Efficient Hardware Acceleration of CNNs using Logarithmic Data Representation with Arbitrary log-base

Sebastian Vogel
Robert Bosch GmbH
Renningen, Germany
sebastian.vogel@de.bosch.com

Mengyu Liang
Technical University of Munich
Munich, Germany
gu47liy@mytum.de

Andre Guntoro
Robert Bosch GmbH
Renningen, Germany
andre.guntoro@de.bosch.com

Walter Stechele
Technical University of Munich
Munich, Germany
walter.stechele@tum.de

Gerd Ascheid
RWTH Aachen University
Aachen, Germany
ascheid@ice.rwth-aachen.de

ABSTRACT

Efficient acceleration of Deep Neural Networks is a manifold task. In order to save memory requirements and reduce energy consumption we propose the use of dedicated accelerators with novel arithmetic processing elements which use bit shifts instead of multipliers. While a regular power-of-2 quantization scheme allows for multiplierless computation of multiply-accumulate-operations, it suffers from high accuracy losses in neural networks. Therefore, we evaluate the use of powers-of-arbitrary-log-bases and confirmed their suitability for quantization of pre-trained neural networks. The presented method works without retraining of the neural network and therefore is suitable for applications in which no labeled training data is available. In order to verify our proposed method, we implement the log-based processing elements into a neural network accelerator on an FPGA. The hardware efficiency is evaluated in terms of FPGA utilization and energy requirements in comparison to regular 8-bit-fixed-point multiplier based acceleration. Using this approach hardware resources are minimized and power consumption is reduced by 22.3%.

CCS CONCEPTS

• **Hardware → Hardware accelerators**; • **Computing methodologies** → *Scene understanding*; *Neural networks*; • **Computer systems organization** → *Real-time system architecture*;

ACM Reference Format:
Sebastian Vogel, Mengyu Liang, Andre Guntoro, Walter Stechele, and Gerd Ascheid. 2018. Efficient Hardware Acceleration of CNNs using Logarithmic Data Representation with Arbitrary log-base. In *IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN (ICCAD '18), November 5–8, 2018, San Diego, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3240765.3240803

ICCAD '18, November 5–8, 2018, San Diego, CA, USA

ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3240803

1 INTRODUCTION

In recent years, a variety of tasks has been solved above human level performance using Deep Neural Networks (DNNs), such as image classification [8]. As more and more attention is paid to these algorithms, many applications of DNNs appear, such as the application in Advanced Driver Assistance Systems (ADASs), surveillance systems, and smart phones. While web-based applications may rely on a powerful high-performance back end, applications in energy constrained mobile devices profit from efficient embedded accelerators. A variety of hardware accelerators for DNNs has been proposed [2, 3, 7], only some focus on the efficiency of the underlying mathematical processing elements. We therefore focus in this paper on the efficient data representation for DNN algorithms. Especially the data representation in logarithmic scale is promising as it allows for the use of barrel shifters instead of bulky multipliers.

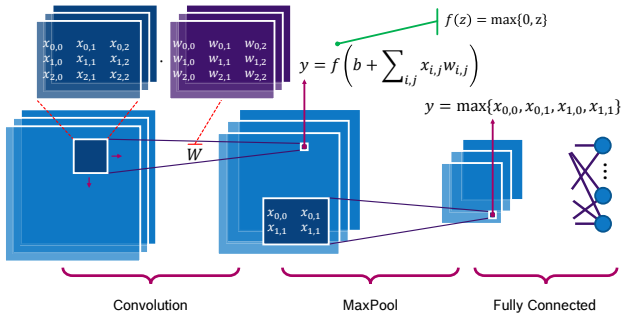


Figure 1: A typical CNN consists of convolutional layers, subsampling (pooling) layers, and potentially fully connected layers. Convolutional layers extract features from the input image. Pooling layers are used to reduce the network size towards the output. For classification tasks fully connected layers predict a respective class label.

With this paper we make the following contributions:

- We propose a novel self-supervised quantization scheme which preserves pre-trained network performance without the need for fine-tuning.
- We evaluate a non-uniform low bit-width quantization which allows hardware-friendly bit shift based processing.

- We develop a novel arithmetic processing element which processes logarithmic-based data with arbitrary log-base.
- We evaluate the proposed arithmetic Processing Element (PE) on a hardware accelerator for DNNs.

The paper is structured as follows: Section 2 presents related work concerning low bit-width and power-of-2 quantization schemes. In Section 3 we describe a linear and a logarithmic based quantization scheme as well as an optimization method to determine quantization parameters. Section 4 discusses the performance of the proposed quantization methods on several networks and tasks before Section 5 goes into details of a dedicated Neural Network Accelerator (NNA) incorporating our proposed data representation. We conclude our findings in Section 6.

2 RELATED WORK

In order to reduce memory requirements of DNNs many techniques have been proposed. While some procedures propose to change the architecture of the network [9, 10] others propose heavy data quantization. In [5, 16] dedicated training methods for CNNs have been proposed with network parameters being constrained to two values (-1, +1). This results in a strong network compression and reduces complexity of respective hardware accelerators [1]. However, these methods require a specialized training procedure and therefore cannot be used on pre-trained neural networks. Furthermore, the methods have not yet been adapted to complex networks and tasks, such as image segmentation. In [15] the quantization of network parameters and intermediate results using power-of-2 is proposed. By this approach not only the computation complexity is reduced but also data quantization can be reduced to a few bit. Furthermore, in [13] the quantization of network parameters using 4 Bit power-of-2 quantization is proposed. As the performance of the CNN drops using this quantization method, the authors retrain the network to restore the previous network accuracy. Additionally, the authors evaluate their proposed approach on a hardware accelerator. While this approach reduces the size of network parameters to a few bit, the intermediate results still require higher bandwidth and memory size. An approach of quantizing pre-trained neural networks has been presented in [6]. However, this procedure requires labeled training data.

3 METHODOLOGY

3.1 DNNs

DNNs typically consist of a number of stacked convolutions, subsampling functions, and optionally fully connected layers (see Figure 1). Mathematically convolutional layers as well as fully connected layers can be expressed as a weighted sum of input values x with weight parameters w. In order to introduce a non-linearity into neural network layers, a bias b and a so called activation function $\Phi(\cdot)$ is typically used. Therefore, a simplified expression for the output y of a neural network layer can be expressed as

$$y = \Phi\left(\sum xw + b\right). \tag{1}$$

A commonly used activation function is the ReLU-function. It sets negative values to 0 and acts as the identity function for positive values (see Equation (2)).

$$\Phi(x) = \mathrm{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

In order to reduce the network size towards the output, subsampling (pooling) functions can be used. These pooling functions usually propagate a single value out of several neighboring values and hence reduce the layer size. A popular pooling function is the maximum function which chooses the highest value out of a number of values.

$$y = \max(x_1, x_2, \ldots, x_n) \tag{3}$$

Neural networks are usually trained on high-performance Graphics Processing Units (GPUs) with a data representation of float 32b. When deploying such a neural network into a resource constrained embedded system, an efficient data representation is profitable. In the following subsections we will discuss possible quantization methods resulting in efficient data representations for embedded systems.

3.2 Linear Quantization

Most efficient embedded computing systems incorporate fixed-point instead of floating-point multipliers to reduce power consumption as well as chip area. Fixed-point multipliers require a data representation with linear quantization. Equation (4) describes the mapping of a value x with potentially high resolution (e.g. float 32b) onto a value x_q using signed linear quantization. The quantization parameters for a linear scheme are the bit-width bw and the full-scale range ($\mathrm{fsr} = 2^{\mathrm{FSR}}$). The full-scale range corresponds to the interval which is divided into 2^{bw} steps. The step size is defined in Equation (5).

$$x_q = \mathrm{clip}\left(\mathrm{round}\left(\frac{x}{step}\right) step, -2^{FSR}, 2^{FSR} - step\right) \tag{4}$$

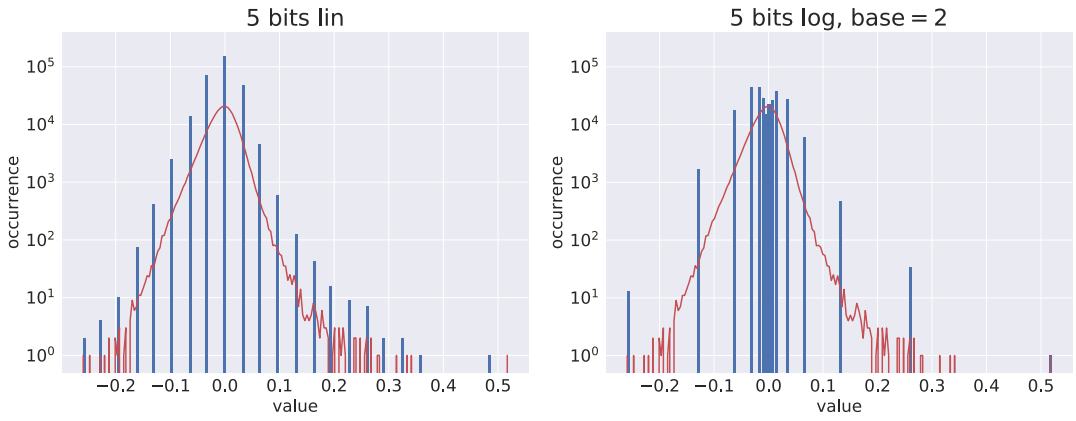
$$step = 2^{FSR+1-bw} \tag{5}$$

$$clip(x, min, max) = \begin{cases} max, & x \geq max \\ x, & min < x < max \\ min, & otherwise \end{cases} \tag{6}$$

Figure 2a visualizes linear quantization using 5 Bit.

3.3 Logarithmic-based Quantization

When quantizing non-uniform data distributions, a non-linear quantization method might be applicable. Most neural network weights w as well as activation values y are non-uniformly distributed. Therefore, we consider the use of a logarithmic-scaled quantization. Equation (7) describes the mapping of an unquantized value x onto a logarithmically quantized value x_q. In addition to the quantization parameters FSR and bw, this mapping has another degree of freedom: the log-base a. In Figure 2b the power-of-2 (i.e. log-base $a = 2$) quantization of a typical weight distribution is illustrated. Instead of saving and calculating with value x_q, the value $\hat{x}_q = \log_a(|x_q|)$ alongside with the sign of x_q are used. Storing this value requires bw Bits. Equation (7) takes a sign bit as well as a code for the zero-value into account. In our design the Most Significant Bit (MSB)



(a) Linear quantization **(b) Power-of-2 quantization**

Figure 2: Linear (a) vs. power-of-2 (b) quantization of a non-uniform data distribution using 5 Bit.

stores the sign of x_q and the values $[110\ldots0]_b$ and $[010\ldots0]_b$ represent the zero value.

For $a = 2$ this quantization methods corresponds to the classical power-of-2 quantization. When using log-bases $a < 2$ the resulting quantized data distribution includes an inherent pruning effect. This can be seen in Figures 3a and 3b for base $a = 2^{1/2}$ and $a = 2^{1/4}$ respectively.

$$\begin{aligned} x_q &= \text{sign}(x)\, a^{\hat{x}_q} \\ \hat{x}_q &= \text{clip}\left(\text{round}\left(\log_a |x| - FSR\right), 2 - 2^{bw-1}, 0\right) + FSR \end{aligned} \tag{7}$$

3.4 Convolution with log-scaled Data

Most state-of-the-art CNNs use ReLU-activations. Therefore, the activation values y are non-negative and can be quantized using Equation (7) where $sign(\cdot)$ is ignored and no separate sign bit is necessary. Concerning network weights Equation (7) can be directly used.

Assuming quantized activations and weights Equation (1) results in the following form.

$$\tilde{y} = \Phi\left(\sum x_q w_q + b\right) \tag{8}$$

We denote the log-bases for x_q as a_x and for w_q as a_w respectively. Then the product in Equation (8) can be written as

$$\begin{aligned} p_q = x_q w_q &= \text{sign}(w_q)\, a_x^{\hat{x}_q} a_w^{\hat{w}_q} \\ &= \text{sign}(w_q)\, 2^{\log_2(a_x)\hat{x}_q + \log_2(a_w)\hat{w}_q}. \end{aligned} \tag{9}$$

From a hardware perspective especially those log-bases $a_{w,x}$ are appealing, which fulfill

$$\log_2(a_{w,x}) = 2^{-z_{w,x}}, \text{ where } z_{w,x} \in \mathbb{Z}. \tag{10}$$

In that case the exponent in Equation (9) consists of a bit-aligned addition. We now assume $z_x \geq z_w$ and define

$$\Delta z := |z_x - z_w|. \tag{11}$$

With this definition we rewrite Equation (9) to

$$\begin{aligned} x_q w_q &= \text{sign}(w_q)\, 2^{\hat{p}_q}, \text{ where} \\ \hat{p}_q &= 2^{-z_x}\left(\hat{x}_q + 2^{\Delta z}\hat{w}_q\right). \end{aligned} \tag{12}$$

From Equation (12) we can conclude that the multiplication of two logarithmic quantized numbers x_q and w_q consists of an addition $(\hat{x}_q + 2^{\Delta z}\hat{w}_q)$ and a bitshift $(2^{\hat{p}_q})$. When $z_x = 0$ the value $\hat{p}_q$ is an integer number and hence the equations describe a simple power-of-2 scheme. However, when $z_x > 0$ value $\hat{p}_q$ results from a bitshift to the right and thus $\hat{p}_q$ has z_x fractional bits.

In order to get a close-to-hardware description of Equation (12), we define the following functions and operator

$$\begin{aligned} Int(v) &:= \text{integer part of value } v, \\ Frac(v) &:= \text{fractional part of value } v, \\ LUT(k) &:= \text{lookup table entry } k, \\ v << b &:= \text{bitshift left of } v \text{ by } b \text{ bits}, \end{aligned} \tag{13}$$

and rewrite Equation (12):

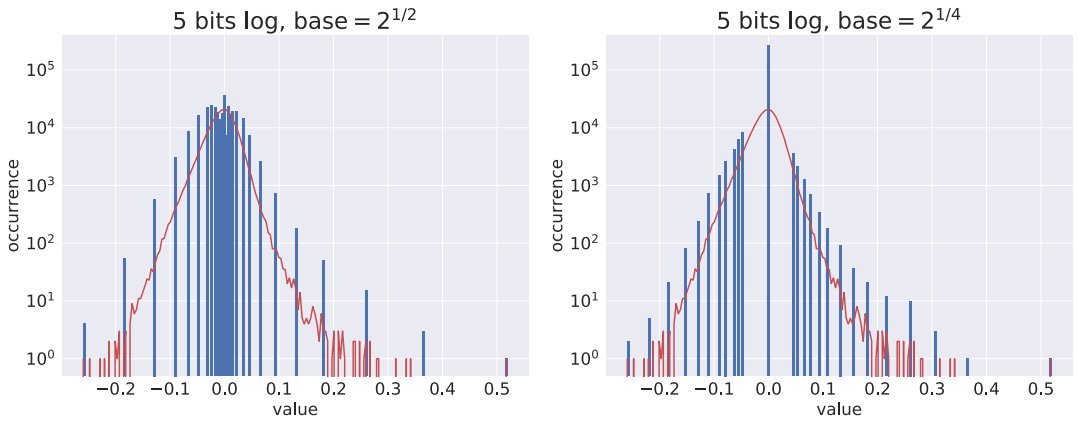
$$\begin{aligned} x_q w_q &= \text{sign}(w_q)\, 2^{Int(\hat{p}_q) + Frac(\hat{p}_q)} \\ &= \text{sign}(w_q)\, 2^{Frac(\hat{p}_q)} \cdot 2^{Int(\hat{p}_q)} \\ &= \text{sign}(w_q)\, \left(LUT(Frac(\hat{p}_q)) << Int(\hat{p}_q)\right). \end{aligned} \tag{14}$$

In order to avoid the computation of $2^{Frac(\hat{p}_q)}$ in hardware, we propose to use a small lookup table with 2^N entries. N is equal to the number of fractional bits of $\hat{p}_q$ and can be derived from Equations (9) and (12):

$$N = \max(z_x, z_w). \tag{15}$$

3.5 Optimization-Scheme

In order to find the best logarithmic base for weights and activations in each layer as well as the right choice of the full scale range parameter FSR, we propose to minimize the Propagated Quantization Error (PQE). Propagated Quantization Error is the resulting error at the output y of a layer of a neural network due to the quantization



(a) Logarithmic quantization with base $2^{1/2}$

(b) Logarithmic quantization with base $2^{1/4}$

Figure 3: Logarithmic quantization with base $2^{1/2}$ (a) and base $2^{1/4}$ (b) using 5 Bit. With decreasing log-base the quantization step size around 0 increases and the granularity at higher values becomes more fine-grained.

error added to the input x of a layer. To evaluate the PQE we use up to 10 batches of training images. We first produce the original intermediate values for each layer. In a second step we evaluate the PQE for each layer under different log-base settings. Both, log-base a_x as well as a_w are to be found by minimizing PQE.

$$\begin{aligned} a &= \underset{a}{\operatorname{argmin}} \|y - \tilde{y}\|_2 \\ a_x &= \underset{a_x}{\operatorname{argmin}} \left\| \sum xw - \sum x_q w \right\|_2 \\ a_w &= \underset{a_w}{\operatorname{argmin}} \left\| \sum xw - \sum x w_q \right\|_2 \end{aligned} \tag{16}$$

In contrast to minimizing the quantization error ΔQ itself, this procedure takes the weight parameters w, respectively x, into account.

$$\begin{aligned} x_q &= x + \Delta Q_x \\ w_q &= w + \Delta Q_w \end{aligned} \tag{17}$$

From Equation (16) follows

$$\begin{aligned} a_x &= \underset{a_x}{\operatorname{argmin}} \left\| \sum \Delta Q_x w \right\| \text{ and} \\ a_w &= \underset{a_w}{\operatorname{argmin}} \left\| \sum x \Delta Q_w \right\|. \end{aligned} \tag{18}$$

The optimization method we propose here does not require finetuning nor retraining of the network after quantization. Furthermore, input data without labels is sufficient as we rely on the data distributions in the network. We therefore, call this optimization scheme self-supervised.

4 EVALUATION

4.1 Image Classification

4.1.1 *AllCNN on CIFAR-10.*

We evaluate the power-of-2 quantization on a lightweight neural network trained as a classifier on the CIFAR-10 dataset[11]. The CIFAR-10 dataset contains 60,000 RGB-images at resolution 32×32 categorized into 10 labeled classes. Our classification results are reported on the test set comprising 10,000 images. The network under consideration has been presented in [18] and consists of 5 convolutional layers with kernel size 3×3 and 2 convolutional layers with kernel size 1×1. The number of network parameters is around 950,000. As can be seen in Table 1 this lightweight network performs well under low bit-width quantization down to 5 Bit linear quantization. When using 4 Bit representation the linear quantization scheme does not perform better than random guessing. For the power-of-2 quantization the network accuracy degrades by only 4.32%. Activation values as well as network weights are quantized using log-base 2 in this experiment.

Table 1: Performance of a quantized CNN for classification of the CIFAR-10 dataset [11]. Comparison of original network performance with low bit-width linear quantization and power-of-2 quantization.

AllCNN [18]	float 32b	lin 5b	lin 4b	$\log_2$ 5b	$\log_2$ 4b
top-1 acc. [%]	88.72	84.68	10.27	85.27	84.40

4.1.2 *VGG16 on ImageNet.*

Classification of the CIFAR-10 dataset is a rather simple task when compared to classification of the ImageNet dataset [12]. The ImageNet dataset contains more than 1 million labeled RGB-images categorized into 1,000 classes. The network we consider for this task is VGG16 proposed by Simonyan and Zisserman in 2014 [17]. It is trained to classify RGB-images at a resolution of 224×224 and consists of 13 convolutional layers with kernel size 3×3 and 3 fully connected layers. The total number of parameters of this network is about 138 million. The pre-trained network with float 32b data representation achieves a top-1 accuracy of 67.53%. Using a power-of-2 quantization scheme for weights and activations reduces this performance by more than 10%. However, when using a log-based quantization method for network weights where the log-base is chosen to be $a = \sqrt{2}$ (see Equation (7)), the top-1 accuracy reduces by 2.72% to 64.81%. We believe this performance difference results

from the higher resolution the log-base-$\sqrt{2}$ quantization provides at higher values compared to log-base-2 quantization (compare Figure 3). The results of this experiment are summarized in Table 2.

Table 2: Performance of a quantized DNN for classification of the ImageNet dataset [12]. Comparison of original network performance with power-of-2 quantization and logarithmic scaled quantization using log-base $\sqrt{2}$.

VGG16 [17]	float 32b	w: $\log_2$ 5b x: $\log_2$ 5b	w: $\log_{\sqrt{2}}$ 5b x: $\log_2$ 5b
top-1 acc. [%]	67.53	56.77	64.81
top-5 acc. [%]	87.99	80.59	86.28

4.2 Semantic Segmentation

Especially for autonomous driving and modern ADASs Deep Neural Networks provide a promising technology in order to get human-level performance for visual environment perception. A challenging task in the field of visual perception is the pixel-wise labeling of high resolution images. The task of labeling each pixel of an image is called semantic segmentation. We evaluate two DNNs trained for semantic segmentation on the Cityscapes dataset [4]. From this dataset we use a subset of the 5,000 fine-grained labeled RGB-images for quantization and report our results on the validation set which consists of 500 images. There are 30 different class labels available of which we take 19 for our experiments. We report the performance in terms of pixel accuracy – i.e. the portion of correctly labeled pixels – and mean Intersection over Union (mIoU). Mean IoU takes the size of labeled areas into account and therefore better reveals small performance degradations.

4.2.1 *FCN-8s on Cityscapes.*

A well-known CNN for semantic segmentation is a fully-convolutional network (FCN) presented by Long et al. in 2016 [14]. This network is based on the convolutional part of VGG16 with 13 convolutional layers. Additionally, the FCN has trailing upsampling layers to reproduce an output resolution equal to the input resolution of the network. The authors suggested to add convolutional layers parallel to the main processing path of the CNN to increase the segmentation performance. The network configuration we consider here, is referred to as FCN-8s [14] and has a about 143 million parameters. The float 32b baseline performance in our experiments is 64.48% mIoU and 93.88% pixel accuracy. In Table 3 the results of our quantization experiments with this network are given. Quantizing the activation values only, using power-of-2 values (i.e. log-base $a_x = 2$) degrades the network performance significantly. However, relying on log-base $a_x = 2^{1/4}$ for activations reduces mIoU by 0.4% and pixel accuracy by 0.11% only. Concerning the weight parameters we found the same log-base $a_w = 2^{1/4}$ to work well. The resulting network performance using 5 Bit quantization with logarithmic scaled data representation is 63.16% mIoU and 93.65% pixel accuracy.

Table 3: Performance of a quantized CNN for semantic segmentation of the Cityscapes dataset [4]. Comparison of original network performance with logarithmic scaled quantization using various log-bases.

FCN-8s [14]	w: x:	float 32b float 32b	float 32b $\log_2$ 5b	float 32b $\log_{2^{1/4}}$ 5b	$\log_{2^{1/4}}$ 5b $\log_{2^{1/4}}$ 5b
mIoU [%]		64.48	57.94	64.08	63.16
pix. acc. [%]		93.88	91.98	93.77	93.65

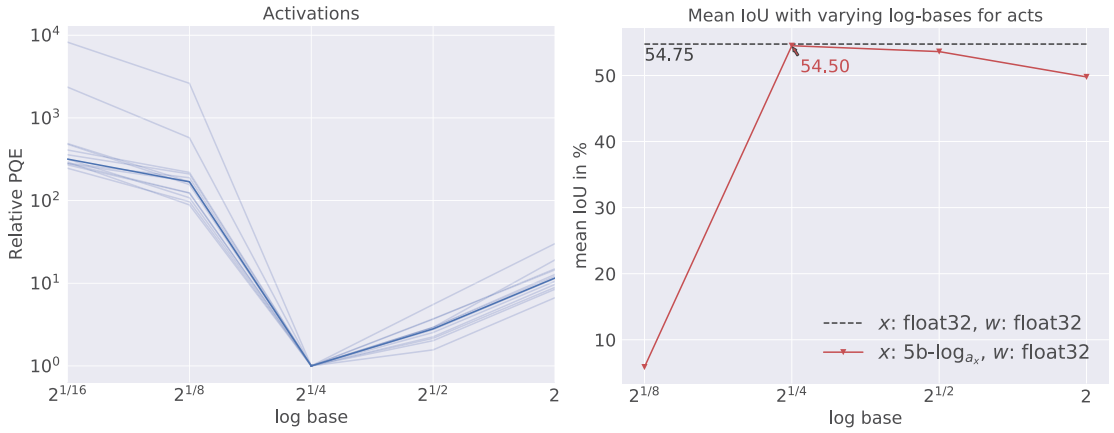
4.2.2 *Dilated Model on Cityscapes.*

Various approaches have been proposed for CNN architectures to achieve best results for semantic segmentation. An architecture which avoids parallel processing paths, in contrast to the FCN-8s network, is the Dilated Model proposed by Yu and Koltun in 2016 [19]. We train a network using the Dilated Model architecture on a downscaled variant of the Cityscapes dataset. The image resolution for our experiment is 512×256. The network we use has 7 regular convolutional layers and 6 convolutional layers with dilation rates {2, 2, 4, 4, 8, 8}. The dilated convolution has a wider receptive field while not increasing the number of parameters. While our network does not achieve the performance of the FCN-8s network on the high resolution images, it has a total of 5.3 million parameters and thus is more than 25× smaller than the FCN-8s network. Therefore, we regard this network as an appropriate candidate for evaluating our hardware accelerator.

In Section 3.5 we presented an optimization scheme for finding the optimal log-base by evaluating the Propagated Quantization Error. We determine the PQE for this network in each layer regarding various log-bases of activations. The normalized PQE is depicted in Figure 5a. For this model the minimal PQE is obtained for log-base $a_x = 2^{1/4}$ in each layer. In Figure 5b the corresponding network performance in terms of mean Intersection over Union can be seen when varying the log-base for all layers. From Figure 5 we conclude that the PQE correlates with the performance of the network. The optimal log-base for network weights is obtained similarly. The evaluation results corresponding to various log-bases a_w of network weights and the respective network performance are depicted in Figure 6.

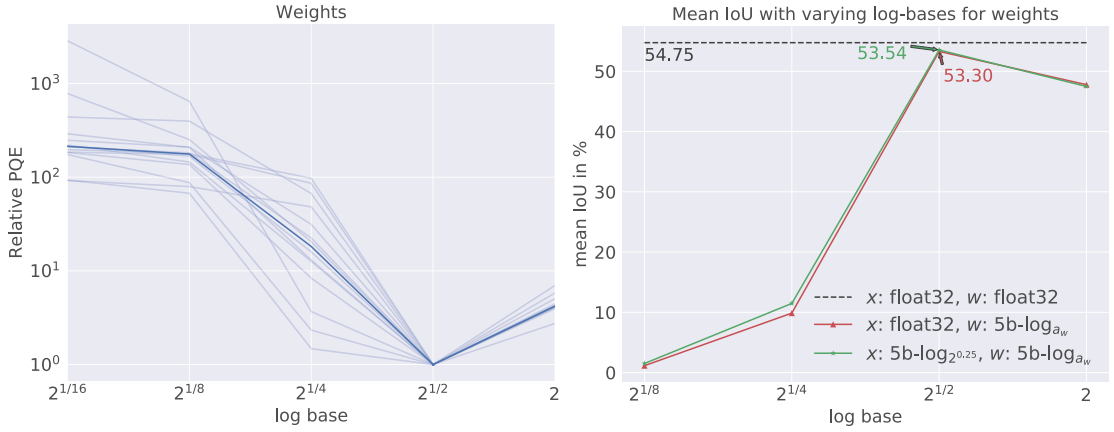
An example output of the logarithmic quantized network using 5 Bit for activations and weights is shown in Figure 4, overlayed with the input image.

Figure 4: Output example of the Dilated Model on the Cityscapes dataset using 5b log-based data representation.



(a) Normalized PQE **(b) mIoU**

Figure 5: Normalized Propagated Quantization Error (a) and mIoU (b) at varying log-base a_x of activations.



(a) Normalized PQE **(b) mIoU**

Figure 6: Normalized Propagated Quantization Error (a) and mIoU (b) at varying log-base a_w of weights.

Table 4: Performance of a quantized CNN for semantic segmentation of the Cityscapes dataset [4]. Comparison of original network performance with logarithmic scaled quantization using various log-bases.

Dilated Model [14] w: / x:	float 32b / float 32b	float 32b / $\log_{2^{1/4}}$ 5b	$\log_{2^{1/2}}$ 5b / float 32b	$\log_{2^{1/2}}$ 5b / $\log_{2^{1/4}}$ 5b
mIoU [%]	54.75	54.50	53.30	53.54
pix. acc. [%]	91.80	91.49	91.21	91.56

5 NEURAL NETWORK ACCELERATOR

5.1 Hardware Architecture

The design of our Neural Network Accelerator (NNA) is based on the parallel computation of convolutions. The NNA processes each layer of a neural network separately. Therefore, the configuration of a layer is loaded and the processing elements are configured accordingly. We instantiate 256 PEs in parallel and feed data through an input buffer and results are stored to an output buffer. As the internally available BRAM size of the Xilinx Virtex 7 FPGA (7VX485T) cannot store the complete network, we load and store data through a memory controller from and to external DDR3 memory. This generic architecture is used to evaluate the efficiency enhancement when replacing fixed-point-based PEs with logarithmic-based PEs.

5.2 Processing Elements

The Processing Elements represent the computational cores for the accelerated processing of a neural network. Therefore, an optimization of PEs will directly effect the efficiency of the overall hardware design.

5.2.1 *Linear PE.*

The design of PEs using fixed-point multiplication is illustrated in Figure 8 (b). The 8 Bit input values w and x are multiplied and

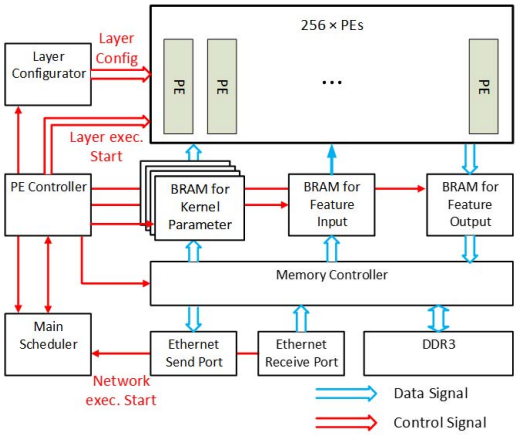


Figure 7: Top level hardware architecture of the Neural Network Accelerator

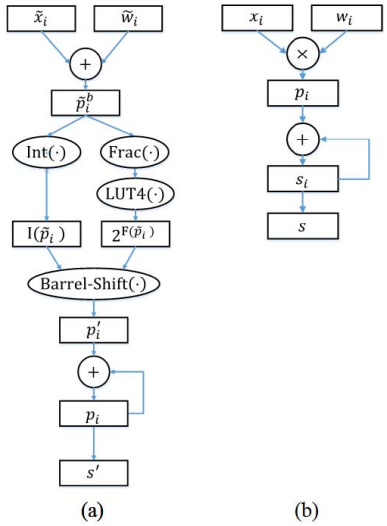


Figure 8: (a) shows a Processing Element for logarithmic quantized data. This PE uses an adder, a lookup table, and a barrel shifter before accumulating the result. (b) shows the multiply-accumulate engine for linear quantized data.

summed up. The accumulation output register has a bit-width of 28 Bits to be able to accumulate up to 4096 values without overflow. Finally, a quantization module extracts an output value of 8 Bit once the convolution is processed. This output value is stored to memory.

5.2.2 Logarithmic PE.
The logarithmic PE processes the input values $\hat{x}_q$ and $\hat{w}_q$ according to Equations (12) and (14). The corresponding design is shown in Figure 8. First the values are added. Afterwards, the fractional part of the addition result is extracted and used as a pointer to an entry in a lookup table. The number of entries in this lookup table is defined by Equation (15). For our experiment, we deploy the Dilated Model for semantic segmentation on our NNA. As analyzed in Section 4, the number of LUT entries is $2^N = 2^{z_a} = 4$ for this model. We found a size of 6 Bits to be sufficient for each LUT entry. As the optimal log-base is identical for all layers (see Figures 5 and 6), we do not need to reconfigure the logarithmic PE during run-time.

The accumulation value p_q results in linear data representation (see Equation (9)). Therefore, we implement a quantization module, which transforms the accumulation result into a 5 Bit log-scaled value. The hardware overhead for this quantization module is negligible.

5.2.3 FPGA Utilization for PEs.
In Table 5 the resulting resource requirements for both PEs are given when (automatic) DSP instantiation is switched off during synthesis. While the number of FPGA LUTs is reduced for log-based PE compared to linear PE, the number of required registers increases. This results from the additional register stages added into the logarithmic computation.

The overall utilization of the NNA is summerized in Table 6. The numbers reported are again based on no DSP use for processing elements. The resulting DSP-slices we instantiate are used for address generation.

Table 5: Hardware utilization of a Processing Element implemented without DSP-slices on Xilinx Virtex 7 FPGA (7VX485T)

PE	Linear PE 8 Bit	Logarithmic PE 5 Bit
#LUTs	189	103
#registers	83	112

Table 6: Hardware utilization of the Neural Network Accelerator implemented on Xilinx FPGA Virtex 7 (7VX485T). The instantiated DSP slices are used for address generation.

Accelerator	with lin. PE 8 Bit	with log. PE 5 Bit
#LUTs	45,290	29.405
#register	32,623	30,850
#BRAM	740	740
#DSP slices	9	9

5.3 Power Measurement

In order to fairly compare the fixed-point multiplier approach using a linear quantization method with the logarithmic quantization approach, we implement two designs where the Processing Elements are constrained to not implement the Virtex7 built-in and highly-optimized fixed-point multipliers of DSP-slices. With this procedure, we want to make sure the logic overhead of fixed-point multipliers compared to barrel shifters is taken into account when measuring power consumption.

We run the NNA to process the convolutional layers of the Dilated Model for semantic segmentation on the Xilinx VC707 Evaluation Board. The Evaluation Board allows access to the power regulators through the Power Management Bus (PMBus). We extract the power consumption from the on-board power management ICs while the system is running. Preprocessing of images (mean shift) is done on a host computer and data transfer is managed through Ethernet. The mean power consumption we measure at run-time is given in Table 7. By using log-based PEs we can reduce the dynamic power consumption of our accelerator by 22.3%.

At this point we want to mention that an actual ASIC design would result in significantly less power consumption compared to the FPGA implementation we measure here. Also, as there are no DSP-slices involved in the PEs, these values appear high. However, the advantages of using barrel shifters instead of fixed-point multipliers can be seen with our experiments already. Furthermore, we do not change the memory bus nor the BRAM configuration for the experiments with log-based data representation. In Figure 7 the parallel structure of PEs can be seen. Therefore, we expect additionally a strong reduction in power consumption when reducing the bit-width of the memory subsystem from 8 Bit down to 5 Bit.

Table 7: Power Consumption

Accelerator	Power	dyn. Power
NNA with lin. PE	4.329W	2.301 W
NNA with log. PE	3.756W	1.788 W
abs. Δ	0.573W	0.5131W
rel. Δ	13.24%	22.3%

6 CONCLUSION

In this paper we present a low bit-width quantization method based on logarithmic data representation. The benefits of this method are reduced bit-width for both, network parameters as well as activation data, reduced complexity of the Processing Elements, lower power consumption, and no need for finetuning a pre-trained neural network after quantization. The measured results of power consumption on a prototypical implementation on an FPGA confirm our expectations of an efficient implementation. The power consumption is successfully reduced by 22.3% compared to an 8 Bit fixed-point multiplier design. Implementing an ASIC design of our Neural Network Accelerator, we expect to even further increase power savings when compared to a design based on fixed-point multipliers.

REFERENCES

[1] R. Andri, L. Cavigelli, D. Rossi, and L. Benini. 2016. YodaNN: An Ultra-Low Power Convolutional Neural Network Accelerator Based on Binary Weights. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 236–241. https://doi.org/10.1109/ISVLSI.2016.111

[2] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*. ACM, New York, NY, USA, 269–284. https://doi.org/10.1145/2541940.2541967

[3] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. 2016. Eyeriss: A Spatial Architecture for Energy-efficient Dataflow for Convolutional Neural Networks. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16)*. IEEE Press, Piscataway, NJ, USA, 367–379. https://doi.org/10.1109/ISCA.2016.40

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 3123–3131.

[6] Philipp Gysel, Jon Pimentel, Mohammad Motamedi, and Soheil Ghiasi. 2018. Ristretto: A Framework for Empirical Study of Resource-Efficient Inference in Convolutional Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* (2018). https://doi.org/10.1109/TNNLS.2018.2808319

[7] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. 2016. EIE: Efficient Inference Engine on Compressed Deep Neural Network. In *Proceedings of the 43rd International Symposium on Computer Architecture (ISCA '16)*. IEEE Press, Piscataway, NJ, USA, 243–254. https://doi.org/10.1109/ISCA.2016.30

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR* abs/1502.01852 (2015). arXiv:1502.01852 http://arxiv.org/abs/1502.01852

[9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR* abs/1704.04861 (2017). arXiv:1704.04861 http://arxiv.org/abs/1704.04861

[10] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *CoRR* abs/1602.07360 (2016). arXiv:1602.07360 http://arxiv.org/abs/1602.07360

[11] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. 2009. CIFAR-10 (Canadian Institute for Advanced Research). (2009). http://www.cs.toronto.edu/~kriz/cifar.html

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12)*. Curran Associates Inc., USA, 1097–1105. http://dl.acm.org/citation.cfm?id=2999134.2999257

[13] Edward H. Lee, Daisuke Miyashita, Elaina Chai, Boris Murmann, and S. Simon Wong. 2017. 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*. 5900–5904. https://doi.org/10.1109/ICASSP.2017.7953288

[14] J. Long, E. Shelhamer, and T. Darrell. 2015. Fully Convolutional Networks for Semantic Segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3431–3440. https://doi.org/10.1109/CVPR.2015.7298965

[15] Daisuke Miyashita, Edward H. Lee, and Boris Murmann. 2016. Convolutional Neural Networks using Logarithmic Data Representation. *CoRR* abs/1603.01025 (2016).

[16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. *XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks*. Springer International Publishing, Cham, 525–542. https://doi.org/10.1007/978-3-319-46493-0_32

[17] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*. http://arxiv.org/abs/1409.1556

[18] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. 2014. Striving for Simplicity: The All Convolutional Net. *CoRR* abs/1412.6806 (2014). arXiv:1412.6806 http://arxiv.org/abs/1412.6806

[19] Fisher Yu and Vladlen Koltun. 2015. Multi-Scale Context Aggregation by Dilated Convolutions. *CoRR* abs/1511.07122 (2015). arXiv:1511.07122 http://arxiv.org/abs/1511.07122