally, an invert transformation of tensor $\mathbf{M}$ obtained on the previous step is made using matrix $\mathbf{A}$. This procedure is performed in parallel for $m \times m$ output pixels and is more efficient than the direct convolution.

Winograd computational bottleneck Winograd transformation matrices $\mathbf{A}$ and $\mathbf{B}$ have a simple structure, so they can be hard coded in the inference code to additionally increase the performance. To estimate the relative complexity of the algorithm stages, we calculate the number of sum and mul operations as if we used a direct matrix multiplication (instead of hard coded). Even though this estimate may be far from reality, as it strongly depends on the type of inference engine (CPU, GPU or DNN accelerator), let us consider it as an approximate estimation (see the percentage in the bottom right corners of Fig. 1a). As you can see, the multiplication in the Winograd domain (obtaining matrix M) is the most computationally intense stage: it takes 52.8% of all computations (calculated for $C = F = 32$). Moreover, as the number of filters F or/and the number of channels C increases, the ratio tends to 100% (see Fig. 1b). Therefore, acceleration of that stage by quantization is very desirable.

Key problem As a rule, the smaller the tile size m, the less the quality drops during quantization of the Winograd convolution F$(m, 3)$; however, acceleration of the quantized Winograd convolution also becomes lower. Using the Winograd algorithm with small tile size ($m = 2$) often allows quantization without a quality drop, while Winograd algorithms with bigger tiles ($m \geqslant 4$) are very difficult to quantize (see Tab. 3 or Tab. 4). There are many studies addressing this issue, however, oftentimes the quality drop still remains unacceptable. We found one of the reasons why the quality drop in the post-training quantization (*PTQ*) cannot be compensated by the quantization-aware training (*QAT*) – it is a significant disbalance of the data ranges in the Winograd domain (see Sec. 4). We propose a technique aimed at solving this problem: to equalize the ranges of the Winograd-domain input channels on the forward pass by scaling the input tensor using special balancing coefficients (the filter channels are balanced offline). As can be seen in Fig. 2a and Appendix C, the balancing is a very cheap operation, but it significantly decreases the accuracy drop, for example, by 1.8 times for ResNet-18 (ImageNet, F$(6, 3)$, 8 bit, Tab. 4).

Table 2. The summary of the main well-known works and frameworks that propose methods for quantization of Winograd convolutions.

Work	Quant	Type	Offset	BN fusing	Scale types	Weights & Activations	Winograd
[17], IEEE-2020	PTQ	Dynamic	Yes	No	scalar	both	F(2, 3)
[11], ICLR-2019	PTQ	Dynamic	No	Yes	scalar	both	F(2, 3)
[22], ARM, 2019	PTQ	Dynamic	Yes	No	tile	weighs only	F(2, 3), F(4, 3), complex Winorgad
[10], MLSyS-2020	QAT	Static	No	No	scalar	both	F(2, 3), F(4, 3), F(6, 3), trainable transforms
[6], BOLT inference framework	PTQ	Dynamic, Static	No	Yes	tile, tile for act.	both	F(4,3)

Our contribution Contributions of this work can be summarized as follows:

1. We introduce a novel method for balancing of the channel ranges of the inputs and filters of the quantized Winograd convolutions. It is characterized by:
   (a) **Lightweight**: a small computational overhead;
   (b) **Compatibility**: compatible with any existing techniques for quantization of Winograd convolutions and works well for both PTQ and QAT ($\mathbf{\Omega}$ is a trainable variable in that case);
   (c) **Universality**: does not depend on the type of Winograd algorithm.
2. Our experiments on super-resolution (SR) and image classification tasks conducted for various bitwidths ($b = 4, 6, 8$), tile sizes m ($m = 4, 6$), quantization types (dynamic/static) and scale types (scalar/tile) show that the proposed balancing technique significantly improves the quality of Winograd quantization.

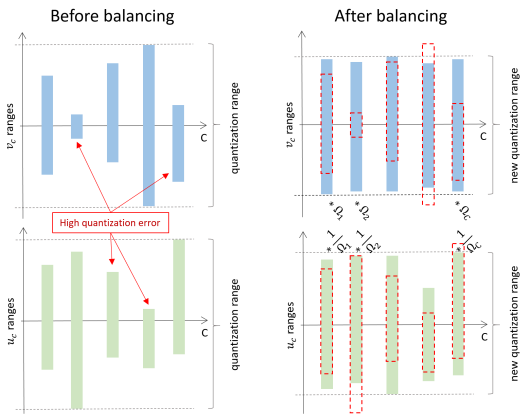
Our technique is not a panacea, but it extends the application area of the quantized Winograd convolutions substantially. We believe that it will have its rightful place in the standard quantization pipeline as an additional technique to further improve the quality of Winograd quantization.

2. Related works

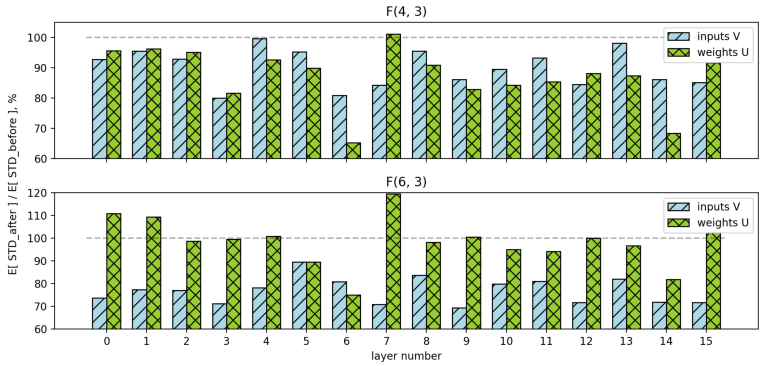
Quantization is a powerful tool for compressing and accelerating neural networks by using low-precision numbers. Quantization without training, also called *Post-Training Quantization* (*PTQ*), is a difficult and demanding task, since such quantization does not require complex calculations, has a high execution speed and can be efficiently performed on mobile devices. Various PTQ methods are actively proposed by the research community [1, 18, 23–25]. Cross-layer equalization and quantization bias correction techniques from paper [24] are effective methods that do not require any data and can help preserve the quality of the quantized model. The cross-layer equalization procedure from [24] is based on channel balancing of weights of subsequent convolutions. Also, the idea of factorization of layer channels to improve the quality of subsequent quantization is investigated in [21]. In these works, the channel balancing technique is applied only to weight tensors of traditional direct convolutions. AdaRound [23] is one of the popular and effective methods for PTQ, which allows to adaptively adjust the round function for different weight components of the model layers. While being the quickest approach, PTQ usually leads to the decrease in quality of a quantized network.

Quantization-Aware Training (*QAT*) is a popular way to obtain a quantized model with a high quality [4, 5, 8, 9, 13, 29]. In many studies on QAT, the authors reach quantized quality close to the original. QAT employs stochastic gradient descent with quantized weights and activations during training. This class of quantization methods has a few drawbacks – these methods require a full training dataset (usually it is private and it's distribution can be restricted) and computationally expensive training (a fine-tuning time of a quantized model can reach a full-precision training time), which sometimes is not available. A popular solution to the problem of non-differentiability of rounding functions during training of quantized models is to estimate the gradients of such functions using straight-through estimators [3].

The Winograd algorithm for acceleration of CNNs was first applied by Lavin et al. in [15]. Since then, a lot of research has been focused on improving Winograd convolutions, in particular on the study of quantization of Winograd convolutions. The most popular works on quantization of Winograd convolutions are summarized in Tab. 2. In [19], Winograd convolutions are extended to Residue Number System (RNS), which enables the use of bigger Winograd transformation tiles. Work [16] proposes a method of post-training quantization of Winograd convolutions, which uses a special optimization of quantization parameters to increase the final quality. Also there are some inference frameworks that support quantized Winograd convolutions,



(a) The idea of Channel balancing. Before balancing: the heterogeneity of channel ranges in $\mathbf{V}$ and $\mathbf{U}$ tensors leads to the increase of quantization error. After balancing: multiplying an element in $\mathbf{V}$ and dividing the corresponding one in $\mathbf{U}$ by same value Ω_i does not change the norm of $\mathbf{M}$, but can equalize the channel ranges and as a result significantly improve the quality of quantization.



(b) ResNet-20 model with Winograd convolutions on CIFAR-10 dataset. The ratio of the expectation of the standard deviation of channel ranges after balancing to the one before balancing for inputs and filters of all Winograd convolutions. In the great majority of cases, the channel balancing significantly reduces the standard deviation of channel ranges of both inputs V and weights U. For F$(6, 3)$ algorithm there is a small number of cases of increasing the dynamic ranges variance for the weight U (see the ratios of layers 0, 1, 7 and 15), but the overall positive impact of the equalization is still strong.

Figure 2. Channel balancing procedure for Quantized Winograd convolutions.

for example, BOLT [6], LANCE [17] or IntelCaffe [11]. As a rule, inference frameworks that support quantized Winograd convolutions automatically perform post-training quantization. More complex and efficient Winograd algorithms are more vulnerable to quantization, and therefore many frameworks [11, 17] do not use the most efficient algorithms due to a significant accuracy drop caused by quantization. Also, many frameworks implement dynamic quantization with online calculation of quantization scales of Winograd-domain inputs for each input sample, since it provides better quality, although it is less efficient than static quantization, for which we need to preliminary evaluate the input quantization scales.

Many of the proposed methods for quantization of Winograd convolutions are based on PTQ, because it is convenient. However, there are several works [2, 10] that address QAT of Winograd convolutions. In [10], in addition to QAT, to prepare a model with quantized Winograd convolutions, Neural Architecture Search (NAS) is used to select different values of tile size m for different Winograd convolutions. Also, different additional techniques can be used to improve the quality of quantized Winograd convolutions, such as using tile scales or affine quantization scheme [14]. However, often the quality drop remains unacceptable, especially in the case of more efficient Winograd algorithms ($m \geq 4$) or in the case of quantization with scalar scale, which is the main quantization format in some existing frameworks [11, 17]. The balancing technique proposed in this paper allows to significantly increase the quality of the quantized Winograd convolutions, and as a result, use more efficient Winograd algorithms and quantization formats without drop in model quality.

3. Preliminaries

For the sake of brevity, we refer the reader to the white paper [25] to learn about the SOTA quantization algorithms and terms. In our paper, we used symmetric uniform quantization [14]:

$$\tilde{\mathbf{Z}} = \lfloor \mathbf{Z} \cdot \mathbf{s} \rceil = \mathbf{Clip}\Big(\mathbf{Round}\big(\mathbf{Z} \cdot \mathbf{s}\big), -B, +B\Big), \quad (3)$$

where the tilde denotes that $\tilde{\mathbf{Z}}$ is a tensor of integers from interval $[-B, B]$ (same notation used in Fig. 1a), $B = 2^{b-1} - 1$, b is the quantization bitwidth and $s = B / \max|\mathbf{Z}|$ is the quantization scale (we discuss it below). There are more precise quantization types, which can improve the results presented in the paper, for example, the affine quantization with offsets [14]. We choose Eq. (3) for simplicity.

Let us focus now on the quantization scales s_v and s_u corresponding to inputs $\mathbf{V}$ and weight $\mathbf{U}$ in Winograd domain accordingly. As can be seen in Fig. 1a, they are used in two quantization ($\hat{\mathbf{V}} \rightarrow \tilde{\mathbf{V}}$ and $\hat{\mathbf{U}} \rightarrow \tilde{\mathbf{U}}$) and one dequantization ($\tilde{\mathbf{M}} \rightarrow \mathbf{M}$) stages. In our work, we considered two types of quantization scales:

1. **Scalar**: $s_v, s_u \in \mathbb{R}$ are scalars, and are used for quantization of tensors $\mathbf{V}$ and $\mathbf{U}$ correspondingly (tensor sizes are given in Fig. 1a). This case is the most difficult for quantization (see Tab. 3), but it was considered in many previous studies (see Tab. 2).

2. **Tile**: $\mathbf{s}_v, \mathbf{s}_u \in \mathbb{R}^{a \times a}$ are square matrices. Each pixel $\mathbf{V}_{ij} \in \mathbb{R}^C$ in tensor $\mathbf{V} \in \mathbb{R}^{C \times a \times a}$ has its own quantization scale $(\mathbf{s}_v)_{ij}$. This means that we quantize pixels independently. Such an approach is much more precise, since the dynamic range of pixels varies significantly. For simplicity, same pixels in filters share the

same quantization scale: $(\mathbf{s}_u)_{ij}$ is used for quantization of tensor $\mathbf{U}_{ij} \in \mathbb{R}^{F \times C}$.

From the quality perspective, the scalar scale $s_v \in \mathbb{R}$ is much worse than the tile scale $\mathbf{s}_v \in \mathbb{R}^{a \times a}$ (see Tab. 3), but from the number of operations they are equal (see step 3 in Fig. 1a): both $\mathbf{s}_v \odot \hat{\mathbf{V}}_c$ and $s_v \cdot \hat{\mathbf{V}}_c$ require a^2 operations, $\hat{\mathbf{V}}_c \in \mathbb{R}^{a \times a}$. Furthermore, this paper covers two quantization types related to quantization scale $\mathbf{s}_v$:

1. **Dynamic**: $\mathbf{s}_v$ is recalculated for every input $\mathbf{V}$ in the inference stage dynamically:

$$(\mathbf{s}_v)^n_{ij} = \frac{B}{\max |\mathbf{V}^n_{ij}|}, \tag{4}$$

where n denotes the n-th processing iteration, $\mathbf{V}^n_{ij} \in \mathbb{R}^C$. The approximate relative cost of that makes up a quarter of the relative cost of the quantization stage in Fig. 1b. In practice, the dynamic approach is rarely used, because it is hard to implement efficiently, however, it is much more precise (see Tab. 3 and Tab. 4).

2. **Static**: $\mathbf{s}_v$ is estimated using a set of inputs $\mathbf{V}$, obtained using the training or the calibration dataset. There are different methods how to estimate $\mathbf{s}_v$ (see [25]). For simplicity, we use the simplest approach:

$$(\mathbf{s}_v)_{ij} = \frac{1}{N} \sum_n^N (\mathbf{s}_v)^n_{ij}, \tag{5}$$

where n denotes the n-th element of the calibration set, and N denotes the number of elements in this set.

Now, we can write the formula of the *Quantized Winograd* convolution (*QW*):

$$\mathbf{Y}_f = \mathbf{A}^T \frac{\sum_c^C \lfloor \mathbf{B}^T \mathbf{X}_c \mathbf{B} \cdot s_v \rceil \odot \lfloor \mathbf{G} \mathbf{W}_{fc} \mathbf{G}^T \cdot s_u \rceil}{s_u s_v} \mathbf{A}. \tag{6}$$

For simplicity, Eq. (6) is written for the scalar scales. A computation in the numerator ($\tilde{\mathbf{M}}$ in Fig. 1a) is done using integer arithmetic. Using INT8 data speeds up the computations from $1.4\times$ to $2.2\times$ (strongly depends on the implementation) by increasing the number of elements processed at the same time, improving the efficiency of the cache utilization, and reducing the number of read/write operations to RAM.

4. Balancing Winograd convolution channels

The following rule holds for any method of calculating convolutions, in particular, for Winograd convolutions: the more balanced the channel ranges of inputs and weights of the convolution, the better its quantization. We can divide the channels of the Winograd-domain input by some balancing coefficients $\mathbf{\Omega}$ and multiply the corresponding channels of the Winograd-domain filter by the same balancing coefficients $\mathbf{\Omega}$ (the rectangles denote the data ranges in Fig. 2a). As a result, the transformed full-precision Winograd convolution (hereinafter the "FP Winograd") is equivalent to the original Winograd convolution. Moreover, if the balancing coefficients are chosen correctly, the new Winograd convolution is easier to quantize, since the channel ranges of the Winograd-domain inputs and filters become more balanced (Fig. 2a: small rectangles are stretched and large ones are squeezed, and as a result, all channels have their ranges equalized). By analogy with formulas from Eq. (6), the *Balanced Quantized Winograd* (*BQW*) convolution can be written as:

$$\mathbf{Y}_f = \mathbf{A}^T \left(\sum_c^C \left\lfloor \frac{\mathbf{B}^T \mathbf{X}_c \mathbf{B}}{\mathbf{\Omega}} \right\rceil \odot \lfloor \mathbf{G} \mathbf{W}_{fc} \mathbf{G}^T \odot \mathbf{\Omega} \rceil \right) \mathbf{A}, \tag{7}$$

where quantization scales s_u and s_v are omitted (but their existence is implied) to improve readability. Here, it is clear that balancing of the FP Winogard does not change the output $\mathbf{Y}_f$, because the variable $\mathbf{\Omega}$ is eliminated. Please, note that division and multiplication by $\mathbf{\Omega} \in \mathbb{R}^{C \times a \times a}$ are the element-wise operations (Hadamard product).

The scheme in Fig. 1a shows the formula from Eq. (7), from the computational stages perspective. Let us first focus on the sizes of tensors: tensor $\mathbf{V} \in \mathbb{R}^{C \times a \times a}$ and tensor $\mathbf{\Omega}$ have equal sizes, but the tensor $\mathbf{U} \in \mathbb{R}^{F \times C \times a \times a}$ is a four-dimensional tensor. So, to balance the weights $\mathbf{U}$, it is necessary to multiply each $\mathbf{U}_f$ element-wise by $\mathbf{\Omega}$ F times. It is not an issue, however, as this mapping $\mathbf{U} \to \hat{\mathbf{U}}$ and the quantization $\hat{\mathbf{U}} \to \tilde{\mathbf{U}}$ are performed offline.

Fig. 1b shows that the relative complexity of balancing is initially small (1.2%, Fig. 1a) and decreases fast with increasing C and F. Moreover, in the case of static quantization, scale $\mathbf{s}_v$ of the $\mathbf{\Omega}$ tensor can be fused in it, and the balancing overhead will be zero (see Sec. 6).

The correct choice of the balancing parameters $\mathbf{\Omega}$ significantly reduces the variance of the channel ranges of the Winograd-domain filter and inputs, and therefore allows maintaining a high model quality during quantization of Winograd convolutions. For the case of post-training quantization of Winograd convolutions, we propose a special direct algorithm for calculating the balancing tensor without additional model training, see Sec. 5. This algorithm uses the distributions of the input tensors of Winograd convolutions – one should just pass a few data samples to the model to calibrate expectations of Winograd-domain inputs (without training). Fig. 2b shows the ratio of the expectation of standard deviations of channel ranges of the Winograd-domain filters and inputs after balancing to the one before balancing for ResNet-20 model with Winograd convolu-

tions on CIFAR-10 dataset for this algorithm. As can be seen from this diagram, the proposed balancing algorithm significantly reduces the standard deviations of the channel ranges for the Winograd-domain filters and inputs. As will be shown below, the proposed algorithm for finding the balancing coefficients can significantly improve the quality of post-training quantization for various computer vision networks with Winograd convolutions, see Sec. 7. Moreover, we demonstrate that it is possible to initialize the balancing coefficients using this algorithm and further train them as trainable variables during Winograd quantization-aware training for greater quality improvement, see Sec. 7.

5. Post-training setup of balancing coefficients

The cross-layer weights equalization method (CLE, [24]) performs balancing of the channel ranges of weights of the two subsequent direct convolutions. The formula of the balancing coefficients is derived as an analytical solution of the optimization problem, aimed to balance the weight tensors of the subsequent layers. We generalize this optimization problem and its solution to the case of joint balancing of the Winograd-domain filters and inputs. We define the precision of channel c in the frequency domain (i, j) in balanced Winograd-domain filter $\hat{\mathbf{U}}$ as p_{ij}^c:

$$p_{ij}^c = \frac{r_{ij}^c}{R_{ij}}, \tag{8}$$

where r_{ij}^c is the quantization range of channel c in frequency domain (i, j) in tensor $\hat{\mathbf{U}}$, and R_{ij} is the total range of $\hat{\mathbf{U}}$ in frequency domain (i, j). In the case of symmetric quantization [14], we have $r_{ij}^c = 2 \cdot \max_f |\hat{U}_{fcij}|$ and $R_{ij} = 2 \cdot \max_{f,c} |\hat{U}_{fcij}|$. Similarly, we define the precision of channel c in frequency domain (i, j) in the balanced Winograd-domain input $\hat{\mathbf{V}}$ as q_{ij}^c:

$$q_{ij}^c = \frac{t_{ij}^c}{T_{ij}}, \tag{9}$$

where t_{ij}^c is the quantization range of channel c in frequency domain (i, j) in $\hat{\mathbf{V}}$ and T_{ij} is the total range of $\hat{\mathbf{V}}$ in frequency domain (i, j). To calculate the values of t_{ij}^c and T_{ij}, we calculate the maximum absolute values of components of tensor $\mathbf{V}$ for each channel c and Winograd-domain frequency (i, j) for several data batches, and then estimate the average value of the obtained maximum absolute values for these batches.

In the best case, the ranges of each channel for each Winograd domain are equal to the total range of this Winograd domain, which means that we use the maximum possible representative power per channel. As a result, we want to find balancing coefficients $\mathbf{\Omega}$ such that the total precision per channel for the Winograd-domain inputs and filters is maximized jointly for each frequency (i, j) in the Winograd domain:

$$\sum_c p_{ij}^c \cdot q_{ij}^c \rightarrow \max_{\Omega}. \tag{10}$$

A similar optimization problem is formulated in [24] with respect to the precision of channels of two adjacent layers, and analytical solution of this problem is given. Analogically, the solution to optimization problem from Eq. (10) can be written out explicitly:

$$\Omega_{ij}^c = \sqrt{\frac{t_{ij}^c}{r_{ij}^c}}. \tag{11}$$

The choice of balancing coefficients using formula from Eq. (11) provides the maximum value of total precision per channel for the Winograd-domain inputs and filters from Eq. (10), and therefore promotes the preservation of the quality of Winograd convolutions after quantization.

6. Fusing of balancing coefficients

Fusing the balancing coefficients into quantization scale is possible for the case of static quantization:

$$\mathbf{M} = \frac{\tilde{\mathbf{U}} \odot \lfloor s_v \cdot \mathbf{V} / \mathbf{\Omega} \rceil}{s_u \cdot s_v} \rightarrow \mathbf{M} = \frac{\tilde{\mathbf{U}} \odot \lfloor s_\Omega \cdot \mathbf{V} \rceil}{s_u \cdot s_v}, \tag{12}$$

where s_Ω is equal to $\frac{s_v}{\mathbf{\Omega}}$ and has dimensions (C, a, a). In this case, the same number of operations are performed in the inference stage as in the case of the traditional Winograd convolution. However, it requires storing a small number of additional parameters – balancing scales s_Ω. In the case of dynamic quantization, there is a need to perform additional operations of dividing the inputs into balancing tensors. These operations make up a very small part of the operations performed during the inference of the neural network, see Fig. 1.

7. Experiments

In this section, we compare the quality of models with QW and BQW convolutions. To this end, we conduct experiments with different architectures for image classification and image super-resolution tasks. In our experiments, we use pre-trained full-precision models, in which we replace all direct 3×3 convolutions (stride 1) with Winograd convolutions and quantize them. Usually, several first and last convolutions are replaced by the Winogard convolutions with the smaller tile size $m = 2$ to boost quality of a quantized model. This trick is not necessary for us, since our aim is to prove the positive impact of balancing.

The symmetric uniform quantization (Eq. (3)) is applied to both weights and inputs of Winograd convolutions. To make a comprehensive ablation study, we made same experiments for different quantization types (dynamic/static), quantization scales (tile/scalar), and bitwidths ($b = 4, 6, 8$).

Table 3. Quantization of ResNet-20 with Winograd convolutions on Cifar-10. The full-precision model accuracy is 91.73%.

Scale type	Quantization type	Winograd algorithm	Bits	PTQ, accuracy, % QW (Baseline)	PTQ, accuracy, % BQW	QAT, accuracy, % QW (Baseline)	QAT, accuracy, % BQW
Tile	Static	F(4, 3)	4	10.00	10.00	33.79	**85.33**
			6	76.30	**79.36**	85.98	**90.74**
			8	89.93	**90.31**	91.18	**91.75**
		F(6, 3)	6	33.74	**39.75**	30.27	**83.79**
			8	81.29	**81.44**	48.77	**90.57**
	Dynamic	F(4, 3)	6	84.28	**86.43**	N/A	
			8	91.45	**91.83**		
		F(6, 3)	6	51.10	**61.37**		
			8	90.90	**91.18**		
Scalar	Static	F(4, 3)	8	16.22	**68.23**	N/A	
	Dynamic	F(6, 3)	8	18.06	**78.96**		

Table 4. Post-training quantization of models with Winograd convolutions on ImageNet into 8 bit (scale type: tile).

Model	Full-precision accuracy	Quantization type	Winograd algorithm	Accuracy, %, QW	Accuracy, %, BQW
ResNet-18	69.76%	Static	F(4,3)	66.15%	**67.54**%
			F(6,3)	53.56%	**60.59**%
		Dynamic	F(4,3)	67.72%	**68.94**%
			F(6,3)	60.17%	**66.08**%
ResNet-34	73.30%	Static	F(4,3)	70.81%	**71.86**%
			F(6,3)	62.34%	**66.52**%
		Dynamic	F(4,3)	71.71%	**72.86**%
			F(6,3)	67.43%	**70.87**%
ResNet-50	76.15%	Static	F(4,3)	75.36%	**75.84**%
			F(6,3)	73.75%	**74.47**%
		Dynamic	F(4,3)	75.84%	**76.10**%
			F(6,3)	75.01%	**75.79**%

Table 5. Static 8-bit PTQ of ESPCNN model for 3x super-resolution with F(4, 3) Winograd convolutions on Set5 and Set14 datasets.

Quantization (scale type: tile)	Set5, PSNR to ground truth	Set14, PSNR to ground truth	Set5, PSNR to full-precision model	Set14, PSNR to full-precision model
Full-precision model	30.74 dB	27.06 dB	∞	∞
Quantized Winograd (baseline)	29.41 dB	26.33 dB	35.89 dB	35.72 dB
Quantized Balanced Winograd	**30.24** dB	**26.76** dB	**39.62** dB	**39.15** dB

To initialize the scale parameters of the Winograd-domain inputs $\mathbf{V}$ in the case of static quantization, we pass several data samples through the model and estimate the average data ranges (see Eq. (3)). In our experiments, we use 10000 samples from training sets for image classification tasks and 3200 samples from the training set for image super-resolution tasks for both calibration of balancing coefficients and estimation of quantization scales of Winograd-domain inputs in the case of static quantization.

PTQ for image classification We have conducted experiments for static and dynamic PTQ of ResNet-20 [12] model on the CIFAR-10 dataset, and ResNet-18, ResNet-34 and ResNet-50 models [12] on the ImageNet dataset. In these experiments, we use F(4, 3) and F(6, 3) Winograd algorithms. For experiments on CIFAR-10, we quantize the models to 6 and 8 bits for the case of the tile scale, and

Figure 3. PTQ of ESPCNN model for 3-x super-resolution: comparison of QW and BQW ($F(4,3)$, static 8 bit, scale type: tile).

for the case of scalar scale we provide only the results of quantization of F$(4,3)$ Winograd algorithm to 8 bits. For experiments on ImageNet, we quantize the models to 8 bits and provide results for the case of the tile scale. Validation accuracy of the quantized models with QW and BQW convolutions is given in Tab. 3 and Tab. 4. We observe that using channel balancing significantly improves the quality of the quantized models with Winograd convolutions for various Winograd algorithms and quantization formats in the case of PTQ.

QAT for image classification We also conducted experiments for static QAT of ResNet-20 model on the CIFAR-10 dataset. We quantize models with traditional and balanced Winograd convolutions to 4, 6 and 8 bits for the case of the tile scale. In the case of balancing, parameters $\mathbf{\Omega}$ are initialized by the algorithm from Sec. 5 and then tuned as trainable variables during training of the model with BQW convolutions. We train all quantized models with Winograd convolutions for 300 epochs using SGD with different configurations of training parameters. In each case, we provide the best result achieved, since the best results for QW and for BQW are achieved with different hyperparameters. We observe that using channel balancing significantly improves the quality of the quantized models with Winograd convolutions and accelerates convergence during training, see Tab. 3 and Fig. 4. Note that the results of QAT are worse than the results of PTQ in the case of F$(6,3)$ Winograd algorithm for the baseline QW, since in this case QAT does not converge and stops at a configuration worse than the one using which it was initialized. See details about training configurations in Appendix B.

PTQ for image Super-Resolution In this section, we provide the results of experiments for quantization of *Efficient Sub-Pixel CNN* (ESPCNN) [26] with global residual connection for the $3\times$ image super-resolution task. The full-precision model was trained on the Vimeo-90K dataset and consists of 6 convolutions. We replace all direct 2D convolutions except for the first one with Winograd convolutions, as the kernel size of the first convolution is 5. We quantize filters and inputs of Winograd convolutions to 8 bits. We conducted experiments with F$(4,3)$ Winograd algorithm for the task of static PTQ with tile scale, see results in Tab. 5. The obtained results show that the quality of the model with BQW convolutions significantly exceeds the quality of the model with QW convolutions and is close to the quality of the full-precision model, see Fig. 3.

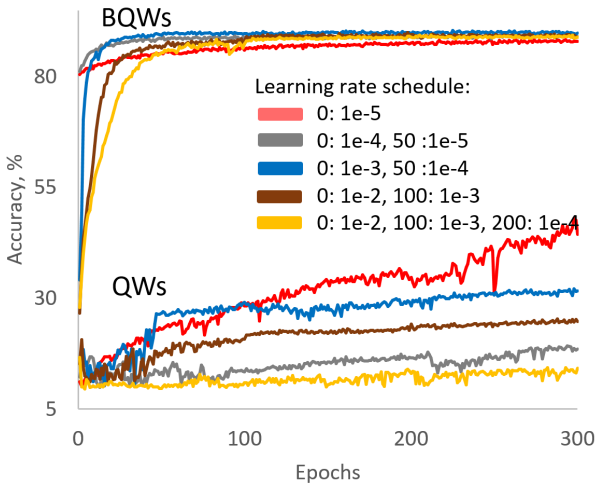


Figure 4. The test accuracy over epochs for F$(6,3)$ Winograd algorithm. We use the SGD optimizer with momentum. The traditional quantized Winograd without balancing (QW) does not converge at all. In contrast, the balanced quantized Winograd (BQW) trains well. Pay attention, that BQW is stable to variation of hyperparameters (LR) – it converges well for any LR.

8. Conclusions

We propose a novel technique of balancing the channel ranges of the Winograd-domain inputs and filters that significantly increases the quality of the quantized Winograd convolutions. The conducted experiments show that the variance of data ranges decreases – data in different channels becomes homogeneous. This has a positive impact on the quality of the quantized model as is confirmed by the experiments. The price to pay to equalize the channel ranges is small, see Appendix C. Moreover, in the case of static scales, the overhead of balancing can be eliminated by fusing the balancing coefficients into the quantization scales. In addition, the strength of the suggested technique lies in the compatibility with any existing quantization method and any Winograd algorithm.

References

[1] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Post-training 4-bit quantization of convolution networks for rapid-deployment. *arXiv preprint arXiv:1810.05723*, 2018. 3

[2] Barbara Barabasz. Quantaized winograd/toom-cook convolution for dnns: Beyond canonical polynomials base. *arXiv preprint arXiv:2004.11077*, 2020. 4

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

[4] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 696–697, 2020. 3

[5] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: parameterized clipping activation for quantized neural networks. *CoRR*, abs/1805.06085, 2018. 3

[6] Huawei Technologies Co. Bolt inference framework. https://github.com/huawei-noah/bolt.git, 2021. 3, 4

[7] Stephen A. Cook. On the minimum computation time of functions. *PhD thesis*, 1966. 1

[8] 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. 3

[9] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 3

[10] Javier Fernandez-Marques, Paul N Whatmough, Andrew Mundy, and Matthew Mattina. Searching for winograd-aware quantized networks. *arXiv preprint arXiv:2002.10711*, 2020. 3, 4

[11] Jiong Gong, Haihao Shen, Guoming Zhang, Xiaoli Liu, Shane Li, Ge Jin, Niharika Maheshwari, Evarist Fomenko, and Eden Segal. Highly efficient 8-bit low precision inference of convolutional neural networks with intelcaffe. In *Proceedings of the 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning*, page 1. 2018. 1, 3, 4

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 7

[13] 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. 3

[14] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 4, 6

[15] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4013–4021, 2016. 1, 3

[16] Guangli Li, Zhen Jia, Xiaobing Feng, and Yida Wang. Lowino: Towards efficient low-precision winograd convolutions on modern cpus. In *50th International Conference on Parallel Processing*, pages 1–11, 2021. 3

[17] Guangli Li, Lei Liu, Xueying Wang, Xiu Ma, and Xiaobing Feng. Lance: efficient low-precision quantized winograd convolution for neural networks based on graphics processing units. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3842–3846. IEEE, 2020. 3, 4

[18] Zhenhua Liu, Yunhe Wang, Kai Han, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *arXiv preprint arXiv:2106.14156*, 2021. 3

[19] Zhi-Gang Liu and Matthew Mattina. Efficient residue number system based winograd convolution. In *European Conference on Computer Vision*, pages 53–68. Springer, 2020. 3

[20] Partha Maji, Andrew Mundy, Ganesh Dasika, Jesse Beu, Matthew Mattina, and Robert Mullins. Efficient winograd or cook-toom convolution kernel implementation on widely used mobile cpus. In *2019 2nd Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*, pages 1–5. IEEE, 2019. 1

[21] Eldad Meller, Alexander Finkelstein, Uri Almog, and Mark Grobman. Same, same but different: Recovering neural network quantization error through weight factorization. In *International Conference on Machine Learning*, pages 4486–4495. PMLR, 2019. 3

[22] Lingchuan Meng and John Brothers. Efficient winograd convolution via integer arithmetic. *arXiv preprint arXiv:1901.01965*, 2019. 1, 3

[23] 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. 3

[24] 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. 3, 6

[25] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. 3, 4, 5

[26] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *CoRR*, abs/1609.05158, 2016. 8

[27] Andrei L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. In *Soviet Mathematics Doklady*, volume 3, pages 714–716, 1963. 1

[28] Shmuel Winograd. *Arithmetic complexity of computations*, volume 33. Siam, 1980. 1

[29] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2016. 3