IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 February 2024, accepted 17 February 2024, date of publication 26 February 2024, date of current version 18 March 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3369889

RESEARCH ARTICLE

CANET: Quantized Neural Network Inference With 8-bit Carry-Aware Accumulator

JINGXUAN YANG, XIAOQIN WANG, AND YIYING JIANG
Institute of Microelectronics of the Chinese Academy of Sciences, Chaoyang, Beijing 100029, China
School of Integrated Circuits, University of Chinese Academy of Sciences, Huairou, Beijing 101408, China
Corresponding author: Xiaoqin Wang (wangxiaoqin@ime.ac.cn)

ABSTRACT Neural network quantization represents weights and activations with few bits, greatly reducing the overhead of multiplications. However, due to the recursive accumulation operations, high-precision accumulators are still required in multiply-accumulate (MAC) units to avoid overflow, incurring significant computational overhead. This constraint limits the efficient deployment of quantized NNs on resource-constrained platforms. To address this problem, we present a novel framework named CANET, which adapts the 8-bit quantized model to execute MAC operations with 8-bit accumulators. CANET not only employs 8-bit carry-aware accumulators to represent overflow data correctly, but also adaptively learns the optimal format per layer to minimize truncation errors. Meanwhile, a weight-oriented reordering method is developed to reduce the transfer length of the carry. CANET is evaluated on three networks in the ImageNet classification task, where comparable performance with state-of-the-art methods is realized. Finally, we implement the proposed architecture on a custom hardware platform, demonstrating a reduction of 40% in power and 49% in area compared with the MAC unit with 32-bit accumulators.

INDEX TERMS Convolutional neural network, quantization, efficient inference, low-precision accumulator.

I. INTRODUCTION

Neural network quantization facilitates the efficient deployment of deep neural networks (DNNs) on resource-constrained platforms, such as drones and Internet-of-Things (IoT) devices [1], [2]. During inference, MAC operations consisting of multiplications and accumulations dominate the arithmetic cost. Quantization successfully maps floating-point parameters to the low-precision fixed-point numbers, drastically reducing the cost of multiplications [3]. However, high-precision accumulators are necessary for deep feature computation, which requires the accumulation of thousands of products. The resulting computational overhead greatly limits the efficacy of quantization and prompts active exploration of inference based on low-precision accumulators [4].

Overflow is problematic when high-precision accumulators are replaced with low-precision counterparts. Because of overflow, out-of-range values are wrapped around to the minimum representable value, resulting in severe data distortion [6]. Some impressive methods were proposed to reduce or eliminate the risk of overflow associated with low-precision accumulators [4], [6], [13], [29], [30]. Xie et al. [4] reduced the risk of overflow in low-precision accumulators by adaptively scaling weights and activations bit widths. Ni et al. [6] mitigated the impact of overflow on accuracy by cyclic activation layers. De Bruin et al. [13] employed a heuristic method to determine the appropriate parameter precision per layer for fixed-precision accumulators. However, frequent overflow imposes tight constraints on the parameter bit width, impairing the model accuracy in such methods.

This paper introduces a novel framework named CANET, which aims to employ 8-bit accumulators within the 8-bit quantization framework without compressing the bit width of weights and activations. CANET incorporates a carry-aware accumulator (C2A) with a fixed-point format to rectify wraparound data. The error introduced by C2A is quantized to truncation and swamping error. Additionally, an adaptive algorithm is designed for learning optimal fixed-point formats to minimize truncation error. Meanwhile,

The associate editor coordinating the review of this manuscript and approving it for publication was Wenming Cao.

a weight-oriented accumulation order is adopted to reduce the transfer length of the carry and mitigate the swamping error. Furthermore, an integer-only method is presented to eliminate other high-precision operations. CANET is trained and validated across three neural networks on ImageNet datasets, demonstrating comparable performance with state-of-the-art quantization methods. Finally, we evaluate the effectiveness of CANET in terms of power and area on a 55nm CMOS custom hardware platform.

In summary, the main advantages of this work are as follows:

- As a novel quantization method, CANET uses the carry-aware algorithm to avoid overflow, enabling efficient inference using 8-bit accumulators in an 8-bit quantization framework.
- Based on the error analysis, CANET utilizes an adaptive format learning algorithm and a weight-oriented accumulation order to improve the accuracy of low-precision accumulation.
- Experiments conducted on three networks for ImageNet datasets demonstrate that CANET achieves comparable performance to state-of-the-art quantization methods while using 8-bit accumulators.
- We implement a MAC unit with the proposed architecture on a 55nm CMOS custom hardware platform and demonstrate the efficacy of CANET in terms of power and area.

The rest of the paper is organized as follows. Section II reviews and analyzes the related works. In Section III, the background of quantization and accumulator are briefly introduced. A detailed demonstration of CANET is presented in Section IV. Experimental results are demonstrated in Section V. Finally, Section VI presents the conclusion of this paper.

II. RELATED WORK

A. NETWORK QUANTIZATION

Quantization has been widely used in network compression [31], [32] with two types of quantizers: uniform and nonuniform. The uniform quantizer linearly maps floating-point numbers to fixed-point domains through a fixed quantization resolution. The nonuniform quantizer allocates more quantization levels to critical value regions, leading to a nonlinear projection [28]. Although nonuniform quantization usually achieves better performance than the uniform strategy, it incurs extra hardware overhead, such as look-up tables (LUT), owing to the complicated projection operations. On the other hand, uniform quantization methods are explored for efficient inference in the following directions.

Some researchers focused on extreme quantization, in which only binary or ternary weights and activations are involved [22], [23], [27]. These methods used bit-shift logic instead of high-precision multiplications to achieve a significant acceleration but often result in substantial performance degradation. Another type of effort used integer-only quantization to accomplish inference by integer arithmetic operations [9], [14], [15], [16], [17]. In integer-only methods, the full-precision multiplications caused by scale factor were emulated to *int32* multiplications [15], *int8* multiplications [9], or bit-shift logic [14]. However, more effort is required to eliminate other high-precision operations, such as 32-bit accumulations, for a lower arithmetic cost.

B. LOW-PRECISION ACCUMULATOR

Several methods are proposed to use low-precision accumulators in network training or inference. Some of the research [4], [6], [13], [30] sought to reduce or avoid the risk of overflow in low-precision accumulators by limiting the range of parameters. However, the limited parameters range impairs model performance. On the other hand, variable bit widths may bring about inefficient utilization of hardware resources (e.g., 4-bit weights still need 8-bit MAC units in the 8-bit inference framework). Another line of work [5], [8] accelerated network training and inference by low-precision floating-point accumulators, which are challenging to deploy on resource-constrained devices.

In summary, the feasibility and gains of the low-precision accumulator have been extensively demonstrated. However, few methods explored the low-precision-accumulator-based inference without constraining the range of parameters, leading to a limited design space.

III. BACKGROUND

A. QUANTIZED MODEL INFERENCE

In fixed-precision quantization, features transfer through layers with a constant bit width via quantization and dequantization. Quantization maps the floating-point values to integers, as shown in (1). In contrast, dequantization recovers real values from quantization using (2).

$$quantization\,(x, s, n, p) = clip(\left\lfloor \frac{x}{S} \right\rceil + Z, n, p). \quad (1)$$

$$dequantization\,(x, s, z) = s \cdot (x - Z). \quad (2)$$

where x represents the input feature map, during quantization, x corresponds to the floating-point features and x denotes the quantized integers in dequantization. The scale factor, denoted by s, determines the quantization resolution. Z represents the zero point. n and p are the positive and negative boundaries of the quantized integers. For the b-bit unsigned quantizer, $n = 0$, $p = 2^{b-1}$, and for the signed quantizer, $n = -2^{b-1}$, $p = 2^{b-1} - 1$.Furthermore, $\lfloor \cdot \rceil$ rounds scaled values to the nearest integers, and *clip* limits the quantized values to the range of $[n, p]$.

This paper focuses on 8-bit integer-only quantization to achieve a better trade-off between accuracy and acceleration by fixing weights and activations at 8-bit. In a convolution layer with weight tensor $\mathcal{W} \in \mathbb{R}^{C_o \times C_i \times k \times k}$, where C_o, C_i are the output and input channels, and k denotes the kernel size. During inference, MAC operations involving N times ($N = C_i \times k^2$) multiplications and additions, need accumulators

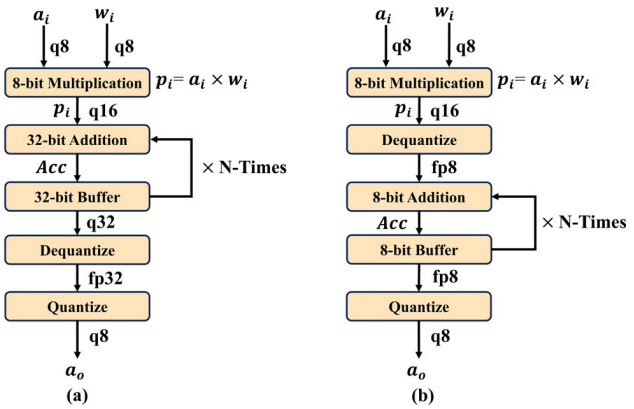

FIGURE 1. (a) Quantized NN inference with 32-bit accumulators. (b) NN Inference with 8-bit accumulators.

with a bit width of $(16 + \log_2 N)$ to prevent overflow. Thus, given the layer shapes in typical network architectures, such as $\mathcal{W} \in \mathbb{R}^{512\times512\times3\times3}$, 32-bit accumulators are necessary.

However, the robustness of the network to reduced arithmetic precision facilitates the deployment of low-precision accumulators [5], [8]. Recalling the specific process of inference, as illustrated in Figure 1 (a), the 16-bit partial sum p_i resulting from the multiplication of 8-bit quantized (q8) weights and activations is accumulated to a 32-bit intermediate result, denoted by Acc. Acc is then converted to the fixed-point (fp) domain by dequantization and subsequently quantized to q8 output a_o. The arithmetic operations are written as:

$$a_o - z_o = clip\left(\left\lfloor \frac{\left(s_a \cdot s_w \cdot \sum_i^N ((a_i - z_a) \cdot w_i\right) + b_i}{s_o} \right\rceil, n, p\right), \tag{3}$$

where, s_a, s_w, and s_o are the scale factors for a_i, w_i, a_o. Bias is denoted by b_i.

Assuming *relu* is adopted as the nonlinear activation function and p_i ($p_i = a_i w_i$) represents the partial sum, (3) can be simplified as:

$$a_o = clip\left(\left\lfloor \frac{\left(s_a \cdot s_w \cdot \sum_i^N p_i\right) + b}{s_o} \right\rceil, n, p\right). \tag{4}$$

The MAC operations $\sum_i^N p_i$ yield high-precision intermediate results, which are quantized to 8-bit a_o by S ($S = s_a \cdot s_w / s_o$). The quantizer preserves only eight bits of the intermediate results as quantized output a_o, leading to a partial waste of computational and storage resources. This motivates us to execute the accumulation process in 8-bit accumulators to minimize the bit width redundancy. A scheme is depicted in Figure 1 (b), where the dequantization maps a 16-bit p_i to an 8-bit fixed-point before accumulation. The corresponding arithmetic operations are as follows:

$$a_o = clip\left(\left\lfloor \frac{\sum_i^N (s_a \cdot s_w \cdot p_i) + b}{s_o} \right\rceil, n, p\right). \tag{5}$$

However, $\sum_i^N (s_a \cdot s_w \cdot p_i)$ still leads to high dynamic range intermediate results, which are prone to incur the overflow of 8-bit accumulators. The overflow results are affected by modulo operations, which discard the overflow bits and wrap the remaining bits to the minimum representable value, resulting in severe distortion [6]. Therefore, correctly expressing the overflow value is crucial for inferences with low-precision accumulators.

IV. METHODS

In this section, we provide a detailed description of CANET, comprising three main components: (1) a carry-aware accumulator with fixed-point formats to rectify wraparound data, (2) an adaptive learning algorithm for optimal fixed-point formats, and (3) a weight-oriented accumulation order to reduce the transfer length of the carry.

A. CHUNK-BASED CARRY-AWARE ACCUMULATION WITH FIXED-POINT FORMAT

In carry-aware accumulation, a fixed-point format (IL, FL) is employed to represent the accumulated data Acc. Specifically, an 8-bit fixed-point accumulated value consists of a 1-bit sign, followed by an n-bit integer length (IL), and the remaining (7)-n)-bit for the fractional length (FL). The arithmetic operations involved in the accumulation chain are depicted as:

$$Acc+ = s_a s_w \sum_i^G p_i, \tag{6}$$

where G represents the chunk size, corresponding to the parallelism of the Processing Element (PE) in hardware. $\sum_i^G p_i$ denotes the high-precision partial sums.

Consider the initial format of accumulators as (IL, FL). Before accumulation, $P_j (P_j = s_a \times s_w \times \sum_i^G p_i)$ is mapped to an 8-bit fixed-point number via (IL, FL), introducing a truncation error. IL and FL are dynamically updated based on the value of c, which is the number of carries in the accumulation chain. During accumulation, if Acc exceeds the maximum representable value of the current format, c increases by one. Subsequently, the format (IL, FL) is updated to (IL+c, FL−c), partially swamping the fractional part of P_j and Acc. When $c > FL$, Acc surpasses the upper bound and saturates at the maximum representable value. The proposed accumulation process is illustrated in Figure 2. In an accumulation chain of length N, the full-precision accumulated results S_N is given by:

$$S_N = S_{N-1} + P_{N-1} = P_1 + P_2 + \ldots + P_N. \tag{7}$$

We consider a case in which Acc is positive. When $Acc \leq 2^{7-FL} - 2^{-FL}$, $c = 0$, $P_j = P'_j + \Delta P_j$, the error is the summation of ΔP_j. Where P'_j is the 8-bit fixed-point partial sum, and the truncation error, denoted by ΔP_j, is indicated by the red dashed line in Figure 2. When $Acc \in \hat{\mathrm{E}}(2^{7-\mathrm{FL}} - 2^{-FL}, 2^7 - 1]$, and $c \in \hat{\mathrm{E}}(0, FL]$, the swamping error ΔP_j^c is introduced, as labeled by the gray dashed box in Figure 2. The swamping error ΔP_j^c increases exponentially with the value of c. When $c > FL$, Acc saturates at the $2^7 - 1$.

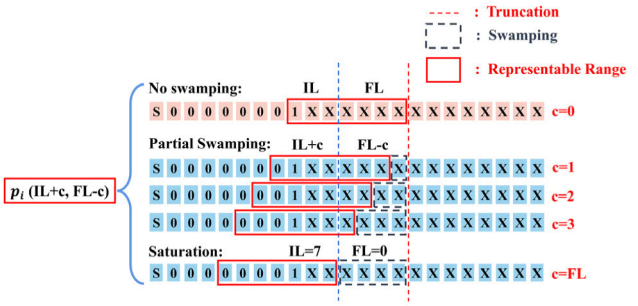

FIGURE 2. Carry-aware accumulation process with initialized format: IL = 3, FL = 4.

The error in carry-aware accumulation is related to the value domain of *Acc*. To quantify the error, we assume that the maximum value of c is $K(K \leq \text{FL})$ in an N-length accumulation chain. Consequently, S_N is divided into K sub-blocks, each maintaining the same value of c. Here, $N = \sum_{k}^{K} n_k$, where n_k corresponds to the length of the k^{th} sub-block. The arithmetic operations in each sub-block are expressed as:

$$S_k = \sum\nolimits_j^{n_k} P'_j + \sum\nolimits_j^{n_k} \Delta P_j^k. \tag{8}$$

Thus, S_N is given by:

$$\begin{aligned} S_N &= \sum\nolimits_k^K S_k + \sum\nolimits_m^M \Delta P''^m \\ &= \sum\nolimits_j^N P'_j + \sum\nolimits_j^{n_k} \Delta P_j^1 + \ldots + \sum\nolimits_j^{n_K} \Delta P_j^K + \sum\nolimits_m^M \Delta P''^m, \end{aligned} \tag{9}$$

where $\left|\Delta P_j^1\right| \leq 2^{-FL}$, $\left|\Delta P_j^k\right| \leq 2^{k-FL}$, $\left|\Delta P''^m\right| \leq 2^{m-FL}$. $\sum_m^M \Delta P''^m$ corresponds to the swamping error of *Acc*. In most cases, $M \ll$ N. Thus, (9) can be simplified as:

$$S_N \approx \sum\nolimits_j^N P'_j + \sum\nolimits_k^K \sum\nolimits_j^{n_k} \Delta P_j^k. \tag{10}$$

In summary, errors in carry-aware accumulation can be categorized into swamping and truncation errors.

B. ADAPTIVE FIXED-POINT FORMAT LEARNING

Truncation from high to low precision leads to irreversible information loss. Preserving critical information in high-precision values is important to minimize this truncation error. One intuitive method is to represent P_j in (6) with an appropriate format. The 8-bit fixed-point format takes *Acc* in the range [−128, 127], beyond which leads to saturation errors. In contrast, the lower bound (0, 7) offers a minimum resolution of 2^{-7}, causing the full truncation of small values. Meanwhile, a fixed data format cannot adapt to partial sums with varying statistical characteristics. An improper format leads to an exponential growth in error, affecting the numerical accuracy of S_N. Numerical tests on quantized networks show that (IL, FL) cannot provide sufficient resolution. In Figure 3, for a convolution layer in ResNet18, the format (IL, FL) yields a minimum relative accumulation error of

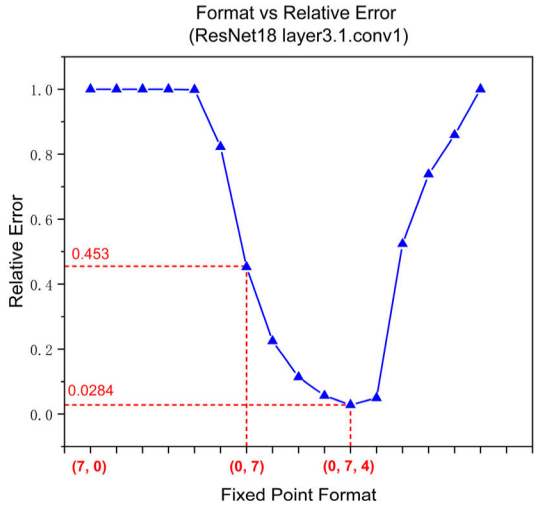

FIGURE 3. Accumulation errors introduced by different fixed-point formats for ResNet18 layer 3.1.conv1.

45%. Whereas, by scaling the original data using a scale factor of 2^4, this error can be reduced to 2.8%.

Therefore, an adaptive fixed-point format learning method is proposed, incorporating an additional scale factor β to enhance the ability to represent small values. The scale factor ensures a minimum resolution of $2^{-7-\beta}$ through scaling P_i by 2^β before accumulation and subsequently restoring the S_N after completion. With β, the fixed-point format is rewritten as (IL, FL, β). In our framework, a coefficient α is introduced to update the fixed-point format based on the relative error of P_j. Specifically, α is learned by:

$$\alpha += sign \times lr_\alpha \times \log_2 (1 + \max (\mathrm{e}_r/\mathrm{e}_m - 1, 0)), \tag{11}$$

where lr_α is the learning rate, e_r corresponds to the relative errors of P_i, and e_m is the target value, which is initialized to 2^{-6}. *sign* is a customized symbol function, as shown in:

$$sign = \begin{cases} -1; & \sum e_a(\left|e_a \geq \frac{1}{\beta}\right|) > \sum e_a\left(|e_a < \frac{1}{\beta}|\right) \\ +1; & otherwise. \end{cases} \tag{12}$$

Equation (11) updates α based on the relative error in low-precision accumulation. If $e_r > e_m$, (11) equals $(sign \times lr_\alpha \times log_2(e_r/e_m))$. *sign* indicates the error composition, where e_a represents the absolute error of P_i, $e_a(|e_a \geq 1/\beta|)$ denotes the saturation error in P_j, and $e_a\,(|e_a < 1/\beta|)$ corresponds to the truncation error. $sign = -1$ indicates saturation dominance, prompting a reduction in α to widen the integer representation range. If $sign = 1$, truncation is considered to be the major part, and α should increase for a greater fraction length. When $\mathrm{e_r} \leq \mathrm{e_m}$, α remains constant to enable stable training. Finally, if $FL < 7$, α updates FL to $\lfloor FL \times \alpha \rfloor$. When $FL \geq 7$, α modifies β to $(\lfloor FL \times \alpha \rfloor - 7)$.

C. WEIGHT-ORIENTED INCREMENTAL ACCUMULATION ORDER

We designate a single accumulation within the k^{th} sub-block as a carry transfer of length k. As shown in (10), a reduction of the total carry transfer length diminishes the swamping error ΔP_i^k. This reduction can be naturally achieved by incremental accumulation order [7], [12]. However, the chronological generation of partial sums in hardware inference makes it challenging to precisely implement an incremental accumulation order. Considering the trade-off between the accuracy and hardware overhead, we propose a weight-oriented sorting method.

In MAC operations, the magnitude of weights plays a role in determining their contribution to the outputs, as weights with small magnitudes tend to produce small partial sums [21], [25], [26]. Based on this insight, (13) is employed to derive the magnitude of each weight group, and the accumulation order is determined according to the magnitude. Specifically, the group with a smaller $Mg(w^l)$ tends to yield a smaller partial sum, which should be accumulated first. Conversely, the group with a larger $Mg(w^l)$ should be accumulated later. Importantly, during inference, the order remains the same owing to the invariance of the weights.

$$Mg\left(wg^l, Co\right) = ||W^l_{Co,G\times(ci-1)+1:G\times Ci,:,:}||_2. \tag{13}$$

$||W^l||_2 = \sqrt{\sum W^l(:)^2}$. Where W^l denotes the weights in l^{th} layer. Since the chunk size G corresponds to the hardware parallelism, the indexing does not affect the computation efficiency of MAC operations.

D. INTEGER-ONLY INFERENCE

Full-precision scale factors and BN layers should be tackled to save computational resources [14], [18]. Specifically, CANET adopts a double-forward method for BN fusion. BN layers are first fused with the convolutional layers to compute the effective weight and bias, as expressed in (14) and (15), respectively. Subsequently, the first inference is performed to update the parameters in BN, such as μ and σ. During the second inference, BN remains invariant, and the output of the fused layer engages in back-propagation. The fused layers avoid additional hardware overhead of BN.

Additionally, power-of-two scale factors are employed to eliminate the 32-bit multiplication, as depicted in (16). Exponential Moving Average (EMA) is employed to update *rmax* and *rmin*. As a result, during inference, full-precision multiplication is substituted with a more efficient bit-shift logic.

$$W_{fuse} = \gamma \frac{W_{conv}}{\sqrt{\sigma^2 + \epsilon}}. \tag{14}$$

$$b_{fuse} = \gamma \frac{-\mu}{\sqrt{\sigma^2 + \epsilon}} + \beta. \tag{15}$$

$$scale \approx 2^{\left\lfloor \log_2\left(\frac{rmax-rmin}{2^b-1}\right)\right\rceil}. \tag{16}$$

where $\gamma, \mu, \beta, \sigma^2$ are learnable parameters in BN layers, ϵ is a tiny constant, W_{conv} denotes the weights of the convolutional layer. W_{fuse} and b_{fuse} correspond to effective weights and biases. *rmax*, *rmin* represent the maximum and minimum values recorded in the calibrator, respectively.

V. EXPERIMENTS

In this section, we first present the details of the experiment settings. Subsequently, numerical experiments are designed to validate the benefits of adaptive format learning and weight-oriented accumulation order for low-precision accumulation. Following this, CANET is compared against state-of-the-art methods that maintain 32-bit accumulators and other high-precision arithmetic operations. Remarkably, CANET is capable of achieving comparable performance. Finally, we demonstrate the effectiveness of CANET in terms of power and area on a 55nm CMOS custom hardware platform.

A. EXPERIMENTAL SETUP

CANET is evaluated on the image recognition task using ImageNet datasets [33]. All images are resized to 256 × 256, then randomly cropped to 224 × 224, with random horizontal flipping and normalization. The evaluation is conducted on three different NNs: VGG-16bn [20], ResNet18 [19], and ResNet34 [19], using the Top-1 and Top-5 accuracy on the validation set as performance metrics.

We begin with training the full-precision models from scratch and then use the quantization-aware training (QAT) framework to train the quantized models [24]. Afterward, CANET is adopted to fine-tune the quantized model. During pre-training, the SGD optimizer is used with 0.1 initial learning rate, 0.9 momentum, and a weight decay of 0.0005. In the QAT phase, the initial learning rate is set to 0.001, and the min-max calibrator is applied for weight and activation. We use the per-tensor and per-channel granularity scale factors for activation and weights, respectively.

The fixed-point formats for all layers are initialized to $(2, 5, 0)$. Additionally, the learning rate lr_α is set to 0.001 to update α, and the weight-oriented accumulation order is only used in the inference phase. For the residual blocks in ResNet, the inconsistent formats in shortcut connection operations incur numerical errors. To this end, we conduct format unification before accumulation. Specifically, we select the largest IL and smallest β between sub-layers as the unified IL and β, then determine FL based on the unified IL.

B. ADAPTIVE FIXED-POINT FORMAT LEARNING

Adaptive fixed-point format learning aims to automatically select the appropriate format for different distributions of partial sums. The experiments are conducted on VGG16-bn, ResNet18, and ResNet34. The fixed-point formats determined by the adaptive learning algorithm are shown in Figure 4 (a), Figure 4 (c), and Figure 4 (e). The formats vary across layers, demonstrating the design of adaptive learning. Furthermore, β plays a role in deep layers, making fixed-point formats adaptable to variable data distribution. Figure 4 (b), (d), and (f) present the distribution of the average

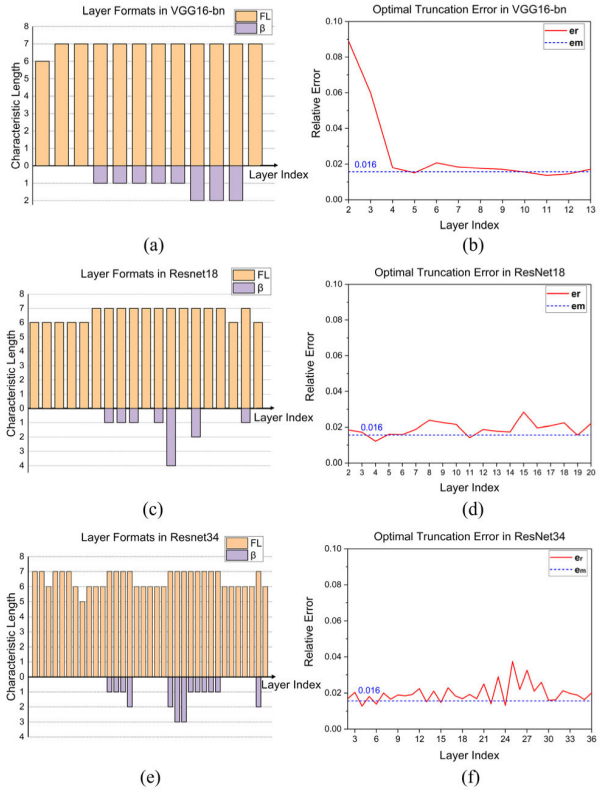

FIGURE 4. (a) Fixed-point format for layers of VGG16-bn. (b) Relative error of each layer in VGG16-bn. (c) Fixed-point format for layers of ResNet18. (d) Relative error of each layer in ResNet18. (e) Fixed-point format for layers of ResNet34. (f) Relative error of each layer in ResNet34.

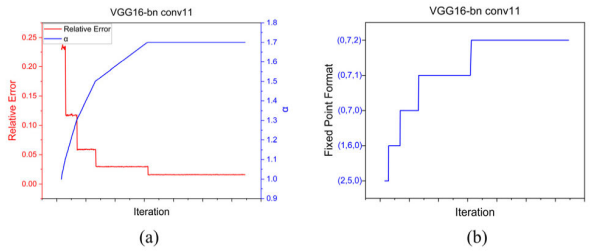

FIGURE 5. (a) Iterative process of learnable α and relative errors during adaptive learning in VGG16-bn conv11. (b) Adaptive tuning process of the fixed-point format in VGG16-bn conv11.

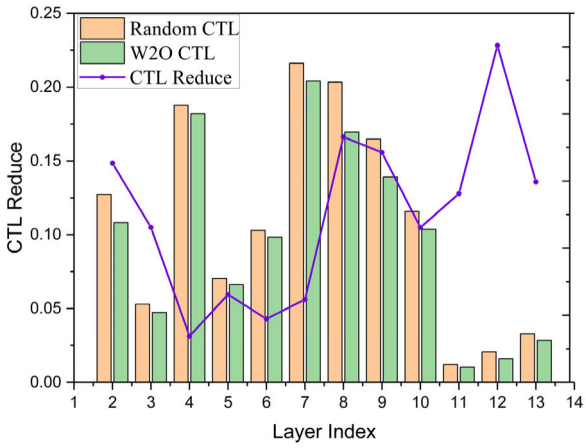

FIGURE 6. Comparison of the carry transfer length (CTL) for random and weight-oriented accumulation order (W2O) across layers in VGG16-bn.

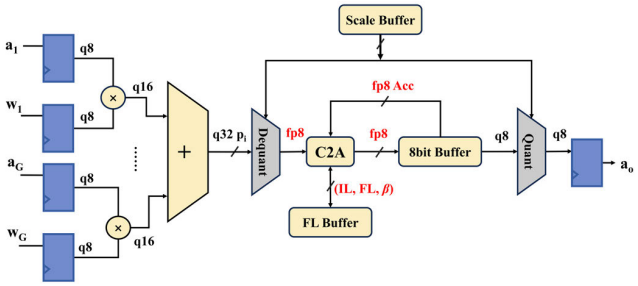

FIGURE 7. Multiply-accumulate unit with 8-bit accumulator and carry-aware logic "C2A" for hardware analysis.

truncation error introduced by the final fixed-point format. The experimental results show that, through adaptive learning, the relative truncation errors from the fixed-point format essentially are close to the ideal relative error e_m.

Subsequently, Figure 5 (a) illustrates the adaptive learning process of conv11 in VGG16-bn. Test nodes capture the iterative process of α and truncation error for analysis. The initialized format $(5, 2, 0)$ results in an average truncation error of 23%. Then, α is updated based on the truncation error, using varying step sizes. Eventually, α remains stable when the error is reduced below the set value. For conv11, the final format is determined to be $(0, 7, 2)$, and the error eventually converges to 1.5%. Numerical results demonstrate the effectiveness of adaptive learning. The proposed method requires only two epochs to fine-tune the quantized networks.

C. CARRY TRANSFER LENGTH COMPARISON

As shown in (10), the accumulation order affects the transfer length of the carry, which results in different swamping errors. Intuitively, a longer transfer distance results in a larger swamping error. The formula for the total carry transfer length (CTL) is given in (17). Figure 7 illustrates the CTL in each layer of VGG16-bn under the two accumulation orders, highlighting varying degrees of CTL suppression with the weight-oriented accumulation order (W2O). In comparison to the random order [7], the proposed method demonstrates improvements on all network layers with various accumulation lengths, achieving an average reduction of 12% on CTL.

$$CTL = \sum_{k=1}^{K} k \cdot n_k. \tag{17}$$

D. BENCHMARK RESULT

CANET is compared with other high-precision baselines that retain 32-bit accumulators and other high-precision arithmetic operations within the 8-bit quantization framework. The experimental results of three NNs are presented in Table 1 to Table 3. BL denotes the full-precision baseline. Bits represent the bit widths of activations and weights, which are

TABLE 1. Top1 and Top5 test results on VGG16-bn.

Method	Bits	Mul	Acc	Top1 / Top5	Drop
BL	32	32	32	73.4 / 91.5	-
LSQ [11]	8	32	32	73.5 / 91.5	0
FAQ [10]	8	8	32	73.7 / 91.6	+0.1
Ours	**8**	**shift**	**8**	**73.4 / 91.4**	**-0.1**

TABLE 2. Top1 and Top5 test results on ResNet34.

Method	Bits	Mul	Acc	Top1 / Top5	Drop
BL	32	32	32	74.1 / 91.8	-
LSQ [11]	8	32	32	74.1 / 91.8	0
FAQ [10]	8	8	32	73.7 / 91.6	-0.2
CPT [34]	8	32	32	73.1 / --	-
Ours	**8**	**shift**	**8**	**73.8 / 92.0**	**0.2**

TABLE 3. Top1 and Top5 test results on ResNet18.

Method	Bits	Mul	Acc	Top1 / Top5	Drop
BL	32	32	32	70.5 / 89.6	-
LSQ [11]	8	32	32	71.1 / 90.1	0.5
FAQ [10]	8	8	32	70.2 / 89.3	-0.3
HAWQ-V3 [35]	8	32	32	71.6 / --	-
RV-Quant [36]	8	32	32	70.0 / 89.3	-0.3
Ours	**8**	**shift**	**8**	**71.0 / 90.1**	**0.5**

fixed at 8-bit in 8-bit quantization methods. Mul corresponds to the precision of multiplication, where 32 indicates the 32-bit multiplication, and shift denotes the use of bit-shift logic instead of multiplication. Acc represents the precision of accumulators and 32 corresponds to the full-precision accumulators.

In the VGG16-bn test, with only 8-bit accumulation and bit-shift logic, CANET experiences only a 0.1% accuracy degradation compared to the full-precision model. On the ResNet18 and ResNet34, CANET achieves 0.5% and 0.2% performance above the full-precision baseline. The experimental results show that with only 8-bit accumulators, CANET can obtain comparable performance with other high-precision baselines.

E. HARDWARE ANALYSIS

A MAC unit structure is implemented on SMIC 55nm CMOS to evaluate the effectiveness of CANET. The structure which is illustrated in Figure 7, comprises the following three parts:

TABLE 4. Hardware implementation results for two architectures of MAC unit in the 55nm CMOS process.

Architecture	Bits	Acc	Power (μW)	Area (μm^2)
Baseline	8	32	20.5	7295
CANET	**8**	**8**	**12.3**	**3718**

(1) an 8-bit multiply-accumulate arithmetic unit, primarily responsible for generating the partial sums $\sum_i^G p_i$; (2) an 8-bit accumulator, consisting of 8-bit full adders and 8-bit registers (in regular structure, this part requires 32-bit full adders and 32-bit registers); and (3) additional control logic for carry-aware logic, and the bit-shift circuits for scaling and rescaling.

We implement a MAC unit with the proposed architecture and a conventional MAC unit with 32-bit accumulators, comparing the power and area overhead through Design Compiler. The hardware analysis results are listed in Table 4. As-proposed 8-bit-accumulator-based MAC reduces power consumption by 40% and area by 49%, visually demonstrating a significant improvement in power efficiency.

Through further discussion, the proposed inference framework effectively optimizes the redundant intermediate information generated by accumulation. The conventional architecture requires a combination of 32-bit and 8-bit memory rather than a single 8-bit memory to overcome the mismatch of the bit widths between the intermediate results and quantizer. In contrast, CANET eliminates the mismatch and motivates the removal of the 32-bit intermediate buffer, allowing accumulation in the feature memory directly. The proposed framework enables a more flexible design boundary for efficient inference of neural networks.

VI. CONCLUSION

This paper presents a novel 8-bit network quantization methodology that, aims to achieve an efficient inference framework using 8-bit accumulators. Unlike previous approaches that sought to reduce the risk of overflow by compressing the parameter range or precision, the proposed framework uses 8-bit carry-aware accumulators to avoid overflow. Additionally, we employ adaptive fixed-point format learning and a weight-oriented accumulation order to improve accumulation accuracy. The experimental results demonstrate that CANET achieves comparable performance with other high-precision methods on three neural networks while using 8-bit accumulators. Furthermore, the evaluation of the customized hardware platforms confirms the effectiveness of CANET in power efficiency.

REFERENCES

[1] O. Weng, "Neural network quantization for efficient inference: A survey," 2021, *arXiv:2112.06126*.

[2] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Boca Raton, FL, USA: CRC Press, 2022, pp. 291–326.

[3] M. Courbariaux, Y. Bengio, and J. David, "Training deep neural networks with binary weights during propagations," 2015, *arXiv:1511.00363*.
[4] H. Xie, Y. Song, L. Cai, and M. Li, "Overflow aware quantization: Accelerating neural network inference by low-bit multiply-accumulate operations," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jan. 2020, pp. 868–875.
[5] C. Sakr, "Accumulation bit-width scaling for ultra-low precision training of deep networks," in *Proc. ICLR*, 2019, pp. 1–15.
[6] R. Ni, H.-M. Chu, O. Castaneda, P.-Y. Chiang, C. Studer, and T. Goldstein, "WrapNet: Neural net inference with ultra-low-precision arithmetic," in *Proc. ICLR*, 2020, pp. 1–16.
[7] T. G. Robertazzi and S. C. Schwartz, "Best ordering," *ACM Trans. Math. Softw.*, vol. 14, no. 1, pp. 101–110, Mar. 1988.
[8] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7685–7694.
[9] Q. Jin, "F8Net: Fixed-point 8-bit only multiplication for network quantization," in *Proc. ICLR*, 2022, pp. 1–19.
[10] J. L. McKinstry, S. K. Esser, R. Appuswamy, D. Bablani, J. V. Arthur, I. B. Yildiz, and D. S. Modha, "Discovering low-precision networks close to full-precision networks for efficient embedded inference," 2018, *arXiv:1809.04191*.
[11] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," in *Proc. ICLR*, 2020, pp. 1–12.
[12] P. Blanchard, N. J. Higham, and T. Mary, "A class of fast and accurate summation algorithms," *SIAM J. Scientific Comput.*, vol. 42, no. 3, pp. A1541–A1557, Jan. 2020.
[13] B. D. Bruin, Z. Zivkovic, and H. Corporaal, "Quantization of deep neural networks for accumulator-constrained processors," *Microprocessors Microsystems*, vol. 72, Feb. 2020, Art. no. 102872.
[14] S. Jain, A. Gural, M. Wu, and C. Dick, "Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks," in *Proc. Mach. Learn. Syst.*, Mar. 2020, pp. 112–128.
[15] 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 *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1966–1976.
[16] S. Kim, A. Gholami, Z. Yao, M. W. Mahoney, and K. Keutzer, "I-BERT: Integer-only BERT quantization," in *Proc. ICML*, 2021, pp. 5506–5518.
[17] Q. Guo, Y. Wang, and X. Cui, "Integer-only neural network quantization scheme based on shift-batch-normalization," 2021, *arXiv:2106.00127*.
[18] X. Zhao, Y. Wang, X. Cai, C. Liu, and L. Zhang, "Linear symmetric quantization of neural networks for low-precision integer hardware," in *Proc. ICLR*, 2020, pp. 1–16.
[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[21] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, Dec. 2016, pp. 2074–2082.
[22] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2625–2631.
[23] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *Proc. ICLR*, Apr. 2017, pp. 1–10.
[24] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.
[25] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," 2019, *arXiv:1902.09574*.
[26] S. Vadera and S. Ameen, "Methods for pruning deep neural networks," *IEEE Access*, vol. 10, pp. 63280–63300, 2022.
[27] J. Y. Kang, C. H. Ryu, and T. H. Han, "Binarized neural network with parameterized weight clipping and quantization gap minimization for online knowledge distillation," *IEEE Access*, vol. 11, pp. 8057–8064, 2023.
[28] Z. Liu, K.-T. Cheng, D. Huang, E. Xing, and Z. Shen, "Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 4932–4942.
[29] I. Colbert, A. Pappalardo, and J. Petri-Koenig, "A2Q: Accumulator-aware quantization with guaranteed overflow avoidance," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 16989–16998.
[30] H. Li, J. Liu, L. Jia, Y. Liang, Y. Wang, and M. Tan, "Downscaling and overflow-aware model compression for efficient vision processors," in *Proc. IEEE 42nd Int. Conf. Distrib. Comput. Syst. Workshops (ICDCSW)*, Jul. 2022, pp. 145–150.
[31] M. Nagel, M. Fournarakis, R. Ali Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A white paper on neural network quantization," 2021, *arXiv:2106.08295*.
[32] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 7948–7956.
[33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[34] Y. Fu, H. Guo, M. Li, X. Yang, Y. Ding, V. Chandra, and Y. Lin, "CPT: Efficient deep neural network training via cyclic precision," 2021, *arXiv:2101.09868*.
[35] Z. Yao and Z. Dong, "HAWQ-v3: Dyadic neural network quantization," in *Proc. ICML*, 2021, pp. 11875–11886.
[36] E. Park, S. Yoo, and P. Vajda, "Value-aware quantization for training and inference of neural networks," in *Proc. ECCV*, 2018, pp. 580–595.

JINGXUAN YANG received the B.S. degree from Chongqing University, Chongqing, China, in 2021. He is currently pursuing the M.S. degree with the Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, China. His research interests include low-power IC design and image classification.

XIAOQIN WANG is currently a Researcher with the Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, China. Her main research interests include low-power processors and microsystems design for intelligent perception, intelligent edge computing, and audio and video processing.

YIYING JIANG received the B.S. degree from Southeast University, Nanjing, China, in 2021, and the M.S. degree from Waseda University, in 2022. She is currently pursuing the Ph.D. degree with the Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, China. Her research interests include compute-in-memory and deep learning accelerator.