Extremely low-bit Representation for DNN Inference

Boris Bukchin - 316466259 Elizabet Khaimov – 318174810

Abstract – Convolutional neural networks are a common solution for image classification problems yet there is hardware that cannot support it because of memory and computational needs. Therefore, quantization is a frequent adjustment of CNN for fitting to the hardware. There are many methods of quantization, and we focus on floating point quantization, because values through the CNN mostly represented as 32-bit floating point. In this paper, we propose an extremely low-precision floating-point quantization-oriented processor for inference optimization. Together with the paper [1] **this method of floating-point quantization led as to our main innovative addition – dynamic bias in exponent representation for representing adjusted range of relevant values for each layer and element. Through the research, we focused on common architecture ResNet50 and analyzed each step of our study by three elements – activations, weights (with biases) and partial sums. As a result, with our additions the floating-point quantization achieves higher accuracy in relation to the traditional quantization (same number of bit representation) and more efficiency from the 32-bit floating point representation.**

I. INTRODUCTION

Deep Learning (DL) has experienced rapid growth. Due to its good performance, deep learning has shown a promising application in many new areas as intelligent surveillance, autonomous driving, and smart home.

Since many applications are highly real-time and power oriented there has been much previous work in terms of hardware-software co-design to make CNN more efficient. Such designs use dedicated accelerators for optimal power consumption and speed, and frequently take advantage of quantization and pruning techniques. Such accelerators are especially important at edge devices for inference stage.

CNNs like ResNet use convolution layers to reduce the images into a form which is easier to process. The convolution operation in these networks involves repeatedly accumulating the multiplication between input image data (activations) and filter weights to produce the output value stored in the partial sum component (PSUM).

A common method to speed up the matrix multiplication process is using a Systolic Array which is a standard in computer architecture for parallel programs. Each cell in the systolic array contains a Processing Engine, i.e., PE, which computes a single cell in the output feature map matrix. Conventional PE architecture contains a 8x8-bit multiplier, a 16-bit wide adder and a Partial Sum component, i.e., PSUM. The PSUM is a register that stores the result of the multiply-accumulate, i.e., MAC operation.

The operation involves two numbers, A is the input image data, i.e., Activations, B is the filter weights, i.e., Weights, when both A and B are quantized to 8-bit wide values. Using the multiplier, we compute the output between A_i and B_i, then accumulate it to the PSUM in the following order: PSUM $=\sum_{i=1}^{n} A_i B_i$. The output of the multiplication between Ai and Bi is up to 16-bit wide due to the 8-bit quantization performed on A and B.

Different methods of quantization can be used to reduce the bit width of activations, filter weights and PSUM. Integer quantization allows usage of integer hardware for calculation. With floating point quantization fine-tuning can be done to achieve high efficiency and accuracy.

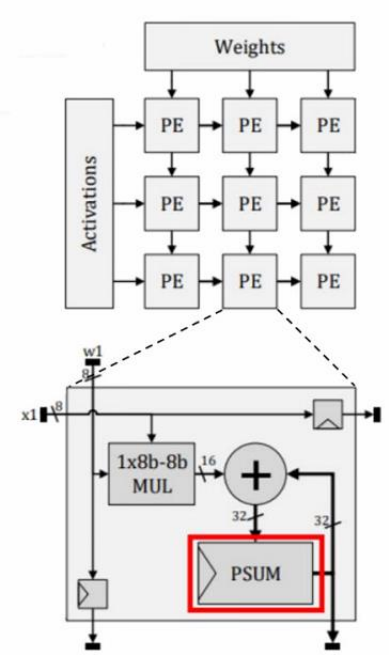

Figure 1 Systolic Array and Its Component [2]

II. METHODS

A. 5-bit Floating point

Similar to the definition of 32-bit floating-point from the IEEE-754 standard [3], the binary representation of 5-bit floating-point number contains sign, mantissa and exponent in order. The decimal value of 5-bit floating-point number is then calculated by:

$$V_{dec} = (-1)^S \times 1.M \times 2^{E-bias} \quad \text{I.}$$

where V_{dec} is the value in decimal, S, M and E are all unsigned values and denote the sign, mantissa and exponent, respectively. For bias in Eq. (1), it is introduced for both positive and negative exponents as

$$bias = 2^{E_w-1} - 1 \quad \text{II.}$$

where E_w is the data width of E. In our work we introduce "Dynamic Bias" where we choose the best bias for each calculation. Floating point representation characterized by having dense number representation for smaller absolute numbers and having higher absolute error when representing large numbers [4]. Therefor adjusting possible exponent values for their usage in NN have significant effect on NN performance. With adjusting the bias to be "Modified Bias" in all numbers where $Modified\ Bias = 2^{E_w} - 1$, we increased NN accuracy with 5 bits from 0.28% to 58.5%.

$$Bias_{dynamic} = \lfloor \log_2(\max |tensor_{element}|) \rfloor \quad \text{III.}$$

Dynamic bias is not possible for designing as realistic PE. But, with one adjustment it can – the bias of the PSUM is fixed to the modified bias. The PSUM calculated per value and the value range of the whole tensor is yet unknown at this step, therefore dynamic bias cannot be calculated without knowing the value range. This bias will be called the "realistic dynamic bias".

There are three special definitions in IEEE-754 standard. The first is subnormal numbers when E = 0, and Eq. (1) is modified to:

$$V_{dec} = (-1)^S \times 0.M \times 2^{1-bias} \quad \text{IV.}$$

Then, Infinity (Inf) and Not a Number (NaN) are the other two special cases. In our work Inf and NaN are not used since this is not used in NN inference computation. By using these special representations for more numbers, we increased 5-bit NN accuracy by 22%.

Standard IEEE Floating-point representations for 32/16/8 are:

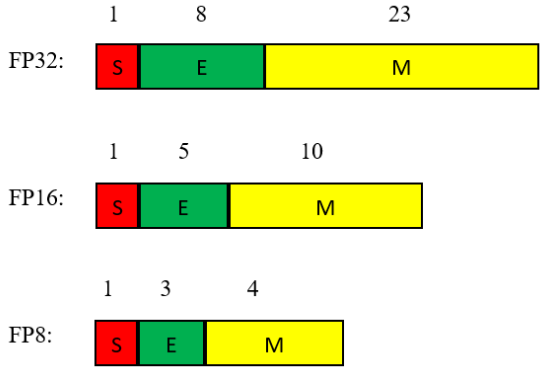

Figure 2 Floating-point standards

In our work we introduce optimized 5-bit FP protocols for each element in NN separately. We

define “Default Configuration” for 5-bit FP to be as in Figure3.

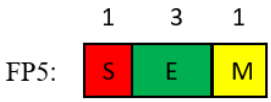

Figure 3 Default Configuration

B. FP MAC

Multipliers and accumulators (MACs) in our PEs are performing operations on FP numbers. Calculation on traditional IEEE FP numbers is done in previous works [1]. We introduce a modified multiplication when using dynamic bias:

A=activations, W=weights, P=PSUM.

1. Multiplying the significand.

$$(1.M_A * 1.M_W)$$

2. Placing the decimal point in the result

3. Adding the exponents.

$$(E_A + E_W - Bias_W - Bias_A + Bias_P)$$

4. Obtaining the sign.

$$S_A \ (xor) \ S_W$$

5. Normalizing the result; i.e. obtaining 1 at the MSB of the results’ significand.

6. Rounding the result to fit in the available bits.

7. Checking for underflow/overflow occurrence

For simplicity in hardware, we use dynamic bias only for activations and weights while keeping PSUM bias at “modified bias”.

C. RESNET50

In this project we discuss the effect of different floating-point protocols on ResNet50 DNN performance with ImageNet dataset. Initial reference is a trained ResNet50 with 76% top-1 accuracy on ImageNet with 32 standard FP representation.

There are many architectures to examine our ideas for improvement. We decided to focus on common group of CNN – ResNet models and particularly on ResNet50. After executing inferences and compare the accuracies for basic quantization (8 standard FP representation) ResNet50 accuracy is 70.5% top1 on ImageNet. Thus, the accuracy is not too low, and we can quantize even more than 8 standard FP representation.

ResNet contains 2 main layers that are quantized:

1. Convolution layers, where kernel weights and activations are multiplied and added with partial sums (PSUM). Notice that in this architecture most of the convolution layers appear after RELU layer. Thus, the values of activation are non-negative.
2. FC layers, fully connected layers where all weights are multiplied with all activations and added with PSUMs.

D. QUANTIZATION

We use quantization for nearest neighbor method, where activations and weights are preprocessed to best representation of 5-bit FP.

All calculations in NN are performed on 5-bit FP MACs, therefor PSUM registers in the NN are 5-bit FP as well.

III. RESULTS AND DISCUSSION

A. Standard Floating-Point Quantization

First, grid search on number of representation bits for activation and weights with the assumption that PSUM is the activation of the next layer therefore the number of representing bits is the same for activations and PSUM. The quantization includes representation of special cases (NaN and Inf).

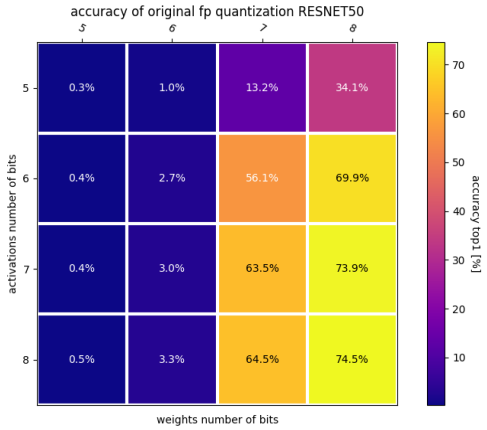

Figure 4 accuracy top1 of ResNet50 by bit number quantization

For all the presentations the number for exponent in the bits vector is the same while the mantissa changes according to number of representing bits.

From Figure3 can be inferred, mantissa length is critical for weights. However, activations do not benefit from detailed mantissa in the same percentage. We decided to improve the smallest representation 5 bits for all the elements.

B. Elements Order Bits

Afterwards, we added another grid search with 5 floating point bit representation with different combinations of bits for exponent and mantissa.

Understanding the architecture brought up the assumption, that activations are mostly non-negative because the convolution layer appear after RELU layer. Therefore, the search included order bits without the sign bit for each element. Except the first activations that is quantized as PSUM because there is no RELU before the first convolution layer.

In this search, more revealed about the relevance of each part (exponent, mantissa, sign). The accuracy is higher for weights and PSUM with larger exponent, hence can represent larger range of small number. Therefor decreasing the exponent from 3 bits is destructive to accuracy. In addition, removing the sign bit for weighs or PSUM also have a destructive effect.

C. Sign Removal

Removing the sign bit led to the idea of removing it dynamically when there are small percentage of negative values by zero them. At last, it derived minor improvement (about 0.2%) and we decided to leave off this idea.

D. Modified Bias

With 5-bit floating point representation, there is 2^5 different numbers as the number of different combinations of bits.

The range of the values in each element is different but stays similar through the NN with small changes. Most of the values are small (less than 1) but the range that can be represented in the default configuration is $[-2^4, 2^4]$ There are unnecessary representations (numbers bigger than 1), and modifications of bias can set this range to represent more detailed range of numbers in range $[-2^0, 2^0]$. Therefore, the modified bias is set to be 7.

Figure 5 is another grid search as in section A but with "modified bias". For most of the configurations there is a big improvement.

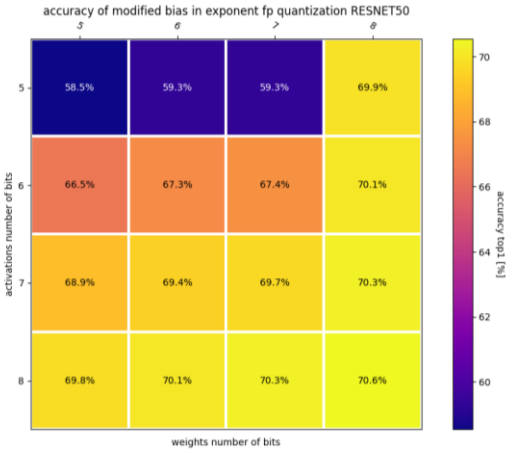

Figure 5 accuracy top1 of ResNet50 by bit number quantization with modified bias

E. NaN and Inf Representation

As described in section 2.A, in NN inference NaN and Inf are not used. Therefor we used these representations for more FP numbers. We

experimented the effect of these additional representations in 5-bit FP, with default exponent, mantissa and sign for all elements. Resulting in 22% increase in accuracy, from 58.5% to 70.5%.

With all above optimizations we performed a sweep through all possible combinations of sign, exponent and mantissa sizes. We found that the best accuracy for 5-bit FP is 73.48% for following configuration in figure 6:

Figure 6 Optimal Configuration

F. Dynamic Bias

An upgrade for the modified bias, the bias dynamically depends on the edges of the range for each layer and each element separately.

The histograms below display the accuracy of our suggested quantization (optimized quantization) against the traditional 5 bits floating point quantization. It can be inferred that our quantization preserves more values of the original tensor with the same number of bits.

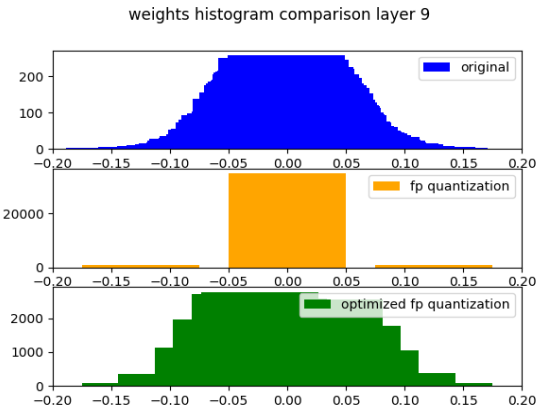

Figure 7 histogram comparison of values for weights in layer 9

G. Real Dynamic Bias

Adapting the dynamic bias to realistic physical design is keeping accuracy higher from the traditional 5 bits floating point but a little lower than the ideal dynamic bias.

For the same bits order (sign 1 bit, exponent 3 bits, mantissa 1 bit):

	Accuracy top1 [%]
Dynamic bias	72.292
Realistic dynamic bias	71.764

H. Overall

In our work we show significant accuracy improvement with described optimizations. In figure 8 we summarize the best results achieved in this project.

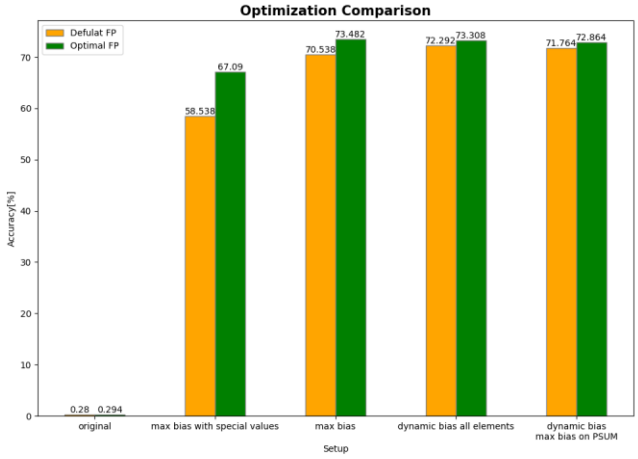

Figure 8 Optimizations summary

While some optimizations are more costly in hardware than others, we show in figure 9 the best options to use 5-bit FP quantization.

In “optimal configuration” – we use activations with different FP configurations than weights. And use “Modified Bias” in all layers and elements, producing best results.

In “realistic configuration” – we use same FP configuration for all elements. And use “realistic dynamic bias”, producing slightly lesser results.

Both configurations achieve better results than default 8FP and showing small degradation in comparison to 32-bit original NN.

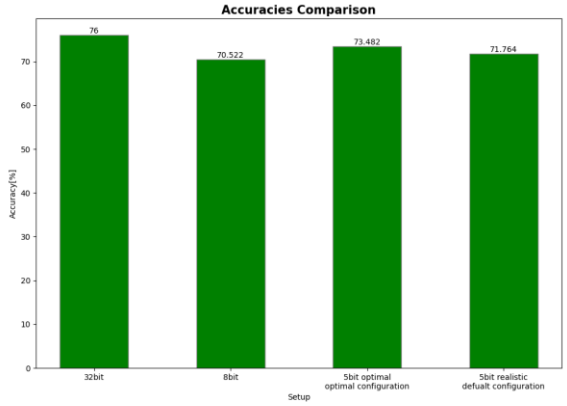

Figure 9 Configuration Comparison

I. Physical Analysis

We compared 32-bit FP MAC [5] and 5-bit FP MAC area and power performance. We used "realistic configuration" for the 5-bit FP MAC. For the comparison we used 45nm technology and used "design vision" by synopsis as synthesis tool. The results are described in table1.

	Total dynamic power[uW]	**Total cell area [half nand gate]**
32bit MAC	495	3779
5bit MAC	45	310

Table 1 Physical Comparison

There is a significant improvement both in power and area smaller 5-bit design. As for the described in table1 we get x11 less power consumption with x12 less area. Featuring much more efficient design than traditional 32-bit NN accelerators.

IV. CONCLUTION

The base of neural networks is tensor multiplication with floating point numbers and optimizing it have huge impact on neural networks performance in terms of power and area. In this paper, we improved the basic floating-point quantization to 5 bits extremely low bit representation and yet preserved the NN real life applicable. Our main contributions are finding the best configuration of 5 bits floating point representation and adjusting the exponent bias value to optimal for each element and layer.

For future works, design compatible MACs to support dynamic bias and different FP configurations as mentioned in this work. Moreover, examine the influence of dynamic bias in different granularities of tensor (such as channels, groups, columns).

V. REFERNCES

[1] C. W. e. al, "Phoenix: A Low Precision Floating Point Quantization Oriented Architecture For Convolutional Neural Networks," arxiv, 2020.

[2] U. W. Gil Shomron, "Non-Blocking Simultaneous Multithreading: Emabracing the Resiliency of Deep Neural Networks," 2020.

[3] D. Mangla, "Quantization Arithmetic," 2020. [Online]. Available: https://heartbeat.comet.ml/quantization-arithmetic-421e66afd842.

[4] B. W. e. al, "Basics of Floating-Point Quantization," in *Quantization Noise*, Cambridge University, 2010, pp. 257-273.

[5] U. M. e. al, "Implementation of Floating Point Multiplier," IJRECE, 2018.