# Self-Supervised Quantization of Pre-Trained Neural Networks for Multiplierless Acceleration

Sebastian Vogel*, Jannik Springer*†, Andre Guntoro*, Gerd Ascheid†
*Robert Bosch GmbH, Renningen, Germany
†RWTH Aachen University, Aachen, Germany
{sebastian.vogel, fixed-term.jannik.springer, andre.guntoro}@de.bosch.com, gerd.ascheid@ice.rwth-aachen.de

***Abstract*—To host intelligent algorithms such as Deep Neural Networks on embedded devices, it is beneficial to transform the data representation of neural networks into a fixed-point format with reduced bit-width. In this paper we present a novel quantization procedure for parameters and activations of pre-trained neural networks. For 8 bit linear quantization, our procedure achieves close to original network performance without retraining and consequently does not require labeled training data. Additionally, we evaluate our method for power-of-two quantization as well as for a two-hot quantization scheme, enabling shift-based inference. To underline the hardware benefits of a multiplierless accelerator, we propose the design of a shift-based processing element.**

***Index Terms*—Quantization, Neural Networks, Hardware, Multiplierless Acceleration**

## I. Introduction

Since Deep Neural Networks (DNNs) achieve super human-level performance on tasks such as image classification [1], they have gained extraordinary attention not only from a machine learning point of view, but also from a hardware perspective. DNNs are compute intensive and benefit from dedicated accelerators for efficient and effective mobile application. A variety of dedicated hardware accelerators has been presented [2]–[4], where a majority focuses on architectural improvements rather than on optimization of the mathematical processing elements (PEs). However, nearly all approaches benefit from a data representation with reduced bit-width. As for training of DNNs typically GPUs are being used, the resulting data representation is Float32. Therefore, we focus in this paper on the transition of pre-trained DNNs to fixed-point variants.

We make the following contributions with this paper:

- We formulate a novel quantization procedure for pre-trained DNNs and evaluate it on a variety of networks for two tasks: classification and semantic segmentation.
- We investigate the effects of our proposed procedure for 8 bit linear quantization of weights, biases, and activations.
- Additionally, we evaluate our procedure for data representations enabling multiplierless acceleration of DNNs.
- We show that networks quantized with our procedure achieve near to original performance without the need for fine-tuning. Therefore, it does not require labeled training data.

This paper is structured as follows: The next section summarizes related work. Section III introduces mathematical principles of DNNs and discusses three quantization schemes. In Section IV we propose a novel quantization procedure. Our results are presented and discussed in Section V.

## II. Related Work

To benefit from reduced bit-widths, many works focus on quantized training [5]–[7]. These methods strongly intervene in the training procedure and cannot be applied to pre-trained neural networks. In [8] a quantization procedure for pre-trained networks based on arbitrary step size and an arbitrary zero-point is proposed. However, to handle the nonsymmetric quantization, additional online calculation is needed (see equation (8) in [8]). Although weights and activations are quantized to 8 bit, biases are represented with 32 bit. The results are reported after a fine-tuning phase. Some approaches use individual bit-widths for different layers and parameters, such as [9]. The drawbacks of such methods are additional hardware requirements regarding individual bit-width handling. Approaches evaluating multiplierless quantization schemes are reported in [10]–[13]. All of them require a fine-tuning step to recover from significant performance drop and none of them investigates complex tasks such as semantic segmentation. In [14] a quantization scheme enabling multiplierless acceleration similar to ours is proposed. However, the authors evaluate their method on rather simple classification tasks and do not provide a detailed quantization procedure for pre-trained networks. Instead, a retraining procedure is proposed. Yet, especially for mapping tools where no training data is available, fine-tuning is not an option. Additionally, their proposed PEs require massive multiplexers in a hardware implementation.

## III. Methodology

### A. Deep Neural Networks

Typical Deep Neural Networks consist of convolutions, pooling operations, and potentially fully-connected layers. The network is built by stacking a number of layers, where the output $y^{(l)}$ of a layer $l$ is input $x^{(l+1)}$ to a following layer $l+1$. Convolutional and fully-connected layers can be expressed as weighted sums, followed by nonlinearity functions $\Phi(\cdot)$ including a bias term $b$:

$$y^{(l)} = \Phi\left(\sum w^{(l)} x^{(l)} + b^{(l)}\right) \tag{1}$$

978-3-9819263-2-3/DATE19/©2019 EDAA

State-of-the art networks typically feature one of the following piecewise linear functions $\Phi(\cdot)$:

$$\begin{aligned}\Phi(x) &= \text{ReLU}(x) = \max(x, 0) \text{ or} \\ \Phi(x) &= \text{sign}(x).\end{aligned} \tag{2}$$

For training of particularly *Deep* Neural Networks, BatchNorm layers may be used [15]. After training, the parameters of these layers can be merged into the weights and biases of preceding convolutional or fully-connected layers [8].

### B. Quantization

In this section we discuss three quantization schemes which enable embedded systems to efficiently deploy neural networks: linear quantization, power-of-two quantization, and a novel generalized quantization scheme for hardware efficient computation using bit-shifts.

*1) Linear Quantization:* Equation (3) describes the quantization function $lin\_quant_N : x \mapsto x_q$ for a signed data representation, where $clip(x, a, b)$ denotes the clipping function to restrict value $x$ to the interval $[a, b]$, and two quantization parameters $N$ the bit-width and $\Delta$ the step size.

$$x_q = \text{clip}\left(\text{round}\left(\frac{x}{\Delta}\right),\ -2^{N-1},\ 2^{N-1}-1\right)\Delta \tag{3}$$

An example is given in Fig. 1 where original data distributions are depicted alongside with quantized variants. The supporting points of a linear quantization scheme are shown with green circles.

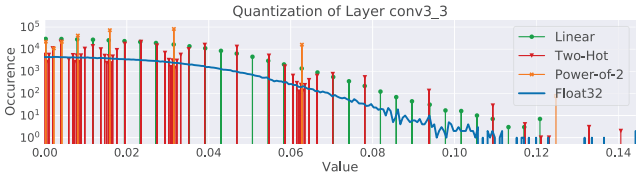
*2) Power-of-Two Quantization:* For the multiplication $a \cdot b$, where the operand $b$ is constrained to a power-of-two value $b \in \{2^n | n \in \mathbb{N}_0\}$, a hardware implementation can be efficiently realized by a simple bit-shift $a \ll \log_2(b)$. Equation (4) shows the mapping $pow2\_quant_N : x \mapsto x_q$ to quantize value $x$ using a signed power-of-two scheme with $N$ bits.

$$\begin{aligned}\hat{x}_q &= \text{clip}\left(\text{round}\left(\log_2\left|\frac{x}{\Delta}\right|\right),\ 0,\ 2^{N-1}\right) - 2 \\ x_q &= \text{sign}(x)\,\text{round}\left(2^{\hat{x}_q}\right)\Delta\end{aligned} \tag{4}$$

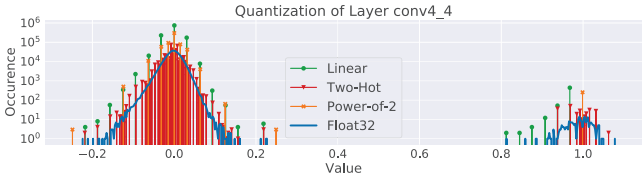
In (4) we subtract 2 after clipping, because we need a separate code for the zero-value of $x_q$. In case $\hat{x}_q = -2$, the value $x_q$ will be rounded to 0 in round $\left(2^{\hat{x}_q}\right)$. For a hardware implementation value $x_q$ will be saved as $\hat{x}_q$ using $N$ bits including the sign bit of $x_q$. This helps to effectively exploit the available bit size.

The resulting supporting points for a power-of-two quantized data distribution are shown in Fig. 1 (marked with orange crosses). It can be seen that the number of sampling points diminishes towards high values.

*3) Two-Hot Quantization:* While from a hardware perspective power-of-two quantization offers efficiency and area advantages, it has a reduced resolution towards high values. To provide an increased resolution while maintaining the hardware advantages of bit-shifts, the factor $b$ can be replaced by two power-of-two values $c, d \in \{2^n | n \in \mathbb{N}_0\}$ where each




(a) Positive range of weights of a layer with unimodal data distribution.




(b) Weights of a dilated convolutional layer with multimodal distribution.

Fig. 1. Quantization of the weights of convolutional layers of a neural network. Linear, power-of-two, and two-hot quantization is shown.

is using half of the bit-width $N$. The multiplication $a \cdot b$ can then be expressed as

$$a \cdot (c + d) = a \ll \log_2(c) + a \ll \log_2(d), \tag{5}$$

which still offers considerable energy and area advantages for a bit-shift-based hardware implementation. The binary representation of value $b = c + d$ carries at most two '1's. We therefore call the quantization scheme "Two-Hot Quantization."

To determine $x_q$, value $x$ is transformed into a power-of-two value $x_q^{MSB}$ based on the $pow2\_quant$ function defined in (4) using half bit-width. A second value $x_q^{LSB}$ is derived from the remainder $x_\delta$:

$$\begin{aligned}x_q^{MSB} &= \frac{pow2\_quant_{N/2}(x)}{\Delta} \\ x_\delta &= \frac{x}{\Delta} - x_q^{MSB} \\ x_q^{LSB} &= \frac{pow2\_quant_{N/2}(x_\delta)}{\Delta}.\end{aligned} \tag{6}$$

The proposed quantization function $two\_hot\_quant_N : x \mapsto x_q$ is then:

$$x_q = \Delta\left(2^\zeta x_q^{MSB} + x_q^{LSB}\right),\ \zeta \in \mathbb{N}_0. \tag{7}$$

We introduce parameter $\zeta$ as it offers an additional degree of freedom and does not contribute to hardware cost once it is fixed. $2^\zeta$ corresponds to a fixed bit-shift of one summand. Instead of saving the two-hot-coded value $x_q^{MSB} + x_q^{LSB}$, the exponent values $\hat{x}_q^{MSB}$, $\hat{x}_q^{LSB}$ will be saved each using $N/2$ bits, where both may include a sign bit. Our two-hot quantization is a generalized representation of the 1-alphabet introduced in [14].

The sampling pattern for our proposed quantization scheme is depicted in Fig. 1 with red triangles. As can be seen, the resolution is increased towards high values due to additional supporting points in the neighborhood of power-of-two values.

## IV. Fixed-point Transformation of Pre-Trained DNN

### A. Quantization Optimization

While the bit-width $N$ is based on a hardware architectural decision, step size $\Delta$ is data dependent. In this section we first discuss two straight-forward approaches to determine the optimum step sizes for activations, biases, and weights. Then, we introduce a novel optimization procedure which we found to be especially suitable for neural networks.

*1) Based on Maximum Absolute Value:* The first approach derives the step size $\Delta$ from the maximum absolute value in a data distribution. Let $\chi$ represent values from a random data distribution. Then, $\Delta_\chi$ can be written as

$$\Delta_\chi = \frac{\max\left(|\chi|\right)}{2^{N-1}} - 1 \ . \tag{8}$$

For linear quantization, this procedure will introduce strong quantization noise in case of far outliers. This approach is used in [8] to quantize weights.

*2) Based on Minimum Mean Squared Error:* When quantizing values $\chi$ with a quantization scheme $quant_N(\chi, \Delta)$, where $N$ is the bit-width and $\Delta$ is the step size, the resulting quantization error can be written as an additive term $\delta_\chi$:

$$\chi_q = quant_N\left(\chi, \Delta_\chi\right) = \chi + \delta_\chi. \tag{9}$$

To reduce quantization noise, the following equation determines step size $\Delta_\chi$ by minimizing the $L^2$-norm of the error vector and hence minimizes the mean squared error (MSE) resulting from the quantization:

$$\Delta_\chi = \underset{\Delta_\chi}{\operatorname{argmin}}\left(\|\chi - \chi_q\|_2\right) = \underset{\Delta_\chi}{\operatorname{argmin}}\left(\|\delta_\chi\|_2\right). \tag{10}$$

This approach has been similarly proposed in [16] to quantize activations.

*3) Based on propagated Quantization Error:* Here, we propose to determine the step size $\Delta$ based on the propagation of the quantization error $\delta$ through the neural network. We write $\tilde{y}$ for the output of a layer with quantized weights $w_q$:

$$\begin{aligned} w_q &= quant_N\left(w, \Delta_w\right) = w + \delta_w, \\ \tilde{y} &= \Phi\left(\sum x w_q + b\right), \\ y &= \Phi\left(\sum x w + b\right). \end{aligned} \tag{11}$$

To determine the optimal step size $\Delta_w$, we minimize the propagated quantization error (propQE) at the output of the layer:

$$\Delta_w = \underset{\Delta_w}{\operatorname{argmin}}\big(\| \underbrace{\tilde{y} - y}_{propQE} \|_2\big). \tag{12}$$

For the step size $\Delta_x$ of quantized input activations $x$ follows:

$$\begin{aligned} \tilde{y} &= \Phi\left(\sum x_q w + b\right), \\ \Delta_x &= \underset{\Delta_x}{\operatorname{argmin}}\big(\|\tilde{y} - y\|_2\big). \end{aligned} \tag{13}$$

This procedure is also used for finding a suitable quantization step size $\Delta_b$ for biases.

Our procedure weights the quantization error with the parameters, respectively activations, of the pre-trained network. While determining the step sizes for a specific layer, we do not quantize any other layer of the network to prevent the influence of other quantization noise on the optimization. For the proposed process, we do not need any training labels. We use the intermediate values of the network based on valid input samples to supervise the quantization procedure. Therefore, we call this approach "self-supervised."

### B. Fixed-Point Transformation

Once the quantization step sizes $\Delta_x, \Delta_w, \Delta_b$ are determined for activations, weights, and biases for each layer, we transform the network into a fixed-point version. Therefore, we round the step sizes $\Delta$ to the next power-of-two value $\hat{\Delta} \in \{2^z | z \in \mathbb{Z}\}$. This allows a fixed-point multiply-accumulate operation where the input values $x$ are multiplied with weights $w$ and the bias $b$ is added after a bit-shift:

$$\sum x_q w_q + \left(b_q \ll \left(\log_2 \hat{\Delta}_b - \log_2 \hat{\Delta}_x - \log_2 \hat{\Delta}_w\right)\right). \tag{14}$$

## V. Evaluation

We evaluate our proposed quantization procedure on a variety of networks and data sets. For an image classification task we use the ILSVRC data set [22]. The data set consists of more than 1 million RGB-images each labeled with one out of 1,000 categories. We use a random subset of training images at a resolution of $224 \times 224$ for our quantization procedure and report our results on the validation set of 50,000 images. Reported performance figures are top-1 (highest rated label correct) and top-5 accuracy (correct classification among five highest rated labels). The networks for classification are VGG16 [17], ResNet50 [18], and InceptionResNetV4 [19].

To include a more complex task, we also report results for semantic scene segmentation in which each input pixel of a given image is labeled with a certain class. As a benchmark, we use the Cityscapes data set [23]. It consists of 5,000 RGB-images with fine-grained annotations for training and contains a validation set of 500 images. The images have a resolution of $2048 \times 1024$. We determine the quantization step size on a subset of training images and report results on the validation set. The network performance of two models (Dilated Model [20] and FCN8s [21]) is reported using mean intersection over union (mIoU) and mean pixel accuracy (pix. acc.) [24]. For the experiments with Dilated Model [20] we use $4\times$-downsampled Cityscapes images with a resolution of $512 \times 256$.

In this section we first compare three procedures for determining quantization step sizes: based on maximum absolute value, minimal MSE, and minimal propagated quantization error. We then evaluate the power-of-two and two-hot quantization schemes for weights of pre-trained networks which enable multiplierless (i.e. bit-shift-based) acceleration. No fine-tuning of network parameters takes place after quantization. We do not use any annotations of images. All of our experiments are conducted using $N = 8$ bit.

TABLE I

PERFORMANCE OF CLASSIFICATION NETWORKS (VGG16, RESNET50, AND INCEPTIONNETV4) AND NETWORKS FOR SEMANTIC SCENE SEGMENTATION (DILATED MODEL AND FCN8S) WITH $N = 8$ BIT QUANTIZED ACTIVATIONS FOR DIFFERENT OPTIMIZATION STRATEGIES (MAX ABS, MIN MSE, MIN PROPQE) TO DETERMINE THE STEP SIZE $\Delta_y^{(l)}$ OF EACH LAYER. FURTHER INCREASING THE NUMBER OF IMAGES DOES NOT INCREASE PERFORMANCE.

| Quantization | Quantization Optimization | VGG16 [17] | | ResNet50 [18] | | InceptionResNetV4 [19] | | Dilated Model [20] | | FCN8s [21] | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | top-1 [%] | top-5 [%] | top-1 [%] | top-5 [%] | top-1 [%] | top-5 [%] | mIoU [%] | pix. acc. [%] | mIoU [%] | pix. acc. [%] |
| Number of training images used | | 100 | | 100 | | 100 | | 36 | | 10 | |
| Float32 Baseline | – | 69.58 | 89.04 | 72.99 | 90.93 | 75.61 | 92.48 | 55.63 | 92.85 | 66.48 | 94.65 |
| $y_q$ lin quant | max abs | 66.36 | 88.82 | 64.75 | 86.69 | 0.00 | 0.02 | 51.70 | 91.14 | 64.68 | 93.41 |
| $y_q$ lin quant | min MSE | 68.51 | 88.79 | 70.08 | 88.95 | 69.66 | 89.40 | 54.23 | 92.00 | 65.04 | 93.29 |
| $y_q$ lin quant | min propQE | 69.09 | 88.97 | 71.31 | 90.61 | 73.89 | 91.67 | 55.65 | 92.79 | 66.49 | 94.46 |
| $y_q(propQE)$ vs baseline | – | -0.49 | -0.07 | -1.68 | -0.32 | -1.72 | -0.81 | +0,02 | -0.06 | +0.01 | -0.19 |

### A. Linear Quantization of Activations

In Table I the results for linear quantization of activations is presented. Quantization step sizes are determined based on three different procedures (max abs, min MSE, and min propQE). For these experiments we kept weights and biases at Float32 precision. The number of input images to create stable and reproducible results is reported.

Distributions of activations tend to have an increased number of outliers. Our method (propQE) helps to correctly clip outliers and effectively use the dynamic range of a linear quantization scheme especially for activations. Particularly, the activation values of InceptionResNetV4 have far outliers which leads to a complete failing of the network after quantizing the activations using max abs. The quantization of some of the networks leads to an increased performance. We suppose this effect emerges from a better generalization due to regularizing effects of the quantization procedure.

### B. Linear Quantization of Weights and Biases

When searching for the optimal quantization step size for parameters, we use the original network with no quantization of activations. This prevents interference with quantization noise from activations. As a consequence, the order of the step size analysis is irrelevant. For the experiments, Batch-Norm layers have been merged into the convolutional layer parameters $w$ and $b$ [8]. Hence, no separate quantization of Batch-Norm layers is necessary.

For the results reported in Table II, we jointly apply parameter and activation quantization, where the quantization of activations is based on minimal propQE. Our proposed procedure performs best compared to max abs and min MSE. When comparing the results for minimal propQE with Table I, we see that parameter quantization does not further decrease the network performance.

We found that it is important to introduce separate quantization parameters for biases. An identical quantization of weights and biases (i.e. $\Delta_b^{(l)} = \Delta_w^{(l)}$) using 8 bit significantly reduces accuracy. While from a hardware perspective an identical quantization of biases and activations (i.e. $\Delta_b^{(l)} = \Delta_y^{(l)}$) is easy to implement, the networks performed poorly under this constraint. For all of our experiments we allowed individual step sizes for activations, weights, and biases.

### C. Power-of-two Quantization of Weights

We also investigate the impact of power-of-two quantization on the network performance. The results are shown in Table III. While our approach of minimizing the propagated quantization error works better than using maximum absolute values, the networks experience a strong performance drop with power-of-two quantization. ResNet50 fails completely and the impact on InceptionResNetV4 is remarkable. We suppose this observation emerges from the specific data representation offered by the quantization scheme: Power-of-two quantization exhibits an extremely high dynamic range $DR_{p2}$:

$$\mathrm{DR}_{p2} = 20 \log_{10} \left( 2^{2^{N-1}-2} \right) \approx \left( 2^{N-1} - 2 \right) 6.02 \text{ dB}. \quad (15)$$

Hence, it either over-resolves values close to zero or offers unused quantization steps outside of the data distribution. Especially for multimodal distributions (typical for layers with dilated convolutions [20]), power-of-two quantization offers few sampling points for modes not centered at zero (see Fig. 1 (b)).

### D. Two-Hot Quantization of Weights

Two-hot quantization offers a significantly different resolution than power-of-two quantization. While it still enables a hardware efficient implementation of multiplications, it also provides a well suited nonuniform quantization for representing weights of neural networks. Two-hot quantization can be regarded as a power-of-two quantization with additional quantization steps in the neighborhood of each supporting point (see Fig. 1).

The results of our experiments are shown in Table III. We observe a performance drop of less than 1.5% for top-1 accuracy compared to a linear quantization scheme (first line). Even for the rather complex task of semantic segmentation, the performance drop is negligible (<0.5% mIoU drop). Specifically, our linear quantization as well as two-hot quantization outperforms previous work on the FCN8s network which included a fine-tuning step [12]. The dynamic range $DR_{2h}$ of this quantization scheme is in the order of $2^{N/2}$ smaller than $DR_{p2}$ for the same bit-width $N$:

$$\mathrm{DR}_{2h} \approx \left( 2^{\frac{N}{2}-1} - 2 + \log_2(1 + 2^{\zeta}) \right) 6.02 \text{ dB}. \quad (16)$$

TABLE II

WE EVALUATE THREE QUANTIZATION ANALYSIS PROCEDURES FOR LINEAR QUANTIZATION OF PARAMETERS: MAX ABS, MIN MSE AND MIN PROPQE. THE RESULTS REPORTED HERE ARE BASED ON NETWORKS WITH QUANTIZED ACTIVATIONS $y_q$ (SEE MIN PROPQE IN TABLE I). WE USE $N = 8$ BIT.

| Quantization | Quantization Optimization | VGG16 [17] | | ResNet50 [18] | | InceptionResNetV4 [19] | | Dilated Model [20] | | FCN8s [21] | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | top-1 [%] | top-5 [%] | top-1 [%] | top-5 [%] | top-1 [%] | top-5 [%] | mIoU [%] | m. pix. acc. [%] | mIoU [%] | m. pix. acc. [%] |
| Number of training images used | | 100 | | 100 | | 100 | | 36 | | 10 | |
| Float32 Baseline | – | 69.58 | 89.04 | 72.99 | 90.93 | 75.61 | 92.48 | 55.63 | 92.85 | 66.48 | 94.65 |
| $w_q$, $b_q$ lin quant | max abs | 68.72 | 88.86 | 70.73 | 90.35 | 73.52 | 91.43 | 55.33 | 92.74 | 66.29 | 94.40 |
| $w_q$, $b_q$ lin quant | min MSE | 68.88 | 88.91 | 70.92 | 90.44 | 73.44 | 91.44 | 55.53 | 92.75 | 66.37 | 94.41 |
| $w_q$, $b_q$ lin quant | min propQE | 69.12 | 89.06 | 71.67 | 90.73 | 73.71 | 91.57 | 55.62 | 92.78 | 66.47 | 94.44 |
| $w_q, b_q(propQE)$ vs baseline | – | -0.46 | +0.02 | -1.32 | -0.20 | -1.90 | -0.91 | -0.01 | -0.07 | -0.01 | -0.21 |

TABLE III

THREE QUANTIZATION SCHEMES (LINEAR, POWER-OF-TWO, AND TWO-HOT QUANTIZATION) FOR WEIGHTS ARE COMPARED. WE QUANTIZE ACTIVATIONS AND BIASES USING LINEAR QUANTIZATION, OPTIMIZED BY MIN PROPQE. ($N = 8$ BIT)

| Quantization | Quantization Optimization | VGG16 [17] | | ResNet50 [18] | | InceptionResNetV4 [19] | | Dilated Model [20] | | FCN8s [21] | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | top-1 [%] | top-5 [%] | top-1 [%] | top-5 [%] | top-1 [%] | top-5 [%] | mIoU [%] | m. pix. acc. [%] | mIoU [%] | m. pix. acc. [%] |
| Number of training images used | | 100 | | 100 | | 100 | | 36 | | 10 | |
| lin-quant Baseline | min propQE | 69.12 | 89.06 | 71.67 | 90.73 | 73.71 | 91.57 | 55.62 | 92.78 | 66.47 | 94.44 |
| $w_q$ power-of-2 quant | max abs | 58.55 | 81.12 | 0.70 | 2.36 | 24.83 | 47.65 | 30.13 | 72.87 | 55.19 | 88.58 |
| $w_q$ power-of-2 quant | min propQE | 63.85 | 86.76 | 0.69 | 2.66 | 37.77 | 64.55 | 49.52 | 90.13 | 60.75 | 92.10 |
| $w_q$ 2-hot quant, $\zeta = 0$ | min propQE | 68.82 | 89.51 | 70.24 | 90.05 | 71.99 | 90.74 | 55.23 | 92.71 | 66.21 | 94.40 |
| $w_q$ 2-hot quant, $\zeta = 2$ | min propQE | 68.91 | 89.54 | 70.84 | 90.35 | 72.47 | 91.11 | 55.34 | 92.74 | 66.24 | 94.41 |
| two-hot vs lin quant baseline | – | -0.21 | -0.52 | -0.83 | -0.38 | -1.24 | -0.46 | -0.28 | 0.04 | -0.23 | -0.03 |

Compared to the fixed dynamic range of a power-of-two scheme, $DR_{2h}$ is dependent on the bit-shift parameter $\zeta$ introduced in (7). We conducted experiments with varying $\zeta$ and found the highest performance for $\zeta = 2$ for all networks. Therefore, a hardware implementation can be realized where the addition of the products $x \ll \hat{w}_q^{MSB}$ and $x \ll \hat{w}_q^{LSB}$ incorporates a fixed bit-offset by 2.

Dilated convolutions used in the Dilated Model typically exhibit a multimodal distribution as shown in Fig. 1 (b). Two-hot quantization offers a higher resolution for the second mode than power-of-two quantization and we assume that consequently a better performance can be seen.

### E. Hardware

To underline the advantages of the two-hot quantization scheme, we propose dedicated processing elements using two-hot quantized weights. These PEs are based on bit-shifts and hence enable an efficient hardware acceleration of the inference of Deep Neural Networks. An overall architecture of hardware accelerators is out of scope of this paper. Here, we merely focus on the arithmetic cores. The design of the proposed PE including an accumulator stage is depicted in Fig. 2 (a). For comparison, a multiplier-based PE plus accumulator register is shown in Fig. 2 (b). The shift-based engine uses two 3-bit values of $W$ and shifts the input value $X$ accordingly. The bit-adjusting parameter $\zeta$ is implemented as a fixed bit-shift resulting in an addition with bit-offset, respectively. The results are added or subtracted depending on the first sign bit. Based on the second sign bit, the value is added or subtracted to the accumulator register (acc reg). For

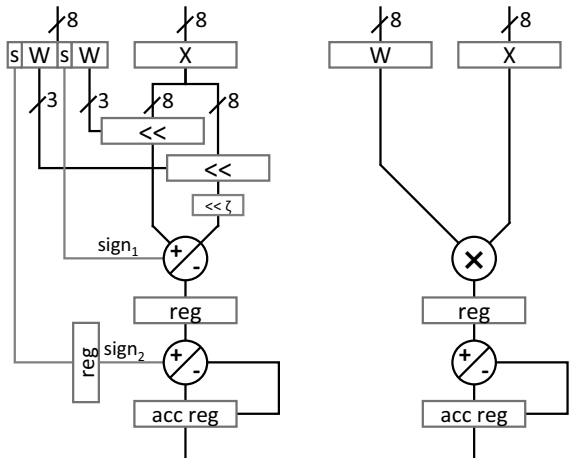



Fig. 2. Two designs for PEs are depicted. (a) shows a PE for two-hot quantized weights. (b) shows a PE for linear quantized weights.

the multiplier design, weights and activations are represented in two's complement representation. In contrast, both 4-bit values of the two-hot weight-composite are represented in a sign-magnitude representation and only the activation value is represented in two's complement format.

Fig. 3 shows an exemplary output of the Dilated Model using linear quantization for activations and biases with (a) linear quantization and with (b) two-hot quantization for weights. The mutual difference is shown in Fig. 3 (c).

(a) (b) (c)

Fig. 3. Exemplary output of the quantized Dilated Model using linear quantized activations and biases with (a) linear quantized (mIoU 55.62%) and with (b) two-hot quantized weights (mIoU 55.34%) for $N = 8$ bit. In (c), differently labeled pixels between (a) and (b) are marked black.

## VI. Conclusion

In this paper, we present a novel quantization procedure for pre-trained neural networks with little to no performance drop for $N = 8$ bit while not fine-tuning nor retraining the network. We evaluate the procedure on several networks for classification as well as semantic segmentation. Additionally, we show that two-hot quantization for weights performs nearly as good as a linear quantized variant. To the best of our knowledge, we are the first to present a multiplierless quantization scheme without fine-tuning while achieving close to original network performance. Therefore, our procedure is a well-suited candidate for DNN quantization and mapping tools with no access to labeled training data. Additionally, we propose the design of processing elements with two-hot quantized weights for the implementation in dedicated DNN accelerators.

## References


[1] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ser. ICCV '15. Washington, DC, USA: IEEE Computer Society, 2015, pp. 1026–1034.

[2] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '15*. ACM Press, 2015.

[3] S. Yin *et al.*, "A high energy efficient reconfigurable hybrid neural network processor for deep learning applications," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 968–982, apr 2018.

[4] S. Han *et al.*, "Eie: Efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 243–254.

[5] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *CoRR*, vol. abs/1606.06160, 2016.

[6] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems (NIPS) 29*, 2016, pp. 4107–4115.

[7] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 3123–3131.

[8] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[9] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 2849–2858.

[10] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *CoRR*, vol. abs/1603.01025, 2016.

[11] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*, 2017, pp. 5900–5904.

[12] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

[13] H. Tann, S. Hashemi, R. I. Bahar, and S. Reda, "Hardware-software codesign of accurate, multiplier-free deep neural networks," in *Proceedings of the 54th Annual Design Automation Conference 2017 on - DAC '17*. ACM Press, 2017.

[14] S. S. Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy, "Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 145–150.

[15] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, 2015, pp. 448–456.

[16] J. Qiu *et al.*, "Going deeper with embedded FPGA platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '16*. ACM Press, 2016.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2017, pp. 4278–4284.

[20] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015. [Online]. Available: http://arxiv.org/abs/1511.07122

[21] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3431–3440.

[22] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, apr 2015.

[23] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] G. Csurka, D. Larlus, and F. Perronnin, "What is a good evaluation measure for semantic segmentation?" in *Procedings of the British Machine Vision Conference 2013*. British Machine Vision Association Press, 2013.