L2L: A Highly Accurate Log_2_Lead Quantization of Pre-trained Neural Networks

Salim Ullah*, Siddharth Gupta§, Kapil Ahuja§, Aruna Tiwari§, Akash Kumar*
*Technische Universität Dresden, Germany
§Indian Institute of Technology Indore, India
*{salim.ullah, akash.kumar}@tu-dresden.de, §{ms1804101006, kahuja, artiwari}@iiti.ac.in

***Abstract*—Deep Neural Networks are one of the machine learning techniques which are increasingly used in a variety of applications. However, the significantly high memory and computation demands of deep neural networks often limit their deployment on embedded systems. Many recent works have considered this problem by proposing different types of data quantization schemes. However, most of these techniques either require post-quantization retraining of deep neural networks or bear a significant loss in output accuracy. In this paper, we propose a novel quantization technique for parameters of pre-trained deep neural networks. Our technique significantly maintains the accuracy of the parameters and does not require retraining of the networks. Compared to the single-precision floating-point numbers-based implementation, our proposed 8-bit quantization technique generates only $\sim 1\%$ and $\sim 0.4\%$, loss in the top-1 and top-5 accuracies respectively for VGG16 network using ImageNet dataset.**

***Index Terms*—machine learning, neural networks, quantization**

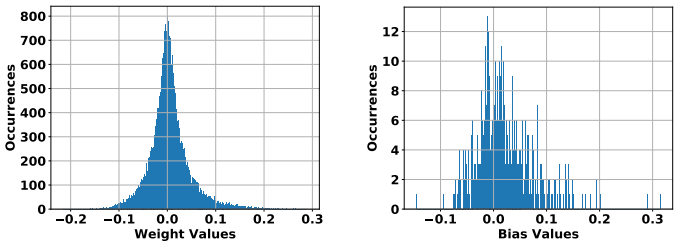
I. Introduction

Deep neural networks (DNNs) are the machine learning models which have achieved promising classification accuracies on different recognition problems such as images, speech, and natural language processing [1]–[3]. However, the DNNs are computationally expensive and have very high memory footprints. For these reasons, high-performance parallel architectures, such as graphics processing units (GPUs), are typically used for the training of DNNs. Further, to provide high computational accuracies, these systems typically use single-precision floating-point numbers. However, the high power consumption and memory requirements of these CPUs and GPUs-based DNN models make them an infeasible choice for embedded devices on edge. A plethora of recent works has proposed different types of data representation techniques and hardware accelerators to reduce the memory and power budgets of trained DNN models. Most of these techniques represent the parameters of a trained network in low precision fixed-point number systems by utilizing different types of quantization schemes. Decreasing the precision of parameters helps in reducing the memory footprint, interconnect bandwidths for transferring the intermediate results, and the required computational complexity. However, the overall output accuracy of a quantized DNN model is degraded due to the errors induced by quantization. For some quantized DNNs, the final output accuracy can be slightly improved by retraining the network with quantized parameters and adjusting the quantized parameters accordingly. However, the retraining of a quantized DNN is a time, energy, and computational resource-consuming operation. Therefore, there is always the need for defining quantization methods which can produce high-quality results without retraining the networks. To this end, we claim the following contributions in this work:

- We present a novel quantization technique for pre-trained DNNs. With our proposed quantization method, higher output accuracies can be achieved than those obtained with state-of-the-art methods.
- Our technique identifies the positions of the leading 1 and the following most significant bits in a fraction to represent a quantized number. Our analysis shows that the identification and processing of the most significant bits can remarkably improve the precision of the quantized number.
- Our proposed quantization technique is highly accurate and significantly retains the precision of the parameters. Therefore, retraining of the quantized network is not required to recover from the quantization errors.
- Our proposed quantization technique replaces the computationally intensive and resource-hungry *Multiply*-operators in a DNN model with *bit-shift* and *add* operators.

II. Related Work

Quantization of neural networks can be achieved either by training the networks with quantized parameters or applying different quantization schemes on the trained network parameters. For example, the works in [4]–[9] utilize specialized methods to train networks with different fixed-point low-precision parameters. The techniques proposed in [6]–[9] have even reduced the precision of trained parameters to only 1-2 bits. As the training of a network is a learning process, the in-training quantization can heal many of the quantization induced errors in the final output of the network. However, this technique cannot be utilized for the networks already trained with floating-point numbers. To quantize the parameters of a pre-trained network, the works in [10]–[14] have proposed different quantization schemes. The technique proposed in [11] utilizes different bit-widths for each layer to reduce the errors in final output accuracy. To replace the computationally costly multiply-operation with bit-shifts, the works in [10], [12]–[14] have used the power of 2 quantizations for pre-trained networks' parameters. However, most of these techniques require a fine-tuning (retraining) step to reduce the errors induced due to quantization. The authors of [14] have avoided the post-quantization fine-tuning step by computing the quantization step size using an iterative approach. In their proposed technique, the optimal quantization step sizes for features and parameters are computed by iteratively adjusting the step size for each data structure in each layer and recording the generated errors in the layer under consideration. The independent



(a) Weights Distribution (b) Biases Distribution

Fig. 1: Distribution of weights and biases for pre-trained VGG-16 [15] (a) Conv1_2 layer (b) Conv4_1 layer

quantization of each data structure and the iterative approach for finding the quantization step size may take a significantly long time for the quantization of very deep neural networks.

The quantization technique proposed in this paper, referred to as *log_2_lead*, can substantially maintain the precision of pre-trained network parameters and therefore, does not require retraining of the quantized network. The quantization technique of *log_2_lead* is also independent of neurons' criticalities and does not require any time consuming iterative process for computing the quantized parameters.

III. Quantized DNNs

In this section, we briefly present a description of the usually employed techniques for the quantization of pre-trained DNNs.

A. Overview of DNNs

A typical DNN consists of multiple stacked layers of primary computational units called *Neurons*. The commonly used layers in a DNN are *convolution*, *pooling*, and optionally a few *fully connected* layers. A DNN is trained using these layers with floating-point numbers. For example, Fig. 1 shows the distribution of weights and biases of two different layers for a trained VGG16 [15] network. Next, we describe the commonly used quantization schemes to reduce the memory map of a trained DNN.

B. Commonly Used Quantization Techniques

Linear Quantization

The linear quantization of a data tensor x from floating-point precision to N-bit fixed-point precision is described by Eq. 1–3. The step size Δ in the Eq. 1 represents the minimum possible increment in the quantized value x_{quant}. Eq. 2 is used to align Δ with the maximum and minimum allowed value in N-bit precision[1]. Finally, the Δ is used in Eq. 3 for computing the quantized value x_{quant}. As the computation of the Δ is based on the maximum absolute value of x, this method creates robust quantization errors for far outliers in x. Fig. 2(a) and Fig. 3(a) show the linear quantization of the weights and biases presented earlier in Fig. 1. The linear quantization provides a limited number of uniformly separated discrete fixed-point values for representing the Float32 values.

$$\Delta = clip\left(\frac{max\ (\mid x \mid)}{2^{N-1}}, 2^{N-1}, 2^{-(N-1)}\right) \tag{1}$$

[1]2** shows power of 2.

$$\Delta = 2\ **\ clip\left(round(log_2(\Delta)), N-1, -N+1\right) \tag{2}$$

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

Power of 2 Quantization (log_2 Quantization)

The power of 2 quantization (also referred to as log_2 quantization) has been used by many state-of-the-art works to replace the multiplication operations in DNNs with bit-shifts. Eq. 4 and Eq. 5 represent an elementary scheme of power of 2 quantization for mapping a floating-point value x to a power of 2 value x_{quant}. Fig. 2(b) and Fig. 3(b) show the power of 2 quantization of the Conv1_2 weights and Conv4_1 biases of pre-trained VGG16 network. Compared to the linear quantization, the power of 2 quantization have many repeatedly occurring values which are not close to 0.

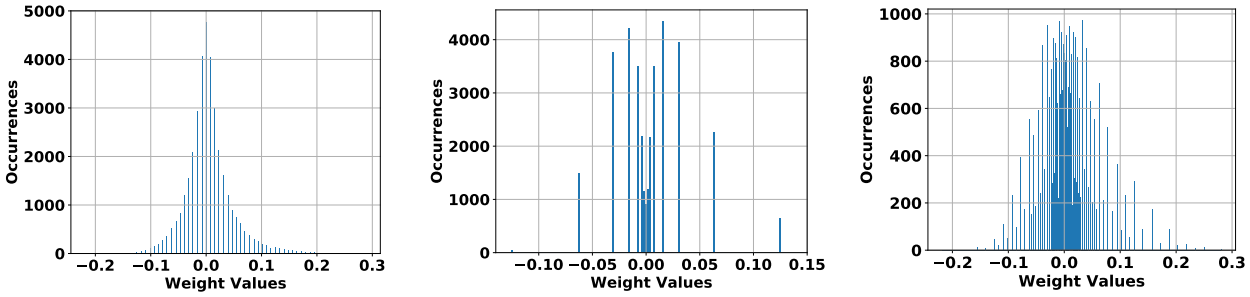
$$\hat{x_{quant}} = clip\left(round\left(log_2(\mid x \mid)\right), 0, N\right) \tag{4}$$

$$x_{quant} = sign(x)2^{\hat{x_{quant}}} \tag{5}$$

IV. Proposed Technique

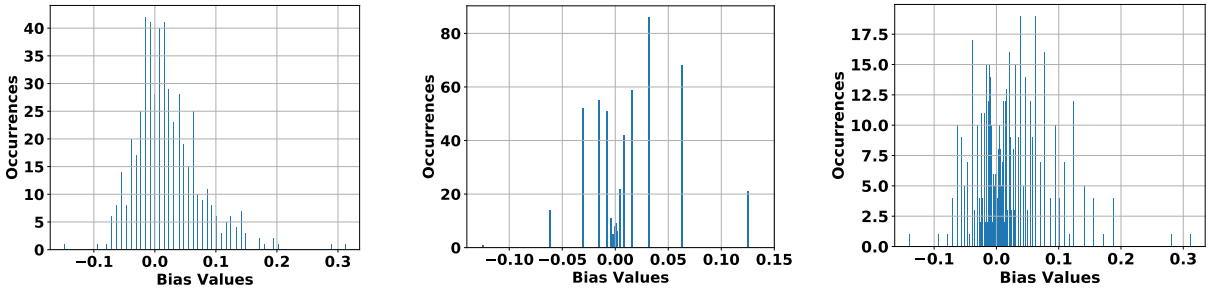
The proposed fixed-point quantization technique attempts to minimize the quantization induced errors by identifying and storing the most significant 1's in the parameters of a pre-trained DNN. As the trained parameters are represented in single-precision floating-point scheme (32 bits), the identification of the 1's which have higher significance can notably reduce the quantization generated errors. To identify the significant 1's in Float32-based parameters, Fig. 4 gives a histogram of the leading 1's in all the weights and biases of two different layers of VGG16 network. As shown in Fig. 4(a), the leading 1 for most of the weights of Conv1_2 layer occur at bit position -6. However, there also some weights with very low-values and having the leading 1 occurring at bit position -15. A typical log_2 quantization as described in Eq. 4 and Eq. 5 would only find a single leading one for each weight and discard all other bit locations. Due to ignoring all other bit locations for computing the quantized value, the log_2 quantization can introduce substantial quantization errors.

In our proposed *log_2_lead* technique, we also apply log_2 to detect the position of the leading 1 in a fraction. However, to heal the quantization errors, *log_2_lead* also observes the following bits locations after the leading one location. For N-bit fixed-point quantization, Fig. 5 shows the template of our proposed *log_2_lead* quantization scheme. The first bit is reserved for showing the sign of the quantized parameter. Following $\lceil\frac{N-1}{2}\rceil$ bits are reserved for storing the location of the leading-1 in the original non-quantized parameter. The remaining bits are used to store the values of the following $\lfloor\frac{N-1}{2}\rfloor$ bits after the leading 1. For example, for the leading-1 histograms in Fig. 4, the leading-1 at bit position -12 would be represented as a binary number '1100' by the '*leading one location*' field in the template shown in Fig. 5. Moreover, to increase the precision of the quantized value, we always analyze $\lfloor\frac{N-1}{2}\rfloor+1$ bits after the leading 1 and round the values to $\lfloor\frac{N-1}{2}\rfloor$ bits accordingly. A similar rounding technique is as also discussed in [16]. The rounding of $\lfloor\frac{N-1}{2}\rfloor+1$ bits further helps in retaining the precision of a rounded number. For



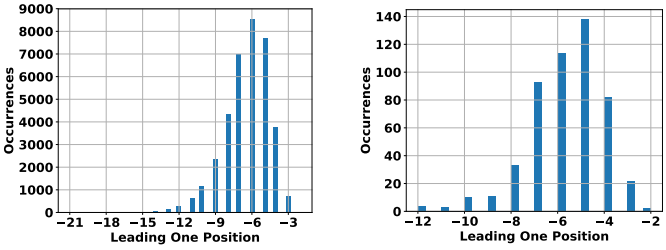
(a) Linear quantization (b) Power of 2 quantization (c) *log_2_lead* quantization

Fig. 2: Quantization of pre-trained weights of Conv1_2 layer of VGG-16 [15]



(a) Linear quantization (b) Power of 2 quantization (c) *log_2_lead* quantization

Fig. 3: Quantization of pre-trained biases of Conv4_1 layer of VGG-16 [15]



(a) Weights of a single layer (b) Biases of a single layer

Fig. 4: Histogram of leading 1's for all weights and biases for pre-trained VGG-16 [15] (a) Conv1_2 layer (b) Conv4_1 layer

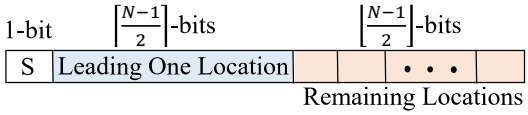


Fig. 5: Template for proposed quantization technique

example, Fig. 6 shows an example of quantizing a fractional number 0.217884 using the proposed *log_2_lead* technique. The leading 1 is found at bit location -3 which is stored in our template as a binary number 0011. After the leading 1, following four bits are analyzed and rounded to binary pattern 110. The corresponding quantized value is 0.21875. Through our testing on the trained parameters of different benchmark DNNs, we have found that quantization errors can be healed significantly in the final output of a DNN by utilizing 8 bits for *log_2_lead* quantization. Fig. 2(c) and Fig. 3(c) shows

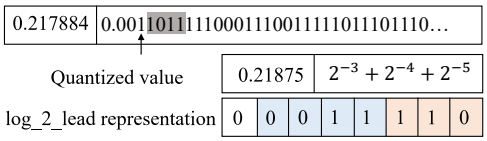


Fig. 6: *log_2_lead* quantization example

the quantization of weights and biases of two different layers of VGG16 (already presented in Fig. 1). It can be observed that the proposed *log_2_lead* technique provides more discrete fixed-point values and better coverage for quantizing Float32-based parameters than the linear and power of 2 quantization schemes.

V. Experimental Setup and Results

For the application-level evaluation of *log_2_lead*, linear quantization, and power of 2 quantization, we have used TensorFlow framework [18]. Using the framework, we have implemented on VGG16 Network [15] with ImageNet dataset [17] to test the efficacy of our proposed quantization technique for classification accuracy of quantized netwroks. These results are discussed in Section V-A. The proposed *log_2_lead* technique replaces the resource-demanding *multiply* operation with *bit-shifts* and *add* operations. Using these two operations, we have implemented a *Processing Element* (PE) for the multiplication of *log_2_lead*-based weights and features. The PE is implemented in VHDL for Xilinx Virtex-7 xc7v585tffg1157-3 FPGA using Xilinx-Vivado-17.4. The resource utilization of our proposed PE is compared with Vivado standard multiplier IPs. The implementation results are presented in Section V-B.

TABLE I: Classification accuracy of VGG16 network [15] on ImageNet dataset [17] with different quantization schemes

Quantization	VGG16 [15]	Top-1 [%]	Top-5 [%]
	Float32	64.72	85.74
weights, biases quantized	8-bit linear	59.8	82.55
	Power of 2	0.1	0.63
	log_2_lead	64.51	85.64
	Float32 vs *log_2_lead*	-0.21	-0.1
weight, biases and activations quantized	8 bit linear	59.83	82.55
	Power of 2	1.16	7.48
	log_2_lead	64.05	85.34
	Float32 vs *log_2_lead*	-0.67	-0.4

TABLE II: Resource utilization of proposed PE and Vivado multiplier IPs along with corresponding classification accuracies for VGG16 [15] network on ImageNet dataset [17]

Design	Quantization Scheme	Feature size (bits)	Weight size (bits)	Resources (LUTs)	Achieved Accuracy	
					Top-1 [%]	Top-5 [%]
PE	*log_2_lead*	8	8	67	64.47	85.63
IP	Linear	8	8	85	59.83	82.55
IP	Linear	8	9	89	64.26	85.34
IP	Linear	8	10	109	64.52	85.56
IP	Linear	8	11	111	64.65	85.72
IP	Linear	8	18	166	64.67	85.7

A. Image Classification

To evaluate the efficacy of our proposed technique on classification task, we have also tested it on ImageNet dataset [17]. For the $50,000$ validation images, the pre-trained network has 64.72% and 85.74% Top-1 and Top-5 classification accuracies using the single-precision Float32-based parameters and activations. We have applied various 8-bit quantization schemes on the pre-trained parameters and evaluated for classification accuracy. These results are presented in Table I. For the first experiment, the activations have Float32 precision, and the weights and biases are quantized to 8-bit precision. Compared to the linear quantization and power of 2 quantization, our proposed technique *log_2_lead* produces better Top-1 and Top-5 classification accuracies. The log_2_lead scheme reduces the Top-1 and Top-5 percentage classification accuracy only by 0.21 and 0.1, respectively. For the second experiment, all data structures are quantized to 8-bit precision. The log_2_lead quantization scheme still produces better classification accuracy than the power of 2 and linear quantization schemes. Compared to the baseline Float32-based classification, log_2_lead has only 0.67 and 0.4 drops in the Top-1 and Top-5 percentage classification accuracies, respectively.

B. Hardware Implementation Results

Utilizing the 6-input lookup tables (LUTs) and the associated carry chains of Xilinx FPGAs, we have implemented a processing element (PE) for the log_2_lead-based multiplication. Table II compares the implementation results of our proposed 8-bit PE with Xilinx Vivado area-optimized multiplier IP for various sizes. It also shows the corresponding classification accuracy of quantized VGG16 for ImageNet dataset for *log_2_lead* and linear quantization with different data sizes for weights. The activations (feature size) are 8-bit linear quantized. The baseline Float32-based Top-1 and Top-5 classification accuracies are 64.72% and 85.74%, respectively. The log_2_lead scheme provides better classification accuracy with a fewer number of utilized LUTs for its PE than the linear quantization with 8×8 multipliers. Even with 18 bits of precision, the accuracy of linearly quantized weights design is only marginally better than log_2_lead, despite consuming more than twice the area of log_2_lead design.

VI. Conclusion

In this paper, we presented a novel quantization scheme for deep neural network. Our proposed technique analyzes Float32-based parameters of a trained network and records the bit positions of the most important bits. Our proposed technique is highly accurate and does not require the traditional accuracy-recovery retraining of the quantized network. Using bit-shifts and addition operations, we have also presented an efficient implementation of a PE for the multiplication of weights and features.

Acknowledgment

This work is supported in part by the German Research Foundation (DFG) within the Cluster of Excellence, "Center for Advancing Electronics Dresden" (CfAED) at the Technische Universität Dresden.

References

[1] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," ICCV 2015.doi: 10.1109/ICCV.2015.123

[2] L. Deng, G. Hinton and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: an overview," ICASSP 2013.

[3] T. Young, D. Hazarika, S. Poria and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing [Review Article]," IEEE Computational Intelligence Magazine, vol. 13, no. 3, pp. 55-75, Aug. 2018.

[4] Z. Lin, M. Courbariaux, R. Memisevic and Y. Bengio, "Neural networks with few multiplications." arXiv preprint arXiv:1510.03009 (2015).

[5] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen and Y. Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," arXiv preprint arXiv:1606.06160 (2016).

[6] M. Courbariaux, Y. Bengio and J. P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," Advances in neural information processing systems (pp. 3123-3131).

[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," In European Conference on Computer Vision, Springer, Cham, 2016.

[8] F. Li, B. Zhang and B. Liu. "Ternary weight networks," arXiv preprint arXiv:1605.04711 (2016).

[9] C. Zhu, S. Han, H. Mao and W. J. Dally, "Trained ternary quantization," arXiv preprint arXiv:1612.01064 (2016).

[10] H. Tann, S. Hashemi, R. I. Bahar and S. Reda, "Hardware-software codesign of accurate, multiplier-free Deep Neural Networks," DAC 2017.

[11] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," ICML 2016.

[12] S. Vogel, M. Liang, A. Guntoro, W. Stechele and G. Ascheid, "Efficient hardware acceleration of CNNs using logarithmic data representation with arbitrary log-base," ICCAD 2018.

[13] D. Miyashita, E. H. Lee and B. Murmann, "Convolutional neural networks using logarithmic data representation," arXiv preprint arXiv:1603.01025 (2016).

[14] S. Vogel, J. Springer, A. Guntoro and G. Ascheid, "Self-Supervised Quantization of Pre-Trained Neural Networks for Multiplierless Acceleration," DATE 2019.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556 (2014).

[16] D. M. Loroch, F. J. Pfreundt, N. Wehn and J. Keuper, "Tensorquant: A simulation toolbox for deep neural network quantization," in Proceedings of the Machine Learning on HPC Environments, p. 1. ACM, 2017.

[17] O. Russakovskyet al., "ImageNet large scale visual recognition challenge," International Journal of Computer Vision, vol. 115, no. 3, pp.211–252, 2015.

[18] M. Abadi, A. Agarwal, P. Barham, E. Brevdo et al. "TensorFlow: Large-scale machine learning on heterogeneous systems, 2015." Software available from tensorflow. org 1, no. 2 (2015).