# Linear and Logarithmic Quantization Approaches for Efficient Inference with Deep Neural Networks

Constantin Berger
*Chair of Data Processing*
*Technical University Munich*
Munich, Germany
constantin.berger@tum.de

***Abstract*—Quantization enables efficient processing of Deep Neural Networks. In this work, the methods of Linear and Logarithmic Quantization are discussed. These methodologies are applied to a Deep Neural Network for controlling an autonomous drone. The trade-off between reduction of computational complexity and loss of accuracy is the main subject of this investigation. Moreover, I propose an approach to overcome the limitations of logarithmic quantization, which requires the specific handling of negative values. This is achieved by storing the sign of the non-quantized value in the sign-bit of the fixed-point value representation after Quantization. This approach allows the application of Logarithmic Quantization to Neural Networks with positive and negative weights. The results show that the given hardware does not allow for significant performance improvements.**

***Index Terms*—Machine Learning, Deep Neural Networks, Quantization**

## I. Introduction

Modern Machine Learning approaches have been applied to various disciplines of engineering including control [1], [2]. Real-time systems being controlled by arbitrarily complex Neural Networks require efficient processing of sensory data. This work is embedded in the utilization of Deep Neural Networks (DNNs) for controlling flight maneuvers of an autonomously acting drone. One such maneuver is to fly a circle while maintaining altitude. To achieve this, a Neural Network computes four values with each corresponding to the power control in form of PWM-signals of one rotor. The inputs of the network consist of several sensory values representing the position and speed of the drone. However, the following investigations apply to any resource-demanding system with a requirement of real-time functionality. For efficient processing of DNNs, the authors of [3] give an extensive overview of hardware as well as software-related measures. Reduced-precision techniques also referred to as Quantization are common tools for an efficient code design.

The authors of [4], [5] and [6] provide rather sophisticated methods for quantizing Neural Networks. However, for the initial investigation of Quantization techniques for the purpose of accelerating the forward-propagation process of a Neural Network, simpler linear and logarithmic techniques are the subject of this work. Quantization in general facilitates the use of datatypes with lower bit-width and the conversion from floating-point to fixed-point value representations. In the special case of logarithmic Quantization, it is possible to replace multiplication operations in the forward-propagation process with bit-shifts for additional acceleration. Recent works in [7] and [8] provide approaches for efficient processing of Neural Networks with logarithmic data representation. However, [8] is limited to the fact it does not address the representation of negative values.

Further works as [9], [10], and [11], address the application of their Quantization methodologies to pre-trained Neural Networks. However, these methods seem to require additional computational effort compared to the basic concepts of Quantization. Their evaluation remains to be further discussed after assessing the fundamental strategies. The authors of [12] propose a method dedicated to pre-trained Neural Networks but do not extensively discuss the computational operations with respect to their Quantization scheme.

In this work, the modalities of Quantization are being analyzed with regard to the aspects of execution time and accuracy of the forward-propagation process. The investigation of accuracy is achieved by applying the techniques of [7] and [8] to pre-trained Neural Networks which require an additional extension similar to the method in [13]. The purpose of this work is not only to compare Quantization methods but also to evaluate if an enhancement for pre-trained networks is reasonable in terms of accuracy and computational complexity.

The rest of this paper is organized as follows. In chapter II the concepts of Linear and Logarithmic Quantization are introduced. Chapter III and IV evaluate runtime and accuracy of four concepts of Quantization: Linear Quantization, Logarithmic Quantization of activations, Logarithmic Quantization of weights, and Logarithmic Quantization of weights and activations.

## II. Linear and Logarithmic Quantization

Converting floating-point representations to fixed-point representations of weights or activations can lead to performance improvements of embedded systems. Thereby, the characteristics of power consumption and storage define the performance. According to [13], the conversion is carried out by a mapping of a value $x \in \mathbb{R}$ to a Linear-Quantization level $x_q \in \mathbb{R}$. This operation is parameterized by the full-scale range $fsr = 2^{FSR}$ which defines the highest possible absolute value for Linear

Quantization and the stepsize $\Delta s$. The bit-width $b$ parameterizes $\Delta s$ which is given by

$$\Delta s = 2^{FSR+1-b}. \tag{1}$$

Given these parameters, the mapping for Linear Quantization is carried out by

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

where $\text{round}(a)$ rounds a value $a$ to the nearest integer and

$$\text{clip}(a, min, max) = \begin{cases} min & , a \leq min \\ max & , a \geq max \\ a & , \text{else} \end{cases}. \tag{3}$$

If $fsr$ is considered to be the maximum possible value of $x$, the outer clipping function becomes redundant. Linear Quantization is a suitable choice for representing uniformly distributed data. However, the distributions of weights in Neural Networks are mostly non-uniformly distributed. Figure 1a shows the weight distribution of a benchmark DNN which aims to control the drone to fly a circle.

To reduce the loss of information, Logarithmic Quantization is more applicable to the approximately normal-distributed weights. According to [13], this method can be parameterized by the same variables $fsr$ and $b$ as for Linear Quantization. The logarithmic data $\hat{x}_q$ are related to the Linear-Quantization data $x_q$ by

$$x_q = \text{sign}(x) \cdot 2^{\hat{x}_q}. \tag{4}$$

Accordingly and without further consideration of values $x > fsr$, the Logarithmic-Quantization data are obtained by

$$\hat{x}_q = \text{clip}\left(\text{round}\left(\log_2|x| - FSR\right), 2 - 2^{b-1}, 0\right) + FSR \tag{5}$$

for $x \neq 0$ and

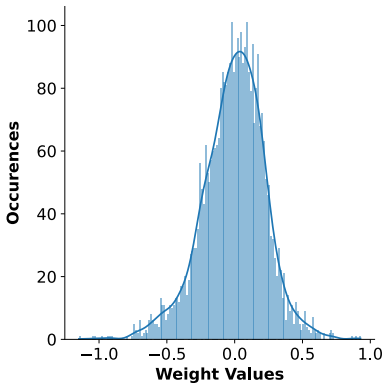
$$\hat{x}_q = 2 - 2^{b-1} \tag{6}$$

for $x = 0$. This procedure allows representing the majority of values with higher accuracy than values with low occurrences within the method of Logarithmic Quantization. The figures 1b and 1c show the weights of the Neural Network with Linear and Logarithmic Quantization, respectively.

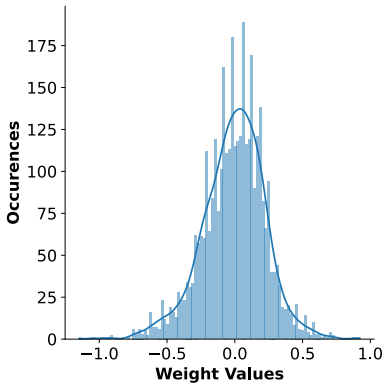
However, the total information about $x_q$ is not solely given by $\hat{x}_q$, but also by $\text{sign}(x)$ as described by equation 4. Since the values are represented by fixed point datatypes an efficient way of encoding $\text{sign}(x)$ is required. From equation 5 it can be observed, that for $FSR = 0$ all values $\hat{x}_q$ are either negative or equal to 0, which can can be achieved by scaling. Thus, the first bit which previously determined the sign of $\hat{x}_q$ can also be used to store this information. Following this, the expression

$$\hat{x}_q^* = \text{sign}(x) \cdot |\hat{x}_q| \tag{7}$$

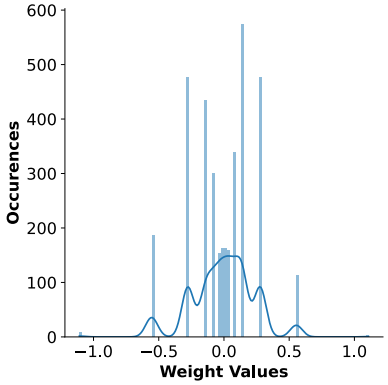
denotes the newly introduced data format. Using this trick, the proposed methodologies for processing logarithmic data in [7] and [8] can be efficiently applied to negative Quantization values.




(a) Floating Point Representation




(b) Linear Quantization




(c) Logarithmic Quantization

Fig. 1: Weight Distributions of a Benchmark Neural Network before and after Linear and Logarithmic Quantization with $b = 8$.

## III. Runtime Experiments

In this section, the runtime performances of the previously shown approaches for Linear and Logarithmic Quantization are determined as the basis for a further extension. This extension aims towards an application of the Logarithmic Quantization to pre-trained Neural Networks with positive and negative weights. The hardware used for these experiments is an STM32F411 microcontroller which is at the same time the controller for the drone. For investigating Linear Quantization based on equation 2, common values for the bit-width $b$ (see table I) and corresponding sizes of fixed-point datatypes were used. In the regime of Logarithmic quantization, the two proposed methods from [8] for quantizing only activations in method (a) and for quantizing activations as well as weights in method (b) were set up for the experiment with $b = 16$. It is to be noted, that only Neural Networks with positive weights were used for both methods (a) and (b).

The benefit of processing forward-propagation computations in Neural Networks with logarithmically represented data is that multiplication operations can be replaced by bit-shifts which reduce computational complexity and thus runtime [3]. The corresponding mechanism is proposed in [8]. In addition, a similar procedure was implemented for quantizing only weights in method (c). Therefore, the computation of an activation $a$ of the $j$-th neuron depending on $n$ activations $x_i$ of neurons in the previous layer and their corresponding weights $w_i$ as well as bias $b_0$ can be approximated as

$$\begin{aligned} a_j &= \sum_{i=1}^{n} w_i x_i + x_0 = \sum_{i=1}^{n} w_i \cdot 2^{\hat{x}_i} + x_0 \\ &\approx \sum_{i=1}^{n} \text{bitshift}(w_i, \text{round}(\hat{x}_i)) + x_0. \end{aligned} \tag{8}$$

The previous expression denotes the logarithmically quantized value $x_i$ as $\hat{x}_i$ and $\text{bitshift}(a, b)$ is a function that bit-shifts a value $a$ by an integer $b$ in fixed-point arithmetic. The quantities $x_i$ and $w_i$ are linearly quantized. Based on this formulation and the derivations in [8], the measurement experiments were carried out by averaging $1 \cdot 10^5$ forward-propagation cycles of the benchmark Neural Network. Table I shows the measured time values and possible execution frequencies.

It can be observed, that with the given hardware the forward-propagation process using Linear Quantization is in general not faster than the same computation using a floating-point representation of weights and activations. This can be due to the Floating Point Unit (FPU) which is featured by the microcontroller. It remains to be discussed if the forward-propagation processing using fixed-point values on a micro-controller without FPU is as expected faster than floating-point representation.

Furthermore, it can be stated that methods (a) and (b) of Logarithmic Quantization cause a runtime improvement where method (b) shows the most significant advance. This is possibly due to the fact that in (a), after the application of the activation function to $a_j$ the logarithmic representation $\hat{a}_j$ has to be computed. In method (b), the value representation never leaves the logarithmic domain which is why the computational step of taking the logarithm of $\hat{a}_j$ is not necessary.

TABLE I: Runtime Measurements

*Linear Quantization of Weights and Activations*

| **Datatype** | **Runtime** | |
|---|---|---|
| | ***Forward-Propagation*** | ***Frequency*** |
| int64 ($b = 64$) | 2.88 ms | 347.54 Hz |
| int32 ($b = 32$) | 1.93 ms | 518.59 Hz |
| int16 ($b = 16$) | 2.23 ms | 449.31 Hz |
| int8 ($b = 8$) | 2.23 ms | 449.31 Hz |

*Logarithmic Quantization using int16 ($b = 16$)*

| **Logarithmic Quantity** | **Runtime** | |
|---|---|---|
| | ***Forward-Propagation*** | ***Frequency*** |
| Activations (a) | 2.20 ms | 453.55 Hz |
| Weights and Activations (b) | 1.97 ms | 507.70 Hz |
| Weights (c) | 2.47 ms | 405.25 Hz |

*Benchmark Measurement*

| **Datatype** | **Runtime** | |
|---|---|---|
| | ***Forward-Propagation*** | ***Frequency*** |
| float | 1.76 ms | 567.45 Hz |

## IV. Accuracy Experiments

The previously determined runtime measurements suggest that Logarithmic Quantization, especially method (b), is applicable for faster processing of Neural Networks compared to linear value representation. This section addresses the question of whether the methodologies proposed in [7] and [8] constitute techniques on the basis of which also pre-trained Neural Networks can be processed more efficiently.

To assess this, the investigated methodologies need to be extended such that they are no longer limited to positive weights and/or activations. Subsequently, the outputs of the benchmark Neural Network are compared to sample outputs without Quantification. This procedure allows investigating the loss of accuracy which constitutes the drawback of hardware acceleration. Subsequently, it has to be evaluated if the extension measures lead to different runtime measurements. The steps for extending [7] and [8] are presented in the following.

### A. Consideration of negative Weights and Activations

As proposed in section II, the information about $\text{sign}(x)$ can be efficiently stored in the sign-bit of $\hat{x}_i^*$. For method (a) the accumulation given by equation 8 of previous-neuron activations has to be altered to

$$\begin{aligned} a_j &= \sum_{i=1}^{n} \text{sign}(\hat{x}_i^*) \cdot w_i \cdot 2^{-|\hat{x}_i^*|} + x_0 \\ &\approx \sum_{i=1}^{n} \text{sign}(\hat{x}_i^*) \cdot \text{bitshift}(w_i, -\text{round}(|\hat{x}_i^*|)) + x_0. \end{aligned} \tag{9}$$

It is to be noted, that the bias-term $x_0$ is not logarithmically represented as it is not weighted and hence no computational operation is performed on it. Furthermore, the computational steps of computing $\text{sign}(\hat{x}_i^*)$ and $|\hat{x}_i^*|$ are not necessary when

the activation functions of the Neural Network are ReLU-functions as then $\hat{x}_i^*$ would always be positive.

The Logarithmic Quantization of weights and activations of (b) is characterized by an efficient approximation of accumulating the previous activations in the logarithmic domain. For distinguishing $a_j$, equation 8 can be altered as

$$\begin{aligned} a_j &= \sum_{i=0}^{n} \operatorname{sign}(\hat{x}_i^*) \cdot 2^{-(|\hat{w}_i^*|+|\hat{x}_i^*|)} \\ &\approx \sum_{i=1}^{n} \operatorname{sign}(\hat{x}_i^*) \cdot \operatorname{bitshift}(1, -\operatorname{round}(|\hat{w}_i^*| + |\hat{x}_i^*|)), \end{aligned} \tag{10}$$

where $\hat{w}_0^* = 0$. In the following, the argument for the bitshift of 1 is denoted as $\hat{p}_i^* = -\operatorname{round}(|\hat{w}_i^*| + |\hat{x}_i^*|)$. In [8], an approximation for the logarithmic accumulation of $\hat{a}_j$ is carried out using $\log_2(1+x) \approx x$ for $0 \leq x < 1$. In this approach, the $n$ summands of $a_j$ can be negative, which why it is not possible to perform a logarithmic accumulation according to

$$\begin{aligned} \hat{a}_j \approx & \max(\hat{a}_{j-1}, \hat{p}_j) \\ & + \operatorname{bitshift}(1, -|\operatorname{round}(|\hat{a}_{j-1}|) - \hat{p}_j|), \end{aligned} \tag{11}$$

where $\hat{a}_j \geq 0 \ \forall j$. Therefore, the accumulation process for each neuron would require an initial transformation of each summand from logarithmic to linear domain and an ultimate back-transformation. This would require an enormous computational effort which is why method (b) is considered inadequate for this type of extension.

Processing the forward-propagation algorithm of Neural Networks with positive and negative weights represented in the logarithmic domain (c) can be achieved similarly as for the logarithmic representation of activations (a). Hereby, equation 8 can be expressed as

$$\begin{aligned} a_j &= \sum_{i=i}^{n} \operatorname{sign}(\hat{w}_i^*) \cdot x_i \cdot 2^{-|\hat{w}_i^*|} + x_0 \\ &\approx \sum_{i=1}^{n} \operatorname{sign}(\hat{w}_i^*) \cdot \operatorname{bitshift}(x_i, -\operatorname{round}(|\hat{w}_i^*|)) + x_0. \end{aligned} \tag{12}$$

### B. Evaluation of Output Accuracies

Having implemented the previously defined accumulation processes for methods (a) and (c), the respective accuracy of the outputs can be evaluated based on the Root Mean Squared Error (RMSE), the R2 Score, and the Correlation Coefficient with respect to the benchmark outputs. The measurement outcomes for Logarithmic Quantization as well as Linear Quantization parameterized by $b$ are listed in table II.

The results show that Linear Quantization, in general, reproduces the output with high accuracy. A fixed point datatype with $b = 8$ still achieves sufficient results. It is obvious that even larger bit widths as for example $b = 32$ or $b = 64$ would make the error vanish. For the case of Logarithmic Quantization, it can be observed that both methods (a) and (c) are not sufficient enough for reproducing the outputs of a benchmark Neural Network. However, the RMSE of method (c) is lower than for method (a), which is due to the fact that representing the weights logarithmically matches their value distribution more beneficially than for Logarithmic Quantization of activations.

TABLE II: Accuracy Measurements

*Linear Quantization of Weights and Activations*

| **Bit-width** | **Accuracy** | | |
|---|---|---|---|
| | ***RMSE*** | ***R2 Score*** | ***Correlation*** |
| $b = 16$ | $1.12 \cdot 10^{-3}$ | 1.00 | 1.00 |
| $b = 10$ | $69.30 \cdot 10^{-3}$ | 0.99 | 0.99 |
| $b = 8$ | 0.22 | 0.99 | 0.99 |
| $b = 6$ | 0.87 | 0.97 | 0.99 |
| $b = 4$ | 3.40 | 0.64 | 0.80 |

*Logarithmic Quantization with Bit-width $b = 10$*

| **Logarithmic Quantity** | **Accuracy** | | |
|---|---|---|---|
| | ***RMSE*** | ***R2 Score*** | ***Correlation*** |
| Activations (a) | 7.05 | $-0.50$ | 0.04 |
| Weights (c) | 6.31 | 0.00 | 0.07 |

### C. Re-evaluation of Runtime Measurements

The extended methods require additional computation steps. Due to this, the runtime has to be re-evaluated. For method (a), the new runtime amounts to $10.77\,\mathrm{ms}$ for a complete forward-propagation process, which exceeds the corresponding value distinguished in section III. For method (c), the new measurement value is $2.32\,\mathrm{ms}$, which is approximately on the same scale as before. Both experiments were conducted with datatype int16 ($b = 16$). The reason for the outlying runtime of method (a) is attributable to the additional computation of each activation in the logarithmic domain. In this case, the logarithm has no longer been approximated according to [8]. In the previous approximation, $\operatorname{round}(|\log_2(x)|)$ is computed by returning the position of the first 1 bit of $x$ seen from the most significant bit. Possibly, the runtime measurement would lie on the same scale as before if the approximation was used. Nevertheless, this additional investigation was left out as the evaluated accuracy of the method has shown poor results anyhow.

### D. Interpretation

The loss of accuracy in the case of Logarithmic Quantization can be attributed to the calculation of activations using the rounded logarithms of activations or weights (see equations 9 and 12). It can be observed, that after the first layer in the benchmark Neural Network the activations only differ slightly, but the propagating errors increase with the depth of the network. In [13], the accumulation of $a_j$ is characterized by the additional multiplication of each summand with some factor that accordingly represents the remaining fraction of the rounded logarithm. To illustrate this procedure, the accumulation of $a_j$ based on logarithmically represented weights and

activations is given by

$$a_j = \sum_{i=0}^{n} \text{sign}(\hat{w}_i^*) \cdot \text{sign}(\hat{x}_i^*) \cdot 2^{-(|\hat{w}_i^*|+|\hat{x}_i^*|)} = \sum_{i=0}^{n} \text{sign}(\hat{w}_i^* \hat{x}_i^*) \cdot \text{bitshift}(-\log_2(\text{frac}(\hat{p}_i^*)), -\text{int}(\hat{p}_i^*)), \tag{13}$$

where $\hat{p}_i^* = |\hat{w}_i^*| + |\hat{x}_i^*|$ and $\text{int}(\hat{p}_i^*)$ is the integer part of $\hat{p}_i^*$, whereas $\text{frac}(\hat{p}_i^*)$ is the fractional part of $\hat{p}_i^*$. However, this technique can only be applied to Neural Networks with logarithmically represented weights and activations which in consequence to named reasons were excluded from further investigations in this work.

## V. Discussion

The outcomes of the conducted experiments suggest that hardware acceleration is not achieved by the proposed Quantization techniques in the proposed hardware setup. However, the runtime is always dependent on the microcontroller and its hardware components. In this case, the FPU of the microcontroller controlling the drone might cause that fixed-point value multiplications are not faster than floating-point. Previous works about quantizing Neural Networks as [5], [9], and [14] invoke energy consumption [15] as measure for hardware optimization. However, the quantity of interest, in this case, is the runtime as it determines the maximum frequency of computing motor outputs for the drone.

The outputs of the Neural Network with extended Logarithmic Quantization show that solely the enhancement for negative weights and activations is not sufficient in terms of accuracy. The investigated methods would require an additional adaptation for better performance. The works in [13], [8] and [10] employ additional fine-tuning which enables their application to pre-trained Neural Networks. However, such additional steps would increase the computational complexity and thus the runtime even more.

## VI. Conclusion

This work evaluated a series of parameterized Quantization methodologies with respect to their runtime on a microcontroller and their accuracy with respect to outputs of a benchmark neural network. Furthermore, an efficient coding scheme for storing quantized values in fixed-point datatypes was proposed. The acceleration of the forward-propagation process by using fixed-point datatypes with lower bit width instead of floating-point datatypes could not be achieved in this specific hardware setup. An extension of Logarithmic Quantization methods for their application to pre-trained Neural Networks is possible but requires additional tuning which in consequence leads to higher computational complexity. In conclusion, reduced precision processing of Neural Networks using Quantization can have several benefits with respect to the use of storage and energy consumption but does not lead to a faster computation of power-controls for each rotor of the drone.

## References


[1] K. Cheon, J. Kim, M. Hamadache, and D. Lee, "On replacing pid controller with deep learning controller for dc motor system," *Journal of Automation and Control Engineering Vol*, vol. 3, no. 6, 2015.

[2] G.-H. Liu and E. A. Theodorou, "Deep learning theory review: An optimal control and dynamical systems perspective," 2019.

[3] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[4] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5918–5926, 2017.

[5] S. O. Settle, M. Bollavaram, P. D'Alberto, E. Delaye, O. Fernandez, N. Fraser, A. Ng, A. Sirasao, and M. Wu, "Quantizing convolutional neural networks for low-power high-throughput inference engines," 2018.

[6] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented approximation of convolutional neural networks," 2016.

[7] 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)*, pp. 5900–5904, 2017.

[8] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," 2016.

[9] H. Tann, S. Hashemi, R. I. Bahar, and S. Reda, "Hardware-software codesign of accurate, multiplier-free deep neural networks," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2017.

[10] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *International conference on machine learning*, pp. 2849–2858, PMLR, 2016.

[11] S. Vogel, J. Springer, A. Guntoro, and G. Ascheid, "Self-supervised quantization of pre-trained neural networks for multiplierless acceleration," in *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1094–1099, 2019.

[12] S. Ullah, S. Gupta, K. Ahuja, A. Tiwari, and A. Kumar, "L2l: A highly accurate log_2_lead quantization of pre-trained neural networks," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 979–982, 2020.

[13] S. Vogel, M. Liang, A. Guntoro, W. Stechele, and G. Ascheid, "Efficient hardware acceleration of cnns using logarithmic data representation with arbitrary log-base," in *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '18, (New York, NY, USA), Association for Computing Machinery, 2018.

[14] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2016.

[15] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014.