Received May 29, 2020, accepted June 17, 2020, date of publication June 26, 2020, date of current version July 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3005286

ALigN: A Highly Accurate Adaptive Layerwise Log_2_Lead Quantization of Pre-Trained Neural Networks

SIDDHARTH GUPTA[1], SALIM ULLAH[2], KAPIL AHUJA[1], ARUNA TIWARI[1], (Member, IEEE), AND AKASH KUMAR[2], (Senior Member, IEEE)

[1]Department of Computer Science and Engineering, Indian Institute of Technology Indore, Indore 453552, India
[2]Department of Computer Science, Technische Universität Dresden, 01062 Dresden, Germany

Corresponding authors: Salim Ullah (salim.ullah@tu-dresden.de) and Akash Kumar (akash.kumar@tu-dresden.de)

This work is supported by the German Research Foundation (DFG) funded Project ReAp under Grant 380524764.

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 and scalable technique with two different modes for the quantization of the parameters of pre-trained neural networks. In the first mode, referred to as *log_2_lead*, we use a single template for the quantization of all parameters. In the second mode, denoted as *ALigN*, we analyze the trained parameters of each layer and adaptively adjust the quantization template to achieve even higher accuracy. Our technique significantly maintains the accuracy of the parameters and does not require retraining of the networks. Moreover, it supports quantization to an arbitrary bit-size. For example, compared to the single-precision floating-point numbers-based implementation, our proposed 8-bit quantization technique generates only $\sim 0.2\%$ and $\sim 0.1\%$, loss in the Top-1 and Top-5 accuracies respectively for VGG-16 network using ImageNet dataset. We have observed similar minimal losses in the Top-1 and Top-5 accuracies for AlexNet and Resnet-18 using the proposed quantization scheme for the 8-bit range. Our proposed quantization technique also provides a higher mean intersection over union for semantic segmentation when compared with state-of-the-art quantization techniques. The proposed technique represents parameters in powers of 2, thereby eliminating the need for resource-computationally intensive multiplier units for the hardware accelerators of the neural networks. We also present a design for implementing the multiplication operation using bit-shifts and addition for the proposed quantization technique.

INDEX TERMS Machine learning, deep neural networks, quantization, multipliers.

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 [4]. 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. For these embedded devices, the DNN accelerators are supposed to provide low-power and real-time inferences. A plethora of recent works has proposed different types of data representation techniques and hardware accelerators to reduce the memory and power budgets of training the DNN models and then

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa Rahimi Azghadi.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

using them as inference engines. For example, the Google cloud tensor processing units (TPUs) utilize the Brain Floating Point Format (BFloat16) for providing high-performance operations. The BFloat16, a subset of the single-precision Float32, utilizes only 7 bits for storing the fraction (significand) [31]. However, 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. This paper extends our previous work presented in [29]. To this end, the overall contributions of this manuscript are:

- *Identification of the Most Significant Fractional Bits:* We present an analysis to identify the locations of the leading 1 and the following most significant bits in a fraction to represent a quantized number. These bits mainly define the precision of a quantized number.
- *log_2_lead Quantization Scheme:* Based on our analysis, we present a novel and highly accurate quantization technique, *log_2_lead* (L2L), to quantize the parameters of pre-trained DNNs. Our technique uses a unique template to store the most significant fractional bits.
- *ALigN Quantization Scheme:* We propose an adaptive layerwise variation, referred to as ALigN, of our L2L quantization scheme for pre-trained DNNs. In this technique, we align the available quantization bit-width according to the occurrences of the leading 1's in the trained parameters of each layer. Compared to the L2L quantization scheme, the ALigN technique achieves higher output accuracy for image classification and semantic segmentation tasks.
- *No Retraining of the Quantized Network:* Our proposed quantization techniques are highly accurate and significantly retain the precision of the parameters. Therefore, retraining of the quantized network is not required to recover from the quantization errors.
- *Multiplier Design:* Utilizing our proposed quantization templates, we propose a novel resource-efficient multiplier design using bit-shifts and add operations.

The rest of the paper is organized as follows: Section II presents a summary of the state-of-the-art quantization techniques for DNNs. Section III describes the preliminaries of DNN along with some observation of data distribution of parameters of in trained DNN. Section IV presents our proposed quantization techniques followed by its utilization results in Section V. Finally, Section VI concludes the paper.

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 [5]–[11] utilize specialized methods to train networks with different fixed-point low-precision parameters. The techniques proposed in [7]–[10] have even reduced the precision of trained parameters to only 1-2 bits. The authors of [11] have used dynamic fixed point technique for quantization along with retraining. 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 [12]–[19] have proposed different quantization schemes. The technique proposed in [13] 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 [12], [14]–[18] 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 [17] 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. However, their iterative process of computing and reducing the quantization generated errors, for each layer, ignores the criticality of neurons in that layer. Further, each data structure is quantized independently in their proposed technique, which may result in attaining some locally optimal quantization for the network. The independent 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 work presented in [18] also uses an iterative approach to find the distribution details of the trained parameters. This technique identifies an optimal number of bits for quantization of individual layer parameters, hence trained parameters are quantized in more accurate way. Moreover, it does not quantize all layers to retain the output accuracy of the DNNs. However, the usage of different bit-widths for different layers increases the computational complexity and overall resource utilization of the implementation. The technique described in [19] has used the Monte Carlo Method for sampling the pre-trained weights of a network, and identification of the number of bits to represent the quantized weights with low precision.

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

III. QUANTIZED DNNs

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

A. OVERVIEW OF DNNs

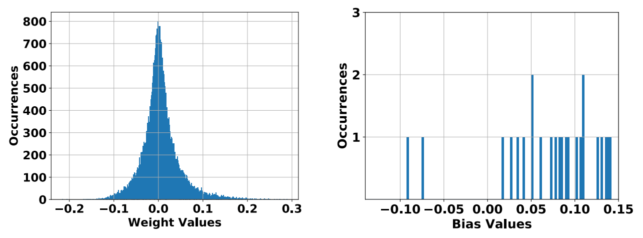
A typical DNN consists of multiple stacked layers of primary computational units called *Neurons*. The commonly used layers in a DNN are *convolution*, *pooling*, and optionally a few *fully connected* layers. Each neuron in each layer receives multiple inputs (features and parameters) and generates a single output (feature). The pooling layers in a DNN are used to subsample and reduce the size of feature maps. The frequently employed pooling function for this purpose is *Max-pooling* which computes the maximum feature *'f'* according to the window size of the neurons in the layer. For an $\frac{n}{2} \times \frac{n}{2}$ window size, the output of a neuron in Max-pooling layer is computed as described in Eq. 1.

$$f_{out} = max(f_1, f_2, f_3, \ldots, f_n) \tag{1}$$

Each neuron of the convolutional and fully-connected layers performs a weighted sum of the features and parameters (weights *'w'*). A bias *'b'* is added with the weighted sum to introduce non-linearity followed by an activation function φ to produce the output of a neuron. Eq. 2 describes this computation. The commonly used activation functions for this purpose are the *ReLU* and *Sigmoid* functions.

$$y = \varphi\left(\sum_{i=1}^{n} f_i w_i + b\right) \tag{2}$$

A DNN is trained using these layers with floating-point numbers. For example, Fig. 1 shows the distribution of weights and biases of a layer for a trained VGG-16 [20] network. As mentioned previously, the memory map of a DNN must be reduced to implement it on resource-constrained and low-power embedded systems. Next, we describe the commonly used quantization schemes to reduce the memory map of a trained DNN.



(a) Weights Distribution (b) Biases Distribution

FIGURE 1. **Distribution of weights and biases for pre-trained VGG-16 [20] Conv1_2 layer.**

B. COMMONLY USED QUANTIZATION TECHNIQUES

1) LINEAR QUANTIZATION

The linear quantization of a data tensor x from floating-point precision to N-bit fixed-point precision is described by Eq. 3–5. The step size Δ in the Eq. 3 represents the minimum possible increment in the quantized value x_{quant}. Eq. 4 is used to align Δ with the maximum and minimum allowed value in N-bit precision.[1] Finally, the Δ is used in Eq. 5 for computing the quantized value x_{quant}. The utilized *clip* function, defined in Eq. 6, ensures that a parameter does not violate the allowed range of values. 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. Most of the quantized numbers have values close to 0.

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

$$\Delta = 2 ** clip\,(round(log_2(\Delta)), N-1, -N+1) \tag{4}$$

$$x_{quant} = clip\left(round\left(\frac{x}{\Delta}\right), -2^{N-1}, 2^{N-1}-1\right)\Delta \tag{5}$$

$$clip(x, Max, Min) = \begin{cases} x, & Min < x < Max \\ Max, & x \geq Max \\ Min, & \text{otherwise} \end{cases} \tag{6}$$

2) 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. 7 and Eq. 8 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 biases of pre-trained VGG-16 network. The elemental power of 2 quantization scheme has a fewer number of discrete fixed-point levels than the linear quantization. Compared to the linear quantization, the power of 2 quantization have many repeatedly occurring values which are not close to 0.

$$\hat{x_{quant}} = clip\,(round\,(log_2(|\,x\,|))\,, 0, N) \tag{7}$$

$$x_{quant} = sign(x)2^{\hat{x_{quant}}} \tag{8}$$

3) DYNAMIC FIXED POINT QUANTIZATION

The dynamic fixed-point quantization technique, used in [11] and [18], is a variation of the fixed-point format. In this

[1] 2** shows power of 2.

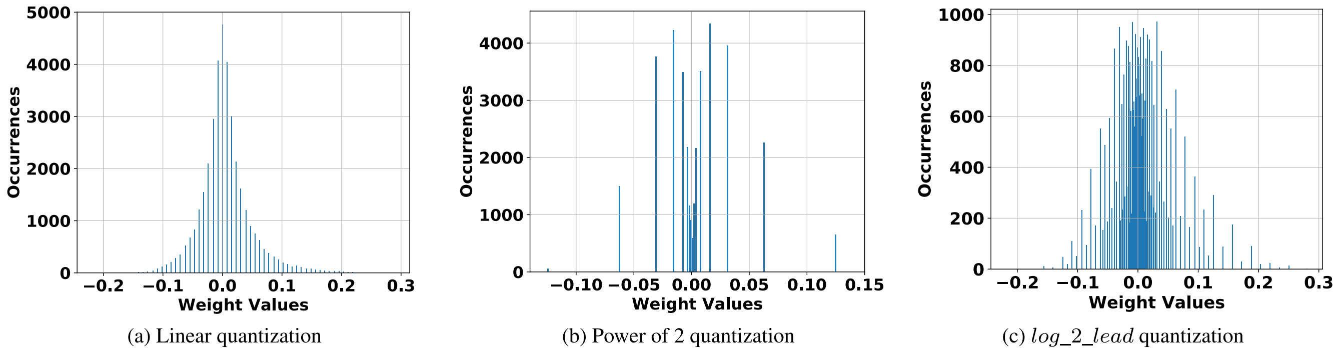

(a) Linear quantization (b) Power of 2 quantization (c) log_2_lead quantization

FIGURE 2. Quantization of pre-trained weights of Conv1_2 layer of VGG-16 [20].

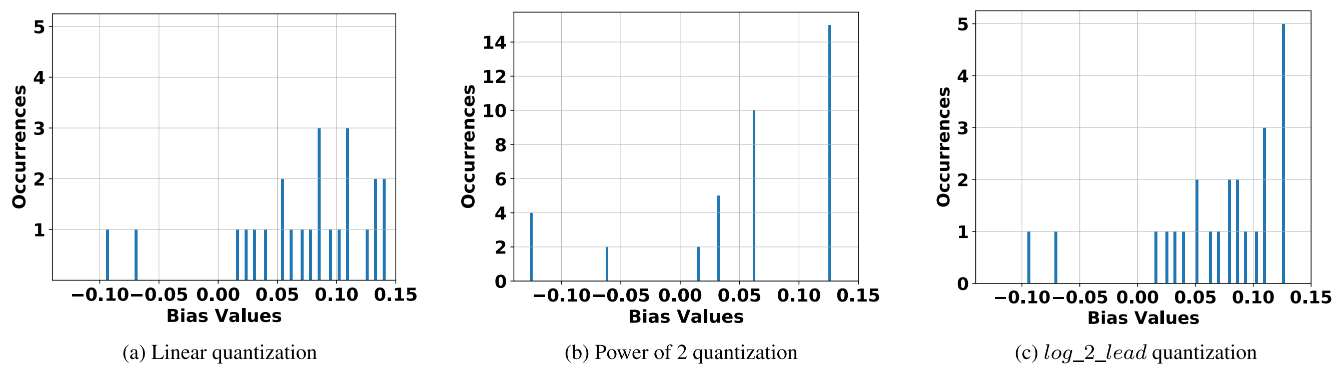

(a) Linear quantization (b) Power of 2 quantization (c) log_2_lead quantization

FIGURE 3. Quantization of pre-trained biases of Conv1_2 layer of VGG-16 [20].

method, the range of trained parameters is analyzed to decide the fractional length (FL) for data representation. An 'FL' value is determined for each layer such that the trained parameters can be expressed with high precision. For quantization word-width 'W' and mantissa value 'm', Eq. 9 describes the number representation for the dynamic fixed-point quantization scheme.

$$Quantized\ value = (-1)^{sign}.2^{-FL} \sum_{i=0}^{W-2} 2^i.m_i \tag{9}$$

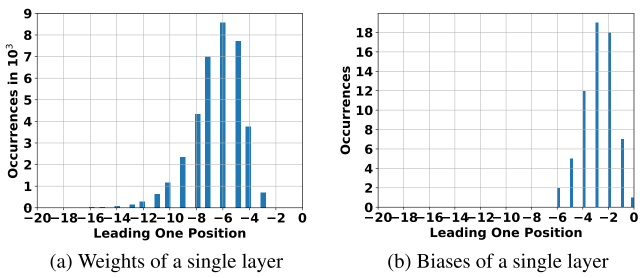
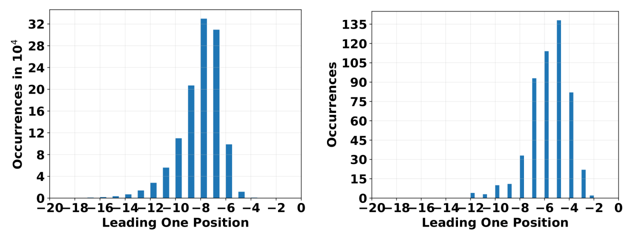

(a) Weights of a single layer (b) Biases of a single layer

FIGURE 4. Histogram of leading 1's for all weights and biases for pre-trained VGG-16 [20] Conv1_2 layer.

IV. PROPOSED TECHNIQUES

A. L2L: *log_2_lead* QUANTIZATION

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 and Fig. 5 give histograms of the leading 1's in all the weights and biases of two different layers of VGG-16 network. As shown, the leading 1 for most of the trained parameters (weights and biases) occurs at some distinct bit positions; for example, for the weights of Conv1_2 and Conv4_1 layers, the leading 1 frequently occurs at bit position -6 and -8 respectively. However, there are also some weights with very low-values and having the leading 1 occurring at bit position -15. In our proposed log_2_lead technique, we utilize log_2 to detect the position of the leading 1 in a fraction. However, to limit the quantization errors, log_2_lead also observes the following bits locations after the leading 1 location. For N-bit quantization, Fig. 6 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



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

FIGURE 5. **Histogram of leading 1's for all weights and biases for pre-trained VGG-16 [20] Conv4_1 layer.**

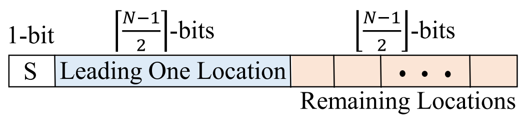

1-bit
$\lceil \frac{N-1}{2} \rceil$-bits
$\lfloor \frac{N-1}{2} \rfloor$-bits
S
Leading One Location
• • •
Remaining Locations

FIGURE 6. **Template for proposed quantization technique.**

binary number '1100' by the '*leading one location*' field in the template shown in Fig. 6. 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 (round towards $+\infty$) to $\lfloor \frac{N-1}{2} \rfloor$ bits. The rounding of $\lfloor \frac{N-1}{2} \rfloor + 1$ bits further helps in retaining the precision of a rounded number. For example, Fig. 7 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.

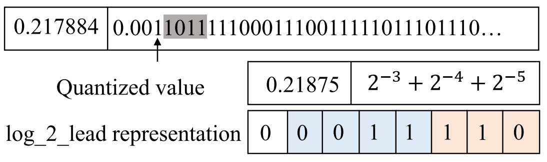


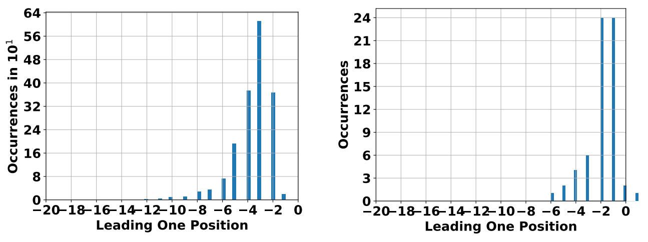
FIGURE 7. ***log_2_lead*** **quantization example.**

Compared to our proposed technique, a typical log_2 quantization as described in Eq. 7 and Eq. 8 would only find a single leading 1 for each weight and discard all other bit locations. Due to ignoring of all other bit locations for computing the quantized value, the log_2 quantization can introduce substantial quantization errors. The quantization technique proposed in [17] divides the float32 parameter into two segments i.e., MSBs and LSBs. It then performs the log_2 quantization of the MSBs and LSBs separately and then adds the MSBs-quantized and LSBs-quantized values to achieve the final quantized value. This technique can recover some quantization induced errors; however, it also ignores all the significant bits after the leading 1 in both segments (MSBs and LSBs).

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 (see Table 6). Fig. 2(c) and Fig. 3(c) shows the quantization of weights and biases of two different layers of VGG-16 (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.

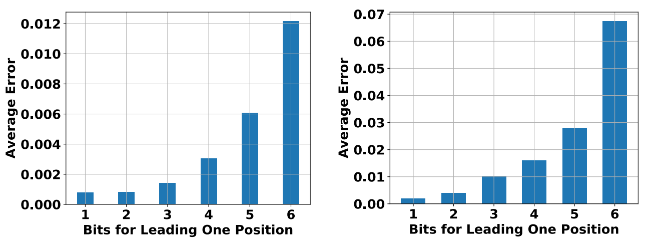
B. ALigN*: ADAPTIVE log_2_lead QUANTIZATION*

The quantization induced errors can be minimized by utilizing the available quantization bit-width intelligently. The proposed *L2L* technique, though minimizes the quantization induced errors, has a fixed template for representing the quantized parameters of all layers in a pre-trained DNN. For example, as shown in Fig. 6, it always utilizes $\lceil \frac{N-1}{2} \rceil$-bits for storing the position of leading 1 in an N-bit quantization. We have observed during our analysis of the parameters' distributions for various layers in different pre-trained DNNs that for some layers, the fixed $\lceil \frac{N-1}{2} \rceil$-bits for storing the leading 1 position are not used efficiently. For example, Fig. 8 presents the leading 1 histogram of the weights and biases of the Conv1_1 layer of a pre-trained VGG-16. It can be observed that the leading 1 for these parameters mostly occurs at bit position -3. To resolve this problem, we propose '*ALigN*', an adaptive log_2_lead quantization technique. In this technique, we analyze the parameters of each layer to identify a suitable number of bits for storing the leading 1 in the original non-quantized parameters of each layer to reduce the corresponding quantization induced errors. Fig. 9 and Fig. 10 show the average



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

FIGURE 8. **Histogram of leading 1's for all weights and biases for pre-trained VGG-16 [20] Conv1_1 layer.**



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

FIGURE 9. **Average error for different number of bits for leading one position for weights and biases for pre-trained VGG-16 [20] Conv1_1 layer.**

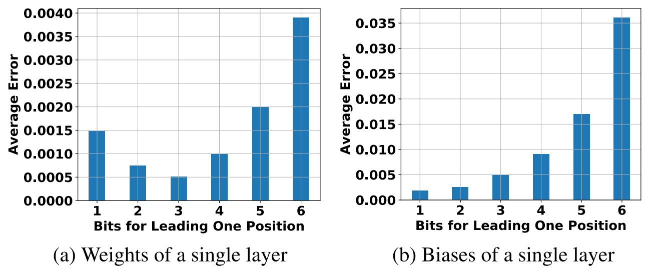


FIGURE 10. **Average error for different number of bits for leading one position all weights and biases for pre-trained VGG-16 [20] Conv1_2 layer.**

quantization induced errors by varying the number of bits reserved for storing the leading 1 location in L2L scheme for the parameters of two different layers of a pre-trained VGG-16 network. For the weights of the Conv1 layer, assigning a single bit for storing the leading 1 location produces minimum quantization errors. However, the quantization of the parameters of the Conv2 layer requires 3-bits for generating the smallest error. Therefore, it is advantageous to align the *L2L* scheme according to the distribution of the parameters of each layer of a pre-trained network. Eq. 10 and Eq. 11 show our technique for finding the optimal number of bits L_w and L_b for storing the leading 1 locations of weights and biases, respectively, for each layer. After determining L_w and L_b adaptively for each layer, we quantize the parameters of each layer by storing leading 1 in L-bits and utilizing the remaining locations for storing $N - L - 1$ following bits. Fig. 11 shows the updated template of our proposed quantization scheme, *ALigN*, to represent the quantized values. As shown, for an N-bit *ALigN* quantization, there is no fixed boundary for storing the leading 1 location.

$$L_w = argmin_{L_w}(Average\{|w_{quant} - w|\}) \tag{10}$$
$$L_b = argmin_{L_b}(Average\{|b_{quant} - b|\}) \tag{11}$$

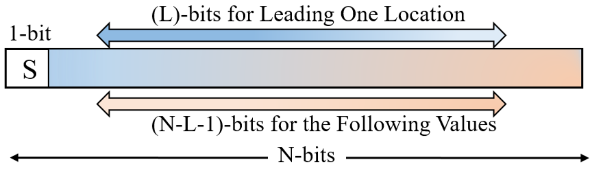

(L)-bits for Leading One Location
1-bit
S
(N-L-1)-bits for the Following Values
N-bits

FIGURE 11. ***ALigN* template for N-bit quantization.**

C. QUANTITATIVE ANALYSIS OF THE PROPOSED QUANTIZATION SCHEMES

To evaluate the efficacy of our proposed quantization schemes, *L2L* and *ALigN* are applied along with various quantization schemes for different layers of pre-trained VGG-16 and AlexNet networks respectively, then average errors are reported in Fig. 12 and Fig. 13. The original weights, in single-precision float32, are quantized to 8-bits. For both networks, our proposed techniques always bear a minimal amount of quantization error in each layer. Further,

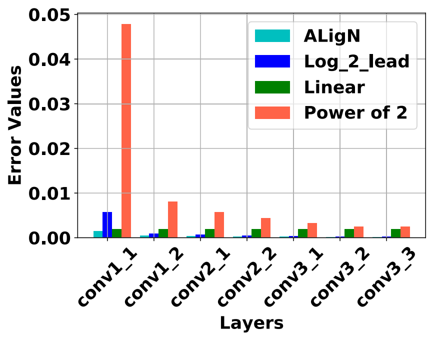


FIGURE 12. **Comparison of average quantization induced errors for quantized weights of convolution layers of VGG-16 [20] between *ALigN*, *log_2_lead*, linear and power of 2 quantization schemes.**

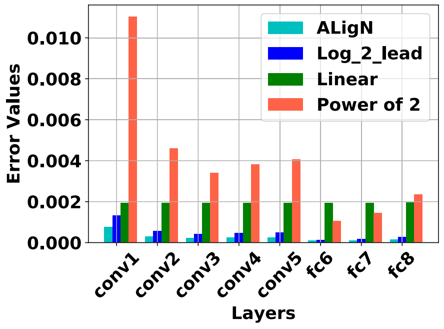


FIGURE 13. **Comparison of average quantization induced errors for quantized weights of convolution layers of AlexNet [28] between *ALigN*, *log_2_lead*, linear and power of 2 quantization schemes.**

ALigN always generates lesser errors than the L2L quantization scheme. This behavior reflects the efficient utilization of the available quantization bit-width by the *ALigN* technique. The traditional power of 2 quantization suffers the most from quantization produced errors. An analysis of Fig. 12 shows that the parameters of the conv1_1 layer have a comparatively higher magnitude than the parameters of the other layers. The allocation of fixed four bits for storing the leading 1 location of the conv1_1 layer has resulted in the underutilization of these bits. This behavior is also evident from Fig. 12, where linear quantization performs better than L2L for the first layer. These parameters do not require more bits for recording the leading 1 location, as also shown in Fig. 9. However, *ALigN* adaptively selects the number of bits for storing the position of the leading 1 for each layer, and it always performs better than the linear quantization. To further investigate the distribution of quantization induced errors, Fig. 16 and Fig. 17 show the relative error distribution of quantized weights for two different layers of VGG-16 and AlexNet networks using different quantization schemes. For both networks, most of the quantization generated errors for *L2L* are limited to a narrow band of values (0 — 0.1), whereas the relative errors generated by linear quantization are spread over a broader spectrum (0 — 0.8). The *ALigN* further reduces these errors and limits them to even a narrower band than *L2L*.

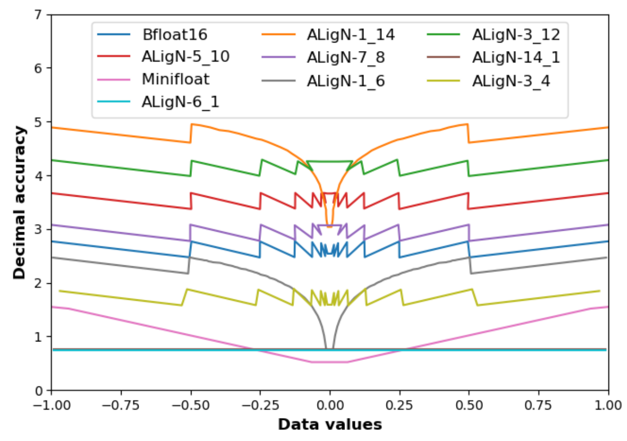


FIGURE 14. Decimal accuracy of different quantization schemes for the active range of values of pre-trained parameters (−1 to +1). *ALigN*-x_y shows x bits reserved for storing the leading 1 location and y bits allocated to store the following values[2].

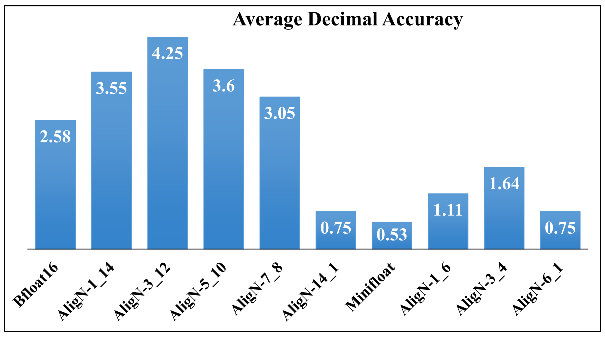


FIGURE 15. Average decimal accuracy of the weights of Conv1_2 layer of VGG-16 [20] network using different quantization schemes. *ALigN*-x_y shows x bits reserved for storing the leading 1 location and y bits allocated to store the following values.

Our proposed quantization schemes efficiently utilize the available bit-width to comprehensively incorporate the dynamic range (−1 to +1) of trained parameters. To show the efficacy of our proposed quantization scheme for the parameters of pre-trained DNNs, we compare its decimal accuracy in the range of −1 to +1 with bfloat16 and minifloat. These results are presented in Fig. 14. The decimal accuracy [21], defined in Eqn. 12, represents the capability of a number system to represent Float32-based numbers. As shown by the results in Fig. 14, the *ALigN-1_14* scheme (1 bit reserved for storing the leading one location and 14 bits allocated to store the following bits) provides the highest decimal accuracy. The *ALigN-7_8* scheme provides slightly better performance than the bfloat16. Similarly, for 8-bit quantization, the *ALigN-1_6* provides the best results among all other 8-bit variations. Finally, Fig. 15 shows the average decimal accuracy of the pre-trained weights of a single layer, Conv1_2, of VGG-16 network using different quantization schemes. The ALigN-3_12 configuration represents the quantized weights with the highest accuracy. Similarly, for 8-bit quantization, the ALigN-3_4 scheme produces better results than other corresponding 8-bit schemes. These results are in line with the observations of Fig. 10, where the allocation of 3 bits for the storage of the leading 1 location was producing the minimum quantization-induced average error. We have shown the results only for the Conv-1_2 layer; however, similar results can be generated for other layers of the network. Nonetheless, for other layers of the network, a different configuration of the ALigN can offer higher decimal accuracy than the ALigN-3_12 configuration. These statistics reiterate our contribution of considerably maintaining the precision of a float32-based fraction.

$$Decimal\ Accuracy = -log_{10}\left| -log_{10}\left(\frac{X_{quant.}}{X_{float}}\right)\right| \quad (12)$$

[2]ALigN-14_1 and ALigN-6_1 schemes have overlapping lines.

D. PROPOSED QUANTIZATION TECHNIQUE-BASED MULTIPLIER

Our proposed technique replaces the computationally expensive multiply operation with *bit-shifts* and *add* operations. Since the quantized weights for a neuron do not represent a single constant number, there is no fixed-amount of shift-operations to realize the multipliers, which can increase overall resource utilization. To answer this challenge, Algorithm 1 describes a novel resource-efficient

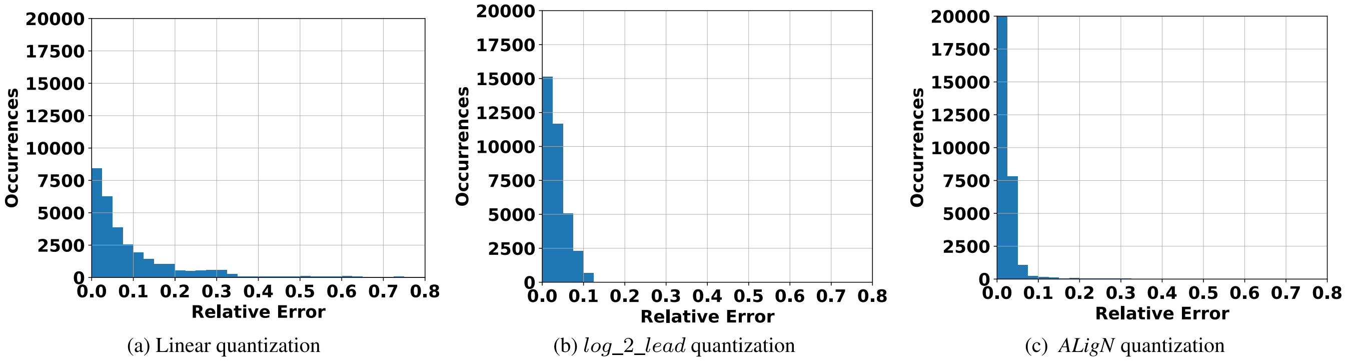


FIGURE 16. Relative error distribution of quantized weights of Conv1 layer AlexNet [28] using different quantization schemes.

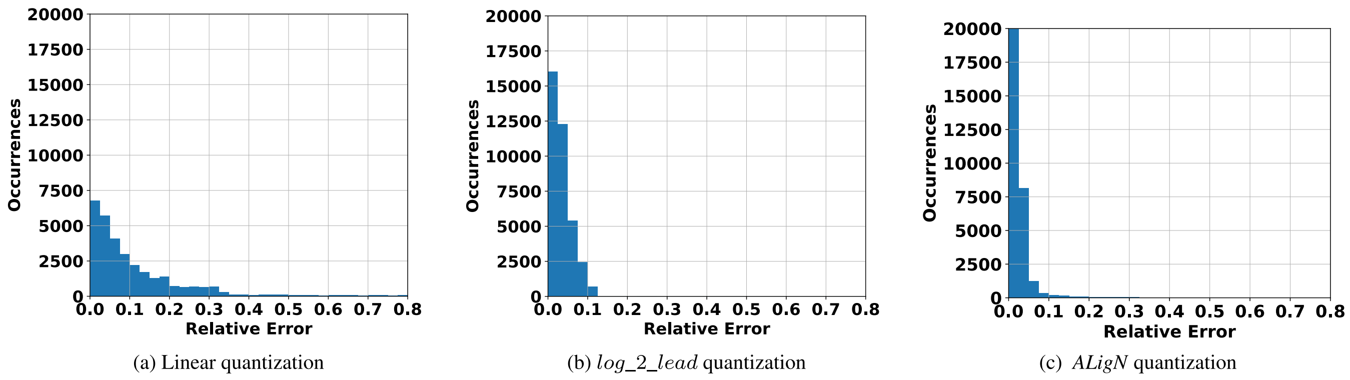


FIGURE 17. Relative error distribution of quantized weights of Conv1_2 layer VGG-16 [20] using different quantization schemes.

Algorithm 1 Multiplication Using *L2L*and *ALigN*

```
Input: Feature: $x$, Weight: $w$ encoded in $L2L$ or
       $ALigN$ format
Output: $y$
1 $y \leftarrow x$
2 for $i \leftarrow 1$ to $\lfloor \frac{N-1}{2} \rfloor$ do
3     if $w\left[\lfloor \frac{N-1}{2} \rfloor - i\right] \neq 0$ then
4         $y \leftarrow y + (x \gg i)$
5 for $j \leftarrow 0$ to $\lceil \frac{N-1}{2} \rceil - 1$ do
6     if $w\left[\lfloor \frac{N-1}{2} \rfloor + j\right] \neq 0$ then
7         $y \leftarrow y \gg 2^j$
8 return $y$
```

implementation of the multiplication of an N-bit *fixed-point feature* with an N-bit *L2L*- or *ALigN*-based quantized weight. Initially, it analyzes and evaluates the least significant $\lfloor \frac{N-1}{2} \rfloor$ bits (lines 2-4). For each bit value equal to 1, it performs the required shifting of the feature and adds the result to the output. Lines 5-7 perform the final shifting of the output according to the value stored for ‘leading one location’. For *ALigN*-based quantization, each layer can have different configurations for storing the leading 1 location and the following bit values. Our generic implementation of Algorithm 1 allows us to adapt the multiplier for each layer independently.

V. EXPERIMENTAL SETUP AND RESULTS

For the application-level evaluation of *log_2_lead*, *ALigN*, linear quantization, and power of 2 quantization, we have used TensorFlow framework [24]. Using the framework, we have implemented a variety of networks with different datasets to test the efficacy of our proposed quantization techniques for image classification and semantic segmentation accuracies of quantized netwroks. These results are discussed in Sections V-A and V-B. The proposed *log_2_lead* technique replaces the resource-demanding *multiply* operation with *bit-shifts* and *add* operations. An algorithm for *log_2_lead*-based multiplication is presented in Algorithm 1. As a test case, we have implemented a *Processing Element* (PE) for Algorithm 1 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-C.

A. IMAGE CLASSIFICATION

For the MNIST dataset [25], we have trained a lightweight neural network consisting of convolution layers (Conv), Maxpool layer,[3] and fully connected layers[4] (FC), as described in Table 1. The MNIST dataset contains 60, 000 and 10, 000 greyscale training and testing images, respectively, at a resolution of 28×28 pixels. After training the network on 60, 000 images, we have applied different 8-bit ($N = 8$) quantization schemes on the trained parameters. The application-level accuracy results are reported in Table 2. The *adaptive log_2_lead* (*ALigN*) quantization technique offers the same classification accuracy as produced by the baseline float32-based implementation. The *log_2_lead* and linear quantization have nearly similar output accuracy compared to the single-precision float32 representation. The power of 2 quantization considers only the most significant 1 in the fraction; therefore, it has produced comparatively reduced output accuracy.

We have also used CIFAR-10 dataset [23] for the lightweight neural network described in Table 1 to assess the accuracy of our proposed quantization schemes. CIFAR-10 dataset contains 50,000 training and 10,000 32×32 RGB images. These images are classified into 10 labeled classes. Lightweight neural network classification accuracy results for 10,000 images using different 8-bit quanitzation schemes are reported in Table 2. The *ALigN* and *log_2_lead* quantization have produced comparable results to the single-precision float32-based results. The power of 2 quantization has shown significant accuracy drop due to

[3]Maxpool layer is only after the second Conv layer.
[4]Size and number of kernels are applied to Conv layers only.

TABLE 1. Description of networks used for classification of MNIST and CIFAR-10 datasets.

Layer	MNIST Network		CIFAR-10 Network	
	Kernel Size	Total Kernels	Kernel Size	Total Kernels
Conv1	5x5	32	5x5	64
Maxpool	-	-	3x3	-
Conv2	5x5	64	5x5	64
Maxpool	2x2	-	3x3	-
FC1	-	512	-	384
FC2	-	512	-	192
Softmax	-	10	-	10

TABLE 2. Classification accuracy of lightweight neural networks on MNIST and CIFAR-10 dataset with 8-bit quantized weights and biases for different quantization schemes.

Quantization	MNIST	CIFAR-10
w, b quantized	Accuracy [%]	Accuracy [%]
Float32	98.18	85.1
8-bit linear	98.13	85.1
Power of 2	97.6	73.5
log_2_lead	98.12	85.1
ALigN	98.18	85.1
Float32 vs *log_2_lead*	-0.06	0.0
Float32 vs ALigN	0.0	0.0

limited coverage of the dynamic range of fractional numbers under consideration.

To evaluate the efficacy of our proposed techniques on a more challenging classification task, we have also tested it on ImageNet dataset [22]. The ImageNet dataset hosts 1.2 million and $50,000$ RGB images for training and validation, respectively. These images are classified into 1000 different categories. For our experimentation, we have taken pre-trained AlexNet [28], VGG-16 [17] and ResNet-18 networks [30]. To improve the inference accuracy of power of 2 quantization scheme, we have also utilized the post-quantization retraining for it to minimize its quantization induced errors. AlexNet network contains 5 convolutional layers followed by 3 fully connected layers. For the ImageNet validation set, the pre-trained AlexNet has 53.89% and 77.62% Top-1 and Top-5 accuracies, respectively, using the baseline single-precision float32 weights, biases, and activations. We have quantized the weights and biases of this network with *log_2_lead*, *ALigN*, linear, power of 2, and dynamic fixed-point quantization schemes. Except for the dynamic fixed-point scheme, all quantization schemes have an 8-bit width. The dynamic fixed-point quantization employs different bit-widths for different layers. For example, for weights and biases of the conv1, conv2, conv3, and conv4 layers, it applies twelve bits, whereas, for the parameters of the conv5 layer, it utilizes ten bits. Similarly, for the parameters of FC6, FC7, and FC8 layers, it uses sixteen, twelve, and six bits, respectively. The Top-1 and Top-5 classification accuracies are shown in Table 3. The *log_2_lead* quantization scheme provides better output accuracies than the linear, power of 2, and dynamic fixed-point quantization schemes. The drop in Top-1 and Top-5 percentage classification accuracies are only 0.01 and 0.12 respectively

TABLE 3. Classification accuracy of AlexNet network [28] on ImageNet dataset [22] with quantization of weights and biases using different schemes.

AlexNet [28]	Top-5 [%]	Top-1 [%]
Float32	77.62	53.89
8-bit linear	76.95	53.11
Power of 2 (without retraining)	65.45	40.72
Power of 2 (with retraining)	75.63	47.19
Dynamic fixed point	74.21	50.24
log_2_lead	77.50	53.88
ALigN	77.67	53.99
Float32 vs *log_2_lead*	-0.12	-0.01
Float32 vs ALigN	+0.05	+0.1

TABLE 4. Classification accuracy of VGG-16 network [20] on ImageNet dataset [22] with quantization of weights and biases using different schemes.

VGG-16 [20]	Top-5 [%]	Top-1 [%]
Float32	85.74	64.72
8-bit linear	82.55	59.8
Power of 2 (without retraining)	0.63	0.1
Power of 2 (with retraining)	56.25	30.78
Dynamic fixed point	83.63	61.43
log_2_lead	85.64	64.51
ALigN	85.71	64.62
Float32 vs *log_2_lead*	-0.1	-0.21
Float32 vs ALigN	-0.03	-0.1

for the *log_2_lead* scheme. Based on the adaptive layerwise analysis of the network, Table 5 shows the number of bits (L_x) delimited to store the leading 1 location for parameters of each layer for an 8-bit *ALigN* quantization scheme. The remaining $8 - L_x - 1$ bits are assigned for storing the bit values following the leading 1. Compared to the baseline float32 accuracy, the *ALigN* scheme produces the same output accuracy.

The VGG-16 network mainly consists of 13 convolution layers with kernel size 3×3 and 3 fully connected layers. 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 4. In this experiment with VGG-16 network, the activations have float32 precision, and the weights and biases are quantized to 8-bit precision. Compared to the linear quantization, power of 2 quantization, and dynamic fixed-point quantization, our proposed techniques *log_2_lead* and *ALigN* produce better Top-1 and Top-5 classification accuracies. The *log_2_lead* scheme reduces the Top-1 and Top-5 percentage classification accuracies only by 0.21 and 0.1, respectively. For the dynamic fixed-point quantization scheme, except for the FC6 layer, which is quantized to 8 bits, all other layers have Float32 precision. The *ALigN* scheme produces better results than others and reduces the Top-1 and Top-5 percentage classification accuracies only by 0.1 and 0.03, respectively. The corresponding number of bits to store the

TABLE 5. No. of bits selected for storing the leading 1 location of parameters of AlexNet [28] and VGG-16 [20] networks using 8-bit *ALigN* technique.

AlexNet	**Layer**	Conv1		Conv2		Conv3		Conv4		Conv5		FC6		FC7		FC8	
	Bits for Leading	2		3		3		3		3		3		3		3	
VGG-16	**Layer**	Conv1_1	Conv1_2	Conv2_1	Conv2_2	Conv3_1	Conv3_2	Conv3_3	Conv4_1	Conv4_2	Conv4_3	Conv5_1	Conv5_2	Conv5_3	FC6	FC7	FC8
	Bits for Leading	1	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3

leading 1 location of the trained parameters of each layer are shown in Table 5.

We have also evaluated our proposed quantization schemes on residual network ResNet-18 with the ImageNet validation set. Table 7 shows the corresponding Top-5 and Top-1 accuracies of the quantized pre-trained network with 8-bit linear, power of 2, *log_2_lead*, and *ALigN* schemes. Our proposed quantization schemes again outperform other quantization techniques in the respective classification accuracies. For example, compared to the baseline float32-based classification accuracies, the *log_2_lead* has only 0.8 and 0.38 drops in the Top-1 and Top-5 percentage classification accuracies, respectively. The *ALigN* scheme improves the Top-1 percentage classification accuracy by 0.12 when compared with the corresponding baseline accuracy. However, the Top-5 percentage accuracy is dropped by 0.1, which is still better than those achieved with the other quantization schemes.

The adaptive nature of the ALigN scheme and the ensuing efficient utilization of quantization bit-widths show that the 8-bit ALigN technique can significantly reduce quantization-induced errors in various DNNs. To further show the efficacy of the ALigN scheme, Table 6 presents the classification accuracies of VGG-16 and ResNet-18 networks for ImageNet dataset using different bit-widths for the ALigN scheme. As shown by the results, the classification accuracy improves for both networks by increasing the quantization bit-width. However, increasing quantization bit-width beyond 8 bits does not have a significant and definite effect on classification accuracy. Therefore, we have selected the 8-bit ALigN for thorough exploration using different applications and DNNs.[5]

B. SEMANTIC SEGMENTATION

To further evaluate the efficacy of our proposed quantization techniques, we have also considered the semantic segmentation task. For this purpose, we have utilized the Fully Convolutional Network (FCN) [26] and PASCAL Visual Object Classes (VOC) 2012 validation set [27]. The FCN utilizes the thirteen convolutional layers of VGG-16 [20] for performing semantic segmentation. The PASCAL VOC 2012 validation dataset contains 1449 images in 20 different classes. Further, we have used the intersection over union (IU) metric for measuring the efficiency of presented quantization techniques for the semantic segmentation task. Table 8 shows the results of applying 8-bit *log_2_lead*, *ALigN*, linear, power

[5]Utilizing 8 bits (a single byte) for storing the quantized parameters (weights and biases) also provides compact storage in the memory.

TABLE 6. ALigN-based quantization of two state-of-the-art DNNs using different bit-widths.

Quantization Bit-width	**ResNet-18**		**VGG-16**	
	Top-5 %	Top-1 %	Top-5 %	Top-1 %
Float32	88.94	69.3	85.74	64.72
6	86.59	65.71	85.13	63.65
7	88.61	68.33	85.64	64.53
8	88.84	69.42	85.71	64.62
9	89.05	69.2	85.74	64.71
10	88.95	69.34	85.7	64.67

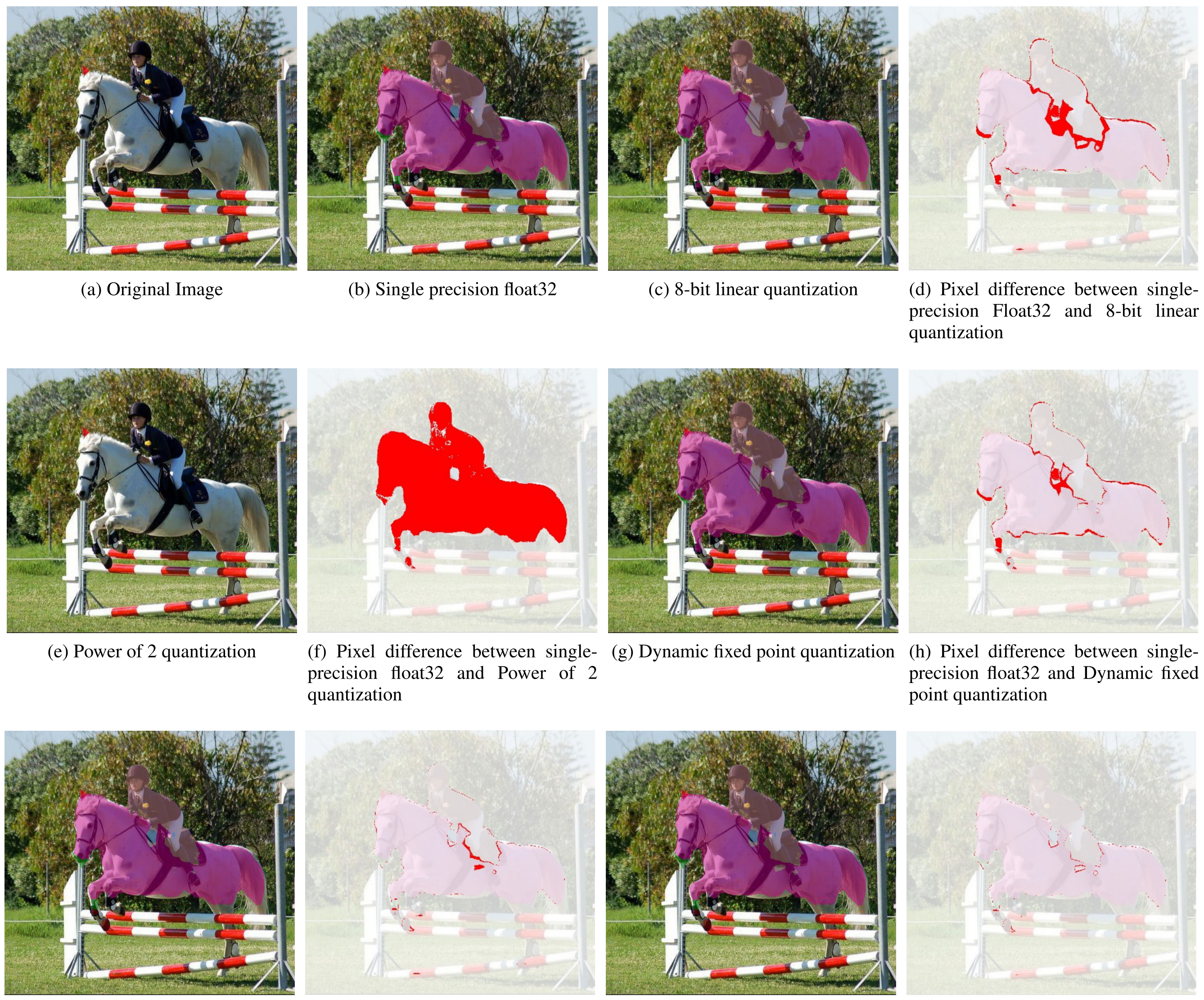
TABLE 7. Classification accuracy of ResNet-18 network [30] on ImageNet dataset [22] with quantization of weights and biases using different schemes.

Resnet-18 [30]	**Top-5 [%]**	**Top-1 [%]**
Float32	88.94	69.30
8-bit linear	88.47	68.70
Power of 2 (without retraining)	5.37	1.49
Power of 2 (with retraining)	86.84	65.45
log_2_lead	88.56	68.41
ALigN	88.84	69.42
Float32 vs *log_2_lead*	-0.38	-0.8
Float32 vs ALigN	-0.1	+0.12

of 2, and dynamic fixed-point quantization schemes on the pre-trained FCN. For the dynamic fixed-point quantization scheme, except for the FC6 layer, which is linearly quantized to 8 bits, all other layers have Float32 precision. As shown by the results, the pre-trained FCN has a mean IU of 61.8% using single-precision float32-based parameters. Compared to this baseline result, the 8-bit linear quantization displays a loss of 1% in the mean IU, whereas the *log_2_lead* scheme has only a 0.5% loss in the mean IU. The *ALigN* and dynamic fixed-point quantization schemes produce no loss in the mean IU. Fig. 18 shows the visual results of applying different quantization schemes on the pre-trained FCN for one single image from the PASCAL VOC 2012 validation dataset. As shown by the images, the *ALigN* produces minimal differences in the pixel values when compared with the float32-based representation.

C. HARDWARE IMPLEMENTATION RESULTS

Xilinx state-of-the-art FPGAs provide 6-input lookup tables (LUTs) for implementing various types of combinational and sequential logic. Utilizing these LUTs, we have implemented a processing element (PE) for the multiplication algorithm defined in Algorithm 1. Table 9 compares the implementation results of our proposed 8-bit PE with Xilinx Vivado area-optimized multiplier IP for various sizes. For simplicity of results, we have presented implementation results for only one configuration i.e., *L2L* (4 bits for



(a) Original Image

(b) Single precision float32

(c) 8-bit linear quantization

(d) Pixel difference between single-precision Float32 and 8-bit linear quantization

(e) Power of 2 quantization

(f) Pixel difference between single-precision float32 and Power of 2 quantization

(g) Dynamic fixed point quantization

(h) Pixel difference between single-precision float32 and Dynamic fixed point quantization

(i) *log_2_lead* quantization

(j) Pixel difference between single-precision float32 and *log_2_lead* quantization

(k) *ALigN* quantization

(l) Pixel difference between single-precision float32 and *ALigN*

FIGURE 18. Object detection comparison with different quantization schemes using FCN8 [26] network on PASCAL VOC 2012 validation set [27].

leading 1 location and 3 bits for the following bits) of our proposed quantization schemes. However, the proposed implementation is generic and can be adapted to support different configurations using the ALigN scheme to quantize different layers. It also shows the corresponding classification accuracies of quantized VGG-16 for ImageNet dataset using *log_2_lead* and linear quantization with different data sizes for weights. 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. To achieve classification accuracy that is comparable to *log_2_lead*, we increased the number of bits for weight quantization. Since the maximum precision of 8-bit *log_2_lead* scheme weights is 18 bits (15 for the leading one location and 3 bits for the following bits), we experimented with a maximum of 18 bits of precision in linear quantization. As expected, while the accuracy roughly increases with an increasing number of bits, so does the total number of utilized LUTs for the Vivado multiplier IP. 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.

Our proposed quantization schemes assume inputs and outputs of a neuron to have the same data types. Therefore, the output data type of our proposed multiplier is decided by the nature of the inputs (feature map). For example, the inputs to VGG-16 in Table 9 are linearly quantized 8-bit fixed-point numbers; therefore, the multiplier output is also a fixed-point

TABLE 8. Mean IU of FCN8 [26] network on PASCAL VOC 2012 validation set [27] with quantization of weights and biases using different schemes.

FCN8 [26]	Mean IU [%]
Float32	61.8
8-bit linear	60.8
Power of 2 (without retraining)	3.5
Power of 2 (with retraining)	21.1
Dynamic fixed point	61.8
log_2_lead	61.3
ALigN	61.8
Float32 vs *log_2_lead*	-0.5
Float32 vs ALigN	0.0

TABLE 9. Comparison of resource utilization of proposed PE and Vivado multiplier IPs along with corresponding classification accuracies for VGG-16 [20] network on ImageNet dataset [22].

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

number. To accumulate the linearly quantized fixed-point numbers, we can use the conventional integer-based adder trees.

VI. CONCLUSION

In this paper, we have proposed two quantization schemes, *log_2_lead* and *ALigN*, for the quantization of pre-trained deep neural networks. Our proposed schemes analyze the Float32-based parameters of a trained deep neural network and record the bit positions of the most important bits. These bits mainly define the precision of a parameter. In the first proposed technique, *L2L*, we always store the $\lfloor \frac{N-1}{2} \rfloor + 1$ most essential bits for an N-bit quantization. However, for the second technique, *ALigN*, we have used an adaptive layerwise analysis to find the number of significant bits to be examined for the quantization of pre-trained parameters. This analysis has reduced the quantization induced errors significantly. Our proposed techniques are highly accurate and do not require the traditional accuracy-recovery retraining of the quantized network. We have presented the efficacy of our quantization schemes for different applications and on various state-of-the-art DNNs. For example, for the ImageNet dataset classification using VGG-16 and ResNet-18 networks, we have achieved around 0.24 and 0.065 drops in Top-5 percentage accuracies using the proposed *L2L* and *ALigN* techniques, respectively, when compared with the Float32-based accuracy. Utilizing our proposed quantization techniques, we have also proposed a multiplier design using bit-shifts and add operations. For similar classification accuracy of the ImageNet dataset using the VGG-16 network, our proposed multiplier implementation has offered a 39% reduction in resource utilization when compared with the Vivado area-optimized multiplier IP. For future work, we intend to use our proposed quantization schemes for other machine learning tasks such as Natural language processing and text generation.

ACKNOWLEDGMENT

(Siddharth Gupta and Salim Ullah contributed equally to this work.)

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1026–1034, 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," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Vancouver, BC, Canada, May 2013, pp. 8599–8603.

[3] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[4] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[5] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio, "Neural networks with few multiplications," 2015, *arXiv:1510.03009*. [Online]. Available: http://arxiv.org/abs/1510.03009

[6] 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," 2016, *arXiv:1606.06160*. [Online]. Available: http://arxiv.org/abs/1606.06160

[7] M. Courbariaux, Y. Bengio, and J. P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[8] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 525–542.

[9] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," 2016, *arXiv:1605.04711*. [Online]. Available: http://arxiv.org/abs/1605.04711

[10] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," 2016, *arXiv:1612.01064*. [Online]. Available: http://arxiv.org/abs/1612.01064

[11] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5784–5789, Nov. 2018.

[12] H. Tann, S. Hashemi, R. I. Bahar, and S. Reda, "Hardware-software codesign of accurate, multiplier-free deep neural networks," in *Proc. 54th Annu. Design Autom. Conf.*, Austin, TX, USA, Jun. 2017, pp. 1–6.

[13] D. D. Lin, S. S. Talathi, and V. S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2849–2858.

[14] 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 *Proc. Int. Conf. Comput.-Aided Design*. New York, NY, USA: ACM, Nov. 2018, p. 9.

[15] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," 2016, *arXiv:1603.01025*. [Online]. Available: http://arxiv.org/abs/1603.01025

[16] S. Shakib Sarwar, S. Venkataramani, A. Raghunathan, and K. Roy, "Multiplier-less artificial neurons exploiting error resiliency for energy-efficient neural computing," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, 2016, pp. 145–150.

[17] S. Vogel, J. Springer, A. Guntoro, and G. Ascheid, "Self-supervised quantization of pre-trained neural networks for multiplierless acceleration," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Florence, Italy, Mar. 2019, pp. 1094–1099.

[18] M. de Prado, M. Denna, L. Benini, and N. Pazos, "QUENN: QUantization engine for low-power neural networks," in *Proc. 15th ACM Int. Conf. Comput. Frontiers*. New York, NY, USA: ACM, May 2018, pp. 36–44.

[19] G. Mordido, M. Van Keirsbilck, and A. Keller, "Instant quantization of neural networks using Monte Carlo methods," 2019, *arXiv:1905.12253*. [Online]. Available: http://arxiv.org/abs/1905.12253

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[21] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomput. Frontiers Innov.*, vol. 4, no. 2, pp. 71–86, 2017.

[22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[23] A. Krizhevsky, V. Nair, and G. Hinton. (2009). *CIFAR-10 (Canadian Institute for Advanced Research)*. Accessed: 2009. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[24] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/

[25] Y. LeCun, C. Cortes, and C. J. Burges. (2010). *MNIST Handwritten Digit Database. 2010*. Accessed: Sep. 2, 2019. [Online]. Available: http://yann.lecun.com/exdb/mnist

[26] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[27] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[29] 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 *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2020, pp. 979–982.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[31] D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen, J. Yang, J. Park, A. Heinecke, E. Georganas, S. Srinivasan, A. Kundu, M. Smelyanskiy, B. Kaul, and P. Dubey, "A study of BFLOAT16 for deep learning training," 2019, *arXiv:1905.12322*. [Online]. Available: http://arxiv.org/abs/1905.12322

SIDDHARTH GUPTA received the B.Tech. degree in computer science and engineering from G. L. B. I. T. M., Greater Noida, India, in 2015. He is currently pursuing the M.S. degree in computer science and engineering from IIT Indore, Indore, India. His current research interests include deep learning, approximation techniques in machine learning, and hardware realization of DNNs.

SALIM ULLAH received the B.Sc. and M.Sc. degrees in computer systems engineering from the University of Engineering and Technology at Peshawar, Pakistan. He is currently pursuing the Ph.D. degree with the Chair of Processor Design, Technische Universität Dresden. His current research interests include the design of approximate arithmetic units, approximate caches, and hardware accelerators for deep neural networks.

KAPIL AHUJA received the B.Tech. degree from IIT (BHU), India, and the M.S. and Ph.D. degrees from Virginia Tech, USA. He was a Postdoctoral Research Fellow with the Max Planck Institute, Germany. He is currently an Associate Professor in Computer Science and Engineering at IIT Indore, India. In the past, he has been a Visiting Professor with TU Braunschweig, Germany, TU Dresden, Germany, and Sandia National Labs, USA. He has a varied background, including degrees in computer science, mathematics, and mechanical engineering. He is working on the mathematics of data science and computational science, specifically machine learning, numerical linear algebra, and optimization.

ARUNA TIWARI (Member IEEE) received the B.E./M.E. degrees in computer science and engineering from SGSITS Indore, India, and the Ph.D. degree in computer science and engineering from RGPV Bhopal, India. She has a background in computer science and engineering. She has more than 20 years of teaching and research experience. She is with IIT Indore, India, since 2012. She is currently an Associate Professor of computer science and engineering. Her research interests include soft-computing learning algorithms, especially with neural networks, fuzzy clustering, and evolutionary computation for different problems for handling big data mainly for disease diagnosis and genomics.

AKASH KUMAR (Senior Member, IEEE) received the joint Ph.D. degree in electrical engineering and embedded systems from the Eindhoven University of Technology, Eindhoven, The Netherlands, and the National University of Singapore (NUS), Singapore, in 2009. From 2009 to 2015, he was with NUS. He is currently a Professor with Technische Universität Dresden, Dresden, Germany, where he is directing the chair of processor design. His current research interests include the design, analysis, and resource management of low-power and fault-tolerant embedded multiprocessor systems.