Mix and Match: A Novel FPGA-Centric Deep Neural Network Quantization Framework

Sung-En Chang*[1], Yanyu Li*[1], Mengshu Sun*[1], Runbin Shi[2], Hayden K.-H. So[2], Xuehai Qian[3], Yanzhi Wang[1], Xue Lin[1]

[1]*Northeastern University,* [2]*The University of Hong Kong,* [3]*University of Southern California*

{chang.sun, li.yanyu, sun.meng, yanz.wang, xue.lin}@northeastern.edu, {rbshi, hso}@eee.hku.hk, xuehai.qian@usc.edu

***Abstract*—Deep Neural Networks (DNNs) have achieved extraordinary performance in various application domains. To support diverse DNN models, efficient implementations of DNN inference on edge-computing platforms, e.g., ASICs, FPGAs, and embedded systems, are extensively investigated. Due to the huge model size and computation amount, model compression is a critical step to deploy DNN models on edge devices. This paper focuses on weight quantization, a hardware-friendly model compression approach that is complementary to weight pruning.**

Unlike existing methods that use the same quantization scheme for all weights, we propose the *first* solution that applies different quantization schemes for different rows of the weight matrix. It is motivated by (1) the distribution of the weights in the different rows are not the same; and (2) the potential of achieving better utilization of heterogeneous FPGA hardware resources. To achieve that, we first propose a hardware-friendly quantization scheme named *sum-of-power-of-2 (SP2)* suitable for Gaussian-like weight distribution, in which the multiplication arithmetic can be replaced with logic shifter and adder, thereby enabling highly efficient implementations with the FPGA LUT resources. In contrast, the existing fixed-point quantization is suitable for Uniform-like weight distribution and can be implemented efficiently by DSP. Then to fully explore the resources, we propose an FPGA-centric mixed scheme quantization (MSQ) with an ensemble of the proposed SP2 and the fixed-point schemes. Combining the two schemes can maintain, or even increase accuracy due to better matching with weight distributions.

For the FPGA implementations, we develop a parameterized architecture with heterogeneous Generalized Matrix Multiplication (GEMM) cores—one using LUTs for computations with SP2 quantized weights and the other utilizing DSPs for fixed-point quantized weights. Given the partition ratio among the two schemes based on resource characterization, MSQ quantization training algorithm derives an optimally quantized model for the FPGA implementation. We evaluate our FPGA-centric quantization framework across multiple application domains. With optimal SP2/fixed-point ratios on two FPGA devices, i.e., Zynq XC7Z020 and XC7Z045, we achieve performance improvement of $\mathbf{2.1\times - 4.1\times}$ compared to solely exploiting DSPs for all multiplication operations. In addition, the CNN implementations with the proposed MSQ scheme can achieve higher accuracy and comparable hardware utilization efficiency compared to the state-of-the-art designs.

***Index Terms*—deep neural network, quantization, FPGA, inference**

I. Introduction

Deep learning or Deep Neural Networks (DNNs) have achieved extraordinary performance in various application domains [1]–[7]. However, the state-of-the-art DNNs may require up to GBs (Giga Bytes) for model size and 10^2 GFLOPs (Giga Floating Point Operations) for inference computation, making it a challenging task to perform on-device inference.

To efficiently execute the diverse DNN inference models for broader applications, the resource-constrained edge computing platforms require two crucial supports. The first one is the specialized hardware acceleration for DNN inference. Extensive research efforts have been dedicated to the efficient implementations of DNN inference models on various edge-computing platforms, such as ASICs [8]–[14], FPGAs [15]–[18], and embedded CPUs/GPUs [19]–[23].

The second is the DNN model compression technique, which not only seeks more efficient hardware implementation based on given models, but also explores the opportunity of algorithm and hardware co-design to achieve better trade-offs among accuracy, hardware cost, and performance. There are two essential techniques for model compression: DNN weight pruning [24]–[30] and weight quantization [31]–[47].

This paper focuses on DNN weight quantization, which becomes imperative to the DNN hardware acceleration especially on the FPGA and ASIC platforms. By representing weights with fewer bits, weight quantization can directly simplify the implementations and accelerate the inference execution speed in a *hardware-friendly manner*. Also, it is supported in GPUs (e.g., PyTorch [22] for NVIDIA GPUs) and mobile devices (e.g., TensorFlow-Lite [23]). In addition, weight quantization yields far less training overhead than weight pruning, let alone the training-heavy network architecture search (NAS)-based model compression techniques. Specifically, in state-of-the-art DNN quantization methods (including our work), retraining process takes usually $1/3 \sim 1/2$ of the epochs as those for the pre-training process, which is totally acceptable training overhead in the exchange for significant inference speedup.

Weight quantization can be considered as a mapping from 32-bit floating-point weights into m-bit weight representations. There are different types of quantization schemes including binary [31]–[34], ternary [35]–[37], low-bit-width fixed-point [38]–[43], and power-of-2 [44]–[47]. In general, binary and ternary quantization schemes result in significant accuracy loss, for example, $> 5\%$ under binary and $2\% - 3\%$ for ternary quantization. The fixed-point quantization can represent the DNN weights using low bit-width, e.g., 4-bit, with negligible accuracy loss. To further simplify hardware implementations, power-of-2 quantization scheme was proposed to replace the

*Equal contribution.

multiplications with bit-shifting operations. However, power-of-2 results in non-negligible accuracy degradation, usually around $1\% - 2\%$, which even cannot be overcome with increasing precision.

To overcome the challenges, instead of using the same quantization scheme for all weights, we propose the *first* solution that applies different quantization schemes for different rows of the weight matrix. It is motivated by (1) the distribution of weights in the different rows are not the same; and (2) the potential of achieving better utilization of heterogeneous FPGA hardware resources. We propose a hardware-friendly quantization scheme named *sum-of-power-of-2 (SP2)* suitable for Gaussian-like weight distribution, in which the multiplication arithmetic can be replaced with logic shifter and adder, thereby enabling highly efficient implementations with the FPGA LUT resources. At the same time, the SP2 quantization enjoys the negligible accuracy loss, just like the fixed-point quantization scheme. In comparison, the fixed-point quantization is suitable for Uniform-like weight distribution and can be implemented efficiently by DSP.

To fully explore the FPGA resources, we propose an FPGA-centric mixed scheme quantization (MSQ) with an ensemble of the proposed SP2 and the fixed-point schemes. Given that each individual scheme can achieve negligible accuracy loss, we demonstrate that combining the two can maintain, or even reach higher accuracy. It is due to the benefit of using two quantization schemes: even within a single layer, the local weight distributions can be diverse, if we assign the right quantization scheme to better fit the local weight distributions, accuracy can be boosted.

For the FPGA implementations, we developed a parameterized architecture with heterogeneous Generalized Matrix Multiplication (GEMM) cores—one using LUTs for computations with SP2 quantized weights and the other utilizing DSPs for fixed-point quantized weights. We first find the partition ratio of the SP2 to fixed-point quantization for weights of a DNN layer through FPGA resource characterization, such that the DSP utilization is kept at 100% and LUT utilization can also be optimized. Given the partition ratio, MSQ quantization training algorithm derives an optimally quantized model for the FPGA implementation. We evaluate our FPGA-centric quantization framework across multiple application domains including image classification, object detection and recognition, machine translation, speech recognition, sentiment classification, and natural language processing, with various DNNs such as convolutional neural networks (CNN), and recurrent neural networks (RNN). With optimal SP2/fixed-point ratios on two FPGA devices, i.e., Zynq XC7Z020 and XC7Z045, we achieve performance improvement of $\mathbf{2.1}\times$ and $\mathbf{4.1}\times$ compared to solely exploiting DSPs for all multiplication operations. In addition, the CNN implementations with the proposed MSQ scheme can achieve higher accuracy and comparable hardware utilization efficiency compared to state-of-the-arts.

The contributions of this work are:

- We propose a novel hardware-friendly SP2 quantization scheme, which enjoys both non-multiplication operations and negligible accuracy degradation.
- We provide the first DNN quantization solution that jointly applies two quantization schemes to achieve better utilization of heterogeneous FPGA hardware resources while not harming the quantized model accuracy.
- Our framework features a novel architecture with heterogeneous GEMM engines and design optimizations, to accommodate our mixed scheme quantization and to optimize FPGA resource allocation.
- The effectiveness of our proposed MSQ is validated across multiple application domains and with FPGA devices, on inference accuracy and FPGA resource utilization efficiency.

Our work is significantly different from existing quantization frameworks that leverage the inter-layer, multi-precision approach. We exploit the previously neglected flexibility on quantization schemes (using both fixed-point and SP2) by adopting a novel *intra-layer, multi-scheme* approach. Specifically, we identify an optimized ratio of the two schemes from FPGA (LUT and DSP) resource characterization, and then assign the different rows of the weight matrix *within a layer* into the two schemes according to the weight distributions. Our method is totally perpendicular to, and can be combined with, the existing inter-layer, multi-precision approaches.

II. Background on DNN Weight Quantization

A. Weight Quantization Schemes

1) Uniform Interval Quantization Schemes: Uniform interval quantization schemes include binary, ternary, and low-bit-width fixed-point. Binary or ternary quantization uses extremely low precision for DNN models, i.e., binarized (e.g., -1, +1) or ternarized (e.g., -1, 0, +1) levels. Representative binary quantization methods include Binaryconnect [31], Binarized Neural Network (BNN) [32], XNOR-net [33], and ABC-Net [34]. With weights constrained to $\{-1, 1\}$, multiplications can be replaced by additions/subtractions. Additions/subtractions can also be eliminated using XNOR and AND operations if activations are quantized to binary as well. On the other hand, ternary quantization schemes are implemented in TWN [35], TTQ [36], and [37]. Ternary representation keeps zero in quantization levels, which requires one more bit to present weights. Ternary networks also benefit from non-multiplication operations while maintaining the natural sparsity (since zero weights are kept). Although binary and ternary quantization can significantly reduce operations and simplify the implementations of hardware accelerators, it introduces non-negligible accuracy loss. For example, based on reports from the above works, accuracy typically degrades by $> 5\%$ under the binary scheme, and $2 - 3\%$ for ternary.

Comparing with binary and ternary quantization, the fixed-point quantization scheme applies the modest and flexible quantization rates to preserve the accuracy as that of the 32-bit floating-point models. For example, 4-bit fixed-point introduces zero or negligible accuracy loss. Fixed-point quantization scheme has been implemented with different meth-

ods/algorithms by DoReFa-Net [38], PACT [39], DSQ [40], QIL [41], μ L2Q [42], and LSQ [43].

With the m-bit fixed-point scheme, quantized weight values are defined as the scaling factor α times quantization levels:

$$\mathcal{Q}^{FP}(m,\alpha) = \pm\alpha \times \{0, \frac{1}{2^{m-1}-1}, \frac{2}{2^{m-1}-1}, ..., 1\}. \quad (1)$$

And the mapping from a 32-bit floating-point weight w into the quantized weight $\hat{w}$ by m-bit fixed-point representation (in sign-magnitude) is given by the following quantizer:

$$\begin{aligned}\hat{w} &= \prod_{\mathcal{Q}^{FP}(m,\alpha)} w \\ &= \alpha \cdot h^{-1}\big(\frac{1}{2^m-1} round((2^m-1)\cdot h(\lceil w,\alpha \rfloor))\big),\end{aligned} \quad (2)$$

where $\prod_{\mathcal{Q}^{FP}(m,\alpha)}(\cdot)$ denotes the quantizer function to project onto $\mathcal{Q}^{FP}(m,\alpha)$; the function $h(\cdot)$ transforms a value within $[-1,+1]$ into the range of $[0,1]$, for example we can use $h(\cdot) = \tanh(\cdot)/2 + 0.5$; and $\lceil w,\alpha \rfloor$ clips w according to

$$\lceil w,\alpha \rfloor = \begin{cases} -1, & w < -\alpha \\ w/\alpha, & -\alpha \le w \le \alpha \\ 1, & w > \alpha \end{cases}. \quad (3)$$

2) Non-Uniform Interval Quantization Schemes: On the other hand, power-of-2 quantization is a non-uniform interval quantization scheme, representative methods including [44]–[47]. Power-of-2 quantization replaces multiplications by bit shifting operations and this number system also possesses higher precision around the mean, which fits the Gaussian distribution of DNN weights better [48], [49]. With an m-bit weight representation (in sign-magnitude), the quantized weight values by the power-of-2 scheme are defined as

$$Q^{P2}(m,\alpha) = \pm\alpha \times \{0, \frac{1}{2^{2^{m-1}-2}}, \frac{1}{2^{2^{m-1}-3}}, ..., 1\}. \quad (4)$$

And the power-of-2 quantizer is then given by

$$\begin{aligned}\hat{w} &= \prod_{\mathcal{Q}^{P2}(m,\alpha)} w \\ &= \begin{cases} \alpha \cdot h^{-1}\big(2^{round(\log_2 h(\lceil w,\alpha \rfloor))}\big) & h(\lceil w,\alpha \rfloor) > 2^{-2^m+1} \\ 0 & h(\lceil w,\alpha \rfloor) \le 2^{-2^m+1} \end{cases}.\end{aligned} \quad (5)$$

With weights quantized into the power-of-2 scheme, multiplications between weight i.e., $2^b (b \in \mathbb{N})$ and activation i.e., a can be implemented by bit shifting as follows:

$$2^b \times a = \begin{cases} a << b, & b > 0 \\ a, & b = 0 \\ a >> b, & b < 0 \end{cases}. \quad (6)$$

Although the power-of-2 quantization scheme can simplify hardware implementation by eliminating multiplications, its precision cannot be increased effectively with increasing m, because increasing m will merely increase resolution around the mean, while the tails are still in low precision. This can also be observed from Eq (5) that when w is a large value, increasing m does not have an effect on $\hat{w}$. In practice, $3 \sim 7$ bits are usually used for power-of-2 quantization, and more bits could not further promote the accuracy of the quantized models. As mentioned in §II-A1 that 4-bit fixed-point results in negligible accuracy degradation, but 4-bit power-of-2 quantization will result in accuracy loss of $1\% - 2\%$.

Algorithm 1: DNN Quantization with ADMM and STE

input : 32-bit floating-point DNN model $\mathcal{M}$, with weights $\mathbf{W}$ to be quantized.
Quantization scheme: $\mathbb{S} \in$ {Fixed-point, Power-of-2, Sum-of-power-of-2}
target: Quantized model $\hat{\mathcal{M}}$

```
// Initialization:
```
$U^0 = 0$; $Z^0 = \mathbf{W}$;

foreach *Epoch* **do**
```
    // Update Z, U:
```
    $Z^t \leftarrow \mathbf{proj}_{\mathbb{S}}(\mathbf{W} + U^{t-1})$;
    $U^t \leftarrow \mathbf{W} - Z^t + U^{t-1}$;
    foreach *Batch* **do**
```
        // STE for activation quantization:
```
        $input \leftarrow \mathbf{proj}_{\mathbb{S}}(input)$;
        $loss \leftarrow \mathcal{M}(input)$;
        $loss \leftarrow loss + \sum \frac{1}{2}\|\mathbf{W} - Z^t + U^t\|^2$;
        Backpropagate $loss$ and update $\mathbf{W}$;

Return $\hat{\mathcal{M}} \leftarrow \mathcal{M}\{\mathbf{proj}_{\mathbb{S}}(\mathbf{W})\}$.

B. Quantization Algorithms

Quantization performs projection from the continuous domain to a discrete number system, which makes the gradients of the loss function unavailable for backpropagation during the training. Two approaches can be applied to solving this unavailable gradient issue. One is employing a Straight Through Estimator (STE) [50], [51] to set the gradient to the constant value of 1 as

$$\begin{aligned}&\textbf{Forward}: y = round(x) \\ &\textbf{Backward}: \frac{\partial y}{\partial x} = \mathbf{1}_{x \in R}\end{aligned}, \quad (7)$$

which is effective in the quantization training. The other approach employs Alternating Direction Method of Multipliers (ADMM) to iteratively solve the parameters with a target quantization scheme as the optimization contraint [47], eliminating the need to backpropagate through the quantizer. In this work, we use a combination of ADMM and STE, as shown in Algorithm 1, which in general follows the ADMM algorithm for weight quantization and where the STE is only applied for activation quantization.

III. Sum-of-Power-of-2 (SP2) Quantization Scheme

In this section, we propose a new hardware-friendly *sum-of-power-of-2 (SP2) quantization scheme*, which enjoys the non-multiplication operations for the inference computation as the binary, ternary, and power-of-2 schemes, while achieving negligible inference accuracy degradation.

TABLE I
ANALYSIS ON THE OPERATIONS FOR WEIGHT-ACTIVATION MULTIPLICATION BY TWO QUANTIZATION SCHEMES OF THE WEIGHTS.

	Weight	Activation	Ops for Weight × Activation
Quantization Scheme	m-bit fixed-point	n-bit fixed-point	n-bit addition for $m-2$ times
Operands	$(m-1)$-bit integer	n-bit integer	
Quantization Scheme	m-bit SP2	n-bit fixed-point	shift by up to $2^{m_1}-2$ bits
Operands	m_1-bit integer, m_2-bit integer $m_1+m_2=m-1, m_1 \geq m_2$	n-bit integer	shift by up to $2^{m_2}-2$ bits up to $(n+2^{m_1}-2)$-bit addition

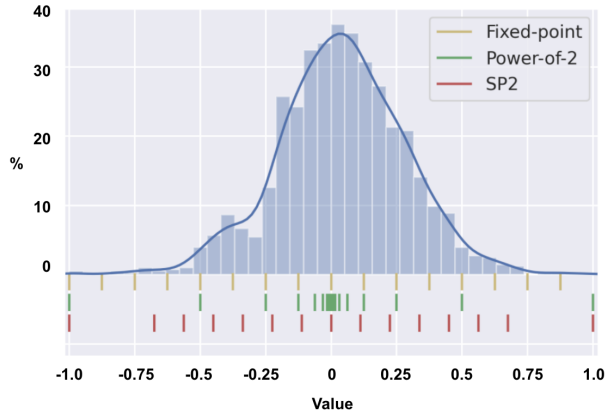

Fig. 1. **Quantization levels by fixed-point, power-of-2, and SP2 in 4-bit weight representation precision, and weight probability distribution of the 4th layer in MobileNet-V2.**

A. SP2 Quantization Scheme

The proposed hardware-friendly sum-of-power-of-2 (SP2) quantization scheme can be considered as a variant of the power-of-2 quantization. SP2 scheme can eliminate multiplication operations in the (quantized) DNN inference models (as the power-of-2 scheme), and at the same time is designed to address the non-negligible accuracy loss of power-of-2 quantization. This is achieved by solving the low precision issue in the tail ends of the weight distribution.

Formally, the quantized weight values by the sum-of-power-of-2 scheme with a total of m-bit representations are

$$\begin{aligned} &Q^{SP2}(m,\alpha) = \pm\alpha \times \{q_1 + q_2\}, \\ &q_1 \in \{0, \frac{1}{2^{2^{m_1}-1}}, \frac{1}{2^{2^{m_1}-2}}, ..., \frac{1}{2}\}, \\ &q_2 \in \{0, \frac{1}{2^{2^{m_2}-1}}, \frac{1}{2^{2^{m_2}-2}}, ..., \frac{1}{2}\}, \end{aligned} \tag{8}$$

where q_1 and q_2 are power-of-2 numbers in similar format as the quantization levels in Eq. (4), and m_1 and m_2 are the number of bits to represent the power-of-2 numbers i.e., q_1 and q_2, respectively. Please note that with a total of m bits to represent an SP2 quantized weight value, 1 bit is still reserved for the sign bit, and therefore we have $m_1+m_2+1=m$ with $m_1 \geq m_2$. In addition, the quantization levels by SP2 i.e., $\pm\{q_1+q_2\}$ are within $[-1,+1]$.

Note that with m-bit representations, the SP2 scheme provides a total of $2^{m_1} \times 2^{m_2} \times 2 - 1 = 2^m - 1$ quantization levels. Although the power-of-2 quantization scheme in m-bit representations also provide $2^m - 1$ quantization levels, the quantization levels resulted from the two schemes scatter distinctly, and therefore the schemes perform differently in preserving the accuracy of the quantized models. In Figure 1, the curve represents the actual probability distribution of DNN weights in a representative layer. Along the x-axis, we label the quantization levels by fixed-point, power-of-2, and our sum-of-power-of-2 quantization schemes. All the three schemes use 4-bit representations and therefore each of them has 15 quantization levels within $[-1,+1]$.

First, let us understand the intuition why the power-of-2 quantized models incur the non-negligible accuracy degradation, while SP2 and fixed-point quantized models can achieve similar accuracy performance. The power-of-2 scheme has very high precision around the mean with only 4-bit weight presentation, but the tail ends present very low precision. In contrast, our SP2 quantization possesses relatively evenly scattered quantization levels, which is close to that of fixed-point quantization levels, except the tail ends where very few weight values are presented. This explains the advantages of SP2 quantization scheme.

Next, we analyze the effect of SP2 quantization scheme on the computation of weight-activation multiplication. In Table I, we compare fixed-point and SP2 quantization schemes of the weights, while throughout this paper we use fixed-point quantization for the activation. In the first scheme with m-bit fixed-point quantization for the weight and n-bit fixed-point quantization on the activation, the weight operand is actually represented as the $(m-1)$-bit unsigned integer, since 1 bit is for the sign. Although a quantization level is within $[-1,+1]$, the actual weight operand is the $(m-1)$-bit unsigned integer. And the activation operand is directly represented as the n-bit unsigned integer, because activations are non-negative. The operations for implementing weight-activation multiplication are therefore n-bit additions for $(m-2)$ times.

For the second scheme with m-bit SP2 quantization on the weight, we have an m_1-bit unsigned integer and an m_2-bit unsigned integer together to encode the quantization level of the quantized weight, and $m_1+m_2=m-1$ because 1 bit is for the sign. The quantization level is then $2^{b_1}+2^{b_2}$, where b_1 and b_2 are encoded with m_1 and m_2 bits, respectively. The weight-activation multiplication is implemented by (1) shift of the activation operand by b_1 bits, (2) shift of the activation operand by b_2 bits, and (3) addition of the two

TABLE II
RESULT FROM DIFFERENT QUANTIZATION SCHEMES FOR THE RESNET-18 AND MOBILENET-V2 DNN MODELS ON CIFAR10, CIFAR100, AND IMAGENET DATASETS.

Quantization Scheme	Bit width (Wght./Actv.)	ResNet-18 Accuracy (%) Top1	Top5	MobileNet-v2 Accuracy (%) Top1	Top5
CIFAR10					
Baseline (FP)	32/32	93.62	-	92.51	-
P2	4/4	92.97 (−0.65)	-	91.34(−1.17)	-
Fixed	4/4	93.43 (−0.19)	-	92.34 (−0.17)	-
SP2	4/4	93.47 (−0.15)	-	92.72 (+0.21)	-
MSQ (half/half)	4/4	93.53 (−0.09)	-	92.57 (+0.06)	-
MSQ (optimal)	4/4	93.65 (+0.03)	-	92.55 (+0.04)	-
CIFAR100					
Baseline (FP)	32/32	74.49	92.70	71.48	91.98
P2	4/4	73.88 (−0.61)	92.14 (−0.56)	68.68 (−2.80)	90.06 (−1.92)
Fixed	4/4	74.37 (−0.12)	92.31 (−0.39)	71.16 (−0.32)	91.63 (−0.35)
SP2	4/4	74.33 (−0.17)	92.49 (−0.21)	71.13 (−0.35)	91.69 (−0.29)
MSQ (half/half)	4/4	74.58 (+0.09)	92.39 (−0.31)	71.21 (−0.27)	91.74 (−0.24)
MSQ (optimal)	4/4	74.60 (+0.11)	92.63 (−0.07)	71.50 (+0.02)	91.82 (−0.16)
		ImageNet		Wght./Actv. 4/32	
Baseline (FP)	32/32	69.76	89.08	71.88	90.29
P2	4/4	68.20 (−1.56)	87.14 (−1.94)	69.93(−1.95)	88.63(−1.66)
Fixed	4/4	69.72 (−0.04)	88.67 (−0.41)	71.26 (−0.62)	90.18 (−0.11)
SP2	4/4	69.74 (−0.02)	88.71 (−0.37)	71.32 (−0.56)	90.17 (−0.12)
MSQ (half/half)	4/4	70.11 (+0.35)	89.41 (+0.33)	71.26 (−0.62)	90.04 (−0.25)
MSQ (optimal)	4/4	70.27 (+0.51)	89.42 (+0.34)	71.31(−0.57)	90.11(−0.18)

shifted operands. Since b_1 and b_2 are encoded by m_1- and m_2-bit unsigned integer, respectively, Operations (1) and (2) can be shift by at most $2^{m_1}-2$ and $2^{m_2}-2$ bits, respectively. The shifted activation operands will be $n+2^{m_1}-2$ and $n+2^{m_2}-2$ bits respectively. Therefore one $(n+2^{m_1}-2)$-bit addition is needed. In summary, with SP2 weight quantization, the weight-activation multiplication can be implemented with two shift operations and one addition operation.

B. Accuracy Performance Analysis

In this section, we discuss the accuracy performance of the fixed-point (Fixed), power-of-2 (P2), and proposed sum-of-power-of-2 (SP2) quantization schemes. Our baseline models use 32-bit floating-point (FP) for both weights and activations. All the quantization schemes apply 4-bit quantization. While different quantization schemes are explored for weights, activations are using fixed-point quantization. Table II summarizes the quantized models' accuracy with the accuracy changes (with respect to the baseline FP models) marked in the brackets. Experiments are conducted with ResNet-18 and MobileNet-v2 models on CIFAR10, CIFAR100, and ImageNet. As for activation quantization, 4-bit fixed-point is used for all the models, except the MobileNet-v2 on ImageNet dataset. MobileNets are a family of specialized and lightweight models, therefore presenting unstable convergence under modifications such as pruning and quantization. Activation quantization of MobileNet-v2 on ImageNet is performed in §IV-C with comparisons to existing works.

In Table II, we now focus on P2, Fixed, and SP2 quantization schemes, and the MSQ scheme will be discussed in §IV. First, power-of-2 (P2) results in significant accuracy degradation, around 1%∼2% in general with extreme case of 2.80% Top-5 accuracy loss of MobileNet-v2 on CIFAR100. For ImageNet, both the fixed-point (Fixed) and sum-of-power-of-2 (SP2) schemes have negligible accuracy loss, $\leq 0.41\%$ for ResNet-18 and $\leq 0.62\%$ for MobileNet-v2 accross the three datasets. These two schemes achieve comparable accuracy of quantized models. In summary, the 4-bit-width Fixed and SP2 quantization schemes are essentially equivalent in terms of the accuracy of the quantized models, and their accuracy losses are negligible.

IV. A FPGA-CENTRIC MIXED SCHEME QUANTIZATION

In this section, we propose our mixed scheme quantization (MSQ) for FPGA implementations of DNN inference models. Based on the analysis in §III-B, fixed-point and SP2 quantization are equivalent in preserving the accuracy of the quantized models when with the same precision, e.g., 4-bit for both. Therefore, in our proposed MSQ, the fixed-point and SP2 schemes with the 4-bit precision are applied in the DNN model quantization, (1) for better FPGA resource allocation, and (2) with negligible accuracy loss.

A. Motivation

The idea of proposed mixed scheme quantization (MSQ) is to partition DNN weights in the same layer into two categories, one is handled by fixed-point quantization, and the other by SP2 quantization. These two schemes use the same precision to facilitate hardware implementation. The motivations to adopt MSQ are: First, let a weight matrix be obtained by transforming the weight tensor of a layer into a 2D GEMM matrix with rows and columns. Weights in different rows of the matrix may present *different distributions*. For rows with more Gaussian-like weight distributions (with smaller variances), SP2 quantization is preferable; while for rows with more Uniform-like weight distributions (with larger variances),

Algorithm 2: FPGA-Centric Mixed Scheme Quantization(MSQ)

input : 32-bit floating-point DNN model $\mathcal{M}$, with weights $\mathbf{W}$ to be quantized.
target: Quantized model $\hat{\mathcal{M}}$

// Initialization:
$U^0 = 0$; $Z^0 = \mathbf{W}$;
Partition rate PR_{SP2} from FPGA resource characterization;
$\mathbb{S}_f$ =Fixed-point; $\mathbb{S}_p$ =SP2;
foreach *Epoch* **do**
  Calculate variance $v_r^{(l)}$ for each r-th row of the layer l weight matrix $\mathbf{W}^{(l)}$;
  Sort $v_{1:R}^{(l)}$ to obtain the threshold $\theta^{(l)}$ such that PR_{SP2} of the rows with variances less than $\theta^{(l)}$;
  if $v_r^{(l)} < \theta^{(l)}$ **then** $\mathbb{S} \leftarrow \mathbb{S}_p$;
  else $\mathbb{S} \leftarrow \mathbb{S}_f$;
  // Update Z, U:
  $Z^t \leftarrow \mathbf{proj}_{\mathbb{S}}(\mathbf{W} + U^{t-1})$;
  $U^t \leftarrow \mathbf{W} - Z^t + U^{t-1}$;
  foreach *Batch* **do**
    $input \leftarrow \mathbf{proj}_{\mathbb{S}}(input)$;
    $loss \leftarrow \mathcal{M}(input)$;
    $loss \leftarrow loss + \sum \frac{1}{2}\|\mathbf{W} - Z^t + U^t\|^2$;
    Backpropagate $loss$ and update $\mathbf{W}$;
Return $\hat{\mathcal{M}} \leftarrow \mathcal{M}\{\mathbf{proj}_{\mathbb{S}}(\mathbf{W})\}$.

fixed-point quantization should be used. Thus, the mixed scheme is necessary at *algorithm level*—it can achieve similar or even potentially higher accuracy than existing schemes. Second, our approach also leads to a better utilization of heterogeneous resources available in FPGA— weights based on the two schemes can be managed by LUT and DSP resources. Specifically, the operations involving SP2 quantized weights should be implemented by LUTs; while those with fixed-point quantized weights can leverage the DSPs, the more limited resources on FPGA for DNN hardware accelerators. Overall, our MSQ achieves a sweet design spot achieving both high accuracy and processing throughput, thanks to the high and optimized utilization of both LUTs and DSPs.

B. Algorithm

In MSQ, each row in a weight matrix should employ either the SP2 or fixed-point scheme. To determine the scheme for each row, the weight variances of all the rows are calculated. We define a threshold θ for the variances, such that for the rows with smaller variances than the threshold, the SP2 quantization is employed; and otherwise, the fixed-point scheme is applied. By setting the proper threshold θ, the desired partition ratio of SP2 to fixed-point can be achieved with improved FPGA resource utilization. Algorithm 2 provides the details.

The optimal ratio of SP2 to fixed-point is determined by the available resources on FPGA devices and resource utilization required to support the design. Generally, the utilization factor of DSPs should be maintained at 100% to take full advantage of the DSP resource for the fixed-point multiplications. When only fixed-point quantization is applied, the LUT utilization is low even though DSP utilization reaches the maximum. Incorporating the SP2 quantization can increase the LUT utilization, and therefore enhancing the throughput. The exploration of the optimal ratio of SP2 to fixed-point among the weight matrix rows is elaborated in §VI.

C. Accuracy Results

1) Experiment Setup: We evaluate our MSQ in three application domains i.e., image classification with convolutional neural networks (CNNs); object detection and recognition with YOLO-v3; machine translation, speech recognition, and sentiment classification with recurrent neural networks (RNNs). We use no extra data augmentations in our quantization, other than those already employed for training the 32-bit floating-point baseline models. Our quantization training algorithm uses step or cosine learning rate decay and ℓ_2 regularization, following training algorithms of the baseline models. Our quantization algorithms are implemented with the PyTorch framework on NVIDIA TITAN RTX GPUs and GeForce RTX 2080Ti GPUs.

For image classification, we evaluate with the deep residual net (ResNet-18) [52], which is a widely used model for computer vision tasks, as well as the lightweight MobileNet-v2 model [53]. We test on CIFAR10 [54], CIFAR100 [54], and ImageNet ILSVRC-2012 [55] datasets. DNN models for CIFAR10 and CIFAR100 datasets are trained from scratch and quantized for 150 epochs. For ImageNet dataset, pre-trained models in 32-bit floating-point are used and quantized for 90 epochs. The initial learning rate is $8e-3$ for CIFAR10, $4e-3$ for CIFAR100, $5e-4$ for ImageNet.

For object detection, we explore the implementation of a fully convolutional neural network (FCNN) called YOLO-v3 [56] on MS COCO 2014 [57] dataset. The learning rate starts from $1e-2$, and decays to $5e-4$ with cosine annealing. We evaluate mean Average Precision (mAP) at an IoU threshold value of 0.5 (mAP@0.5), as well as average mAP over the IoU threshold range from 0.5 to 0.95 (mAP@$(0.5:0.95)$).

For RNNs, we evaluate three networks. The first one is an LSTM network with 256 hidden neurons in two layers [58] on Penn Tree Bank (PTB) [59] dataset for the machine translation application with perplexity (PPL) as the evaluation metric (lower PPL is better). The second is a network based on GRU with 1024 hidden neurons in two layers [60] on TIMIT acoustic-phonetic continuous speech corpus [61] dataset for the speech recognition application. The evaluation metric is Phoneme Error Rate (PER) and lower PER is better. Finally, we use another LSTM network with three hidden layers each having 512 neurons on IMDB [62] dataset for sentiment classification. Our learning rate is $1e-3$ for all the RNNs.

2) Result Analysis: Tables II, III, and IV summarize quantization results for the image classification. Table II compares different quantization schemes including power-of-2 (P2), fixed-point (Fixed), sum-of-power-of-2 (SP2), and our mixed scheme quantization (MSQ). Two partitioning ratios are tested for MSQ, the first one being $PR_{SP2:Fixed} = 1:1$, and the second one being $PR_{SP2:Fixed} = 2:1$ that is the optimal

TABLE III
COMPARISONS WITH EXISTING WORKS WITH RESNET-18 MODEL ON IMAGENET DATASET.

Methods	Bit-width (W/A)	Top-1 (%)	Top-5 (%)
Baseline(FP)	32/32	69.76	89.08
Dorefa [38]	4/4	68.10	88.10
PACT [39]	4/4	69.20	89.00
DSQ [40]	4/4	69.56	N/A
QIL [41]	4/4	70.10	N/A
μL2Q [42]	4/32	65.92	86.72
LQ-NETS [44]	4/4	69.30	88.80
MSQ	**4/4**	**70.27**	**89.42**

TABLE IV
COMPARISONS WITH EXISTING WORKS WITH MOBILENET-V2 MODEL ON IMAGENET DATASET.

Methods	Bit-width (W/A)	Top-1 (%)	Top-5 (%)
Baseline(FP)	32/32	71.88	90.29
PACT [39]	4/4	61.40	N/A
DSQ [40]	4/4	64.80	N/A
MSQ	**4/4**	**65.64**	**86.98**

ratio from FPGA characterizations. On Top-1 accuracy, MSQ has the minimum accuracy loss for most cases.

The accuracy increase of MSQ compared to sole SP2 or Fixed results from several aspects. First, combining SP2 and Fixed makes the quantized DNN weights fit the original weight distribution better. In addition, model compression could slightly increase accuracy when weight bit-width ≥ 4, as the quantization resolution is high enough so that the inference results of DNNs are not affected, and quantization noise can potentially act as regularization that benefits generalization and addresses overfitting.

Tables III, and IV compare our MSQ with existing DNN quantization works including Dorefa [38], PACT [39], DSQ [40], QIL [41], μL2Q [42], and LQ-NETS [44]. Those works and our MSQ start with the same pre-trained models with the same baseline accuracy. Here we use the optimal $PR_{SP2:Fixed} = 2:1$ in MSQ. Note that this optimal ratio is from hardware characterization, not for increasing accuracy. Table III shows that Dorefa, PACT, DSQ, μL2Q, and LQ-NETS have up to 3.84% accuracy degradation, only QIL reports lossless accuracy performance. Our MSQ increases accuracy by 0.49% compared with the floating-point model. Table IV shows that the lightweight model MobileNet-v2 is much harder to quantize with 4-bits (for both weight and activation), our MSQ achieves the highest accuracy of the quantized models.

On the even larger YOLO-v3 model for object detection, we apply 4-bit quantization, which is equivalent to $8\times$ compression rate. We test on two image sizes i.e., 320×320 and 640×640. Our MSQ performs very well in preserving the mAP values i.e., with negligible mAP degradation, and for the case of 640×640 input size and mAP@0.5, MSQ can even increase the mAP value. We notice a slightly higher mAP degradation when the input size is small. This is because the smaller feature maps are more sensitive to quantization error. There is no existing quantization methods reporting about YOLO network quantization. To provide an idea about the mAP degradation by our MSQ, we can compare with the weight pruning method on YOLO with also $8\times$ compression rate [64], which decreases the mAP@0.5 by ~ 3.0 on a simpler dataset than COCO 2014. In general, at the same compression rate, and especially when the dataset is simpler, weight pruning should have less accuracy degradation than weight quantization. But our MSQ can have comparable or even smaller mAP degradation. It demonstrates our MSQ works very well on YOLO networks.

TABLE V
YOLO-V3 ON COCO 2014 DATASET WITH 4-BIT QUANTIZATION. ($8\times$ COMPRESSION RATE)

Image Size	Scheme	mAP @0.5 : 0.95	mAP @0.5
320	Baseline(FP)	37.7	56.8
	MSQ	35.8	53.9
640	Baseline(FP)	45.6	64.7
	MSQ	44.1	64.8

TABLE VI
RNN ON MACHINE TRANSLATION, SPEECH RECOGNITION, AND SENTIMENT CLASSIFICATION.

Scheme	Bit Width (W/A)	Evaluation Metric	Result
LSTM on PTB			
EQMBaseline(FP)	32/32	Perplexity (PPL) lower better	109
EQM [63]	4/4	PPL	114
OurBaseline(FP)	32/32	PPL	110.89
Fixed	4/4	PPL	113.03
SP2	4/4	PPL	113.42
MSQ(half/half)	4/4	PPL	112.74
MSQ(optimal)	4/4	PPL	112.72
GRU on TIMIT			
OurBaseline(FP)	32/32	Phoneme Error Rate (PER) lower better	19.24%
Fixed	4/4	PER	20.14%
SP2	4/4	PER	20.09%
MSQ(half/half)	4/4	PER	19.58%
MSQ(optimal)	4/4	PER	19.53%
LSTM on IMDB			
EQMBaseline(FP)	32/32	Accuracy	89.54%
EQM [63]	4/4	Accuracy	88.47%
OurBaseline(FP)	32/32	Accuracy	86.37%
Fixed	4/4	Accuracy	86.12%
SP2	4/4	Accuracy	86.02%
MSQ(half/half)	4/4	Accuracy	86.28%
MSQ(optimal)	4/4	Accuracy	86.31%

Table VI shows that our MSQ scheme outperforms the Fixed and SP2 quantization for all the three RNN tasks. We also compare our method with existing work EQM [63] on the PTB and IMDB datasets. Because we do not have the same pre-trained models as in EQM [63], we also need to report on their pre-trained (32-bit floating-point) baseline models. On the PTB dataset, EQM [63] increases perplexity (PPL) by 5.00 (the lower the better), while our MSQ only increases by < 2.00.

For the IMDB dataset, EQM loses near 1% accuracy and MSQ only loses 0.06% accuracy. Note that we have not found any DNN quantization works investigating the TIMIT dataset, so we could not compare with existing works on TIMIT.

V. FPGA Implementation: Design and Optimization

Besides obtaining accuracy advantage, the proposed MSQ assembling the fixed-point and SP2 quantization schemes significantly promotes the efficiency of the FPGA deployment. Specifically, the newly joined SP2 quantization provides two apparent advantages in the hardware aspect: (i) the multiplication arithmetic involving the SP2 quantized weights can be implemented with simple logic shifter and adder, instead of the conventional multiplier; and (ii) since the FPGA underlying components include DSP and LUT, the rest LUTs can be leveraged for computations with SP2 weights while the DSPs are simulatenously fully utilized for conventional multiplication. Therefore, with the proposed MSQ as an *ensemble* of fixed-point and SP2, the same device can possibly deliver higher performance than existing designs, in which the throughput is theoretically bounded by the DSP count.

This section addresses the hardware design challenges with mixed number systems. Please note that the hardware benefit from SP2 is orthogonal to prior research efforts (e.g., dataflow [65] and locality [66] optimization), and therefore can be employed by *any* existing DNN accelerator.

A. FPGA Resource Characterization

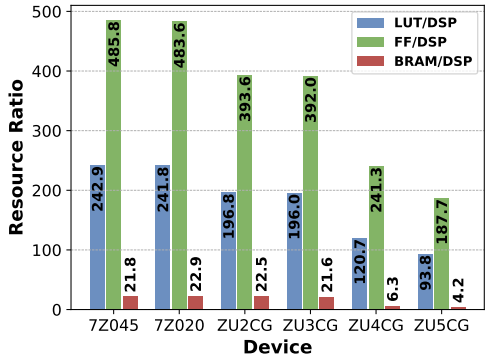

Fig. 2. **Resource ratio of different FPGA devices.** For each device, LUT, FF, and BRAM numbers are all normalized with respect to DSP number.

FPGA devices provide different types of resources, i.e., DSP, LUT, BRAM, and FF, for computation and storage, and the resource amount ratios vary in different FPGA devices. Figure 2 presents the resource ratios of Zynq series devices (each device name starts with "XC" that is omitted for simplicity), with each bar normalized by the DSP count on the corresponding device. The ratio of LUT to DSP attracts our attention, since this number directly decides the building block for multiplications with fixed-point and SP2 quantized weights, respectively. Apparently, the ratio of LUT/DSP in XC7Z045/XC7Z020 devices are larger than that in XCZU4CG/XCZU5CG devices. This also occurs in FPGA devices of other types. Specifically, since the multiplications with fixed-point and SP2 weights consume the DSP and LUT, respectively, the LUT/DSP ratio decides the parallel PE counts for these two operation types. For different devices, we select different proper ratios of PE counts for fixed-point and SP2 according to the available resource amount. Importantly, the PE ratio is used as the desired SP2/fixed-point ratio and sent to Algorithm 2 to obtain the properly quantized models with the novel MSQ scheme.

B. Architecture with Heterogeneous GEMM Engines

This section provides a design based on the versatile tensor accelerator (VTA) [67]. The hardware framework contains four modules as shown in Figure 3(a), where the `Instruction` module loads the instructions and provides control signals to other modules. `Load` and `Store` modules control the input/output activation and weight data communication between on-chip buffers and DRAM. The `Compute` module executes the workloads, with the `RegFile` as the scratchpad memory for partial sum accumulation and `TensorALU` computing the element-wise operations (e.g., activation). The major computation components are the general purpose matrix multiplication (GEMM) cores. Different from VTA, there are two heterogeneous `GEMM` cores, $\texttt{GEMM}_{\texttt{fixed}}$ for conventional multiplications, and $\texttt{GEMM}_{\texttt{sp2}}$ for SP2 operations. Besides conventional GEMM acceleration framework, our $\texttt{GEMM}_{\texttt{fixed}}$ can be naturally combined with advanced GEMM acceleration frameworks with architectural optimizations on the fixed-point operations (and uses DSP resources on FPGA). An example is Bit-Fusion [11], which is orthogonal and can be combined with our MSQ. Firstly, the fixed point operations executed on DSP in our MSQ framework can be accelerated by Bit-Fusion. Secondly, MSQ assigns a large portion (beyond 50%) of computations in each layer to SP2 and leverages LUTs for computation, which are previously not fully exploited by fixed-point acceleration techniques like Bit-Fusion. A doubling performance can be anticipated as fixed-point and SP2 are computed in parallel on FPGA.

The detailed workflow of two `GEMM` cores is illustrated in Figure 3(b). A tiled block of input activation data with a size of $Bat \times Blk_{in}$ is read from the input buffer to the register array, where Bat is the batch size and Blk_{in} is the input channel count of the tile that will be computed in parallel. Note that the input activation will be broadcasted to both `GEMM` cores. As Figure 3(c) displays, the $\texttt{GEMM}_{\texttt{fixed}}$ core is composed of multipliers implemented with DSPs on FPGA, while the $\texttt{GEMM}_{\texttt{sp2}}$ uses LUTs to realize shift and addition for the novel SP2 based computations. Meanwhile, two weight buffers provide the weight values in fixed-point and SP2 formats, respectively. The partial results will be accumulated and stored in individual register filers, and the final results are written to individual output buffers. Because the filters are allocated to heterogeneous `GEMM` cores depending on their weight representation format, two filter index buffers are set to instruct the `Store` unit to write the output data to the

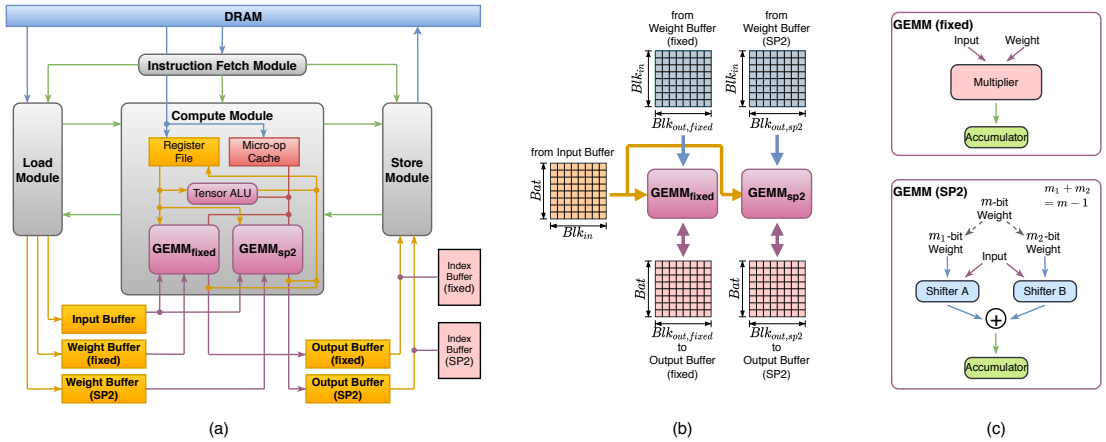

Fig. 3. **Hardware architecture of convolution for MSQ number system.** (a) Overall framework with $\texttt{GEMM}_{fixed}$ core for fixed-point operations and $\texttt{GEMM}_{fixed}$ core for SP2 operations; (b) Dataflow in heterogeneous GEMM cores; and (c) Computations in heterogeneous GEMM cores.

proper global addresses. Figure 3(c) gives a detailed structure to handle fixed-point and SP2 operations in two GEMM cores.

Two design parameters $Blk_{out,fixed}$ and $Blk_{out,sp2}$ indicate the parallel PE count in each GEMM core and size of corresponding registers array, as illustrated in Figure 3(b). Two factors are considered in selecting $Blk_{out,fixed}$ and $Blk_{out,sp2}$. One is that the ratio of $Blk_{out,fixed}$ to $Blk_{out,sp2}$ should be equivalent to that of different weight types (fixed-point/SP2). An imbalanced ratio may result in under-utilization of the certain GEMM core. The other is that the on-chip resources (DSP and LUT count) should yield a particular ratio of design parameters, i.e., a proper number facilitates fully utilization of FPGA resources, which is the key motivation of this work. Specifically, we develop an FPGA-centric MSQ quantization method as mentioned in §IV-B that automatically trains quantized DNN models to achieve a particular ratio that meets the resource allocation on FPGA devices. Additionally, we incorporate the processing operations after the convolution computations into the GEMM cores, including batch normalization, activation (ReLU) and pooling, as these operations consume few resources and incur negligible latency increase compared with convolution computations.

VI. Evaluation

A. Experiment Setup

To demonstrate the improvement of the proposed SP2 (and MSQ) scheme in the hardware aspect, we implemented the architecture with heterogeneous GEMM cores on the embedded FPGA device, in which a high efficiency is usually in demand under resource limitation. As Table VII shows, we implemented the architecture on the Zynq XC7Z020 and XC7Z045 devices with different design parameters that result in different throughput and resource utilization results. Note that for each device, we set up different ratios between the PE array sizes of the $\texttt{GEMM}_{\texttt{fixed}}$ and $\texttt{GEMM}_{\texttt{sp2}}$ cores. Specifically, we progressively increase the size of $\texttt{GEMM}_{\texttt{sp2}}$ core ($Blk_{out,sp2}$), till the LUT utilization reaches 70%. For example, on XC7Z020 the desired fixed/SP2 ratio is 1:1.5 and on XC7Z045 it is 1:2. For all implementations, the quantization bit-width for the CNN models is fixed to 4, and the working frequency is set to 100MHz.

TABLE VII
Hardware implementation parameters with different devices and settings. Bat, Blk_{in}, and $Blk_{out,fixed}$ are set such that the DSP utilization could reach maximum. $Blk_{out,sp2}$ is increased until the LUT utilization is high enough and optimized.

Impl.	Device	Bat	Blk_{in}	Blk_{out} Fixed	Blk_{out} SP2	Ratio (fixed/SP2)	Peak Thrpt. (GOPS)
D1-1	XC7Z020	1	16	16	0	1:0	52.8
D1-2		1	16	16	16	1:1	106
D1-3		1	16	16	24	1:1.5	132
D2-1	XC7Z045	4	16	16	0	1:0	208
D2-2		4	16	16	16	1:1	416
D2-3		4	16	16	32	1:2	624

B. Evaluation with FPGA

1) Resource Utilization: Figure 4 presents the resource utilization with different implementations. Apparently, with the size increase of $\texttt{GEMM}_{\texttt{sp2}}$, more on-chip LUT can be leveraged for a better peak throughput (GOPS). As Table VII shows, on XC7Z020 device (D1-1,2,3), the peak throughput was improved to 2.5× (from 52.8 to 132 GOPS) with the extra $\texttt{GEMM}_{\texttt{sp2}}$ core. The maximum size of the PE array for SP2 is 1.5× of that for fixed point. This peak throughput improvement is 3× on XC7Z045, from 208 to 624 GOPS. Although the ratio of available LUT/DSP is the same between the two devices, the optimal proportion of PE count for SP2 on XC7Z020 (1.5× of fixed-point) is a smaller than that on XC7Z045 (2× of fixed-point). This is because a portion of LUTs is consumed by Load and Store modules to accommodate the $\texttt{GEMM}_{\texttt{sp2}}$ core. The proposed architecture design is general for all devices through adjusting the $Blk_{out,sp2}$ to fully utilize the LUT resource and quantizing the models with the corresponding fixed-point/SP2 ratio.

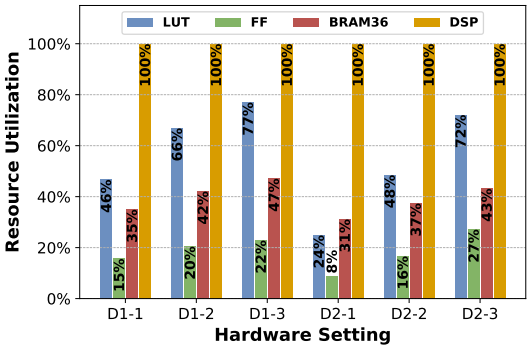

Fig. 4. **FPGA resource utilization with different devices and settings.** In the three designs for each of the two devices, the DSP utilization is maintained at 100% and the LUT utilization is raised to 70%−80% with FF and BRAM resources.

2) Real-world Performance and Comparison: To present the performance with real-world applications, we employed different CNN and RNN models with the proper SP2/fixed-point ratios on the two devices. The networks ResNet-18 and MobileNet-v2 are implemented based on the ImageNet dataset. The performance results of each network under various hardware configurations are displayed in Table VIII. For some layers in CNNs like the first convolutional layer, the peak throughput cannot be reached since the number of input channels is less than Blk_{in} so that the data cannot fill all of the PEs. Generally, for CNN models, the overall PE utilization reaches 52.4% to 70.1%, and the heterogeneous $\texttt{GEMM}_{\texttt{fixed}}$ and $\texttt{GEMM}_{\texttt{sp2}}$ cores improve the throughput by $2.1\times - 2.5\times$ with the optimal design compared to utilizing the $\texttt{GEMM}_{\texttt{fixed}}$ core only. Compared with the design with only 4-bit fixed-point (fixed4/SP2 $= 1:0$) quantization, the optimal design with the ratio of fixed4/SP2 $= 1:1.5$ on XC7Z020 decreases the latency per image from 100.7ms to 47.1ms ($2.13\times$) for ResNet-18, and the optimal design with the ratio of fixed4/SP2 $= 1:2$ on XC7Z045 decreases the latency from 25.1ms to 10.1ms ($2.49\times$) for ResNet-18. The latency improvement is more significant when compared with the 8-bit fixed-point design, as the optimal design on XC7Z020 achieves latency decrease from 181.3ms to 47.1ms ($3.83\times$), and the optimal design on XC7Z045 achieves latency decrease from 45.2ms to 10.1ms ($4.48\times$). As for RNN models, the PE utilization is $42.9\% - 59.2\%$, and the performance is increased by $2.4\times - 4.1\times$.

The optimal MSQ implementations of CNNs based on ImageNet and previous designs are compared in Table IX, from which it can be observed that our ResNet-18 implementations achieve the highest accuracy and enjoy comparable hardware utilization efficiency represented by GOPS/DSP and GOPS/kLUT with designs in [68], [69]. The work [70] acquires higher utilization efficiency but much lower accuracy. MobileNet-v2 has the most complicated structure among all these networks, making it difficult to deploy on hardware platforms, but our designs can still achieve high performance, especially in terms of frame rate. We do not find implementations with ResNet-18 and MobileNet-v2 in other work, so we compare it with other CNNs.

Our proposed solution is beneficial over low-precision GPU for the following two reasons: (1) Current low-precision GPU (Tensor-RT solution) relies on 8-bit, while we can go to 4-bit and further assisted by SP2; (2) FPGA solution is dataflow-based and energy-efficient in general [71]. Comparing with a state-of-art energy-efficient GPU (NVIDIA Jetson AGX, power consumption 10-15W) with Tensor-RT support, we use ResNet-18 as example, measured under the same accuracy. Our FPGA solution (XC7Z045) is slightly higher performant (99FPS vs. 78FPS), but more than $3\times$ higher energy efficiency as the FPGA only consumes around 4W power.

VII. Related Work

This section introduces the DNN weight quantization methods/algorithms for fixed-point and P2 quantization schemes, and discusses DNN weight quantization on FPGA platforms.

A. DNN Quantization Methods

Zhou et al. [38] first explored the potential of fixed-point quantization by introducing hyperbolic tangent transformation to weights and activations, with scaling factors to minimize quantization error. Choi et al. [39] improved this method by adding a parameterized clipping threshold to activations. As alternatives for solving the non-differentiable problem, DSQ [40] developed an evolving training method to gradually approximate STE. QIL [41] parameterized the quantization interval and trained it with task loss, avoiding access to the original training data. μL2Q [42] introduced data distribution loss during training to minimize quantization error. LQ-Net [44] and LSQ [43] proposed a differentiable method to learn the quantizer for each layer jointly with parameters. Miyashita et al. [45] replaced fixed-point quantizer with logarithmic representation to exploit bit shift operations to accelerate inference. INQ [46] splits weights into groups and iteratively quantize the model to low bit-width. Leng et al. [47] employed ADMM training technique to increase the accuracy of extremely low bit-width DNNs.

In addition to these quantization methods for inference acceleration, Zhu et al. [72] proposed a low-bit training framework for the training acceleration. They used the direction sensitive gradient clipping and the deviation counteractive learning rate scaling to ensure a unified 8-bit (INT8) training with minor accuracy degradation.

B. Weight Quantization in FPGA Implementations

Weight quantization has been widely applied to DNN implementations on FPGAs [73]. Some work studies fixed-point quantization. The work [68] utilizes greedy solution to determine the radix position of each layer for quantization. [70] investigates a hybrid quantization scheme that allows different bit-widths for weights, providing more flexibility. For Binarized Neural Networks (BNNs), multiplications can be executed with XNOR gates [74]–[76]. A fully binarized neural

TABLE VIII
PERFORMANCE OF VARIOUS DNN APPLICATIONS ON HARDWARE UNDER DIFFERENT SETTINGS.

Device	Ratio (fixed/SP2)	Utilization				Throughput (GOPS)					
		LUT	DSP	BRAM36	FF	ResNet-18 on ImageNet	MobileNet-v2 on ImageNet	YOLO-v3 on COCO	LSTM on PTB	GRU on TIMIT	LSTM on IMDB
XC7Z020	1:0	12160	220	39	9403	36.0	33.0	36.6	26.1	22.6	25.0
	1:1	22912	220	49	14523	74.4	65.7	74.1	52.9	49.2	58.7
	1:1.5 (opt.)	28288	220	56	17083	77.0	71.8	84.0	77.2	77.2	59.7
XC7Z045	1:0	41830	900	160	31293	144.7	129.6	143.6	91.3	89.6	108.0
	1:1	93440	900	194	65699	285.5	258.1	283.7	183.2	212.5	217.2
	1:2 (opt.)	145049	900	225.5	111575	359.2	326.9	390.0	318.2	369.2	340.7

TABLE IX
COMPARISONS OF CNNS ON IMAGENET WITH PREVIOUS IMPLEMENTATIONS.

Implementation	VGG [68]			AlexNet [70]	DiracDeltaNet [69]	ResNet-18 (Our opt.)		MobileNet-v2 (Our opt.)	
Device	XC7Z045	XC7Z045	XC7Z020	XC7Z045	XCZU3EG	XC7Z020	XC7Z045	XC7Z020	XC7Z045
Bit-width (W/A)	16/16	8/8	8/8	8/8	1/4	4/4		4/4	
Top-1 Accuracy	67.84%	67.72%	67.62%	54.6%	68.5%	70.27%		65.64%	
Frequency (MHz)	150	150	214	200	250	100		100	
LUT	182616	139385	29867	86262	24130	28288	145049	28288	145049
DSP	780	900	190	808	37	220	900	220	900
BRAM36	486	390.5	85.5	303	170	56	225.5	56	225.5
Throughput (GOPS)	187.8	292	84.3	493	47.09	77.0	359.2	71.8	326.9
Frame Rate (FPS)	6.06	9.42	2.72	340	96.5	21.3	99.1	120.7	549.3
GOPS/DSP	0.241	0.324	0.444	0.610	1.273	0.350	0.391	0.326	0.363
GOPS/kLUT	1.029	2.096	2.825	5.747	1.953	2.725	2.475	2.538	2.252

network accelerator is implemented in [76] through utilizing odd-even padding to replace the zero padding values. Another scheme called logarithmic quantization using power of 2 is explored in [77]. In addition, weight quantization could be employed with a two-stage arithmetic unit for low bit-width CNNs [78], a fast matrix and Winograd algorithm [79], a novel CNN architecture for software-hardware codesign [69], a design flow of DNN implementations for more flexible quantization schemes [80], and an OpenCL-based framework Deep Learning Accelerator (DLA) to accomodate designs with different bit-widths [81]. In addition, dynamic quantization with bit fusion in [11] improves the bit-level flexibility by matching various bit-widths for different DNN layers.

VIII. CONCLUSION

This paper investigates efficient DNN inference engine on FPGA devices through DNN quantization, and proposes the *first* solution that applies different quantization schemes for different rows of the weight matrix. We propose a hardware-friendly quantization scheme named *SP2* suitable for Gaussian-like weight distribution, in which the multiplication arithmetic can be replaced with logic shifter and adder, thereby enabling highly efficient implementations with the FPGA LUT resources. In contrast, the fixed-point quantization is suitable for Uniform-like weight distribution and can be implemented efficiently by DSP. To fully explore the FPGA resources, intra-layer, multi-scheme quantization framework with an ensemble of the SP2 and fixed-point schemes. We evaluate our FPGA-centric quantization framework across multiple application domains with various DNNs such as convolutional neural networks (CNN) and recurrent neural networks (RNN). With optimal SP2/fixed-point ratios on two FPGA devices, i.e., Zynq XC7Z020 and XC7Z045, we achieve performance improvement of $2.1\times - 4.1\times$ compared to solely exploiting DSPs for all multiplication operations.

ACKNOWLEDGMENT

This work is partly supported by the National Science Foundation CCF-1901378, CCF-1919117, CCF-1919289, CNS-1909172 and DARPA-HR00112090055.

REFERENCES

[1] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence (AAAI)*, 2017.

[2] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, pp. 2117–2125.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[4] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5934–5938.

[5] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th international conference on Machine learning (ICML)*, 2008, pp. 160–167.

[6] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez, "A survey on deep learning in medical image analysis," *Medical image analysis*, vol. 42, pp. 60–88, 2017.

[7] J. De Fauw, J. R. Ledsam, B. Romera-Paredes, S. Nikolov, N. Tomasev, S. Blackwell, H. Askham, X. Glorot, B. O'Donoghue, D. Visentin, P. A. Keane, and O. Ronneberger, "Clinically applicable deep learning for diagnosis and referral in retinal disease," *Nature medicine*, vol. 24, no. 9, pp. 1342–1350, 2018.

[8] H. Mao, M. Song, T. Li, Y. Dai, and J. Shu, "Lergan: A zero-free, low data movement and pim-based gan architecture," in *Proceedings of the 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018, pp. 669–681.

[9] A. Ren, Z. Li, C. Ding, Q. Qiu, Y. Wang, J. Li, X. Qian, and B. Yuan, "Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing," *Proceedings of the 22nd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, vol. 51, no. 2, pp. 405–418, 2017.

[10] R. Cai, A. Ren, N. Liu, C. Ding, L. Wang, X. Qian, M. Pedram, and Y. Wang, "Vibnn: Hardware acceleration of bayesian neural networks," in *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, 2018, pp. 476–488.

[11] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks," in *Proceedings of the 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE Press, 2018, pp. 764–775.

[12] C. Deng, F. Sun, X. Qian, J. Lin, Z. Wang, and B. Yuan, "Tie: energy-efficient tensor train-based inference engine for deep neural network," in *Proceedings of the 46th Annual International Symposium on Computer Architecture (ISCA)*, 2019, pp. 264–278.

[13] R. Cai, A. Ren, O. Chen, N. Liu, C. Ding, X. Qian, J. Han, W. Luo, N. Yoshikawa, and Y. Wang, "A stochastic-computing based deep learning framework using adiabatic quantum-flux-parametron superconducting technology," in *Proceedings of the 46th Annual International Symposium on Computer Architecture (ISCA)*, 2019, pp. 567–578.

[14] A. Ren, T. Zhang, S. Ye, J. Li, W. Xu, X. Qian, X. Lin, and Y. Wang, "Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers," in *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2019, pp. 925–938.

[15] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2015, pp. 161–170.

[16] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable fpgas," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2017, pp. 15–24.

[17] Z. Chen, A. Howe, H. T. Blair, and J. Cong, "Fpga-based lstm acceleration for real-time eeg signal processing," in *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2018, pp. 288–288.

[18] R. Shi, Y. Ding, X. Wei, H. Liu, H. So, and C. Ding, "Ftdl: An fpga-tailored architecture for deep learning systems," in *The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2020, pp. 320–320.

[19] W. Niu, X. Ma, S. Lin, S. Wang, X. Qian, X. Lin, Y. Wang, and B. Ren, "Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning," in *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020, pp. 907–922.

[20] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze, C. Guestrin, and A. Krishnamurthy, "Tvm: An automated end-to-end optimizing compiler for deep learning," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 578–594.

[21] https://github.com/alibaba/MNN.

[22] A. Paszke, S. Gross, S. Chintala, and G. Chanan, "Pytorch," 2017.

[23] https://www.tensorflow.org/mobile/tflite/.

[24] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *International Conference on Learning Representations (ICLR)*, 2019.

[25] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems (NeurIPS)*, 2016, pp. 2074–2082.

[26] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances in neural information processing systems (NeurIPS)*, 2016, pp. 1379–1387.

[27] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 875–886.

[28] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9194–9203.

[29] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4340–4349.

[30] X. Dong and Y. Yang, "Network pruning via transformable architecture search," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 759–770.

[31] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems (NeurIPS)*, 2015, pp. 3123–3131.

[32] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[33] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision (ECCV)*. Springer, 2016, pp. 525–542.

[34] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 345–353.

[35] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[36] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *International Conference on Learning Representations (ICLR)*, 2017.

[37] Z. He and D. Fan, "Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 438–11 446.

[38] 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.

[39] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[40] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 4852–4861.

[41] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4350–4359.

[42] G. Cheng, L. Ye, L. Tao, Z. Xiaofan, H. Cong, C. Deming, and C. Yao, "μl2q: An ultra-low loss quantization method for dnn," *The 2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.

[43] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *International Conference on Learning Representations (ICLR)*, 2019.

[44] D. Zhang, J. Yang, D. Ye, and G. Hua, "Lq-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 365–382.

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

[46] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," in *International Conference on Learning Representations (ICLR)*, 2017.

[47] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with admm," in *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[48] C. Baskin, E. Schwartz, E. Zheltonozhskii, N. Liss, R. Giryes, A. M. Bronstein, and A. Mendelson, "Uniq: Uniform noise injection for non-uniform quantization of neural networks," *arXiv preprint arXiv:1804.10969*, 2018.

[49] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML)*, 2015, pp. 1613–1622.

[50] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[51] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," in *International Conference on Learning Representations (ICLR)*, 2018.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

[53] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2018, pp. 4510–4520.

[54] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[56] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[57] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[58] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[59] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," 1993.

[60] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[61] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[62] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics, 2011, pp. 142–150.

[63] Q. He, H. Wen, S. Zhou, Y. Wu, C. Yao, X. Zhou, and Y. Zou, "Effective quantization methods for recurrent neural networks," *arXiv preprint arXiv:1611.10176*, 2016.

[64] P. Zhang, Y. Zhong, and X. Li, "Slimyolov3: Narrower, faster and better for real-time UAV applications," *CoRR*, vol. abs/1907.11093, 2019. [Online]. Available: http://arxiv.org/abs/1907.11093

[65] Q. Sun, T. Chen, J. Miao, and B. Yu, "Power-driven dnn dataflow optimization on fpga," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–7.

[66] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, and J. Cong, "Fp-dnn: An automated framework for mapping deep neural networks onto fpgas with rtl-hls hybrid templates," in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2017, pp. 152–159.

[67] T. Moreau, T. Chen, L. Vega, J. Roesch, E. Yan, L. Zheng, J. Fromm, Z. Jiang, L. Ceze, C. Guestrin, and A. Krishnamurthy, "A hardware–software blueprint for flexible deep learning specialization," *IEEE Micro*, vol. 39, no. 5, pp. 8–16, 2019.

[68] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto embedded fpga," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 37, no. 1, pp. 35–47, 2017.

[69] Y. Yang, Q. Huang, B. Wu, T. Zhang, L. Ma, G. Gambardella, M. Blott, L. Lavagno, K. Vissers, J. Wawrzynek, and K. Keutzer, "Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded fpgas," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2019, pp. 23–32.

[70] J. Wang, Q. Lou, X. Zhang, C. Zhu, Y. Lin, and D. Chen, "Design flow of accelerating hybrid extremely low bit-width neural network in embedded fpga," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 163–1636.

[71] J. Cong, Z. Fang, M. Lo, H. Wang, J. Xu, and S. Zhang, "Understanding performance differences of fpgas and gpus: (abtract only)," ser. FPGA '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 288. [Online]. Available: https://doi.org/10.1145/3174243.3174970

[72] F. Zhu, R. Gong, F. Yu, X. Liu, Y. Wang, Z. Li, X. Yang, and J. Yan, "Towards unified int8 training for convolutional neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1969–1979.

[73] K. Guo, W. Li, K. Zhong, Z. Zhu, S. Zeng, S. Han, Y. Xie, P. Debacker, M. Verhelst, and Y. Wang, "Neural network accelerator comparison," https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/.

[74] H. Nakahara, H. Yonekawa, T. Sasao, H. Iwamoto, and M. Motomura, "A memory-based realization of a binarized deep convolutional neural network," in *2016 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2016, pp. 277–280.

[75] H. Nakahara, T. Fujii, and S. Sato, "A fully connected layer elimination for a binarized convolutional neural network on an fpga," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–4.

[76] P. Guo, H. Ma, R. Chen, P. Li, S. Xie, and D. Wang, "Fbna: A fully binarized neural network accelerator," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 51–513.

[77] C. Luo, W. Cao, L. Wang, and P. H. Leong, "Rna: An accurate residual network accelerator for quantized and reconstructed deep neural networks," *IEICE Transactions on Information and Systems*, vol. 102, no. 5, pp. 1037–1045, 2019.

[78] L. Jiao, C. Luo, W. Cao, X. Zhou, and L. Wang, "Accelerating low bit-width convolutional neural networks with embedded fpga," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–4.

[79] D. Wu, J. Chen, W. Cao, and L. Wang, "A novel low-communication energy-efficient reconfigurable cnn acceleration architecture," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 64–643.

[80] X. Zhang, J. Wang, C. Zhu, Y. Lin, J. Xiong, W.-m. Hwu, and D. Chen, "Dnnbuilder: an automated tool for building high-performance dnn hardware accelerators for fpgas," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2018, pp. 1–8.

[81] P. Colangelo, N. Nasiri, E. Nurvitadhi, A. Mishra, M. Margala, and K. Nealis, "Exploration of low numeric precision deep learning inference using intel® fpgas," in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2018, pp. 73–80.