Memory Requirement Reduction of Deep Neural Networks for Field Programmable Gate Arrays Using Low-Bit Quantization of Parameters

Niccolò Nicodemo*, Gaurav Naithani†, Konstantinos Drossos†, Tuomas Virtanen†, Roberto Saletti*
*University of Pisa, Italy
n.nicodemo1@studenti.unipi.it, roberto.saletti@unipi.it
† Audio Research Group, Tampere University, Finland
{firstname.lastname}@tuni.fi

***Abstract*—Effective employment of deep neural networks (DNNs) in mobile devices and embedded systems, like field programmable gate arrays, is hampered by requirements for memory and computational power. In this paper we propose a method that employs a non-uniform fixed-point quantization and a virtual bit shift (VBS) to improve the accuracy of the quantization of the DNN weights. We evaluate our method in a speech enhancement application, where a fully connected DNN is used to predict the clean speech spectrum from the input noisy speech spectrum. A DNN is optimized, its memory requirement is calculated, and its performance is evaluated using the short-time objective intelligibility (STOI) metric. The application of the low-bit quantization leads to a 50% reduction of the DNN memory requirement while the STOI performance drops only by 2.7%.**

***Index Terms*—neural network quantization, memory footprint reduction, FPGA, hardware accelerators**

I. Introduction

Field programmable gate arrays (FPGAs) are highly desirable in mobile devices as they allow for the design of highly efficient systems, with low-latency and low-power requirements. FPGAs are particularly useful for speeding up signal processing by using specific designed hardware (called hardware accelerator) that runs in parallel with the main CPU, usually embedded in the FPGA itself. Deep neural networks (DNNs) often set the state-of-the-art in many signal processing tasks, e.g., speech separation [1]–[3], speech recognition [4], etc. However, memory footprint and memory bandwidth requirements, and the associated power consumption of DNNs are an issue to be solved for the deployment of a DNN on an FPGA.

Two main approaches have been used to decrease the memory requirements for DNNs: i) *changing the architecture* of the network in order to reduce the parameter number [5]–[8], and ii) *quantizing the parameters* of the network to directly reduce the amount of memory needed for storing them (i.e. reducing the memory footprint) and the memory bandwidth needed to read them [9]–[17]. The first approach involves methods like *parameter pruning and sharing* [5], i.e., removing redundant weights or layers, *knowledge distillation* [6], i.e., retrieving a smaller network from a pre-trained bigger one, and the use of *low-rank factorization* [7] or specific convolutional filters [8]. All these methods produce networks with less computational needs but require a modification in the architecture of the network itself. This is less desirable from an hardware perspective, since each change in the DNN may require a revision of the architecture and a new hardware design.

The second approach comprises methods that employ quantization strategies like normalization [11], uniform and non-uniform quantization for different ranges of values [12], hybrid *floating point blocks* [13], using minimum mean squared error [10], weights clipping and bias correction [14], and per-channel or per-layer different scaling [14]–[16]. Mixed approaches came up too, like *binarized neural network* [17], in which weights and activations are forced to -1 and +1 values. These methods can lead to quantizing the parameters of the network from floating-point (e.g. 32-bit) to n-bit fixed-point representation. The parameters of a DNN are usually stored in external memories (i.e. flash memories) in FPGAs. The access time for a flash memory can be a bottleneck and severely slow down the corresponding calculations. Reducing the bit width of the stored parameters of the DNN reduces memory requirements and improves execution speed. Furthermore, smaller, slower, and cheaper memories can be used by employing low-bit fixed point arithmetic, also resulting in a reduction of the power consumption [9]. However, the parameter quantization can lead to a degradation of the DNN performance and very poor results if too few bits are used (i.e. less than 8) [10].

This paper deals with parameter quantization for reducing the memory requirements. We focus on the use-case of FPGAs, proposing a low-bit quantization method which is based on the non-uniform and dynamic quantization methods [12], [15], [18]–[20]. Our method encodes the parameters of the DNN employing a hardware-oriented approach, using codes that can be stored in slow, external memories, while the actual values can be kept in FPGA-mapped lookup tables (LUT).

The authors wish to acknowledge CSC-IT Center for Science, Finland, for computational resources. Part of the research leading to these results has received funding from the European Research Council under the European Union's H2020 Framework Programme through ERC Grant Agreement 637422 EVERYSOUND.

Our approach distinguishes itself from earlier similar works by the introduction of a Virtual Bit Shift (VBS) scheme that allows the dynamic adjustment of the bit representation of the DNN parameters. VBS mitigates the drawbacks of fixed-point quantization scheme by exploiting the non uniform distribution of the parameters of the DNN, increasing the accuracy and reducing performance loss. The quantization scheme is applied to a speech separation task, achieving a reduction of the memory footprint up to 50% if compared to an 8-bit fixed point representation of the parameters, and yielding a performance reduction of only 2.7% in terms of STOI.

The rest of the document is organized as follows. We present our method in Section II and we briefly introduce the task on which we evaluate our method in Section III. Section IV describes the evaluation procedure, whereas the obtained results are presented and discussed in Section V. Section VI concludes the paper and proposes future research directions.

II. Proposed Quantization Method

Our method takes as input the set of optimized parameters Θ of a trained deep neural network (DNN), and quantizes them with fixed point values of m-bit by applying a non-uniform quantization. The m-bit values are separately encoded, using n-bit codes. The m-bit values and their association with the n-bit codes are placed in a lookup table (LUT), stored in the FPGA-embedded and fast memory blocks (e.g. M9K in Intel's Cyclone IV devices). The n-bit wide codes are stored in an external, relatively slow memory.

A. Quantization of parameters

The quantization of the parameters Θ first starts by quantizing the range of values of Θ into discrete intervals. Then, we assign to each parameter as new value the median of the interval to which the original parameter belongs. This process introduces quantization errors to the values of the parameters, resulting to performance loss. Observing that the values of Θ are generally non-uniformly distributed, we apply a non-uniform scheme for calculating the intervals in order to reduce the quantization error. Figure 1 illustrates the cumulative distribution ϕ of the values of Θ, from a feedforward neural network where the values of Θ are clamped to the range $[-1, 1]$.

Specifically, we express the range of the parameters Θ of a DNN as $A = [a^l, a^h]$, where $a^l \leq \theta \leq a^h,\ \forall\, \theta \in \Theta$. We quantize the range A into discrete intervals using an n-bit encoding scheme, resulting into 2^n intervals $\mathbb{B} = \{B_i\}_{i=1}^{2^n}$, with $B_i = [b_i^l, b_i^u)$, $b_{i-1}^u \leq b_i^l < b_i^u$, and interval span $\Delta_i = b_i^u - b_i^l$. We organize the intervals $\mathbb{B}$ in two disjoint groups: the *internal* partition, $\mathbb{B}^{\text{int}} \subset \mathbb{B}$, where the parameters are densely concentrated, and the *external* partition, $\mathbb{B}^{\text{ext}} \subset \mathbb{B}$, where the parameters are sparsely distributed. We draw all $\mathbb{B}$ intervals belonging to $\mathbb{B}^{int}$ with the same interval span Δ_{int}. On the contrary, each interval $\mathbb{B}$ belonging to $\mathbb{B}^{\text{ext}}$ uses a different interval span. This results to a uniform partitioning of $\mathbb{B}^{int}$ and a non-uniform partitioning of $\mathbb{B}^{ext}$, targeting to a reduced quantization error for all high frequently appearing values of parameters (in $\mathbb{B}^{int}$), at the cost of increasing it for less frequent values (in $\mathbb{B}^{ext}$).

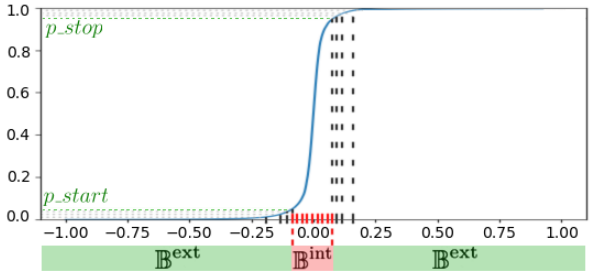

Fig. 1. Cumulative distribution ϕ and partition ranges

We use uniform quantization in $\mathbb{B}^{\text{int}}$ and non-uniform quantization in $\mathbb{B}^{\text{ext}}$. We can define the ratio of number of intervals in the internal and external partitions $R_{\text{B}} = |\mathbb{B}^{\text{int}}|/|\mathbb{B}^{\text{ext}}|$, where $|\cdot|$ is the number of elements in a set, and the probability values p_{start} and p_{stop} denoting the lower and upper boundaries of $\mathbb{B}^{int}$, respectively. We define the number of intervals $|\mathbb{B}^{\text{int}}|$ and $|\mathbb{B}^{\text{ext}}|$ as

$$|\mathbb{B}^{\text{ext}}| = \left\lfloor \frac{2^n}{1+R_{\text{B}}} \right\rfloor + c, \text{ and} \tag{1}$$

$$|\mathbb{B}^{\text{int}}| = 2^n - |\mathbb{B}^{\text{ext}}|, \text{ where} \tag{2}$$

$$c = \begin{cases} 1, & \text{if } \left\lfloor \frac{2^n}{1+R_{\text{B}}} \right\rfloor \text{ is odd} \\ 0, & \text{if } \left\lfloor \frac{2^n}{1+R_{\text{B}}} \right\rfloor \text{ is even.} \end{cases} \tag{3}$$

For the external partition, we uniformly split the range of ϕ and invert it back to get the interval B_i^{ext} as

$$B_i^{\text{ext}} = [\phi^{-1}(i \cdot \Delta_i^{\phi}), \phi^{-1}((i+1) \cdot \Delta_i^{\phi})], \text{ where} \tag{4}$$

$$i \in \left[0, \frac{|\mathbb{B}^{\text{ext}}|}{2}\right) \cup \left[\frac{|\mathbb{B}^{\text{ext}}|}{2} + |\mathbb{B}^{\text{int}}|, |\mathbb{B}^{\text{int}}| + |\mathbb{B}^{\text{ext}}|\right), \text{ and} \tag{5}$$

$\Delta_i^{\phi} = \frac{2 \cdot p_{\text{start}}}{|\mathbb{B}^{\text{ext}}|}$ is the interval span in the range of ϕ and hence the corresponding Δ_i is non uniform. Finally, we uniformly divide $\mathbb{B}^{\text{int}}$ with step

$$\Delta_{int} = \frac{\phi^{-1}(p_{\text{stop}}) - \phi^{-1}(p_{\text{start}})}{|\mathbb{B}^{\text{int}}|}, \tag{6}$$

thereby obtaining the interval B_i^{int} as

$$B_i^{\text{int}} = [\phi^{-1}(p_{\text{start}}) + i \cdot \Delta_{int}, \\ \phi^{-1}(p_{start}) + (i+1) \cdot \Delta_{int}]. \tag{7}$$

For any such interval B_i, the quantized level $\tilde{\theta}_i$ is computed as the m-bit quantized mean of the parameters lying in the B_i interval as

TABLE I
LUT OF THE n-BIT CODE, THE m-BIT WIDE $\tilde{\theta}_i$, AND THE CORRESPONDING PARTITION.

Code $\|_{10}$	$\tilde{\theta}_i$	Partition
0000	-0.3359375	ext
0001	-0.15625	ext
0100	-0.0703125	int
⋮	⋮	⋮

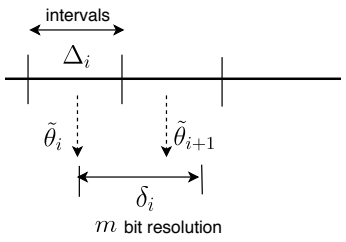

Fig. 2. Example with the resolution of quantization Δ_i being smaller than the resolution of m-bit encoding δ_i.

$$\tilde{\theta}_i = \frac{\sum_{\theta\in\Theta} \mathbb{1}_{\theta\in B_i}\cdot\theta}{\sum_{\theta\in\Theta}\mathbb{1}_{\theta\in B_i}}\Big|_{m\text{-bit}}, \quad \text{where} \tag{8}$$

$$\mathbb{1}_{\Xi} = \begin{cases} 1, & \text{if } \Xi, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

and $\cdot|_{m\text{-bit}}$ means the m-bit representation. Using $\tilde{\theta}_i$ from Eq. (8), we can quantize the parameters Θ of a DNN with an m-bit representation and then represent i indexes with n-bit codes $\tilde{i}_n$, where $n < m$. At this point, the memory requirement is reduced by storing n-bit $\tilde{i}_n$ codes instead of m-bit $\tilde{\theta}_i$ values. The latter are retrieved by reverting the above described correspondence. A lookup table (LUT) which stores the relationship between the n-bit $\tilde{i}_n$ code and its corresponding m-bit value (and the partition it belongs to, i.e. external or internal) is used. An example of such LUT is shown in Table I.

The advantage of this approach is that the external memory is asked to store $\tilde{i}_n$, with lesser bit-width if compared to $\tilde{\theta}_i$. This leads to reduced memory dimension and reduced access time. On the other side, an additional LUT is needed. However, due to its small dimensions (n m-bit words), it can easily be implemented by using the FPGA-embedded memory blocks, which are usually faster than any external memory.

B. Virtual bit shift

If the range from p_{start} to p_{stop} is very narrow, we may end up to an interval span Δ_i which is smaller than δ_i. Given that the smallest number that can be represented using signed m-bit encoding is $2^{-(m-1)}$, then we might end up to a case where adjacent intervals map to the same m-bit value. Figure 2 depicts such a situation, where $\tilde{\theta}_i$ and $\tilde{\theta}_{i+1}$ are the values for quantized parameters corresponding to the i-*th* and (i+1)-*th* interval, respectively, that share the same m-bit representation. To avoid this problem, there should be $\delta_i < \Delta_i$.

TABLE II
AN EXAMPLE OF VIRTUAL BIT SHIFT ENCODING FOR $n = 4$, $m = 8$, AND $k = 4$.

Code	value	binary value^{12bit}	LUT value
0100	0.02099609	.000001010110	01010110

Let us adopt in our method a quantization scheme for $\mathbb{B}^{\text{int}}$ that solves the above problem. Since δ_i depends on the bit-width of the encoding process, a higher resolution can in principle be achieved by quantizing $\tilde{\theta}_i$ using a larger number (e.g. $m + k$) of bits. Since for all $\tilde{\theta}_i \in \mathbb{B}^{\text{int}}$, we have

$$\tilde{\theta}_i \leq \max(|\phi^{-1}(p_{start}), \phi^{-1}(p_{stop})|), \tag{10}$$

we can find $k \in \mathbb{N} : \tilde{\theta}_i < 2^{-k}$. This implies that the k most significant bits will contain either zeros or the sign bit and are thus redundant for storage purposes. The sign bit can be stored in the n bit indexing code $\tilde{i}_n$ itself (e.g. values with code $\tilde{i}_n = \{0000, 0001...0111\}$ will have negative value). Storing in LUT only m least significant bits from an $m + k$-bit representation of $\tilde{\theta}_i$ is like a k bit left shift, which means a 2^k multiplication in binary arithmetic. Let us denote m least significant bits for $\tilde{\theta}_i$ as $\tilde{\theta}_i^m$, so that we have

$$\tilde{\theta}_i^m = \tilde{\theta}_i \cdot 2^k. \tag{11}$$

We can store $\bar{\theta}_i^m$, from which the actual parameter values $\tilde{\theta}_i$ can be retrieved using Eq. (11). Basically we perform a range adjustment by virtual bit shift of the actual parameter values. An example is shown in Table II. The resolution error can now be avoided by complying to a less stringent condition than before, namely, $\delta_i^{m+k} < \Delta_i$, where δ_i^{m+k} is the resolution of $m + k$ bit encoding. Thus, absolute values, signs, and representation range information can be stored at once in the $\tilde{i}_n$ code (e.g. values with $\tilde{i}_n = 0000, ..., 0111$ are negative, with $\tilde{i}_n = 1000, ..., 1111$ are positive, and with $\tilde{i}_n = \{0000, ..., 0011\} \cup \{1011, ..., 1111\}$ have δ_i^m resolution). At this point an n-bit quantization scheme is obtained which, by means of a LUT and a proper $\tilde{i}_n$ code, uses two different range representations allowing the reduction of the quantization errors.

III. DNN-BASED SPEECH ENHANCEMENT

We applied the proposed quantization scheme to a speech enhancement task using a feedforward DNN. The DNN takes as input a noisy voice signal, and predicts the clean voice. The input noisy signal is represented by using the magnitude short-time Fourier transform (STFT) and is then scaled in order to properly calculate the spectrum magnitude and phase by using a coordinate rotation digital computer (CORDIC) algorithm [21] based on integer arithmetic. CORDIC was chosen due to its low need of computational resources, constituting a further advantage in optimizing system's architecture for an actual hardware implementation of the presented method. N frames, $\{\tilde{\mathbf{x}}_{t-N+1}, \tilde{\mathbf{x}}_{t-N+2}, \tilde{\mathbf{x}}_{t-N+3} \ldots, \tilde{\mathbf{x}}_t\}$, of these features are first stacked together and then fed to the DNN to estimate

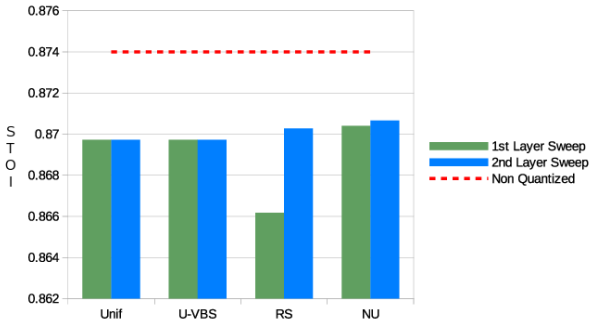

Fig. 3. STOI enhancement performance of different 8-bit quantization approaches being swept on first/second DNN layer keeping the other with 8-bit uniform quantization.

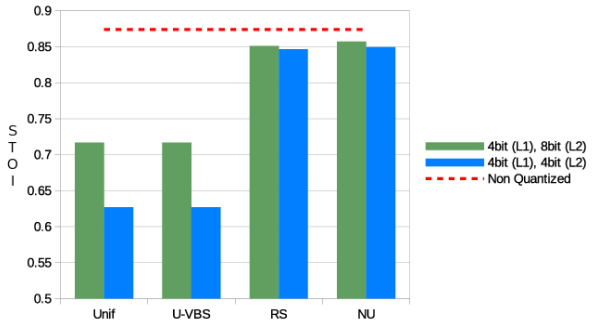

Fig. 4. STOI enhancement performance of different quantization approaches with 4/8-bit quantization for first and second layer.

the clean speech magnitude spectrum $\mathbf{x}_t$. The stacking of features is done to allow the DNN to implicitly model temporal dependencies. The CORDIC algorithm is then applied again on $\mathbf{x}_t$ to restore unmodified phase information. The values thus obtained are scaled back and converted back to the time domain by the inverse fast Fourier transform (IFFT) and overlap-add operations.

IV. Evaluation

Synthetic mixtures of voice and noise are created using the Wall Street Journal (WSJ0) dataset for speech and TUT Acoustic scenes 2016 development dataset [22] for noise, in order to validate and evaluate the novel quantization scheme adopted. The disturbing signals consist of sound recordings from 15 real-world environments, e.g., cafe, train, metro station, etc. A random speech signal is selected from the dataset and an equal-length noise segment is superimposed to it. The training and validation data consist of about 12,000 (around 20 hours) and 5000 mixtures (around 8 hours), respectively. Similarly, the test data consists of about 2800 mixtures (around 5 hours). The speech and noise signals are mixed with a randomly chosen signal to noise ratio (SNR) in the set {0, 5} dB. The native sampling rate for noise signals was 44.1 kHz, so that it was down-sampled to 8 kHz, to match the native sampling rate of WSJ0 audio. The short term objective intelligibility (STOI) [23] metric was used as a measure of intelligibility of the extracted speech.

The STFT features are extracted with Hann window of 128 sample (16 ms) with 50% overlap. Eight input frames are stacked and fed to a two-layer feedforward network with 256 and 129 neurons in input and hidden layer, respectively. The rectified linear unit is used as non-linearity for each layer. The Adam optimization [24] with default parameters is used. For training networks, PyTorch [25] library is used, and for audio processing, Librosa [26] library is used. Since our focus is to fixed point arithmetic devices, the network weights and biases are clamped to the range (-1, +1) in order to avoid overflow and reduce the number of bits needed for correct numeric representation. The DNN weights have been quantized using $n = 4$, $m = 8$, and $k_{1^{st}layer} = 3$, $k_{2^{nd}layer} = 2$, obtained by choosing $ratio = 1$, $p_{\text{start}} = 0.04$ and $p_{\text{stop}} = 0.96$.

TABLE III
Evaluation results in terms of STOI and memory reduction. n-U is for n-bit uniform and n-NU for n-bit non-uniform quantization. Memory reduction is with respect to 8 bit fixed-point uniform quantization as reference.

Quantization scheme			
1st layer	**2nd layer**	**STOI/STOI Loss**	**Memory usage (bytes) /reduction**
8-U	8-U	0.87/-0.002 (00.2%)	297216/ –
4-U	4-U	0.63/-0.246 (28.2%)	148608/ -50.0%
4-NU	4-NU	0.85/-0.024 (02.7%)	148608/ -50.0%
4-NU	8-U	0.86/-0.016 (01.8%)	165120/ -44.4%
No quantization		0.87/ –	
Mix		0.84/ –	

The $ratio$, p_{start} and p_{stop} have been chosen empirically after optimizing for the test data. The partition used was symmetric ($p_{stop} = 1 - p_{start}$). DNN's output was calculated by sweeping both $ratio$ and p_{start} and finding the best results for the above-mentioned values. The biases used are the 8-bit-uniform quantized values.

V. Results

Table III compares the STOI values obtained with different approaches over 2800 noisy samples. 8-bit-uniform (8-U) quantization gives good results with a very small degradation in STOI (0.21%) compared to non-quantized network, while 4-bit-uniform (4-U) quantization leads to a drastic fall in the performance, ending up in a reconstructed speech even less intelligible than the original noisy signal. On the other hand, the 4-bit non-uniform (4-NU) quantization proposed in this paper yields a very good STOI result that is only 2.7% worse as compared to the non-quantized network. In addition, the memory requirement is halved in comparison to the 8-bit uniform quantization, and the memory bandwidth is also decreased at the same time.

Figure 3 compares the different approaches by using 8-bit quantization (uniform and non uniform) for different layers, and how the proposed quantization scheme consisting of range split (RS), i.e. the partitioning of $\mathbb{B}^{int}$ and $\mathbb{B}^{ext}$, and virtual bit shift (VBS) affects the performance. For each simulation, one of the two layer is kept at 8-bit uniform quantization while the other is swept between the following four approaches: uniform 8-bit quantization (U), 8-bit uniform quantization with virtual

bit shift (U-VBS), 8-bit range split (RS), and 8-bit range split with virtual bit shift (NU). It can easily be noticed how the performance suffers, for the first layer sweep, as $\delta_i > \Delta_i$ when range split is used without VBS. Figure 4 shows the effect of these approaches for two cases: 4-bit quantization for both layers, and, 4-bit for the first and 8-bit for the second layer. The usage of range split instead of an uniform scheme clearly enables better performances when low-bit quantization are used and further improvements are allowed when both RS and VBS are used(NU). Finally, a subjective listening examination highlighted an almost non audible difference between the non quantized and 8-bit uniform quantized output while using 4-bit uniform quantization resulted in having the noise still clearly present in the output and an audible high-pitched distortion on both signal and noise. Results of the 4-bit non uniform(NU) quantization present instead an appreciable reduction of the background noise and less audible distortion compared to the 4-bit uniform quantization.

VI. Conclusions

In this paper we presented a method for quantizing the values of the parameters of a deep neural network, focusing on the use-case of FPGAs. Our method employs a companding approach and therefore a variable quantization step depending on the distribution of the values of the parameters, having smaller quantization intervals where there are many values of parameters but less quantization intervals where there are not many values of parameters. In particular, the results presented above shows how the combined usage of LUTs and VBS technique enables a reduction of quantization errors even when low-bit quantization of weights are used. The method does not require any change or pruning of the network, so no retrain is needed. The case studied shows a two-layer feed-forward neural network, from which it emerges that a dramatic reduction of the memory requirements is obtained (50%) with only a slight reduction of the performance. Further research should concern the application of the method to deeper networks and the usage of non symmetrical range split or of a customized multiplying architecture for the weighting of the input values.

References

[1] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 27, no. 8, pp. 1256–1266, 2019.

[2] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," IEEE/ACM Transactions on Audio, Speech, and Language Processing , vol. 26, no. 10, pp.1702–1726, 2018.

[3] Z.-Q. Wang, J. Le Roux, and J. R. Hershey, "Alternative objective functions for deep clustering," in Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018, pp. 686–690.

[4] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina et al., "State-of-the-art speech recognition with sequence-to-sequence models," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE,2018, pp. 4774–4778.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.

[6] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2006, pp. 535–541.

[7] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in 2013 IEEE international conference on acoustics, speech and signal processing. IEEE, 2013, pp. 6655–6659.

[8] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," arXiv preprint arXiv:1710.09282, 2017.

[9] W. Dally, "High-performance hardware for machine learning," NIPS Tutorial, 2015

[10] Y. Choukroun, E. Kravchik, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference," arXiv preprint arXiv:1902.06822, 2019.

[11] R. A. Solovyev, A. A. Kalinin, A. G. Kustov, D. V. Telpukhov, and V. S. Ruhlov, "FPGA implementation of convolutional neural networks with fixed-point calculations," arXiv preprint arXiv:1808.09945, 2018.

[12] E. Park, S. Yoo, and P. Vajda, "Value-aware quantization for training and inference of neural networks," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 580–595.

[13] M. Drumond, T. LIN, M. Jaggi, and B. Falsafi, "Training DNNs with hybrid block floating point," in Advances in Neural Information Processing Systems 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 453–463.

[14] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post training 4-bit quantization of convolution networks for rapid-deployment," CoRR, abs/1810.05723, vol.1, p.2, 2018.

[15] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song et al., "Going deeper with embedded fpga platform for convolutional neural network," in Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. ACM, 2016, pp. 26–35.

[16] P. Judd, J. Albericio, T. Hetherington, T. Aamodt, N. E. Jerger, R. Urtasun, and A. Moshovos, "Reduced-precision strategies for bounded memory in deep neural nets," arXiv preprint arXiv:1511.05236, 2015.

[17] 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 preprint arXiv:1602.02830, 2016.

[18] S. Seo and J. Kim, "Efficient weights quantization of convolutional neural networks using kernel density estimation based non-uniform quantizer," Applied Sciences, vol. 9, p. 2559, 06 2019.

[19] N. Liss, C. Baskin, A. Mendelson, A. M. Bronstein, and R. Giryes, "Efficient non-uniform quantizer for quantized neural network targeting reconfigurable hardware," arXiv preprint arXiv:1811.10869, 2018.

[20] 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), June 2017, pp. 1–6.

[21] J. E. Volder, "The cordic trigonometric computing technique," IRE Transactions on Electronic Computers, no. 3, pp. 330–334, 1959.

[22] A. Mesaros, T. Heittola, and T. Virtanen, "TUT acoustic scenes 2016, development dataset," Feb 2016.

[23] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in 2010 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2010, pp. 4214–4217.

[24] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. International Conference on Learning Representations, 2014.

[25] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in NIPS Autodiff Workshop, 2017.

[26] B. McFee et al., "librosa 0.5.0," Feb. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.293021