# Histogram-equalized quantization for logic-gated residual neural networks

Van Thien Nguyen, William Guicquero, Gilles Sicard

# Histogram-Equalized Quantization for logic-gated Residual Neural Networks

Van Thien Nguyen, William Guicquero and Gilles Sicard
CEA-LETI, F-38000, Grenoble, France. [Email: vanthien.nguyen@cea.fr]

***Abstract*—Adjusting the quantization according to the data or to the model loss seems mandatory to enable a high accuracy in the context of quantized neural networks. This work presents Histogram-Equalized Quantization (HEQ), an adaptive framework for linear and symmetric quantization. HEQ automatically adapts the quantization thresholds using a unique step size optimization. We empirically show that HEQ achieves state-of-the-art performances on CIFAR-10. Experiments on the STL-10 dataset even show that HEQ enables a proper training of our proposed logic-gated (OR, MUX) residual networks with a higher accuracy at a lower hardware complexity than previous work.**

***Index Terms*—CNN, quantized neural networks, histogram equalization, skip connections, logic-gated CNN**

## I. Introduction

Designing low-precision networks [1] is a promising area of research aiming at reducing the bit width to represent weights and activations, thus reducing the overall computational complexity and memory-related costs, namely for performing inference at the edge. The advantages of quantization have been demonstrated on several resource-efficient low-precision CNN accelerators [2], [3], [4], [5], [6].

Quantization-Aware Training (QAT) is the common approach to preserve the performance of quantized models and avoid unacceptable accuracy degradation due to the limited precision. QAT usually consists in using real-valued proxies of the model weights that are on-the-fly quantized during the forward pass while being updated during the backward pass [7]. Although several nonlinear quantization mappings [8], [9] and [10] have demonstrated remarkable algorithmic performances, they are not fully compliant with a simple hardware implementation. On the contrary, a linear symmetric mapping [11] naturally matches a streamlined hardware, making it a more relevant and reasonable choice for model quantization.

In the case of linear symmetric quantization, the thresholds are derived from a unique step size. The calibration of this scaling factor plays a key role and we state that it is likely intractable to find the optimal a priori value, given that it deeply depends on the model topology, its initialization, the inference task and the training procedure. Therefore, using an adjustable scaling factor during the training has demonstrated to be more favorable because of taking into consideration the evolution of weight/activation layer-wise distributions. State-of-the-art methods propose to optimize these parameters under the minimization of the quantization error and/or the loss function. For example, [12] aims at minimizing the mean squared error between the floating-point weights and their ternarization while [11] updates the step size through a simulated gradient in which the descent direction is based on the quantization error. On the other hand, [13] and later [14] propose to learn the scaling factor using the task loss backpropagation. Furthermore, [15] introduces a Straight-Through-Estimated (STE [16]) gradient of the step size with respect to the loss. Based on this work, [17] and [18] take advantage of bitwidth-dependent regularizations to optimize the layer-wise bit allocation given a target model size or a computational budget.

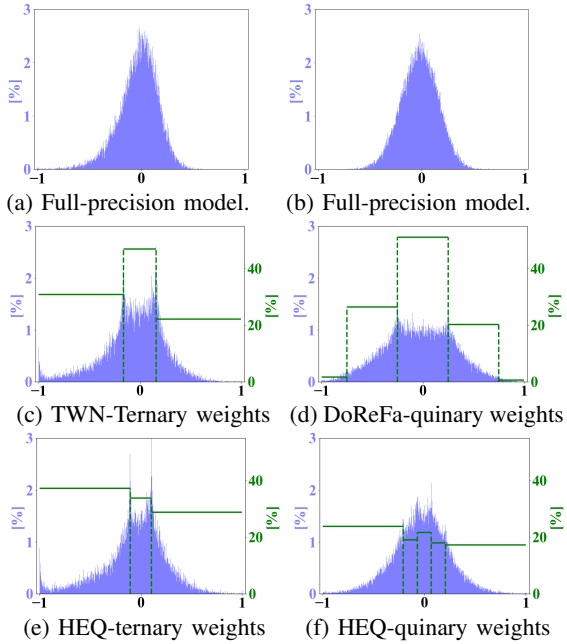
In this paper, we claim that –in most cases– a proper quantization scheme should cover all the available data representation space, somehow maximizing the entropy of the weights [19]. Based on this hypothesis, [20] presents a 2-bit quantization methods for recurrent models where the step size equals to a constant multiple of the mean value of the proxy weights. Similarly, [21] determines the thresholds of 3-value and 4-value quantizations according to the mean and the standard deviation of the proxy weights. However, these approaches are not generic and applied only to under 3-bit quantization. On the contrary, our proposed method – Histogram-Equalized Quantization (HEQ)– automatically adjusts the step size of an $n$-value quantization according to its $n$-quantiles such that the resulting quantized values are more balanced, without any further regularization. We empirically show that our method provides a better accuracy than previous methods on different topology variants, from the baseline plain model to our proposed logic-gated residual networks. Indeed, HEQ advantageously enables a proper training of quantized models that embed OR and MUX logic gates to replace floating-point types of skip connections, this way simplifying the Hardware mapping of layer interconnections.

## II. Linear symmetric quantization

This paper focuses on the linear symmetric quantization to a restricted range of odd $n > 2$ discrete values. We consider the mapping $g : \mathbb{R} \to [[-1, +1]]$ applied to the weight $w$ as:

$$g(w; s) = \frac{2}{n-1}\text{Clip}\left(\left\lfloor \frac{w}{s} \right\rceil, \frac{1-n}{2}, \frac{n-1}{2}\right), \qquad (1)$$

where $[[-1, +1]]$ is discretized with an output step size of $\frac{2}{n-1}$, $s$ is the input step size and $\text{Clip}(x; a, b) = \min(\max(x, a), b)$ with $a < b$. While existing works usually keep the range of floating values by the factor $s$ outside of the clipping function, here we use the $\frac{2}{n-1}$ scale factor so that all the quantized values are explicitely shrunk in the interval $[-1, +1]$. During backpropagation, we use the straight-through-estimated (STE [16]) gradient $\frac{\partial g}{\partial w} = \mathbb{1}_{\{|x|\leq 1\}}$ to update the proxy weights.



(a) Full-precision model. (b) Full-precision model.

(c) TWN-Ternary weights (d) DoReFa-quinary weights

(e) HEQ-ternary weights (f) HEQ-quinary weights

Fig. 1: Weight distributions of 2 layers (in 2 columns) after training of: full-precision model ($1^{st}$ line), existing ternary-weight and quinary-weight model ($2^{nd}$ line), and our proposed HEQ method ($3^{rd}$ line) along with quantization thresholds.

This formulation thus depends on the definition of $s$ that deeply impacts on the model accuracy. Let us consider Ternary Weight Networks (TWN) [12] as the baseline where $s = 2\tau\frac{\sum_{i=1}^{n}|W_i|}{n}$, with a fixed norm factor $\tau = 0.7$. Although the optimal $s$ may change depending on the data distribution, this method cannot be applied to higher precisions and the predefined $\tau$ limits the adaptability of the quantizer. Similarly, DoReFa [22] forces the real values into the range of $[-1, +1]$ by a mapping adapted to the data, but the thresholds remain fixed. Fig. 1 depicts the histogram of proxy weights (blue bars) and quantized weights (horizontal green lines) in the case of 3-value (Figs. 1c, 1e) and 5-value (Figs. 1d, 1f) quantization whose initialization (from full-precision model) is shown in Figs. 1a and 1b, respectively. We can observe that in both cases of TWN (Fig. 1c) and DoReFa (Fig. 1d), the proxy weights are mainly concentrated around zero and the distributions between thresholds (vertical green lines) are unbalanced. In particular, the quinary weights (3-bit) in Fig. 1d can be approximated by only 3 values (2-bit). Consequently, the quantized weights fail to exploit all available values which may cause the model to be sub-optimal. This motivates the use of more proper quantizers, favoring the balance of quantized weights.

## III. Histogram-Equalized Quantization (HEQ)

To resolve the aforementioned imbalance between quantized values, we propose HEQ to automatically adjust $s$ during training. Assuming that a proper quantizer should optimize the balanced use of available discrete values in the data representation space, we iteratively tune $s$ based on the histogram of the proxy weights to equi-distribute quantized weights.

### A. Method

In Fig. 2 we denote $\{(q_i, q_{-i})\}_{i\in[\![1,\frac{n-1}{2}]\!]}$ as $n-1$ points which divide the histogram of weights into $n$ intervals with equiprobabilities (namely $n$-quantiles). Observing that the weights distribution may change during the training procedure but with a median value that usually stays around zero, we assume that these quantiles are symmetrically distributed around zero, *i.e.* $q_{-i} \approx -q_i$ with $q_i > 0$ or $|q_i| = q_i$.

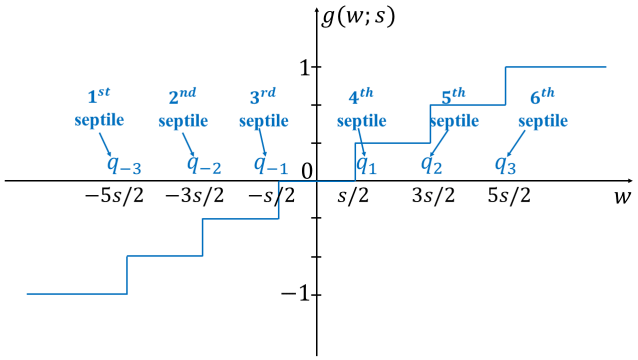



Fig. 2: Symmetric linear quantization with histogram bin equalization when $n$-quantiles $(q_{-i}, q_i)$ are symmetrical and coincide with the quantized thresholds.

In order to equalize the histogram bins of quantized values, we thus re-estimate and update $s$ such that the resulting thresholds used by the quantization function (see Fig. 2) are getting closer to these quantiles. Therefore, $s$ can be approximated by a weighted sum of the quantiles such that $q_i$ approximately coincides with $\frac{(2i-1)s}{2}$. Concretely, we assume that the sum of the absolute value of $n$-quantiles is approximately equal to that of the thresholds:

$$\sum_{i=1}^{\frac{n-1}{2}}(|q_{-i}| + q_i) = 2\sum_{i=1}^{\frac{n-1}{2}}\frac{(2i-1)s}{2}, \tag{2}$$

from which we can derive the following updating formula:

$$s = \frac{4\sum_{i=1}^{\frac{n-1}{2}}(|q_{-i}| + q_i)}{(n-1)^2}. \tag{3}$$

This approach has the great advantage of being generic, compatible with almost all use cases regardless the quantization level, the position of the layer and its type (with a possible extension to an even $n$). To maintain the stability of the model during optimization, we compute and update $s$ only at the beginning of each epoch. The formal training procedure is detailed in Algorithm 1. The proxy weight distributions obtained after a training stage that are reported in Figs. 1e and 1f clearly demonstrate that HEQ provides a more balanced distribution of quantized weights.

Note that with symmetric quantiles, the resulting quantized weights become more equalized. This can be enhanced by forcing the weight median to zero, which has not been applied in the scope of this work for the sake of clarity since it seems to have only a slight impact on the performance of the model.

**Algorithm 1** Training QNN with Histogram-Equalized Quantization (HEQ)

**Input:** Initial proxy weights $\{\mathbf{W}_l\}_{l=1}^{L}$ and training dataset
**Output:** Optimized $\{\mathbf{W}_l\}_{l=1}^{L}$, $\{s_l\}_{l=1}^{L}$
// $B$, $I$, $L$: #batches, #epochs, #layers
// $\mathbf{W}_l$: full-precision proxy weights of the $l^{th}$ layer,
// $s_l$: quantization step size used at the $l^{th}$ layer.

**for** $i = 1$ to $I$ **do**
  **for** $l = 1$ to $L$ **do**
    Find $n$-quantiles of layer $l$
    Compute and update $s_l$ (Eq. 3)
  **end for**
  **for** $b = 1$ to $B$ **do**
    Forward pass using $\{g(\mathbf{W}_l; s_l)\}_{l=1}^{L}$ (Eq. 1)
    Backward pass and update $\{\mathbf{W}_l\}_{l=1}^{L}$
  **end for**
**end for**

### B. State-of-the-art benchmark

HEQ has been evaluated on the CIFAR-10 dataset [23] with $32 \times 32$ RGB images and using the VGG-Small model like in [10]. A combination of a scale-invariant random crop (performed on all-sided 4-pixel padded images) combined with a random horizontal flip is used for data augmentation. Initial proxy weights are from a pre-trained full-precision network. Motivated by Hardware considerations, a 2-bit activation scheme as detailed in DoReFa [22] has been used. Our model is trained during 100 epochs with a small batch size of 50 to favor exploration. The learning rate is set to $10^{-3}$ during the first 50 epochs, then exponentially rescaled by a factor of 0.9 at each epoch. Finally, a very last epoch with a larger batch size of 100 and a smaller learning rate of $10^{-5}$ is performed for fine-tuning. Fig. 3 allows a comparison between TWN [12] and our HEQ method with respect to the resulting weight distributions. While the zero values dominate all layers in the case of TWN, our method reduces the variance and limits the number of weights at 0 to nearly $1/3$ as shown in Fig. 3. The number of $-1$ values slightly dominates as more proxy weights are concentrated on the negative side.

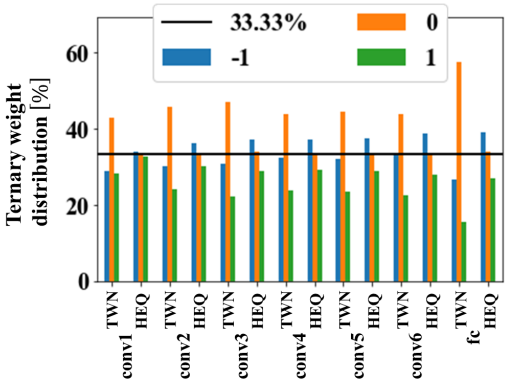
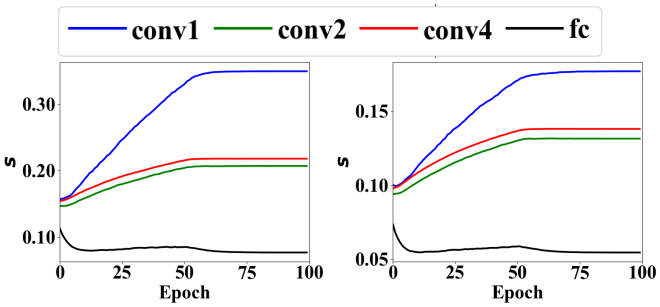



Fig. 3: Comparison of the ternary-weight distribution using TWN and our HEQ method.

Fig. 4 depicts the variation of $s$ during training in both ternary and quinary cases. It shows that the evolution of $s$ depends on the layer and has different convergence values.




(a) Ternary quantization. (b) Quinary quantization.

Fig. 4: Evolution of the step size $s$ during training.

Table I reporting the average accuracy of each configuration over 5 realizations, demonstrates the competitiveness of HEQ compared to the state-of-the-art quantization methods. For instance, when quantizing both weights (W) and activations (A) into 2-bit, we obtain 93.51% accuracy while having only 3 values $\{-1, 0, +1\}$ over 4 values possible like LQ [10] and LLSQ [11]. Compared to the full-precision model, we observe mostly no degradation in the case of quinary weights ($n = 5$) and even a gain with septenary weights ($n = 7$). Moreover, while other works give rise to a full-precision scaling factor besides the integer weights which demands the fusion into Batch Normalization [24] (BN) for later hardware implementation, our models trained with HEQ-ternary and HEQ-quinary obtain directly integer values $0, \pm 1$ (logical operations) and $\pm 0.5$ (bitshifts) which is already compatible for an easy hardware deployment.

TABLE I: Comparison with the state-of-the-art low-precision quantization methods on CIFAR-10.

| Method | HW-Compatibility | Bitwidth W/A | Accuracy(%) |
|---|---|---|---|
| TWN [12] | + | 2/32 | 92.56 |
| STTN [25] | + | 2/2 | 92.93 |
| TRQ [14] | + | 2/2 | 91.2 |
| LQ [10] | - | 2/32<br>2/2 | 93.8<br>93.50 |
| LLSQ [11] | + | 2/2 | 93.31 |
| FP32 baseline | | 32/32 | 93.68 |
| **HEQ-ternary** | ++ | 2/2 | 93.51 |
| **HEQ-quinary** | ++ | 3/2 | 93.66 |
| **HEQ-septenary** | + | 3/2 | 93.75 |

## IV. Extensions to Binarized Skip Connections

### A. Logic-gated Residual Neural Networks

Although quantization methods have been mainly applied to a wide range of DNN topologies, their usage is mainly focused on reducing the weight and activation bit widths. On the other hand, the element-wise addition in the case of skip connections (ResNet [26]) is still performed using a full-precision like in [27], [28] and [29]. The reason is that apart from improving feature map reusability, these full-precision

additions are mostly used to handle the gradient vanishing and mismatching issues, which seem to be even more crucial in the context of quantized models. However, it results in additional costs with respect to the corresponding hardware implementation. In this section, we focus on the compression of those skip connections, including the residual addition and the attention-like multiplication [30], such that these element-wise operations can be implemented by only OR and MUX logic gates rather than 32-bit arithmetic hardware.

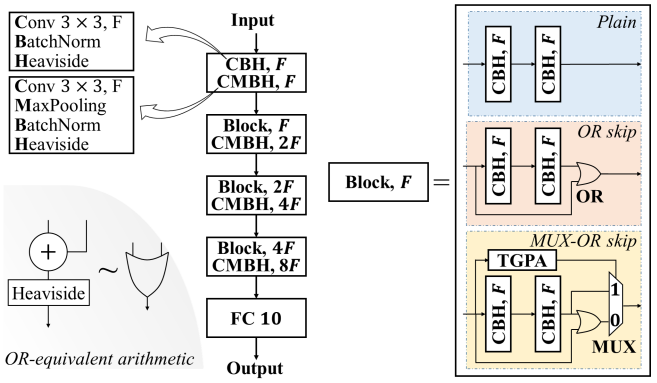



Fig. 5: Models with the plain block (11-hidden layer VGG [31]-variant), OR-gated block and MUX-OR gated block.

Fig. 5 depicts the proposed model design with 3 consecutive convolution blocks with different variants: plain block (denoted as VGG-11), OR block (ORNet-11) and MUX-OR block (MUXORNet-11), where $F$ denotes the basis number of convolutional output feature maps. All the activations are binarized using the Heaviside function $H(x) = \mathbb{1}_{\{x>0\}}$ where $\mathbb{1}$ is the indicator function. The logical OR operation between two binary inputs is arithmetically performed as $x_1 \vee x_2 = H(x_1 + x_2)$. The MUX-OR block additionally embeds an attention-alike branch (called MUX branch) along with the OR skip connection. This MUX branch is composed of a channel-wise Thresholded Global Average Pooling (TGPA) that corresponds to a Global Average Pooling (GPA) followed by the (re)binarization $T(x) = \mathbb{1}_{\{x>0.5m\}}$, where $m$ is set to the maximum of the GPA's outputs in full-precision model and to 1 in quantized model. When deploying the quantized model, this operation can be basically implemented via a bitcount followed by a comparison with a threshold level equal to half the number of pixels. Concretely, the OR-skip connection will be performed for each input feature map channel that has more zeros than ones. Otherwise, the MUX will simply keep the straightforward output of the second Convolution-BatchNorm-Heaviside (CBH) module. One interesting aspect of such a MUX-skip connection is that it favors to balance the number of 1 with respect to the number of 0 throughout the networks, this without any other specific regularization. In terms of hardware deployment, while existing approaches with 32-bit additions and multiplications require hundreds of Xilinx FPGA slices [32], a 1-bit OR only costs a single slice and consumes much less energy [33].

### B. Experimental results

In this section, we evaluate the aforementioned Neural Network topology variants using the proposed HEQ with ternary weights on STL-10 dataset [34] of $96 \times 96$ RGB images. To limit the overfitting, we used the following data augmentation scheme: random crop from all-sided 12-pixel padded images combined with random horizontal flips and cutouts of $32 \times 32$ pixel patches [35]. All the parameters of the quantized model are initialized from its pre-trained full-precision network counterpart, in which all Heaviside functions are replaced by ReLU. We set $F = 64$ instead of 128 in VGG-7, resulting in smaller-sized model. All propositions are implemented using Tensorflow [36] and Larq [37].

TABLE II: Comparison with the state-of-the-art low-precision quantization methods on STL-10 dataset.

| Model | Training | Regularization | # params. (M) | Bitwidth W/A | Acc. (%) |
|---|---|---|---|---|---|
| VGG-7 | LSQ [15] | #params×bit [17] | 4.57 | 2.5/8 | 83.6 |
| | | #MACs×bit [18] | | 2.2/8 | 83.8 |
| VGG-11 | HEQ | None | 3.14 | 2/1 | 83.34 |
| ORNet-11 | | | | | 83.82 |
| MUXORNet-11 | | | | | **84.17** |

Table II summarizes the performance of our proposed models compared to previous work [18] which uses the LSQ [15] method to jointly adapt the step size and the layer-wise bitwidths under a model size-based [17] or a MAC×bit-based regularization. Note that we only took into account the number of convolution parameters for the sake of a fair comparison. While the plain baseline obtains only $83.3\%$ accuracy, ORNet-11 achieves $83.8\%$ and the MUXORNet-11 even achieves up to $84.2\%$, *i.e.* a noticeable improvement without increasing the overall model size and with a negligible extra cost of 2-input MUX, OR gates and thresholded bitcounts. In terms of model size, all our 3 model variants contain less parameters at lower precision compared to [18]. However, the proposed ORNet-11 optimized by HEQ already achieves the same level of accuracy while MUXORNet-11 even obtains a better accuracy ($0.37\%$). These results demonstrate the effectiveness of HEQ on different DNN designs, from VGG-like to the proposed ORNet and MUXORNet. It also shows the possibility of compressing the skip connections via logic gates in order to significantly simplify the hardware mapping of more sophisticated ternarized neural network topologies than VGG-like.

## V. Conclusion

We introduce a novel QAT method based on the equalization of layer-wise weight histograms. During the training process, the step size is adaptively changed according to the proxy weight distribution through its $n$-quantiles, such that the quantized levels are approximately equalized. We empirically show that the models trained with our HEQ can achieve not only state-of-the-art accuracy on CIFAR-10, but even a better accuracy on STL-10 dataset thanks to the proposed logic-gated residual networks, while using a lower precision than previous works on budget-aware learned quantization.

## REFERENCES


[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, 2017.

[2] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: An Energy-Efficient Deep Neural Network Accelerator With Fully Variable Weight Bit Precision," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 173–185, Jan. 2019.

[3] P. C. Knag, G. K. Chen, H. E. Sumbul, R. Kumar, M. A. Anders, H. Kaul, S. K. Hsu, A. Agarwal, M. Kar, S. Kim, and R. K. Krishnamurthy, "A 617 TOPS/W All Digital Binary Neural Network Accelerator in 10nm FinFET CMOS," in *2020 IEEE Symposium on VLSI Circuits*, Jun. 2020, pp. 1–2, iSSN: 2158-5636.

[4] T.-H. Kim and J. Shin, "A Resource-Efficient Inference Accelerator for Binary Convolutional Neural Networks," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 451–455, Jan. 2021, conference Name: IEEE Transactions on Circuits and Systems II: Express Briefs.

[5] R. Andri, G. Karunaratne, L. Cavigelli, and L. Benini, "Chewbaccann: A flexible 223 tops/w bnn accelerator," *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2021.

[6] X. Zhou, L. Zhang, C. Guo, X. Yin, and C. Zhuo, "A convolutional neural network accelerator architecture with fine-granular mixed precision configurability," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[7] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," *arXiv: Learning*, 2016.

[8] S. Han, H. Mao, and W. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *arXiv: Computer Vision and Pattern Recognition*, 2016.

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

[10] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," *ArXiv*, vol. abs/1807.10029, 2018.

[11] X. Zhao, Y. Wang, X. Cai, C. Liu, and L. Zhang, "Linear symmetric quantization of neural networks for low-precision integer hardware," in *ICLR*, 2020.

[12] F. Li and B. Liu, "Ternary weight networks," *ArXiv*, vol. abs/1605.04711, 2016.

[13] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[14] Y. Li, W. Ding, C. Liu, B. Zhang, and G. Guo, "TRQ: Ternary Neural Networks With Residual Quantization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, pp. 8538–8546, May 2021, number: 10.

[15] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," in *8th International Conference on Learning Representations, ICLR*, 2020.

[16] Y. Bengio, N. Léonard, and A. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," *arXiv:1308.3432 [cs]*, Aug. 2013.

[17] S. Uhlich, L. Mauch, F. Cardinaux, K. Yoshiyama, J. A. García, S. Tiedemann, T. Kemp, and A. Nakamura, "Mixed precision DNNs: All you need is a good parametrization," in *8th International Conference on Learning Representations, ICLR, Addis Ababa, Ethiopia, April 26-30, 2020*.

[18] K. Nakata, D. Miyashita, J. Deguchi, and R. Fujimoto, "Adaptive Quantization Method for CNN with Computational-Complexity-Aware Regularization," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021, pp. 1–5, iSSN: 2158-1525.

[19] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[20] Q. He, H. Wen, S. Zhou, Y. Wu, C. Yao, X. Zhou, and Y. Zou, "Effective quantization methods for recurrent neural networks," *ArXiv*, vol. abs/1611.10176, 2016.

[21] M. Z. Alom, A. T. Moody, N. Maruyama, B. C. Van Essen, and T. M. Taha, "Effective Quantization Approaches for Recurrent Neural Networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro: IEEE, Jul. 2018, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/document/8489341/

[22] 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," *ArXiv*, vol. abs/1606.06160, 2016.

[23] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," p. 60, 2009. [Online]. Available: https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf

[24] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ArXiv*, vol. abs/1502.03167, 2015.

[25] W. Xu, X. He, T. Zhao, Q. Hu, P. Wang, and J. Cheng, "Soft threshold ternary networks," in *IJCAI*, 2020.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[27] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," in *ECCV*, 2018.

[28] H. T. Phan, D. Huynh, Y. He, M. Savvides, and Z. Shen, "Mobinet: A mobile binary network for image classification," *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 3442–3451, 2020.

[29] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. W. Mahoney, and K. Keutzer, "Hawqv3: Dyadic neural network quantization," in *ICML*, 2021.

[30] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6450–6458, 2017.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[32] M. Beauchamp, S. Hauck, K. Underwood, and K. Hemmert, "Embedded floating-point units in FPGAs," in *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate array*, 01 2006, pp. 12–20.

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

[34] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *AISTATS*, 2011.

[35] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *ArXiv*, vol. abs/1708.04552, 2017.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[37] L. Geiger and P. Team, "Larq: An open-source library for training binarized neural networks," *J. Open Source Softw.*, vol. 5, p. 1746, 2020.