Up or Down? Adaptive Rounding for Post-Training Quantization

Markus Nagel [* 1]   **Rana Ali Amjad** [* 1]   **Mart van Baalen** [1]   **Christos Louizos** [1]   **Tijmen Blankevoort** [1]

Abstract

When quantizing neural networks, assigning each floating-point weight to its nearest fixed-point value is the predominant approach. We find that, perhaps surprisingly, this is not the best we can do. In this paper, we propose AdaRound, a better weight-rounding mechanism for post-training quantization that adapts to the data and the task loss. AdaRound is fast, does not require fine-tuning of the network, and only uses a small amount of unlabelled data. We start by theoretically analyzing the rounding problem for a pre-trained neural network. By approximating the task loss with a Taylor series expansion, the rounding task is posed as a quadratic unconstrained binary optimization problem. We simplify this to a layer-wise local loss and propose to optimize this loss with a soft relaxation. AdaRound not only outperforms rounding-to-nearest by a significant margin but also establishes a new state-of-the-art for post-training quantization on several networks and tasks. Without fine-tuning, we can quantize the weights of Resnet18 and Resnet50 to 4 bits while staying within an accuracy loss of 1%.

1. Introduction

Deep neural networks are being used in many real-world applications as the standard technique for solving tasks in computer vision, machine translation, voice recognition, ranking, and many other domains. Owing to this success and widespread applicability, making these neural networks efficient has become an important research topic. Improved efficiency translates into reduced cloud-infrastructure costs and makes it possible to run these networks on heterogeneous devices such as smartphones, internet-of-things appli-

cations, and even dedicated low-power hardware.

One effective way to optimize neural networks for inference is neural network quantization (Krishnamoorthi, 2018; Guo, 2018). In quantization, neural network weights and activations are kept in a low-bit representation for both memory transfer and calculations in order to reduce power consumption and inference time. The process of quantizing a network generally introduces noise, which results in a loss of performance. Various prior works adapt the quantization procedure to minimize the loss in performance while going as low as possible in the number of bits used.

As Nagel et al. (2019) explained, the practicality of neural network quantization methods is important to take into consideration. Although many methods exist that do quantization-aware training (Jacob et al., 2018; Louizos et al., 2019) and get excellent results, these methods require a user to spend significant time on re-training models and hyperparameter tuning.

On the other hand, much attention has recently been dedicated to *post-training quantization* methods (Nagel et al., 2019; Cai et al., 2020; Choukroun et al., 2019; Banner et al., 2019), which can be more easily applied in practice. These types of methods allow for network quantization to happen on-the-fly when deploying models, without the user of the model spending time and energy on quantization. Our work focuses on this type of network quantization.

Rounding-to-nearest is the predominant approach for all neural network weight quantization work that came out thus far. This means that the weight vector $\mathbf{w}$ is rounded to the nearest representable quantization grid value in a fixed-point grid by

$$\widehat{\mathbf{w}} = \mathrm{s} \cdot clip\left(\left\lfloor \frac{\mathbf{w}}{\mathrm{s}} \right\rceil, \mathrm{n}, \mathrm{p}\right), \tag{1}$$

where s denotes the quantization scale parameter and, n and p denote the negative and positive integer thresholds for clipping. We could round any weight down by replacing $\lfloor \cdot \rceil$ with $\lfloor \cdot \rfloor$, or up using $\lceil \cdot \rceil$. But, rounding-to-nearest seems the most sensible, as it minimizes the difference per-weight in the weight matrix. Perhaps surprisingly, we show that for post-training quantization, rounding-to-nearest is not optimal.

Our contributions in this work are threefold:

- We establish a theoretical framework to analyze the effect of rounding in a way that considers the characteristics of both the input data as well as the task loss. Using this framework, we formulate rounding as a per-layer Quadratic Unconstrained Binary Optimization (QUBO) problem.

- We propose AdaRound, a novel method that finds a good solution to this per-layer formulation via a continuous relaxation. AdaRound requires only a small amount of unlabelled data, is computationally efficient, and applicable to any neural network architecture with convolutional or fully-connected layers.

- In a comprehensive study, we show that AdaRound defines a new state-of-the-art for post-training quantization on several networks and tasks, including Resnet18, Resnet50, MobilenetV2, InceptionV3 and DeeplabV3.

Notation   We use $\mathbf{x}$ and $\mathbf{y}$ to denote the input and the target variable, respectively. $\mathbb{E}\left[\cdot\right]$ denotes the expectation operator. All the expectations in this work are w.r.t. $\mathbf{x}$ and $\mathbf{y}$. $\mathbf{W}_{i,j}^{(\ell)}$ denotes weight matrix (or tensor as clear from the context), with the bracketed superscript and the subscript denoting the layer and the element indices, respectively. We also use $\mathbf{w}^{(\ell)}$ to denote flattened version of $\mathbf{W}^{(\ell)}$. All vectors are considered to be column vectors and represented by small bold letters, e.g., $\mathbf{z}$, while matrices (or tensors) are denoted by capital bold letters, e.g., $\mathbf{Z}$. Functions are denoted by $f(\cdot)$, except the task loss, which is denoted by $\mathcal{L}$. Constants are denoted by small upright letters, e.g., s.

2. Motivation

To gain an intuitive understanding for why rounding-to-nearest may not be optimal, let's look at what happens when we perturb the weights of a pretrained model. Consider a neural network parametrized by the (flattened) weights $\mathbf{w}$. Let $\Delta\mathbf{w}$ denote a small perturbation and $\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{w})$ denote the task loss that we want to minimize. Then

$$\mathbb{E}\left[\mathcal{L}\left(\mathbf{x}, \mathbf{y}, \mathbf{w} + \Delta\mathbf{w}\right) - \mathcal{L}\left(\mathbf{x}, \mathbf{y}, \mathbf{w}\right)\right] \quad (2)$$

$$\overset{(a)}{\approx} \mathbb{E}\left[\Delta\mathbf{w}^T \cdot \nabla_{\mathbf{w}}\mathcal{L}\left(\mathbf{x}, \mathbf{y}, \mathbf{w}\right)\right.$$
$$\left. + \frac{1}{2}\Delta\mathbf{w}^T \cdot \nabla_{\mathbf{w}}^2\mathcal{L}\left(\mathbf{x}, \mathbf{y}, \mathbf{w}\right) \cdot \Delta\mathbf{w}\right] \quad (3)$$

$$= \Delta\mathbf{w}^T \cdot \mathbf{g}^{(\mathbf{w})} + \frac{1}{2}\Delta\mathbf{w}^T \cdot \mathbf{H}^{(\mathbf{w})} \cdot \Delta\mathbf{w}, \quad (4)$$

where (a) uses the second order Taylor series expansion. $\mathbf{g}^{(\mathbf{w})}$ and $\mathbf{H}^{(\mathbf{w})}$ denote the expected gradient and Hessian of the task loss $\mathcal{L}$ w.r.t. $\mathbf{w}$, i.e.,

$$\mathbf{g}^{(\mathbf{w})} = \mathbb{E}\left[\nabla_{\mathbf{w}}\mathcal{L}\left(\mathbf{x}, \mathbf{y}, \mathbf{w}\right)\right] \quad (5)$$

$$\mathbf{H}^{(\mathbf{w})} = \mathbb{E}\left[\nabla_{\mathbf{w}}^2\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{w})\right]. \quad (6)$$

All the gradient and Hessian terms in this paper are of task loss $\mathcal{L}$ with respect to the specified variables. Ignoring the higher order terms in the Taylor series expansion is a good approximation as long as $\Delta\mathbf{w}$ is not too large. Assuming the network is trained to convergence, we can also ignore the gradient term as it will be close to 0. Therefore, $\mathbf{H}^{(\mathbf{w})}$ defines the interactions between different perturbed weights in terms of their joint impact on the task loss $\mathcal{L}(\mathbf{x}, \mathbf{y}, \mathbf{w} + \Delta\mathbf{w})$. The following toy example illustrates how rounding-to-nearest may not be optimal.

Example 1. Assume $\Delta\mathbf{w}^T = \begin{bmatrix} \Delta w_1 & \Delta w_2 \end{bmatrix}$ and

$$\mathbf{H}^{(\mathbf{w})} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \quad (7)$$

then the increase in task loss due to the perturbation is (approximately) proportional to

$$\Delta\mathbf{w}^T \cdot \mathbf{H}^{(\mathbf{w})} \cdot \Delta\mathbf{w} = \Delta\mathbf{w}_1^2 + \Delta\mathbf{w}_2^2 + \Delta\mathbf{w}_1\Delta\mathbf{w}_2. \quad (8)$$

For the terms corresponding to the diagonal entries $\Delta\mathbf{w}_1^2$ and $\Delta\mathbf{w}_2^2$, only the magnitude of the perturbations matters. Hence rounding-to-nearest is optimal when we only consider these diagonal terms in this example. However, for the terms corresponding to the $\Delta\mathbf{w}_1\Delta\mathbf{w}_2$, the sign of the perturbation matters, where opposite signs of the two perturbations improve the loss. To minimize the overall impact of quantization on the task loss, we need to trade-off between the contribution of the diagonal terms and the off-diagonal terms. Rounding-to-nearest ignores the off-diagonal contributions, making it often sub-optimal.

The previous analysis is valid for the quantization of any parametric system. We show that this effect also holds for neural networks. To illustrate this, we generate 100 stochastic rounding (Gupta et al., 2015) choices for the first layer of Resnet18 and evaluate the performance of the network with only the first layer quantized. The results are presented in Table 1. Among 100 runs, we find that 48 stochastically sampled rounding choices lead to a better performance than rounding-to-nearest. This implies that many rounding solutions exist that are better than rounding-to-nearest. Furthermore, the best among these 100 stochastic samples provides more than 10% improvement in the accuracy of the network. We also see that accidentally rounding all values up, or all down, has an catastrophic effect. This implies that we can gain a lot by carefully rounding weights when doing post-training quantization. The rest of this paper is aimed at devising a well-founded and computationally efficient rounding mechanism.

3. Method

In this section, we propose AdaRound, a new rounding procedure for post-training quantization that is theoretically

Rounding scheme	Acc(%)
Nearest	52.29
Ceil	0.10
Floor	0.10
Stochastic	52.06±5.52
Stochastic (best)	63.06

Table 1. Comparison of ImageNet validation accuracy among different rounding schemes for 4-bit quantization of the first layer of Resnet18. We report the mean and the standard deviation of 100 stochastic (Gupta et al., 2015) rounding choices (Stochastic) as well as the best validation performance among these samples (Stochastic (best)).

well-founded and shows significant performance improvement in practice. We start by analyzing the loss due to quantization theoretically. We then formulate an efficient per-layer algorithm to optimize it.

3.1. Task loss based rounding

When quantizing a pretrained NN, our aim is to minimize the performance loss incurred due to quantization. Assuming per-layer weight quantization[1], the quantized weight $\widehat{\mathbf{w}}_i^{(\ell)}$ is

$$\widehat{\mathbf{w}}_i^{(\ell)} \in \left\{ \mathbf{w}_i^{(\ell),floor}, \mathbf{w}_i^{(\ell),ceil} \right\}, \tag{9}$$

where

$$\mathbf{w}_i^{(\ell),floor} = \mathbf{s}^{(\ell)} \cdot clip\left(\left\lfloor \frac{\mathbf{w}_i^{(\ell)}}{\mathbf{s}^{(\ell)}} \right\rfloor, \mathbf{n}, \mathbf{p} \right) \tag{10}$$

and $\mathbf{w}_i^{(\ell),ceil}$ is similarly defined by replacing $\lfloor \cdot \rfloor$ with $\lceil \cdot \rceil$ and $\Delta\mathbf{w}_i^{(\ell)} = \mathbf{w}_i^{(\ell)} - \widehat{\mathbf{w}}_i^{(\ell)}$ denotes the perturbation due to quantization. In this work we assume $\mathbf{s}^{(\ell)}$ to be fixed prior to optimizing the rounding procedure. Finally, whenever we optimize a cost function over the $\Delta\mathbf{w}_i^{(\ell)}$, the $\widehat{\mathbf{w}}_i^{(\ell)}$ can only take two values specified in (9).

Finding the optimal rounding procedure can be formulated as the following binary optimization problem

$$\underset{\Delta\mathbf{w}}{\arg\min} \quad \mathbb{E}\left[\mathcal{L}\left(\mathbf{x}, \mathbf{y}, \mathbf{w} + \Delta\mathbf{w} \right) - \mathcal{L}\left(\mathbf{x}, \mathbf{y}, \mathbf{w} \right) \right] \tag{11}$$

Evaluating the cost in (11) requires a forward pass of the input data samples for each new $\Delta\mathbf{w}$ during optimization. To avoid the computational overhead of repeated forward passes throught the data, we utilize the second order Taylor series approximation. Additionally, we ignore the interactions among weights belonging to different layers. This, in

---

[1]Note that our work is equally applicable for per-channel weight quantization.
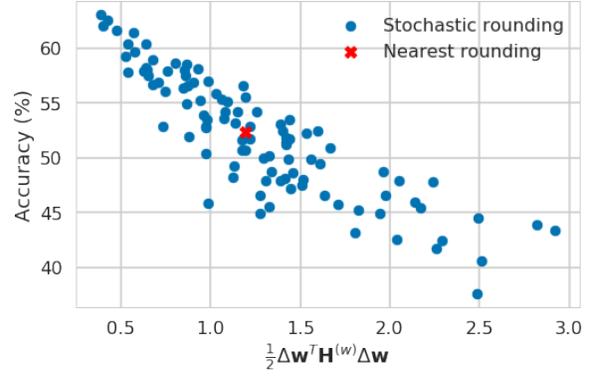

Figure 1. Correlation between the cost in (13) vs ImageNet validation accuracy (%) of 100 stochastic rounding vectors $\widehat{\mathbf{w}}$ for 4-bit quantization of only the first layer of Resnet18.

turn, implies that we assume a block diagonal $\mathbf{H}^{(\mathbf{w})}$, where each non-zero block corresponds to one layer. We thus end up with the following per-layer optimization problem

$$\underset{\Delta\mathbf{w}^{(\ell)}}{\arg\min} \quad \mathbb{E}\left[\mathbf{g}^{(\mathbf{w}^{(\ell)})^T} \Delta\mathbf{w}^{(\ell)} + \frac{1}{2}\Delta\mathbf{w}^{(\ell)T}\mathbf{H}^{(\mathbf{w}^{(\ell)})}\Delta\mathbf{w}^{(\ell)} \right]. \tag{12}$$

As illustrated in Example 1, we require the second order term to exploit the joint interactions among the weight perturbations. (12) is a QUBO problem since $\Delta\mathbf{w}_i^{(\ell)}$ are binary variables (Kochenberger et al., 2014). For a converged pretrained model, the contribution of the gradient term for optimization in (13) can be safely ignored. This results in

$$\underset{\Delta\mathbf{w}^{(\ell)}}{\arg\min} \quad \mathbb{E}\left[\Delta\mathbf{w}^{(\ell)T}\mathbf{H}^{(\mathbf{w}^{(\ell)})}\Delta\mathbf{w}^{(\ell)} \right]. \tag{13}$$

To verify that (13) serves as a good proxy for optimizing task loss due to quantization, we plot the cost in (13) vs validation accuracy for 100 stochastic rounding vectors when quantizing only the first layer of Resnet18. Fig. 1 shows a clear correlation between the two quantities. This justifies our approximation for optimization, even for 4 bit quantization. Optimizing (13) show significant performance gains, however its application is limited by two problems:

1. $\mathbf{H}^{(\mathbf{w}^{(\ell)})}$ suffers from both computational as well memory complexity issues even for moderately sized layers.

2. (13) is an NP-hard optimization problem. The complexity of solving it scales rapidly with the dimension of $\Delta\mathbf{w}^{(\ell)}$, again prohibiting the application of (13) to even moderately sized layers (Kochenberger et al., 2014).

In section 3.2 and section 3.3 we tackle the first and the second problem, respectively.

3.2. From Taylor expansion to local loss

To understand the cause of the complexity associated with $\mathbf{H}^{(\mathbf{w}^{(\ell)})}$, let us look at its' elements. For two weights in the same fully connected layer we have

$$\frac{\partial^2 \mathcal{L}}{\partial \mathbf{W}_{i,j}^{(\ell)} \partial \mathbf{W}_{m,o}^{(\ell)}} = \frac{\partial}{\partial \mathbf{W}_{m,o}^{(\ell)}} \left[\frac{\partial \mathcal{L}}{\partial \mathbf{z}_i^{(\ell)}} \cdot \mathbf{x}_j^{(\ell-1)} \right] \qquad (14)$$

$$= \frac{\partial^2 \mathcal{L}}{\partial \mathbf{z}_i^{(\ell)} \partial \mathbf{z}_m^{(\ell)}} \cdot \mathbf{x}_j^{(\ell-1)} \mathbf{x}_o^{(\ell-1)}, \qquad (15)$$

where $\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)} \mathbf{x}^{(\ell-1)}$ are the preactivations for layer ℓ and $\mathbf{x}^{(\ell-1)}$ denotes the input to layer ℓ. Writing this in matrix formulation (for flattened $\mathbf{w}^{(\ell)}$), we have (Botev et al., 2017)

$$\mathbf{H}^{(\mathbf{w}^{(\ell)})} = \mathbb{E} \left[\mathbf{x}^{(\ell-1)} \mathbf{x}^{(\ell-1)T} \otimes \nabla_{\mathbf{z}^{(\ell)}}^2 \mathcal{L} \right], \qquad (16)$$

where $\otimes$ denotes Kronecker product of two matrices and $\nabla_{\mathbf{z}^{(\ell)}}^2 \mathcal{L}$ is the Hessian of the task loss w.r.t. $\mathbf{z}^{(\ell)}$. It is clear from (16) that the complexity issues are mainly caused by $\nabla_{\mathbf{z}^{(\ell)}}^2 \mathcal{L}$ that requires backpropagation of second derivatives through the subsequent layers of the network. To tackle this, we make the assumption that the Hessian of the task loss w.r.t. the preactivations, i.e., $\nabla_{\mathbf{z}^{(\ell)}}^2 \mathcal{L}$ is a diagonal matrix, denoted by $diag\left(\nabla_{\mathbf{z}^{(\ell)}}^2 \mathcal{L}_{i,i}\right)$. This leads to

$$\mathbf{H}^{(\mathbf{w}^{(\ell)})} = \mathbb{E} \left[\mathbf{x}^{(\ell-1)} \mathbf{x}^{(\ell-1)T} \otimes diag(\nabla_{\mathbf{z}^{(\ell)}}^2 \mathcal{L}_{i,i}) \right]. \quad (17)$$

Note that the approximation of $\mathbf{H}^{(\mathbf{w}^{(\ell)})}$ expressed in (17) is not diagonal. Plugging (17) into our equation for finding the rounding vector that optimizes the loss (13), we obtain

$$\underset{\Delta \mathbf{W}_{k,:}^{(\ell)}}{\arg\min} \quad \mathbb{E} \left[\nabla_{\mathbf{z}^{(\ell)}}^2 \mathcal{L}_{k,k} \cdot \Delta \mathbf{W}_{k,:}^{(\ell)} \mathbf{x}^{(\ell-1)} \mathbf{x}^{(\ell-1)T} \Delta \mathbf{W}_{k,:}^{(\ell)T} \right]$$

$$(18)$$

$$\overset{(a)}{=} \underset{\Delta \mathbf{W}_{k,:}^{(\ell)}}{\arg\min} \quad \Delta \mathbf{W}_{k,:}^{(\ell)} \mathbb{E} \left[\mathbf{x}^{(\ell-1)} \mathbf{x}^{(\ell-1)T} \right] \Delta \mathbf{W}_{k,:}^{(\ell)T} \qquad (19)$$

$$= \underset{\Delta \mathbf{W}_{k,:}^{(\ell)}}{\arg\min} \quad \mathbb{E} \left[\left(\Delta \mathbf{W}_{k,:}^{(\ell)} \mathbf{x}^{(\ell-1)} \right)^2 \right], \qquad (20)$$

where the optimization problem in (13) now decomposes into independent sub-problems in (18). Each sub-problem deals with a single row $\Delta \mathbf{W}_{k,:}^{(\ell)}$ and (a) is the outcome of making a further assumption that $\nabla_{\mathbf{z}^{(\ell)}}^2 \mathcal{L}_{i,i} = c_i$ is a constant independent of the input data samples. It is worthwhile to note that optimizing (20) requires no knowledge of the subsequent layers and the task loss. In (20), we are simply minimizing the Mean Squared Error (MSE) introduced in the preactivations $\mathbf{z}^{(\ell)}$ due to quantization. This is the same layer-wise objective that was optimized in several neural network compression papers, e.g., Zhang et al. (2016); He

et al. (2017), and various neural network quantization papers (albeit for tasks other than weight rounding), e.g., Wang et al. (2018); Stock et al. (2020); Choukroun et al. (2019). However, unlike these works, we arrive at this objective in a principled way and conclude that optimizing the MSE, as specified in (20), is the best we can do when assuming no knowledge of the rest of the network past the layer that we are optimizing. In the supplementary material we perform an analogous analysis for convolutional layers.

The optimization problem in (20) can be tackled by either precomputing $\mathbb{E}\left[\mathbf{x}^{(\ell-1)} \mathbf{x}^{(\ell-1)T} \right]$, as done in (19), and then performing the optimization over $\Delta \mathbf{W}_{k,:}^{(\ell)}$, or by performing a single layer forward pass for each potential $\Delta \mathbf{W}_{k,:}^{(\ell)}$ during the optimization procedure.

In section 5, we empirically verify that the constant diagonal approximation of $\nabla_{\mathbf{z}^{(\ell)}}^2 \mathcal{L}$ does not negatively influence the performance.

3.3. AdaRound

Solving (20) does not suffer from complexity issues associated with $\mathbf{H}^{(\mathbf{w}^{(\ell)})}$. However, it is still an NP-hard discrete optimization problem. Finding good (sub-optimal) solution with reasonable computational complexity can be a challenge for larger number of optimization variables. To tackle this we relax (20) to the following continuous optimization problem based on soft quantization variables (the superscripts are the same as (20))

$$\underset{\mathbf{V}}{\arg\min} \quad \left\| \mathbf{W}\mathbf{x} - \widetilde{\mathbf{W}}\mathbf{x} \right\|_F^2 + \lambda f_{reg}\left(\mathbf{V}\right), \qquad (21)$$

where $\|\cdot\|_F^2$ denotes the Frobenius norm and $\widetilde{\mathbf{W}}$ are the soft-quantized weights that we optimize over

$$\widetilde{\mathbf{W}} = \mathbf{s} \cdot clip\left(\left\lfloor \frac{\mathbf{W}}{\mathbf{s}} \right\rfloor + h\left(\mathbf{V}\right), \mathbf{n}, \mathbf{p} \right). \qquad (22)$$

In the case of a convolutional layer the $\mathbf{W}\mathbf{x}$ matrix multiplication is replaced by a convolution. $\mathbf{V}_{i,j}$ is the continuous variable that we optimize over and $h\left(\mathbf{V}_{i,j}\right)$ can be any differentiable function that takes values between 0 and 1, i.e., $h\left(\mathbf{V}_{i,j}\right) \in [0,1]$. The additional term $f_{reg}\left(\mathbf{V}\right)$ is a differentiable regularizer that is introduced to encourage the optimization variables $h\left(\mathbf{V}_{i,j}\right)$ to converge towards either 0 or 1, i.e., at convergence $h\left(\mathbf{V}_{i,j}\right) \in \{0,1\}$.

We employ a rectified sigmoid as $h\left(\mathbf{V}_{i,j}\right)$, proposed in (Louizos et al., 2018). The rectified sigmoid is defined as

$$h\left(\mathbf{V}_{i,j}\right) = clip(\sigma\left(\mathbf{V}_{i,j}\right)(\zeta - \gamma) + \gamma, 0, 1), \qquad (23)$$

where $\sigma(\cdot)$ is the sigmoid function and, ζ and γ are stretch parameters, fixed to 1.1 and -0.1, respectively. The rectified sigmoid has non-vanishing gradients as $h\left(\mathbf{V}_{i,j}\right)$ approaches 0 or 1, which helps the learning process when we
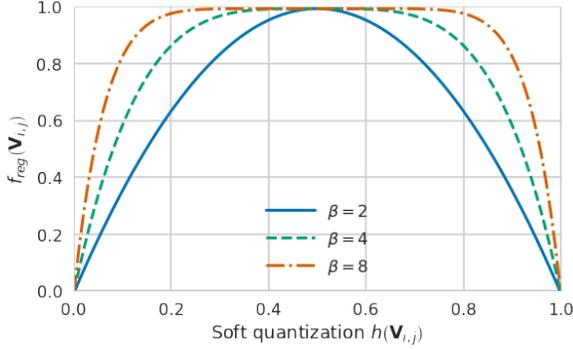
Figure 2. Effect of annealing b on regularization term (24).

Figure 3. Comparison of $h\left(\mathbf{V}_{i,j}\right)$ before (x-axis, corresponding to floating point weights) vs after (y-axis) optimizing (21). We see that all $h\left(\mathbf{V}_{i,j}\right)$ have converged to 0 or 1. Top left and lower right quadrants indicate the weights that have different rounding using (21) vs rounding-to-nearest.

encourage $h\left(\mathbf{V}_{i,j}\right)$ to move to the extremities. For regularization we use

$$f_{reg}\left(\mathbf{V}\right) = \sum_{i,j} 1 - \left|2h\left(\mathbf{V}_{i,j}\right) - 1\right|^{\beta}, \qquad (24)$$

where we anneal the parameter β. This allows most of the $h\left(\mathbf{V}_{i,j}\right)$ to adapt freely in the initial phase (higher β) to improve the MSE and encourages it to converge to 0 or 1 in the later phase of the optimization (lower β), to arrive at the binary solution that we are interested in. The effect of annealing β is illustrated in Fig. 2. Fig. 3 shows how this combination of rectified sigmoid and f_{reg} leads to many weights learning a rounding that is different from rounding to the nearest, to improve the performance, while ultimately converging close to 0 or 1.

This method of optimizing (21) is a specific instance of the general family of Hopfield methods used for binary constrained optimization problems. These types of methods are commonly used as an efficient approximation algorithm for large scale combinatorial problems (Hopfield & Tank, 1985; Smith et al.).

To quantize the whole model, we optimize (21) layer-by-layer sequentially. However, this does not account for the quantization error introduced due to the previous layers. In order to avoid the accumulation of quantization error for deeper networks as well as to account for the activation function, we use the following asymmetric reconstruction formulation

$$\arg\min_{\mathbf{V}} \left\| f_a\left(\mathbf{W}\mathbf{x}\right) - f_a\left(\widetilde{\mathbf{W}}\hat{\mathbf{x}}\right) \right\|_F^2 + \lambda f_{reg}\left(\mathbf{V}\right), \quad (25)$$

where $\hat{\mathbf{x}}$ is the layer's input with all preceding layers quantized and f_a is the activation function. A similar formulation of the loss has been used previously in (Zhang et al., 2016; He et al., 2017), albeit for different purposes. (25) defines our final objective that we can optimize via stochastic gradient descent. We call this algorithm AdaRound, as it adapts to the statistics of the input data as well as to (an approximation of) the task loss. In section 5 we elaborate on the influence of our design choices as well as the asymmetric reconstruction loss on the performance.
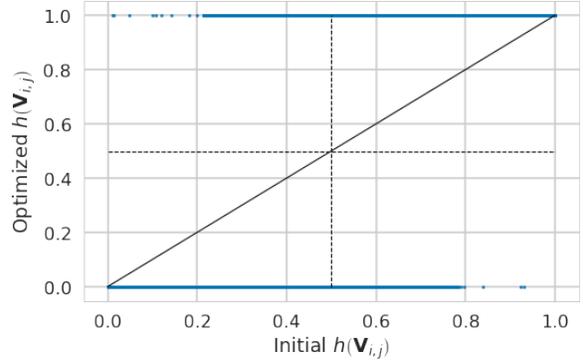
4. Background and related work

In the 1990s, with the resurgence of the field of neural networks, several works designed hardware and optimization methods for running low-bit neural networks on-device. Hammerstrom (1990) created hardware for 8 and 16-bit training of networks, Holi & Hwang (1993) did an empirical analysis on simple neural networks to show that 8 bits are sufficient in most scenarios, and Hoehfeld & Fahlman (1992) developed a stochastic rounding scheme to push neural networks below 8 bits.

More recently, much attention has gone to quantizing neural networks for efficient inference. This is often done by simulating quantization during training, as described in Jacob et al. (2018) and Gupta et al. (2015), and using a straight-through estimator to approximate the gradients. Many methods have since then extended these training frameworks. Choi et al. (2018) learns the activations to obey a certain quantization range, while Esser et al. (2020); Jain et al. (2019) learn the quantization min and max ranges during training so that they do not have to be set manually. Louizos et al. (2019) also learn the grid and formulate a probabilistic version of the quantization training procedure. Uhlich et al. (2020) learn both the quantization grid, and the bit-width per layer, resulting in automatic bit-width selection during training. Works like Kim et al. (2019); Mishra & Marr (2017) exploit student-teacher training to improve quantized model performance during training. Although quantization-aware training is potent and often gives good results, the process is often tedious and time-consuming. Our work seeks to get high accuracy models without this hassle.

Several easy-to-use methods for quantization of networks without quantization-aware training have been proposed as of recent. These methods are often referred to as *post-training quantization* methods. Krishnamoorthi (2018) show several results of network quantization without fine-

tuning. Works like Banner et al. (2019); Choukroun et al. (2019) optimize the quantization ranges for clipping to find a better loss trade-off per-layer. Zhao et al. (2019) improve quantization performance by splitting channels into more channels, increasing computation but achieving lower bit-widths in the process. Lin et al. (2016); Dong et al. (2019) set different bit-widths for different layers, through the information of the per-layer SQNR or the Hessian. Nagel et al. (2019); Cai et al. (2020) even do away with the requirement of needing any data to optimize a model for quantization, making their procedures virtually parameter and data-free. These methods are all solving the same quantization problem as in this paper, and some like Zhao et al. (2019) and Dong et al. (2019) could even be used in conjunction with AdaRound. We compare to the methods that improve weight quantization for 4/8 and 4/32 bit-widths without end-to-end fine-tuning, Banner et al. (2019); Choukroun et al. (2019); Nagel et al. (2019), but leave out comparisons to the mixed-precision methods Cai et al. (2020); Dong et al. (2019) since they improve networks on a different axis.

5. Experiments

To evaluate the performance of AdaRound, we conduct experiments on various computer vision tasks and models. In section 5.1 we study the impact of the approximations and design choices made in section 3. In section 5.2 we compare AdaRound to other post-training quantization methods.

Experimental setup For all experiments we absorb batch normalization in the weights of the adjacent layers. We use symmetric 4-bit weight quantization with a per-layer scale parameter $s^{(\ell)}$ which is determined prior to the application of AdaRound. We set s so that it minimizes the MSE $||\mathbf{W} - \overline{\mathbf{W}}||_F^2$, where $\overline{\mathbf{W}}$ are the quantized weights obtained through rounding-to-nearest. In some ablation studies, we report results when quantizing only the first layer. This will be explicitly mentioned as "First layer". In all other cases, we have the weights of the whole network quantized using 4 bits. Unless otherwise stated, all activations are in FP32. Most experiments are conducted using Resnet18 (He et al., 2016) from torchvision. The baseline performance of this model with full precision weights and activations is 69.68%. In our experiments, we report the mean and standard deviation of the (top1) accuracy on the ImageNet validation set, calculated using 5 runs with different initial seeds. To optimize AdaRound we use 1024 unlabeled images from the ImageNet (Russakovsky et al., 2015) training set, Adam (Kingma & Ba, 2015) optimizer with default hyper-parameters for 10k iterations and a batch-size of 32, unless otherwise stated. We use Pytorch (Paszke et al., 2019) for all our experiments. It is worthwhile to note that the application of AdaRound to Resnet18 takes only 10 minutes on a single Nvidia GTX 1080 Ti.

Rounding	First layer	All layers
Nearest	52.29	23.99
$\mathbf{H}^{(\mathbf{w})}$ task loss (cf. (13))	68.62±0.17	N/A
Local MSE loss (cf. (20))	69.39±0.04	65.83±0.14
Cont. relaxation (cf (21))	**69.58±0.03**	**66.56±0.12**

Table 2. Impact of various approximations and assumptions made in section 3 on the ImageNet validation accuracy (%) for Resnet18. N/A implies that the corresponding experiment was computationally infeasible.

5.1. Ablation study

From task loss to local loss We make various approximations and assumptions in section 3.1 and section 3.2 to simplify our optimization problem. In Table 2, we look at their impact systematically. First, we note that optimizing based on the Hessian of the task loss (cf. (13)) provides a significant performance boost compared to rounding-to-nearest. This verifies that the Taylor expansion based rounding serves as a much better alternative for the task loss when compared to rounding-to-nearest. Similarly, we show that, although moving from the optimization of Taylor expansion of the task loss to the local MSE loss (cf. (20)) requires strong assumptions, it does not degrade the performance. Unlike the Taylor series expansion, the local MSE loss makes it feasible to optimize all layers in the network. We use the cross-entropy method (Rubinstein, 1999) to solve the QUBO problems in (13) and (20), where we initialize the sampling distribution for the binary random variables $\widehat{\mathbf{w}}_i$ as in (Gupta et al., 2015)[2]. Finally, the continuous relaxation for the local MSE optimization problem (cf. (21)) not only reduces the optimization time from several hours to a few minutes but also slightly improves our performance.

Design choices for AdaRound As discussed earlier, our approach to solve (21) closely resembles a Hopfield method. These methods optimize $h(\mathbf{V}_{i,j}) = \sigma\left(\frac{\mathbf{V}_{i,j}}{T}\right)$ with a version of gradient descent with respect to $V_{i,j}$, and annealing the temperature T (Hopfield & Tank, 1985; Smith et al.). This annealing acts as an implicit regularization that allows $h(\mathbf{V}_{i,j})$ to optimize for the MSE loss initially unconstrained, while encouraging $h(\mathbf{V}_{i,j})$ to converge towards 0 or 1 in the later phase of optimization. In Table 3 we show that even after an extensive hyper-parameter search for the annealing schedule of T, using the sigmoid function with our explicit regularization term (24) outperforms the classical method. Using explicit regularization also makes the optimization more stable, leading to lower variance as shown in Table 3. Furthermore, we see that the use of the rectified sigmoid also provides a consistent small improve-

---

[2] In the supplementary material we compare the performance of different QUBO solvers on our problem.

Rounding	First layer	All layers
Sigmoid + T annealing	69.31±0.21	65.22±0.67
Sigmoid + f_{reg}	**69.58±0.03**	66.25±0.15
Rect. sigmoid + f_{reg}	**69.58±0.03**	**66.56±0.12**

Table 3. Impact of different design choices for optimizing (21), on the ImageNet validation accuracy (%) for Resnet18.

Optimization	Acc (%)
Layer wise	66.56±0.12
Asymmetric	68.37±0.07
Asymmetric + ReLU	**68.60±0.09**

Table 4. The influence on the ImageNet validation accuracy (%) for Resnet18, by incorporating asymmetric reconstruction MSE loss and activation function in the rounding optimization objective.

Optimization	Acc (%)
Nearest	23.99
STE	66.63±0.06
AdaRound	**68.60±0.09**

Table 5. Comparison between optimizing (25) using STE (without explicit regularization f_{reg}) vs AdaRound. We report ImageNet validation accuracy (%) for Resnet18.

Grid	Nearest	AdaRound
Min-Max	0.23	61.96±0.04
$\left\|\mathbf{W} - \overline{\mathbf{W}}\right\|_F^2$	23.99	**68.60±0.09**
$\left\|\mathbf{Wx} - \overline{\mathbf{W}}\widehat{\mathbf{x}}\right\|_F^2$	42.89	**68.62±0.08**

Table 6. Comparison between various quantization grids in combination with rounding-to-nearest and AdaRound. We report ImageNet validation accuracy (%) for Resnet18.

ment in accuracy for different models.

Table 4 shows the gain of using the asymmetric reconstruction MSE (cf. section 3.3). We see that this provides a noticeable accuracy improvement when compared to (21). Similarly, accounting for the activation function in the optimization problem provides a small gain.

Optimization using STE Another option we considered is to optimize $\widetilde{\mathbf{W}}$ directly by using the straight-through estimator (STE) (Bengio et al., 2013). This is inspired by quantization-aware training (Jacob et al., 2018), which optimizes a full network with this procedure. We use the STE to minimize the MSE loss in (21). This method technically allows more flexible movement of the quantized weights $\widehat{\mathbf{W}}$, as they are no longer restricted to just rounding up or down. In Table 5 we compare the STE optimization with AdaRound. We can see that AdaRound clearly outperforms STE-based optimization. We believe this is due to the biased gradients of the STE, which hinder the optimization in this restricted setting.

Influence of quantization grid We studied how the choice of weight quantization grid affects the performance gain that AdaRound brings vs rounding-to-nearest. We looked at three different options for determining the scale parameter s; using minimum and maximum values of the weight tensor $\mathbf{W}$, minimizing the MSE $\left\|\mathbf{W} - \overline{\mathbf{W}}\right\|_F^2$ introduced in the weights, and minimizing the MSE $\left\|\mathbf{Wx} - \overline{\mathbf{W}}\widehat{\mathbf{x}}\right\|_F^2$ introduced in the preactivations. $\overline{\mathbf{W}}$ denotes the quantized weight tensor obtained through rounding-to-nearest for a given s. Note, we do not optimize step size and AdaRound jointly as it is non-trivial to combine the two tasks: any change in the step size would result in a different QUBO problem. The results in Table 6 clearly show that AdaRound significantly improves over

rounding-to-nearest, independent of the choice of the quantization grid. Both MSE based approaches are superior to the Min-Max method for determining the grid. Since there is no clear winner between the two MSE formulations for AdaRound, we continue the use of $\left\|\mathbf{W} - \overline{\mathbf{W}}\right\|_F^2$ formulation for all other experiments.

Optimization robustness to data We also investigate how little data is necessary to allow AdaRound to achieve good performance and investigate if this could be done with data from different datasets. The results can be seen in Fig. 4. We see that the performance of AdaRound is robust to the number of images required for optimization. Even with as little as 256 images, the method optimizes the model to within 2% of the original FP32 accuracy. We also see that when using unlabelled images that are from a similar domain but do not belong to the original training data, AdaRound achieves competitive performance. Here, we observe a less than 0.2% degradation on average. It is worthwhile to note that both Pascal VOC and MS COCO only contain a small subset of the classes from Imagenet, implying that the optimization data for AdaRound does not need to be fully representative of the original training set.

5.2. Literature comparison

Comparison to bias correction Several recent papers have addressed a specific symptom of the problem we describe with rounding-to-nearest (Banner et al., 2019; Finkelstein et al., 2019; Nagel et al., 2019). These works observe that quantizing weights often changes the expected value of the output of the layer, i.e., $\mathbb{E}[\mathbf{Wx}] \neq \mathbb{E}[\widehat{\mathbf{W}}\mathbf{x}]$. In order to counteract this, these papers adjust the bias terms for the preactivations by adding $\mathbb{E}[\mathbf{Wx}] - \mathbb{E}[\widehat{\mathbf{W}}\mathbf{x}]$. This "bias correction" can be viewed as another approach to minimize

Optimization	#bits W/A	Resnet18	Resnet50	InceptionV3	MobilenetV2
Full precision	32/32	69.68	76.07	77.40	71.72
DFQ (Nagel et al., 2019)	8/8	69.7	-	-	71.2
Nearest	4/32	23.99	35.60	1.67	8.09
OMSE+opt(Choukroun et al., 2019)	4*/32	67.12	74.67	73.66	-
OCS (Zhao et al., 2019)	4/32	-	66.2	4.8	-
AdaRound	4/32	**68.71±0.06**	**75.23±0.04**	**75.76±0.09**	**69.78±0.05**[†]
DFQ (our impl.)	4/8	38.98	52.84	-	46.57
Bias corr (Banner et al., 2019)	4*/8	67.4	74.8	59.5	-
AdaRound w/ act quant	4/8	**68.55±0.01**	**75.01±0.05**	**75.72±0.09**	**69.25±0.06**[†]

Table 7. Comparison among different post-training quantization strategies in the literature. We report results for various models in terms of ImageNet validation accuracy (%). *Uses per-channel quantization. [†]Using CLE (Nagel et al., 2019) as preprocessing.
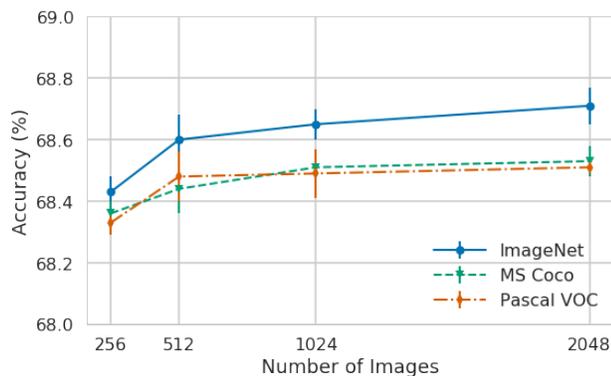

Figure 4. The effect on ImageNet validation accuracy when using different number of images belonging to different datasets for AdaRound optimization.

Rounding	Acc(%)
Nearest	23.99
Bias correction	38.87
AdaRound	**68.60±0.09**

Table 8. Comparison between AdaRound and empirical bias correction, which also counteracts a symptom of the quantization error introduced by rounding to nearest. We report ImageNet validation accuracy (%) for Resnet18.

the same MSE loss as AdaRound (20), but by adjusting the bias terms as

$$\mathbb{E}\left[\mathbf{W}\mathbf{x}\right] - \mathbb{E}[\widehat{\mathbf{W}}\mathbf{x}] = \underset{\widehat{\mathbf{b}}}{\arg\min}\,\mathbb{E}\left[\left\|\left\|\mathbf{W}\mathbf{x} - \left(\widehat{\mathbf{W}}\mathbf{x} + \widehat{\mathbf{b}}\right)\right\|\right\|_F^2\right].$$
(26)

Our method solves this same problem, but in a better way. In Table 8 we compare empirical bias correction from Nagel et al. (2019) to AdaRound, under the exact same experimental setup, on ResNet18. While bias correction improves performance over vanilla quantization without bias correction, we see that for 4 bits it only achieves 38.87% accuracy, where AdaRound recovers accuracy to 68.60%.

ImageNet In Table 7, we compare AdaRound to several recent post-training quantization methods. We use the same experimental setup as described earlier, with the exception of optimizing AdaRound with 2048 images for 20k iterations. For both Resnet18 and Resnet50, AdaRound is within 1% of the FP32 accuracy for 4-bit weight quantization and outperforms all competing methods, even though some rely on the more favorable per-channel quantization

and do not quantize the first and the last layer. Similarly, on the more challenging networks, InceptionV3 and MobilenetV2, AdaRound stays within 2% of the original accuracy and outperforms any competing method.

To be able to compare to methods that also do activation quantization, we report results of AdaRound with all activation tensors quantized to 8 bits. For this scenario, we quantized the activations to 8 bits and set the scaling factor for the activation quantizers based on the minimum and maximum activations observed. We notice that activation quantization, in most cases, does not significantly harm the validation accuracy. AdaRound again outperforms the competing methods such as DFQ (Nagel et al., 2019) and bias correction (Banner et al., 2019).

Semantic segmentation To demonstrate the wider applicability of AdaRound, we apply it to DeeplabV3+ (Chen et al., 2018) evaluated on Pascal VOC (Everingham et al., 2015). Since the input images here are significantly bigger, we only use 512 images to optimize AdaRound. All other aspects of the experimental setup stay the same. To the best of our knowledge, there are no other post-training quantization methods doing 4-bit quantization for semantic segmentation. DFQ works well for 8 bits, however performance drastically drops when going down to 4-bit weight quantization. AdaRound still performs well for 4 bits and has only a 2% performance decrease for 4-bit weights and

Optimization	#bits W/A	mIOU
Full precision	32/32	72.94
DFQ (Nagel et al., 2019)	8/8	72.33
Nearest	4/8	6.09
DFQ (our impl.)	4/8	14.45
AdaRound	4/32	70.89±0.33
AdaRound w/ act quant	4/8	70.86±0.37

Table 9. Comparison among different post-training quantization strategies, in terms of Mean Intersection Over Union (mIOU) for DeeplabV3+ (MobileNetV2 backend) on Pascal VOC.

8-bit activations quantization.

6. Conclusion

In this paper we proposed AdaRound, a new rounding method for post-training quantization of neural network weights. AdaRound improves significantly over rounding-to-nearest, which has poor performance for lower bit widths. We framed and analyzed the rounding problem theoretically and by making appropriate approximations we arrive at a practical method. AdaRound is computationally fast, uses only a small number of unlabeled data examples, does not need end-to-end fine-tuning, and can be applied to any neural network that has convolutional or fully-connected layers without any restriction. AdaRound establishes a new state-of-the-art for post-training weight quantization with significant gains. It can push networks like Resnet18 and Resnet50 to 4-bit weights while keeping the accuracy drop within 1%.

References

Banner, R., Nahshan, Y., and Soudry, D. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Neural Information Processing Systems (NeuRIPS)*, 2019.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Botev, A., Ritter, H., and Barber, D. Practical gauss-newton optimisation for deep learning. *International Conference on Machine Learning (ICML)*, 2017.

Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. Zeroq: A novel zero shot quantization framework. *arXiv preprint arXiv:2001.00281*, 2020.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. *The European Conference on Computer Vision (ECCV)*, 2018.

Choi, J., Wang, Z., Venkataramani, S., Chuang, P. I., Srinivasan, V., and Gopalakrishnan, K. PACT: parameterized clipping activation for quantized neural networks. *arXiv preprint arxiv:805.06085*, 2018.

Choukroun, Y., Kravchik, E., and Kisilev, P. Low-bit quantization of neural networks for efficient inference. *International Conference on Computer Vision (ICCV)*, 2019.

Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. HAWQ: hessian aware quantization of neural networks with mixed-precision. *International Conference on Computer Vision (ICCV)*, 2019.

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. Learned step size quantization. *International Conference on Learning Representations (ICLR)*, 2020.

Everingham, M., Eslami, S., Van Gool, L., Williams, C., Winn, J., and Zisserman, A. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 1 2015.

Finkelstein, A., Almog, U., and Grobman, M. Fighting quantization bias with bias. *arXiv preprint arxiv:1906.03193*, 2019.

Guo, Y. A survey on methods and theories of quantized neural networks. *arXiv preprint: arxiv:1808.04752*, 2018.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. *International Conference on Machine Learning, ICML*, 2015.

Hammerstrom, D. A vlsi architecture for high-performance, low-cost, on-chip learning. *International Joint Conference on Neural Networks (IJCNN)*, 1990.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition, CVPR*, 2016.

He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. *International Conference on Computer Vision (ICCV)*, 2017.

Hoehfeld, M. and Fahlman, S. E. Learning with limited numerical precision using the cascade-correlation algorithm. *IEEE Transactions on Neural Networks*, 3(4):602–611, 1992.

Holi, J. L. and Hwang, J. N. Finite precision error analysis of neural network hardware implementations. *IEEE Trans. Comput.*, 42(3):281–290, 1993.

Hopfield, J. J. and Tank, D. W. "neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52(3):141–152, 1985.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Jain, S. R., Gural, A., Wu, M., and Dick, C. Trained uniform quantization for accurate and efficient neural network inference on fixed-point hardware. *arxiv preprint arxiv:1903.08066*, 2019.

Kim, J., Bhalgat, Y., Lee, J., Patel, C., and Kwak, N. QKD: quantization-aware knowledge distillation. *arxiv preprint arxiv:1911.12491*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference for Learning Representations (ICLR)*, 2015.

Kochenberger, G., Hao, J.-K., Glover, F., Lewis, M., Lü, Z., Wang, H., and Wang, Y. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28(1):58–81, Jul 2014.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Lin, D. D., Talathi, S. S., and Annapureddy, V. S. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, 2016.

Louizos, C., Welling, M., and Kingma, D. P. Learning sparse neural networks through l_0 regularization. *International Conference on Learning Representations (ICLR)*, 2018.

Louizos, C., Reisser, M., Blankevoort, T., Gavves, E., and Welling, M. Relaxed quantization for discretized neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Mishra, A. K. and Marr, D. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *arXiv preprint arxiv:1711.05852*, 2017.

Nagel, M., van Baalen, M., Blankevoort, T., and Welling, M. Data-free quantization through weight equalization and bias correction. *International Conference on Computer Vision (ICCV)*, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NeuRIPS)*. 2019.

Rubinstein, R. The cross-entropy method for combinatorial and continuous optimization. *Methodology And Computing In Applied Probability*, 1(2):127–190, Sep 1999.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Smith, K. A., Palaniswami, M., and Krishnamoorthy, M. Neural techniques for combinatorial optimization with applications. *IEEE Trans. Neural Networks*, 9(6):1301–1318.

Stock, P., Joulin, A., Gribonval, R., Graham, B., and Jégou, H. And the bit goes down: Revisiting the quantization of neural networks. In *International Conference on Learning Representations*, 2020.

Uhlich, S., Mauch, L., Yoshiyama, K., Cardinaux, F., García, J. A., Tiedemann, S., Kemp, T., and Nakamura, A. Mixed precision dnns: All you need is a good parametrization. *International Conference on Learning Representations (ICLR)*, 2020.

Wang, P., Hu, Q., Zhang, Y., Zhang, C., Liu, Y., and Cheng, J. Two-step quantization for low-bit neural networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4376–4384, 2018.

Zhang, X., Zou, J., He, K., and Sun, J. Accelerating very deep convolutional networks for classification and detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(10):1943–1955, 2016.

Zhao, R., Hu, Y., Dotzel, J., Sa, C. D., and Zhang, Z. Improving neural network quantization without retraining using outlier channel splitting. *International Conference on Machine Learning, ICML*, 2019.