Received 17 February 2023, accepted 9 March 2023, date of publication 13 March 2023, date of current version 16 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3256540

RESEARCH ARTICLE

ANS: Assimilating Near Similarity at High Accuracy for Significant Deduction of CNN Storage and Computation

WANG WANG, (Graduate Student Member, IEEE), XIN ZHONG, MANNI LI, ZIXU LI, AND YINYIN LIN, (Member, IEEE)
State Key Laboratory of ASIC and System, School of Microelectronics, Fudan University, Shanghai 201203, China

Corresponding author: Yinyin Lin (yylin@fudan.edu.cn)

ABSTRACT Activation data size has been roaring with the development of convolutional neural networks, which accounts for the boosting storage requirements. Our insight indicates that non-zero values dominate activations, of which the patterns demonstrate near similarity. We propose ANS method to compress activations in real time during both training and inference. High compression ratio with less accuracy loss is achieved by our optimization strategies, including determination of selection box (SB) size according to the amount of zero values of layer, learning and calibrating threshold dynamically, using the mean value of similar SB as compression value. Over 49% of compression ratio is achieved with accuracy loss of less than 0.892%, as well as reduction of multiplications by more than 60%. Comparing to three state-of-art compressed methods under five mainstream CNN models, ANS provides compression ratio improvement of 3.2x over RLC5, 1.9x over GRLC and 1.7x over ZVC. The ANS compressor and decompressor are implemented in Verilog and synthesized in 28nm node, which indicates that ANS has less cost of performance and hardware overburden. ANS modules could be seamlessly attached at the interface or deeply coupled into DNN accelerator with changed data path in the MAC array, which achieve 38% and 56% reduction in energy consumption, respectively.

INDEX TERMS Compression, convolutional neural networks, energy consumption, memory footprint of accelerator.

I. INTRODUCTION

Among models of various deep neuron networks (DNNs), convolutional neural networks (CNNs) have always been predominant in the fields of computer vision. Despite the rapid development of diversified new networks, the use of CNN still experienced approximately a 3 times increase for DNN workloads in data centers from 2016 to 2020 [1].

During the past ten years since CNNs emerged, the demand for memory capacity has been soaring. From AlexNet in 2012 to ResNet152 in 2015, the data size of intermediate activations increased 48x, from less than 8 MB up to nearly 380 MB [2], [3]. And the batch operations further make activations increase linearly with the batch size. Compared with the skyrocketing of activations size, the weights size has even decreased instead (see Section III-A in detail). As a result, the activation size dominates the soaring demand for memory storage of CNNs. For training, the size of activations is about 12 GB at 256 batch size for ResNet50, which occupies more than 80% of the total memory demand [4].

From the perspective of input datasets, ImageNet datasets have 1.34 million image and require 138 GB of storage space [5], far exceeding any available on-chip capacity. Training datasets of many application scenarios such as auto-driving are also continuously expanding [6].

For inference, at least one layer of activations has to be stored for the computation of next layer. Take MobileNet_v2 as an example. Its maximum of activations with fixed point (FXP) 8bits reaches 1.15MB [7]. Limited on-chip capacity at the edge, e.g., hundreds of KB [8], sometimes cannot even

The associate editor coordinating the review of this manuscript and approving it for publication was Mario Donato Marino.

satisfy the activation storage of single layer. And simple reuse of activation on chip is usually not the optimal solution of performance and energy consumption for accelerator [9].

The surge of memory capacity for both training and inference has led to a huge challenge to the energy consumption required by CNN tasks, of which the majority comes from inter-chip data movement. The energy consumption due to accessing data off-chip is two orders of magnitude higher than that of on-chip computing [1].

There are various hardware-based efforts aiming at the challenge of storage explosion. Each one has their own advantages and bottlenecks, e.g., (a) Increasing on-chip memory, which encounters fundamental limitations of on-chip SRAM capacity, due to the reticle limit and Moore's law stagnating [10], [11]; (b) Processing in/near DRAM (PIM), which takes advantage of the large capacity of DRAM, but suffers from the interaction with manufacturers and customers, which involves changes of the memory sub-system and application code for non-standard PIM [12], [13]; (c) 2.5D/3D package, which is beneficial for high bandwidth and high capacity of DRAM, but very costly to further increase the I/O pins under the serious physical constraints of a package, such as power, temperature, form factor and integrity etc [14]. (d) Computation in memory (CIM) of cell-level, including early analog CIM (ACIM) [15] and later digital CIM(DCIM) [16]. Both decrease the energy of data movement, but can cover very limited DNN trainings because only FXP operations can be handled up to now. ACIM faces reliability challenges due to unavoidable variations among cells and high cost of area/power overburden from analog-digital converter. DCIM has to implement digital design besides memory cells, which leads to low area efficiency [17].

Software-based efforts usually aim to reduce the data size of networks in order to decrease CNN energy consumption, including data quantization, rematerialization, network pruning, and sparsity compression, etc.

Data quantization is to replace the original floating-point (FP) precision with a lower one, which brings benefits of both a smaller memory footprint and an amount reduction of corresponding computation [18].

Rematerialization is a method of recalculating activations instead of storing them during training [19]. In addition, due to operations such as ReLU and Pooling in CNN without parameter updates during backward propagation, the activation values can be stored using binarization encoding format [20].

Fine-grained pruning weights are not valid for reducing the data size of activations [21]. Coarse-grained pruning of some non-significant filters can structurally clip associated activations, in which the accuracy loss caused by pruning requires additional iterations to recover [22], [23]. Consequently, even more energy and longer training time are consumed.

Sparsity can be used to compress the activations during training and inference. The basic idea of mainstream compression methods, including RLC [8], ZVC [24] and GRLC [25], is to eliminate the storage space of zero values (ZVs) and store only non-zero values (NZVs) and indexes. It is obvious that the benefits of these method are dependent on the percentage of ZVs in activations.

Actually, NZVs occupy most part of activations for CNN models. Our insight shows that the activation patterns of NZVs have the feature of near similarity (see Section III-C in detail). More than that, the region with a consecutive zero pattern can also be regarded as a special case of near similarity.

We propose ANS method, which can be applied to compress activations of both NZVs and ZVs. Not only the memory footprint but also the amount of associated computation is significantly reduced. The deployment of ANS into inference and training are discussed. And ANS compressor (ANS_C) and decompressor (ANS_D) are implemented in Verilog and synthesized in 28nm node. The cost of performance and hardware overburden are analyzed. Based on five wide-used networks, our evaluations indicate excellent trade-off between compression ratio and accuracy as well as significant reduction of energy consumption due to the decrease in both data size and amount of computation, and we thoroughly compare the compression results with three state-of-art popular sparsity compression methods.

The remainder of this paper is organized as follows: Section II provides a short survey on related work. Section III is about insights which motivate ANS. Section IV describes the detailed approach of ANS. In Section V, two architectures utilizing ANS modules are presented. Section VI evaluates the algorithm and hardware architecture. Section VII discusses the root cause why ANS can achieve both high compression ratio and less accuracy loss. Finally, Section VIII concludes the paper.

II. RELATED WORK

There are prior arts aiming at reduction of data size of activations. Jain et al. [26] observes that the ReLU outputs, that has to be stored for the backward pass, can be encoded using just 1-bit values, leading to 32× compression of the ReLU outputs. Backprop [20] only stores the activations generated by batch normalization (BN) in the computation chain, while the unstored activations are recalculated according to the stored activations in the backward propagation, which is also known as rematerialization. Checkmate [19] formalizes tensor rematerialization as a constrained optimization problem, which is optimized by using off-the-shelf numerical solvers. This type of methods doesn't involve data compression and still has to store massive uncompressed activation values.

Mao et al. [21] analyze that the model size deployed in inference can be reduced by pruning the network at various granularity during the training stage. There are three major pruning methods: (a) Irregularly fine-grained pruning (FGP) of some individual small weight values. This method has no effect on activation reduction [23]. (b) Coarse-grained pruning (CGP) of some weight filters, as well as the relevant activation channels. Luo and Wu [22] prune 25%-50% of filters and activations within the network, which reduces by half the loading of data storage and computation in each iteration. (c) Block-grained pruning (BGP) [27] divides weight into 1 × 4 block, in which 50% of the minimum value and multiply-and-accumulate (MAC) operations are trimmed. Vooturi et al. [28] propose HBsNN to combine boxes of various sizes to segment the sparse matrix to achieve a higher cutting ratio. Since the above method is lossy to crop off some values, ''Retraining'' operations are required to restore the lost precision, which results in an 60% increase of iteration number, and brings additional burden of not only cycles but also energy consumption in training [22].

Sparsity compression (SCP) has been explored to reduce ZVs of activation during training or real-time inference scenarios. In Eyeriss [8], RLC method encodes consecutive ZVs into a single run, which compresses and decompresses the NZVs at the interface of the accelerator, thereby reducing the DRAM accesses. Based on spatial correlation of activation values, GRLC [25] exploits RLC on the 2 × 2 grid of activations, and further distinguishes outliers and non-outliers for NZVs, which achieves a 1.73x compression ratio improvement over RLC. ZVC [24] stores non-zero activations with mask bits to indicate whether the activation is zero after encoding a window of 32 elements. CSR [29] compresses a matrix by processing each row as a sparse vector, by which compression enables random accesses to any row. CSC [30] is similar to CSR, except that NZVs are stored columnwise. The SCP methods mainly focus on ZVs, so the compression ratio depends on the proportion of ZVs in the activations.

Along with the deduction of the activation size, the multiplications by zero are skipped too, which further optimize energy consumption and performance in Snapea [31]. However, the control logic is so complicated that sparse processing can only be applied to some special layers, e.g. minority of convolutional layers in SCNN [32] or fully connected layers in EIE [30], even though direct computation with encoded indices is implemented by dedicated MAC array in order to avoid decompressing of weight and activation [33].

Values are generally quantified as FXP format in inference, which provides an opportunity to explore the sparsity within independent elements. Alameldeen and Wood [34] divide FXP16 activation values into four patterns according to the position of ZVs. Four patterns with different bit widths are a challenge to the accelerator implementation. Song et al. [35] propose

DRQ to judge the local area sensitivity of activations, which divides the activations into regions with large or small influence on the task results. Then activations are quantified as high bit width for sensitive areas and low bit width for insensitive areas. Huang et al. [36] split activations by structured dynamic block. L1-norm of values in the block is used to compare with the learnable threshold to determine whether the low bits needs to be calculated. These methods use a block-grained approach to save energy of data access and multiplication, but are limited to quantized models.

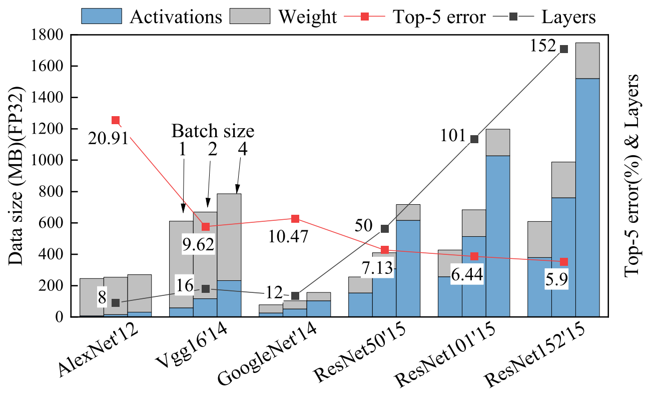


FIGURE 1. Trends of data size for CNN models.

There are prior arts which involves local similarity of input image. Miguel et al. [37] proposes a cache for tasks with close pixel data, of which the multiplication results are reused to reduce the amount of computation. In COREx [38], the calculation results of adjacent frames with similar pixels are reused, which is further applied to DNN tasks to reduce multiplication operations [39], [40], [41], [42]. However, this type of prior arts is only for eliminating the amount of multiplications within Euclidean metric without the consideration of data compression.

III. INSIGHTS WHICH MOTIVATE ANS

In this section we discuss three insights which motivate and guide ANS. The related experiments are implemented on the PyTorch framework with ImageNet verification dataset, and the pre-training models used in inference are from the PyTorch website [48].

A. ACTIVATIONS DOMINATE THE STORAGE REQUIREMENTS

The soaring requirement of CNN model for storage comes from the pursuit of high accuracy. The rapid growth in both depth and width of network leads to a rocketing rise in model parameters and intermediate activations. We summarize the trend of CNN models in Fig. 1 [2], [3], [43], [44]. At an early stage of CNN emergency, weights occupy the majority of total data size. Afterwards, the activations increase much more rapidly than weights.

There are 8 layers for AlexNet in 2012 with 238 MB of weights in FP32, which accounts for most part of total data. In comparison with AlexNet, the weight size of Vgg16 [43] in 2014 increased by 2.3x, while activation size increased by 7.3x. The difference in growth rate originates from two aspects. Firstly, the layers of Vgg16 go up to 16 in order to improve accuracy. Secondly, the network becomes wider. The inception modules of GoogleNet [44] in 2014 utilized 1 × 1 filters to reduce the dimensions of activation, enabling a few 3 × 3 filters to extract features, while only one fully connected layer with large parameters is retained. Consequently, the weight size is only 21% of that of AlexNet.

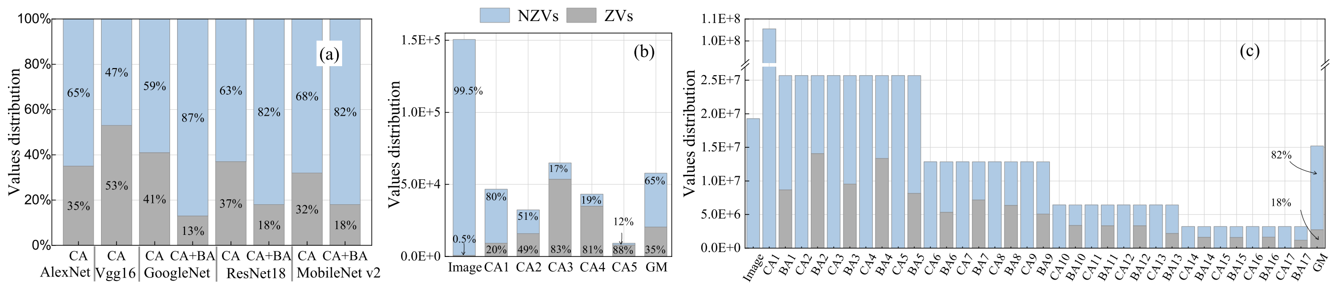


FIGURE 2. ZVs and NZVs percentage in activations and input image for (a) five typical CNN networks, (b) AlexNet and (c) ResNet18.

But the introduction of additional intermediate layers leads to the growth of network depth. Thus, the activation size increased by 3.25x in comparison with AlexNet. As for ResNet152 [3] in 2015, the error rate decreased to 5.9%, nearly 3.6x smaller than that of AlexNet in 2012. But the depth increases to 152-layer, and the residual modules make previous activations to be stored for shortcut connection with the current layer, both of which lead to nearly 48x increase of activation size from 8 MB to 380 MB. The percentage of activation in total data size increases from 3% to 62%. On the contrary, weight size changes slightly from 238 MB with AlexNet to 229 MB with ResNet152 due to introduction of residual module.

The batch operators, which are used to improve the processing efficiency and training accuracy, are another important reason for surge of activations which increase linearly with batch size. For ResNet50, the perfect batch size for training reaches up to 8192 [45]. Thus, the activation size is usually over 10x greater than weight size [1].

B. NZVS DOMINATE ACTIVATIONS

We evaluate the NZVs/ZVs percentage of activations for the popular CNN networks, as shown in Fig. 2. Actually, not all activations need to be stored. At early stage of CNN emergency, Conv-ReLU-Pooling chain is usually used in AlexNet and Vgg16. ReLU and pooling do not update parameters in the training, so the input activations do not have to be stored. Only the the intermediate activations, which are the inputs and outputs of the chain, need to be stored. We named this type as "CA" in Fig. 2. For later GoogleNet and other models, BN is added to the chain, that is, Conv-BN-ReLU-Pooling chain. Since BN needs to update parameters in training, activations after Conv and before BN need to be stored during training. We named this type as "BA". Both CA and BA need to be stored during training for these later models. But only CA has to be stored during edge inference for them. That is because fuse operation is employed for the data package processing of Conv-BN-ReLU-Pooling chain in order to handle the limited capacity of on-chip memory [46].

Fig. 2(a) shows that NZVs occupy most part of activations for all five popular CNN models. MobileNet_v2 achieves a NZV of 68% for CA (for inference scenario),

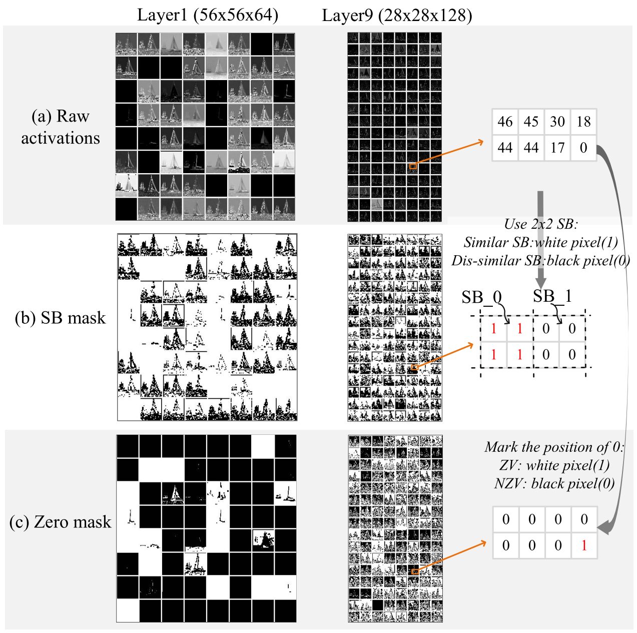


FIGURE 3. Visual activations of layer1 and layer9 for ResNet18. (a) Raw activation pattern, (b) The pattern is divided by 2 × 2 SB, and SBs with similar values inside are marked as white, (c) ZVs in the channel are visualized as white pixels and NZVs as black pixels.

while GoogleNet achieves a NZV of 87% for CA and BA (for training scenario).

For AlexNet without BN layers and ResNet18 with BN layers, we further analyze the NZVs/ZVs percentage in input image and activations of each layer, as shown in Fig. 2(b) and 2(c), respectively. For AlexNet, the geometric mean (GM) of NZVs for images and total activations reaches up to 65%, which is attributed to two reasons: (a) Pooling layers shrink the shapes of output activation layers. Consequently, the activation size in shallow layers is larger than that in deeper layers, e.g. the output activation size of AlexNet layer1 (55 × 55 × 96) is 6.7x greater than that of layer5 (13 × 13 × 256). (b) The ZVs are heavily accumulated in the deeper layers (e.g. 88% for layer5) due to ReLU, but occupy very tiny percentage in the input image (0.5%) and shallow layers (e.g. 20% for layer1). For ResNet18, since the value distribution is uniform after BN layers, the proportion of ZV

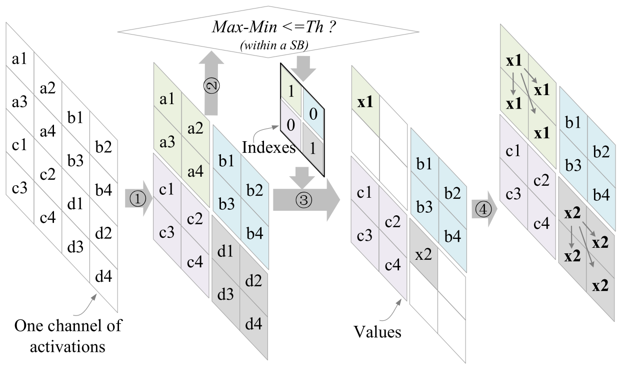


FIGURE 4. An overview of ANS algorithm.

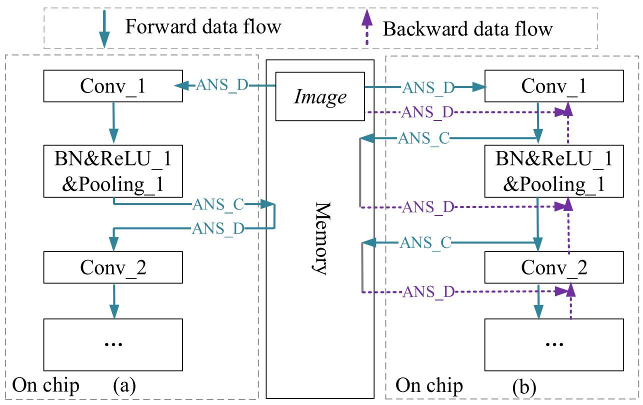


FIGURE 5. ANS_C and ANS_D embedded into data flow of hardware including (a) inference and (b) training. The image is compressed with ANS offline in advance.

of CA is 32%-56%. After ReLU, the ZV of about 50% is obtained. However, BA have almost no ZVs due to no use of ReLU before it. Therefore, 82% of the total activations is NZV.

C. NEAR SIMILARITY IN ACTIVATION PATTERNS

It is well known that the pixel values corresponding to a similar local region of an image are close [37], [38]. We further observe the similar feature in the intermediate activation patterns of CNNs. Take ResNet18 as an example, we visualize the activation channels of layer1 (a representative of shallow layer) and layer9 (a representative of deep layer), as shown in Fig. 3(a). 2×2 selection boxes (SBs) are used to segment the channels in Fig. 3(b). White pixels indicate a SB with similar values inside, such as SB_0 with values range from 44 to 46. And black pixels indicate a SB with dis-similar values inside, such as SB_1 with values range from 0 to 30. It can be seen that the white SBs scatter all over the activation pattern, but generally occupy a large part of area.

In Fig. 3(c), ZVs in the channel are further visualized as white pixels and NZVs as black pixels. It can be seen that the general white area in Fig. 3(b) is bigger than that in Fig. 3(c). That is because the SBs with similar values inside exist not only in the region of ZVs, but also in the region of NZVs. The motivation of ANS is to compress the similar SBs in both regions.

IV. PROPOSED ANS METHOD

We elaborate the ANS method in this section. Firstly, an overview of ANS framework is given, then the crucial points for trade-off between accuracy and compression ratio are addressed one by one. Finally, the inherent method of reducing multiplication operations of ANS is presented.

In this section, ANS are implemented by extending PyTorch framework. The pre-training models for inference are all from the PyTorch website [48]. ImageNet verification dataset is used as input unless special instruction is given. It is assumed that more than 1% drop in top-1 accuracy is unacceptable.

A. ANS OVERVIEW

ANS could be applied to compress not only input image but also activations, only if the local region has the feature of similarity, including non-zero similar data and consecutive-zero ones. The compressed algorithm includes three key steps illustrated in Fig. 4 (①, ② and ③), and decompressed is shown in ④. In order to obtain both high accuracy and strong compression, there are key points for each step as follows:

Step ① Dividing: The image and activation patterns are divided into continuous local regions by using SB. The key problem in this step is how the SB size impacts the compression ratio, which is discussed in detail in Section IV-B.

Step ② Judging: The threshold (Th) is used as criteria to judge the similarity of SB. Compare Th with difference of the maximum and the minimum in the SB. If the Th is bigger, SB is a similar SB (SSB). Otherwise, SB is a dissimilar SB (DSSB). The key problem in this step is how the Th impacts the accuracy after compression. The details are discussed in Section IV-C.

Step ③ Compressing: The SSB is compressed into a single value "x" with index "1", while the values in the DSSB would be totally reserved with index "0". The key problem in this step is how the "x" value is chosen, which is discussed in detail in Section IV-D.

ANS could be seamlessly embedded in the data flow during hardware running CNNs, so images and activations enable to be compressed in real time. Fig. 5 shows the typical data flow with ANS for inference and training. For training phase, the activations generated during forward propagation are compressed by ANS_C and utilized during backward propagation. ANS can also compress the CA stored off-chip memory due to capacity limitation of on-chip memory. See Section VI-A for details on ANS deployment in inference and training.

B. OPTIMIZATION OF SB SIZE

The first step of ANS is to divide activation patterns into consecutive local regions of SB size (height $*$ width

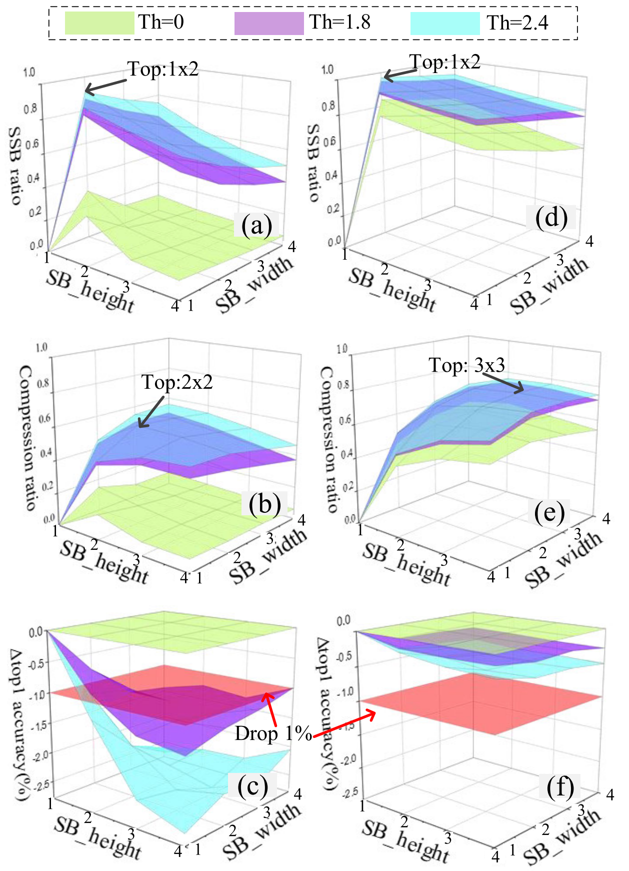


FIGURE 6. SSB ratio ((a), (d)), compression ratio ((b), (e)) and accuracy ((c), (f)) vs. Th under different SB size for AlexNet layer1 ((a), (b), (c)) and layer5 ((d), (e), (f)).

in Fig 6). Before exploring how SB Size affects the accuracy and compression ratio, we define the three concepts as follows:

$$\text{SSB\_ratio} = \frac{\text{SSB\_number}}{\text{SB\_number}} \tag{1}$$

$$\text{Compression\_efficiency} = \frac{\text{SB\_size} - 1}{\text{SB\_size}} \tag{2}$$

$$\text{Compression\_ratio} = \frac{\text{Compressed\_activations}}{\text{Total\_activations}} \tag{3}$$

The compression efficiency is positively associated with SB size. For example, 2×2 SB has a 75% (3/4) compression efficiency. And 4×4 SB has a maximum compression efficiency of 94% (15/16). There is the following correlation among the 3 concepts:

$$\text{Compression\_ratio} = \text{SSB\_ratio} \times \text{Compression\_efficiency} \tag{4}$$

Now we give the statistical analysis results for AlexNet layer1 (about 20% ZVs, representing a layer with few ZVs) and layer5 (about 88% ZVs, representing a layer with rich ZVs), as shown in Fig. 6. Three Th of 0, 1.8 and 2.4 were selected in the experiment (see Section IV-C for the detail of Th). For layer1 and layer5, the SSB ratio is the largest at 1×2 SB and decreases with the increase of SB size. For layer1, however, the maximum compression ratio is at 2×2 SB instead of 1×2. This is because SB with 1×2 or 2×1, has a compression efficiency as small as 50%. For SB size larger than 2×2, although with more than 75%(83% for 2×3 or 3×2 SB) compression efficiency, it also results in small SSB ratio due to the big percentage of NZVs of layer1, which leads to decrease of compression ratio.

For layer5, when SB is 3×3, the compression ratio reaches the maximum. This is because it has a large number of ZVs, so a large SSB ratio can be achieved under a large SB size. Due to the high compression efficiency of 3×3 SB (89%), the compression ratio of layer5 is larger than that of layer1 under the same Th. In addition, the accuracy and compression ratio are the same when the height and width dimensions are reversed in Fig. 6. Therefore, in order to achieve a large compression ratio, we use 2×2 SB for layers with rich NZVs, and 3×3 SB for layers with zero-rich values (over 80%).

For layer1, initially, the accuracy loss increases with the increase of SB size, 0.97% at 2×2 but 1.4% at 3×3 (Th $= 1.8$), which is because compression value in a small SSB effectively represents the uncompressed value, while a large SSB will cause more numerical variation for the local activations. With the further increase of SB size, the accuracy loss becomes small, 0.99% at 4×4 SB, which is caused by the limited SSB ratio, that is, it is more difficult for large SB to be determine as a SSB under a constant Th. For layer5, the accuracy loss decreases to less than 1% with the increase of SB size. For example, when the Th is 2.4, the maximum compression ratio can reach 77%, and the accuracy loss is less than 0.46%. This is because there are rich ZVs in layers, and ZV compression is lossless.

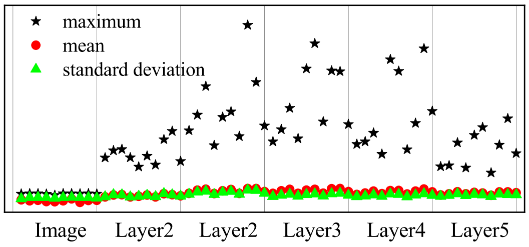


FIGURE 7. The numerical distribution of images and activations of AlexNet layers, including maximum, mean and standard deviation.

C. TH OPTIMIZED BY LEARNING AND DYNAMIC CALIBRATION

Th is the key parameter to judge if one SB is SSB, which has significant impact on the trade-off between compression ratio and accuracy. As described in Section IV-A, we compare Th with the difference between maximum and minimum in a SB to determine if it is a SSB. The bigger the difference is, the more dissimilar the data in the SB is. The opposite conclusion would be got for one single SB under different Th values. This can explain why the accuracy decrease with the increase of Th in Fig. 6. An inappropriate big Th would lead to

TABLE 1. The learning results from ten images respectively.

Images number	1	2	3	4	5	6	7	8	9	10	**Mean**
Th	1.237	1.167	1.356	1.23	1.349	0.928	1.039	1.326	1.064	1.133	**1.183**
Normalized to Mean	+4.6%	-1.4%	+14.6%	+4%	+14%	-21.6%	-12.2%	+12.1%	-10.1%	-4.2%	**0%**
Accuracy(%)	75.15	75.15	74.85	75.35	75.05	75.30	75.05	74.95	75.25	74.95	**75.25**
Normalized to Mean	-0.1%	-0.1%	-0.5%	-0.1%	-0.3%	+0.1%	-0.3%	-0.4%	0	-0.4%	**0%**

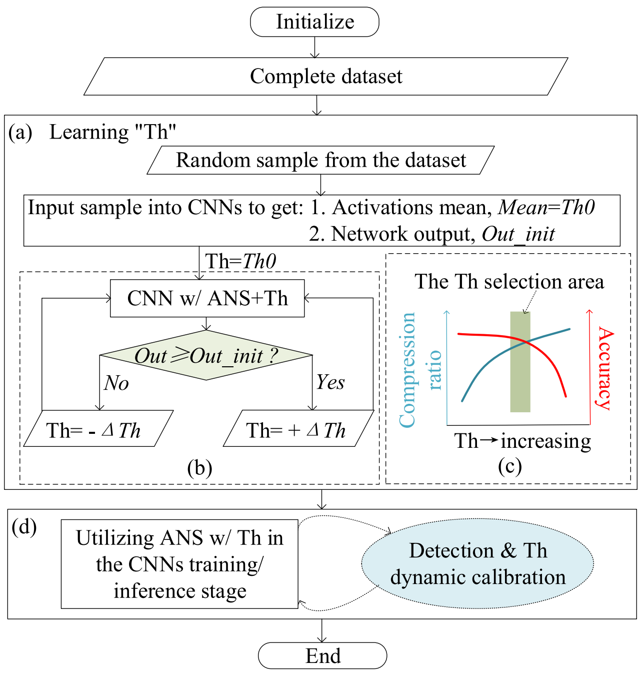

Initialize
Complete dataset
(a) Learning "Th"
Random sample from the dataset
Input sample into CNNs to get: 1. Activations mean, Mean=Th0
2. Network output, Out_init
Th=Th0
CNN w/ ANS+Th
Out≥Out_init ?
No
Yes
Th= - Δ Th
Th= + Δ Th
(b)
The Th selection area
Compression ratio
Accuracy
Th→increasing
(c)
(d)
Utilizing ANS w/ Th in the CNNs training/ inference stage
Detection & Th dynamic calibration
End

FIGURE 8. The flow diagram of learning for Th.

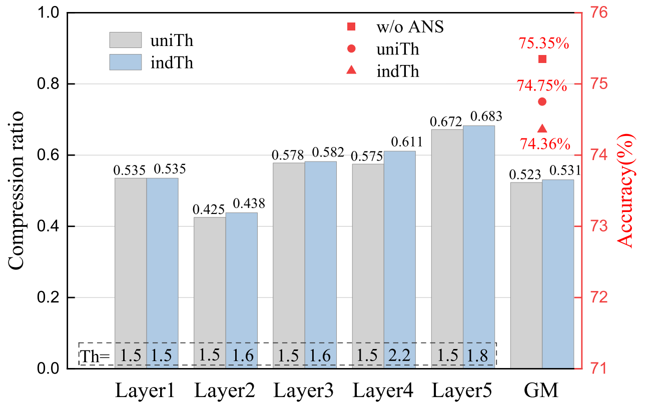


FIGURE 9. Compression ratio vs. layers of AlexNet under different Th and the impact on accuracy.

unacceptable accuracy loss even if the big compression ratio could be achieved.

1) FEASIBILITY FOR USING ONE CERTAIN TH ACROSS THE TOTAL NETWORK

First of all, we need to analyze whether it is feasible using specific Th value to judge local similarity of SB across the total network. Ten images are randomly selected as the input of AlexNet to obtain statistical results of output activations of each layer, including the maximum value, mean and standard deviation, as shown in Fig. 7. The activations of ten images have similar mean and standard deviation. The feature would be more obvious if there are BN layers in the network. Therefore, the distributions of activations, including mean value and standard deviation, are very close instead of being scattered in the following three levels: (a) among different images in the same one network; (b) among different layers in the same one network, (c) among activations in the same one layer. Therefore, it is feasible to use one certain Th to judge the local similarity of activations across the network.

2) LEARNING FOR TH

Here we propose to obtain appropriate Th by learning and dynamic calibration. Our experimental analysis shows that this method can realize a good trade-off between high compression ratio and less accuracy loss.

As shown in Fig. 8, two stages are included: the first stage is learning. A group of samples are randomly extracted from the dataset. And Th would finally be obtained after a series of iterations. The second stage is calibration, which includes: (a) Th is utilized in the training and inference. (b) The accuracy is detected in real time, (c) Th is calibrated dynamically.

Now we further discuss the learning stage in more detail. As shown in Fig. 8(a), a certain amount of image is randomly chosen from the data set as samples. Each sample is input into network to obtain the parameters, including *Mean* (the mean value of activations), and *Out_init* (the inference result). *Th0*, the initial threshold, is set to *Mean*. Then ANS with *Th0* is applied to CNN. And *Out*, the output result, is compared with *Out_init* so as to dynamically tune Th under the increment ΔTh, which is set to one tenth of *Th0* in our experiment. A smaller ΔTh enables to obtain a more accurate Th at the cost of more hardware resources. The iterations continue until the *Out* does not decrease compared with *Out_init*, and optimized Th is determined finally. The preferred region of Th is highlighted in green in Fig. 8(c), which is a schematic diagram of Fig. 6. According to different tolerance of accuracy loss for various tasks, Th has to be chosen in the green region in order to get as high compression ratio as possible.

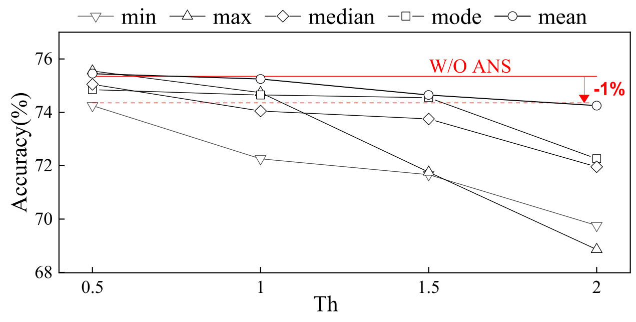


FIGURE 10. Accuracy vs. Th with different compression values.

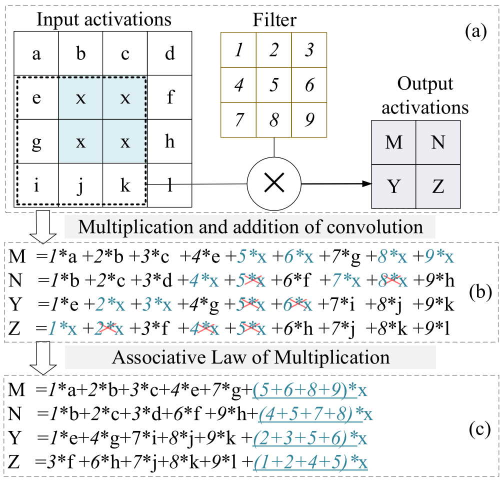

Input activations
Filter
Output activations
(a)
Multiplication and addition of convolution
M =1*a +2*b +3*c +4*e +5*x +6*x +7*g +8*x +9*x
N =1*b +2*c +3*d +4*x +5*x +6*f +7*x +8*x +9*h
Y =1*e +2*x +3*x +4*g +5*x +6*x +7*i +8*j +9*k
Z =1*x +2*x +3*f +4*x +5*x +6*h +7*j +8*k +9*l
(b)
Associative Law of Multiplication
M =1*a+2*b+3*c+4*e+7*g+(5+6+8+9)*x
N =1*b+2*c+3*d+6*f+9*h+(4+5+7+8)*x
Y =1*e+4*g+7*i+8*j+9*k +(2+3+5+6)*x
Z =3*f +6*h+7*j+8*k+9*l +(1+2+4+5)*x
(c)

FIGURE 11. The amount reduction of multiplication with ANS. (a) A schematic of Conv. (b) Conv formula. There are total 36 multiplications, of which 7 marked with a red cross are skipped or gated by reuse. (c) Conv formula with the associative law. 12 multiplications are skipped.

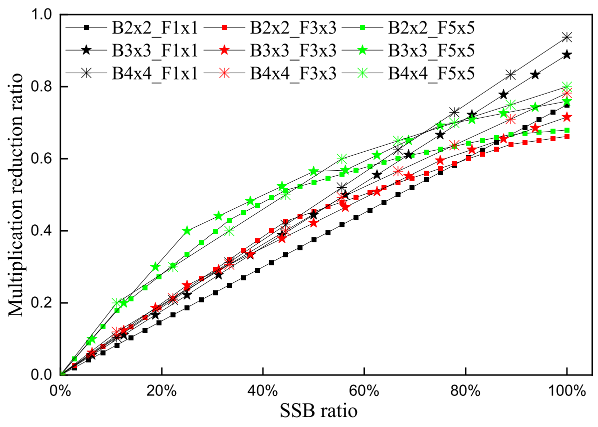


FIGURE 12. Multiplication reduction ratio vs. similarity ratio with 2 × 2, 3 × 3, 4 × 4 SB sizes and 1 × 1, 3 × 3, 5 × 5 filter sizes. In "B_*_F_*", "B" means the size of SB and "F" means the size of filter.

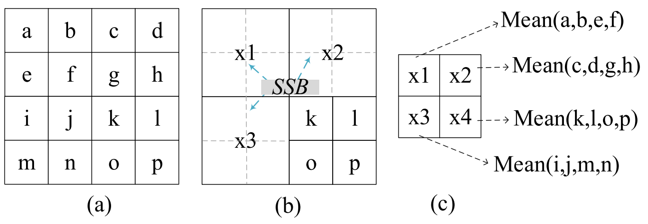


FIGURE 13. ANS vs. average pooling. (a) raw activation pattern, (b) pattern after ANS, (c) pattern after average pooing.

Table 1 shows the results learned from ten images respectively. The value of Th varies between −21.6% and +14.6%, and the corresponding accuracy varies between −0.5% and +0.1%. The values among images are close, which is consistent with the analysis in Section IV-C1. And we use the average value of Th learned from the ten images as the final Th.

During the second stage of calibration, Th is utilized to inference or training. Then the accuracy with ANS is detected on field regularly. Th could decrease in real time in case of accuracy decline.

It is noticeable that no label corresponding to dataset is required during both stages of learning and deployment. The outputs with and without ANS are compared directly to achieve the accuracy change.

3) HOW ABOUT FINE-GRAINED LEARNING FOR TH LAYER BY LAYER?

Actually, Th has different impact on accuracy of the rich and poor zero-value layers, as has been indicated in Fig. 6(c) and 6(f). Taking a 2 × 2 SB as an example, under the same accuracy drop of 1%, Th for layer1 is 1.8, while that for layer5 reaches 2.4. The latter can get bigger compression ratio. Then, will fine-grained learning Th layer by layer lead to better trade-offs between accuracy and compression ratio?

The compression ratio and accuracy between the entire AlexNet with a uniform Th (uniTh) and each layer with an independent Th (indTh) are compared. And the indTh is learned when the original accuracy is reduced by 1%. The experimental results are shown in Fig. 9. The GM of compression ratio for the indTh scenario only increased by 0.8% compared with that of the uniTh scenario, while the accuracy decreased by 0.39%. As a result, the learning of indTh, as an optimization problem with layer-dimensions complexity, consumes much more time and resources than that of uniTh across the network, but nearly no additional benefit of compression ratio is acquired.

D. SELECTION OF COMPRESSION VALUE

In ANS method, the data in SSB would be compressed into one single value "x" with an index "1". It is optimal that "x" selection can reflect the value distribution of the entire SSB, and minimizes the activation changes before and after ANS, so as to decrease accuracy loss due to compression.

We achieve the accuracy of AlexNet with ANS by using the minimum, maximum, mode, median and mean values as "x" respectively, as shown in Fig. 10. It is indicated that mean value of "x" has the minimum of accuracy loss. Now we discuss the reasons case by case. (a) It is obvious that the maximum and minimum lead to the greatest accuracy loss due

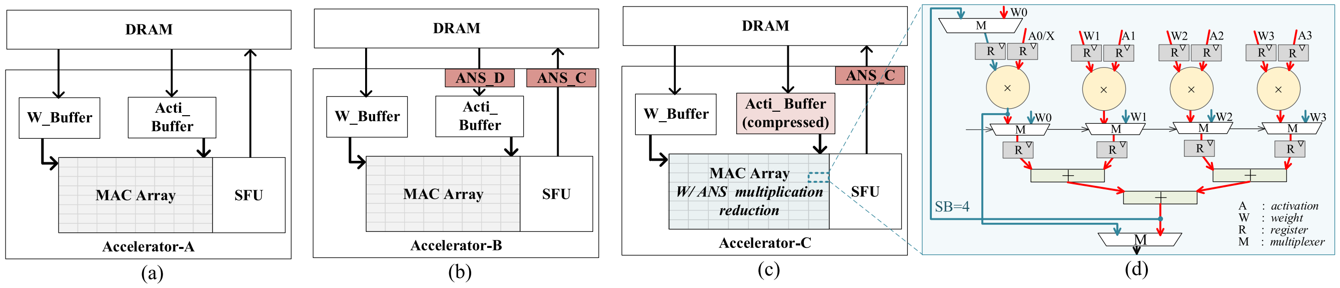


FIGURE 14. Three cases of accelerator: (a) Acc-A, w/o ANS. (b) Acc-B w/ ANS_C and ANS_D at the interface of the chip. (c) Acc-C w/ ANS_C and a special-designed MAC array to compute compressed data with multiplication reduction method. (d) MAC subarray implementation with the multiplication reduction method for 2 × 2 SB.

to numerical deviation. (b) When values in SB are clustered around the maximum or minimum, the median would be more on the side of the maximum or minimum. (c) Mode cannot represent the entire region when the data in SSB has wide distribution with less duplicate values. (d) In comparison, the mean value can better balance the distribution across entire local region. That is why the mean value of SSB is chosen as compression value "x".

E. REDUCTION OF MULTIPLICATIONS

ANS reduces not only the data size, but also the amount of multiplications. Fig. 11 illustrates an example of 4 × 4 activation pattern, in which a 2×2 SSB is included. A 2×2 output is obtained after the convolution by a 3 × 3 filter. As shown in Fig. 11(b), 36 multiplications are needed, in which there are many same operations due to duplicate "x" in SSB, e.g., four "5∗x". By reusing the result of the first multiplication, the latter same ones can be skipped or gated. In this case, 19% (7/36) multiplications is reduced.

When the associative law is further used for multiplications with "x" in the same formula, more amount of multiplication can be reduced, as shown in Fig. 11(c). 33% (12/36) of the multiplications are skipped under the 25% (4/16) SSB ratio. It is noticeable that neither approximation is taken for the amount reduction of multiplication, nor additional addition operation is introduced.

We further analyze the trend of the amount reduction of multiplications under three sizes of filter (1 × 1, 3 × 3, 5 × 5) and three sizes of SB (2 × 2, 3 × 3, 4 × 4). And Fig. 12 shows the ratio of multiplication reduction for 12 × 12 input activation pattern under different SSB ratio. It is assumed that there is no padding, and stride is 1. As the SSB ratio increases, the multiplication reduction ratio increases almost linearly. And for 1 × 1 filter, when SSB ratio reaches 100%, the multiplication reduction ratio can reach 93.8% at most. In addition, when the SSB ratio is less than 70%, the multiplication reduction ratio is positively correlated with filter size. When the SSB ratio is larger, the multiplication reduction ratio is positively correlated with SB size. This is because more data are compressed into one compressed value in a large SB. Thus, more multiplications in a formula are reduced.

F. COMPARISON OF ANS AND AVERAGE POOLING

Outwardly, average pooling and ANS both divide the activation pattern into windows and then compress the window into one single data, the mean of the window. Actually, the two have big differences. Pooling reduces the dimension of the activation pattern in width and height without distinction of local similarity. As illustrated in Fig. 13(a) and 13(c), the 4 × 4 activation pattern shrinks into 2 × 2 one after average pooling, 75% of the information is lost no matter if the data in the window are similar or not. As a result, only a minority of layers could be pooling ones in a network in order to ensure accuracy, e.g., ResNet152 has only 2 pooling layers among total 152 layers. In comparison, ANS only compresses the SSB while reserve all data in DSSB. When decompressing, the compression value in SSB is broadcast to keep the original shape of the activation pattern, as shown in Fig 13(b). These strategies ensure the least disturbance on accuracy under big compression ratio. That is why ANS can be applied to each layer in the network.

V. ANS APPLIED INTO DNN ACCELERATORS

In this section, we discuss the methodology how the ANS method can be applied into DNN accelerators. We proposed two ways of AI- device implementation: the first way does not disturb the internal structure of original accelerator at all, that ANS can be seamlessly attached to various accelerators, which is described in Part A. The other way is analyzed in Part B, of which the motivation is to harvest the benefits of multiplication reduction by ANS. Novel MAC sub-array with changed data path is further proposed.

A. SEAMLESSLY ATTACHED ANS MODULES

Acc-A in Fig. 14(a) is an abstraction of various DNN accelerators such as GPUs [27], TPUs [45], NVDLA [46] CIMs [15], [16] etc. Acc-A serves as baseline in the discussion of this section. Weights and activations are loaded from DRAM to the on-chip weight buffer(W_Buffer) and activation buffer (Acti_Buffer), respectively. Next, the calculations are carried out in MAC array based on vector multiplication data path, of which the results need to be processed by special function units (SFU) such as ReLU, Pooling, BN, etc.

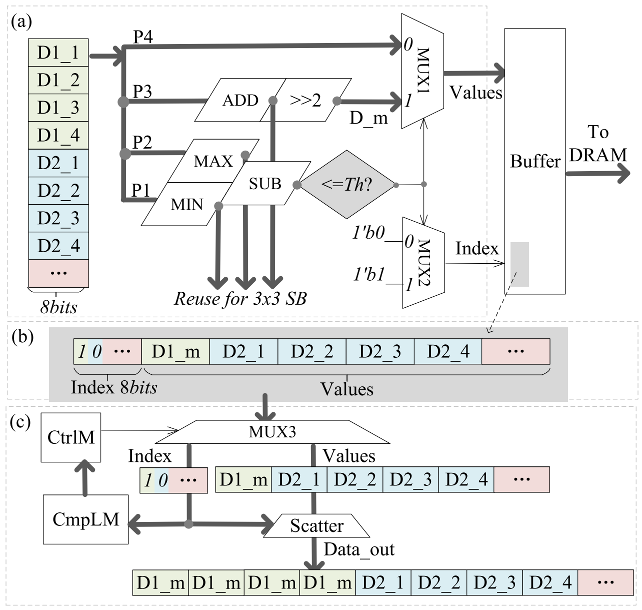


FIGURE 15. (a) ANS_C, (b) memory mapping of compressed activations and (c) ANS_D.

And the final output activations are stored in off-chip DRAM. Accelerators in this section are all designed with weight stationary dataflow, which makes activations, the most memory-accessed data, full utilization of ANS benefits [9].

On the base of Acc-A, we suggest the way of Acc-B, in which the ANS modules are seamlessly attached at the chip interface, including ANS _C and ANS _D. ANS _C works before data is moved into DRAM. ANS _D works after data is transferred into the chip. Acc-B neither changes the data path and data flow of Acc-A, nor disturb the internal structure of Acc-A, which is beneficial to seamlessly apply ANS modules into various accelerators.

Fig. 15 shows a hardware implementation of ANS_C and ANS_D. The signed FXP8 activation values are entered into ANS_C in the order of SB, and there are four paths to process these values. P1 and P2 search the maximum and minimum values in SB respectively, and then carry out subtraction. The results are compared with the Th to determine whether SB is similar, and the index is determined by MUX2, which is a multiplexer. When SB is DSSB, activation values on P4 are not compressed. When SB is SSB, data on path P3 will be added first and then divided, which is replaced by a shift operation. In addition, the intermediate results produced for 2×2 SB, including ADD, MAX and MIN, enable to be reused as SB is 3×3, meaning that one 3×3 SB ANS_C can be programmed to be two 2×2 SB ANS_C.

The memory mapping format of activations after compression is shown in Fig. 15(b). The bit width of index is consistent with the bit width of one value, indicating the number of SB compressed in a period. For FXP8 data, a period can achieve 8 bits of index, that is, 8 SBs are compressed.

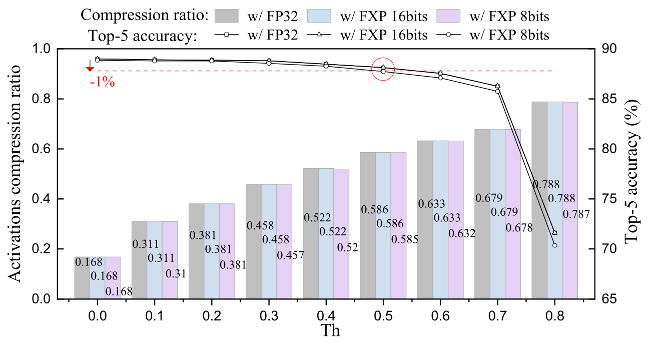


FIGURE 16. Compression ratio & top-5 accuracy for ResNet18 VS. Th, including FP32, FXP16, FXP8 @SB size 2×2.

ANS_D utilizes indexes to calculate the values length in a period in the Computing Length Module (CmpLM). Controller Module (CtrlM) retrieves values for one period according to the calculated data length. Finally, values are decompressed with the scatter operation.

B. DEEPLY COUPLED ANS MODULES

Different from Acc-B, Acc-C in Fig. 14(c) has ANS modules deeply coupled inside Acc-A with changed data path in the MAC array, which means directly processing the compressed format of activations and indexes. Thus, Acti_Buffer is used to store the compressed data, no need to utilize ANS_D to decompress. This way can harvest the benefits of multiplication reduction due to ANS, and thus significantly reduces computational energy consumption.

Take 2×2 SB as an example, we propose the data path design of MAC sub-array, as shown in Fig. 14(d). The design for 3×3 SB is just expanded on the basis of 2×2 SB. Additional multiplexers are used for configuration between SSB and DSSB. DSSB operation corresponds to red arrow flow, while the order of calculation between weight and activation is multiplication before addition. All multipliers and adders have to be used for the operation. SSB operation corresponds to blue arrow flow. Contrary to DSSB, the order of calculation for SSB is addition before multiplication. As depicted in Section IV-E, first add the weights and then multiply it with the compressed activations. It is notable that only one multiplier out of four is used due to multiplication reduction, which decreases energy consumption.

VI. EXPERIMENTS

In this section, the effects of ANS on compression ratio and accuracy for models are analyzed experimentally and compared with three mainstream compression methods. Further, the ANS hardware modules are implemented and the energy efficiency of the accelerators combined with ANS is evaluated.

A. ALGORITHM EVALUATION

1) EXPERIMENTAL SETUP

ResNet18 is taken as an example of popular CNN networks to discuss how ANS is deployed into practical inference and training. ANS is further applied into five widely used CNNs

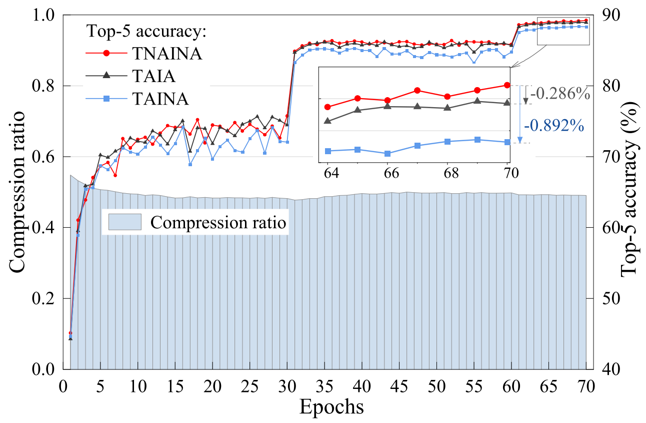


FIGURE 17. The results of accuracy and compression ratio for training ResNet18 VS. epochs.

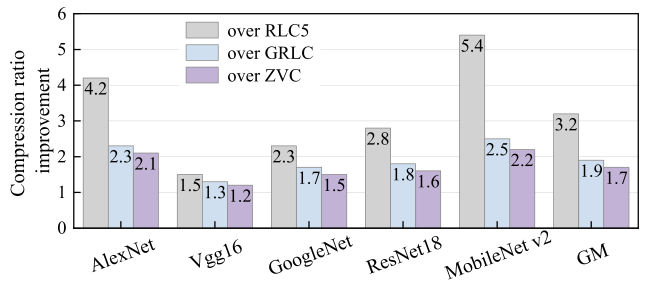


FIGURE 18. ANS compression ratio improvement over RLC5, GRLC and ZVC.

to evaluate the compression results. Finally, the comparison with the three mainstream compression methods, RLC5/GRLC/ZVC, is carried out. The ImageNet validation dataset with 50K labelled images is used for inference. The ImageNet training dataset with 1.34M images is applied to the training phase.

In our experiments, ANS modules are implemented by extending PyTorch framework, which can also be implemented by other popular frameworks, such as TensorFlow. The pre-training models for inference are all from the PyTorch website [48].

2) APPLIED TO INFERENCE WITH FXP QUANTIZATION

We linearly quantize the ResNet18 pre-training parameters and activations into FXP16 and FXP8, which is a usual strategy for inference [8], [18]. And the experimental results of quantized networks utilizing ANS are shown in Fig. 16. The accuracy of quantified FXP16 ResNet18 remains same compared with that of unquantified FP32 network, while that of the FXP8 decrease slightly. Set a 1% drop of accuracy as unacceptable, then the accuracy of the network quantized FXP8 decreases 0.384% at the worst case of Th 0.5, which is lower slightly than that of the unquantified network. The quantized and unquantified networks have same compression ratio, which increases linearly with the Th. There is still a

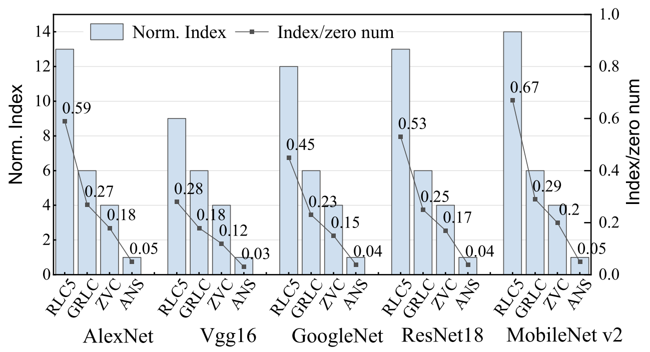


FIGURE 19. Indexes required by the four compression methods (normalized to ANS), and the ratio of index to ZVs of activation in the five network.

TABLE 2. Final Th learned from five networks.

Models name	Th	Accuracy(%) (Top1/Top5)	
		w/o ANS	w/ ANS
AlexNet	1.5	56.638/79.058	55.658/78.476
Vgg16	1.1	71.628/90.368	70.728/89.370
GoogleNet	0.4	69.744/89.544	68.848/89.01
ResNet18	0.5	69.644/88.984	68.672/88.352
MobileNet_v2	1.3	71.844/90.334	70.948/89.768

compression ratio of 0.168 at Th 0, thanks to some SSB with total 0 values.

3) APPLIED TO TRAINING

There are three general scenarios as training and deploying a model with or without ANS:

TAIA (Training with ANS and Inference with ANS): ANS is applied in both training and inference. This means that the inference accelerator includes real-time compression modules of ANS.

TAINA (Training with ANS and Inference with No ANS): ANS is applied in training, but not in inference. In this scenario, there is no ANS modules in the inference terminal.

TNAINA (Training with No ANS and Inference with No ANS): This is the uncompressed scenario as a baseline for comparison.

We utilize ANS in activations of the layer that requires training parameters, including CA and BA. The works in [20] can realize BA by recalculating in the backward propagation, so as to reduce activations storage, which is orthogonal to ANS, and is not considered in our experiment. The ResNet18 training results with ImageNet training dataset are shown in Fig. 17. Without increasing the number of iterations, ANS can only reduce the accuracy of ResNet18 by 0.286% in TAIA scenario and by 0.892% in TAINA scenario. In addition, the compression ratio with ANS has little fluctuation in the whole training period, and the GM of compression ratio is 49.4%, which proves once again that the Th learned by infinitesimal samples enables to achieve reliable compression ratio.

TABLE 3. Specification and unit energy consumption of each modules of the accelerator.

Modules	Size	Bits width	Unit energy*
MAC array	4096x FP16 MAC	to SFU: 1024b	0.34pJ/MAC
W_Buffer	512KB	to MAC array: 1024b	3.5pJ/16bits
Acti _Buffer	512KB	to MAC array: 1024b	3.5pJ/16bits
SFU	ReLU Pool BN etc.	to DRAM: 1024b	None
DRAM	HBM2 8GB	to Chip: 1024b	112.5pJ/16bits

**unit energy refer to [1].*

TABLE 4. Hardware simulation results of ANS_C and ANS_D.

28nm HKMG	ANS_C		ANS_D	
SB size	2x2	3x3	2x2	3x3
Area(um^2)	729	1689	110	237
Power(mW)	0.2381	0.5234	0.0884	0.1949
Latency(ns)	0.89	1.01	0.40	0.56

4) DEPLOYMENTS AND COMPARISON OF COMPRESSION RESULTS

Five networks, including AlexNet, Vgg16, GoogleNet, ResNet18 and MobileNet_v2, which cover a wide range of classic and modern CNNs with different parameter sizes, are selected as the benchmark to evaluate the ANS effects. Table 2 shows the Th of ANS for five networks when the Top1 and Top5 accuracy decreases by less than 1%. We compare compression ratio of ANS with three mainstream compression methods, including RLC5, GRLC, and ZVC.

Fig. 18 shows the compression ratio improvement of ANS over three compression methods. ANS achieves the highest compression ratio improvement of 5.4x over RLC5, 2.5 x over GRLC and 2.2x over ZVC for benchmarks MobileNet_v2, and achieves the GM of compression ratio improvement of 3.2x over RLC5, 1.9x over GRLC and 1.7x over ZVC. ANS shows high compression capability, because not only it can compress ZVs and locally similar NZVs, but also its indexes have low overhead. As shown in Fig. 19, indexes for ANS are reduced by 8x-13x compared with RLC, 5x compared with GRLC, and 3x compared with ZVC.

B. HARDWARE EVALUATION

1) EXPERIMENTAL SETUP

In this section, we evaluate the overburden of ANS realization, as well as impact on performance and energy efficiency.

As analyzed in Section V-A, Acc-B does not disturb the internal structure of accelerator at all, just seamlessly attach ANS_C and ANS_D at the interface. In order to evaluate the over-burden from the attached modules, ANS_C and ANS_D are implemented by using Verilog RTL code and synthesized in 28nm HKMG technology node with Design Compiler and IC compiler tool chain of Synopsys. Based on this, we can obtain the area, power and delay of ANS_C and ANS_D within one single ANS module.

TABLE 5. DOP of ANS modules.

Normal to 28nm*		Ascend310	FSD HW3	TPU v1
Area(mm^2)		572*	1040*	332
Power(W)		20*	179*	75
MAC numbers		16384	18432	65536
Cycles for Layer1		6409	5697	1602
Out data per cycle		43	51	181
ANS cost	DOP (2x2/3x3)	11/5	13/6	46/21
	Area(mm^2)	0.0096	0.0116	0.0404
	Power(W)	0.004	0.005	0.015

**scaling method refers to [49].*

In practical DNN accelerators, MAC arrays would operate in parallel and output a batch of values in one cycle. Correspondingly, multiple ANS modules, instead of single one, should be combined into the Acc-B in order to process these values in parallel in the next cycle. Here the number of ANS modules is defined as the degree of parallelism (DOP). In order to evaluate the area/power over-burden of multiple ANS modules in real accelerators, we use AlexNet layer 1 as benchmark and quantify the ANS DOP in three commercial accelerators, Ascend310 [50], FSD HW3 [51] and TPU v1 [18], next the area/power of multiple ANS modules could be achieved.

We set up a cycle-level simulation platform based on NVDLA [46], an open source accelerator, for the comparison of energy efficiency among Acc-A, Acc-B and Acc-C (detailed in Section V).

t is obvious from Section V-A that the difference of energy consumption between Acc_A and Acc_B come from the attached ANS_C and ANS_D modules, and is independent of the accelerator type and structure. So the trend based on our NVDLA platform would be consistent with other type of accelerator.

Next, according to Section V-B, the difference of energy consumption between Acc_B and Acc_C comes from Acti_Buffer and MAC array, which is independent of all other modules such as W_Buffer, SFU etc. Thus, only if Acc_B and Acc_C are realized based on the same accelerator structure, that means all same except Acti_Buffer and MAC array, the energy difference due to ANS can be achieved through comparison between Acc_B and Acc_C. Thus, the trends based on our NVDLA platform are representative.

In our experiment, the energy consumption of each module (Energy_Module) can be obtained as:

$$\text{Energy\_Module} = \text{Unit\_Energy} \times \text{Num\_Operations} \quad (5)$$

where Unit Energy is listed in the last column of Table 3, and Num_Operations refers to the number of MAC computations or memory data accesses, which can be obtained through our simulation platform when running the benchmark network.

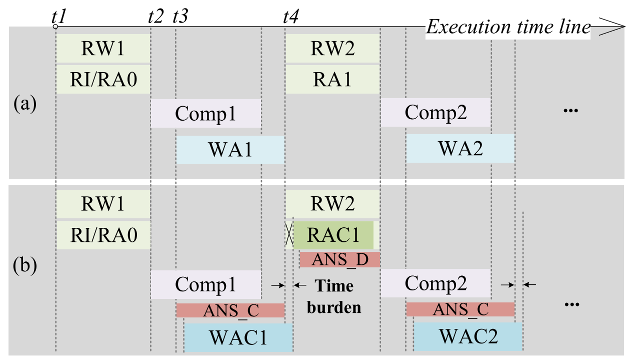


FIGURE 20. The timing of accelerator (a) w/o ANS and (b) embedded transparently with ANS modules.

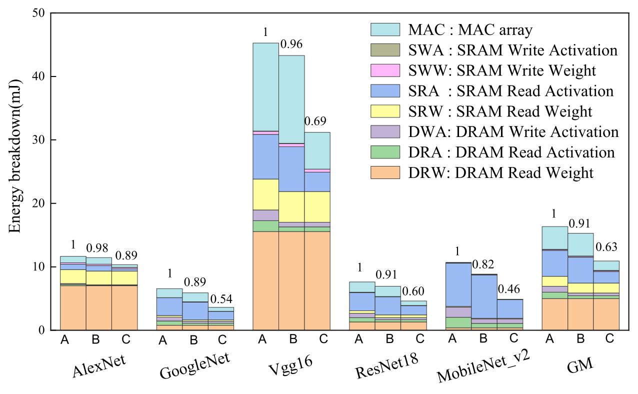


FIGURE 21. Energy breakdown of the three accelerators at batch size 1.

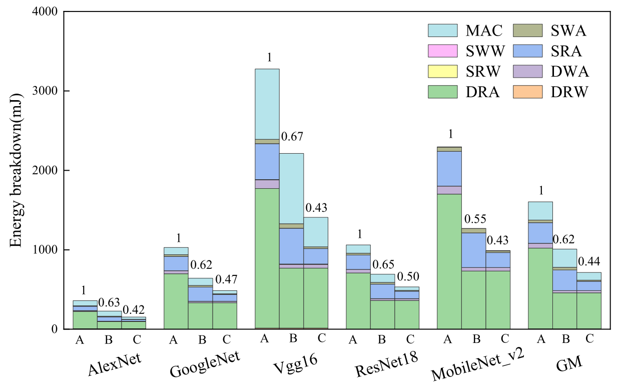


FIGURE 22. Energy breakdown of the three accelerators at batch size 64.

Table 3 lists the detail information of each module for energy evaluation in our experiment. 512 KB Acti_Buffer size is selected to store the activation. Correspondingly, the array consisting of 4096 MACs is used to realize convolution operations of different tile sizes. Batch size 1 and 64 are selected, which represents two different scenarios. For batch size of 1, 512KB of Acti_Buffer can store almost activations of all layers. For batch size of 64, on-chip memory is far less than data requirement.

2) OVERBURDEN OF ANS MODULES

The area, power and latency are shown in Table 4. A combination of ANS_C and ANS_D (configured for one 3×3 SB or two 2×2 SBs) has an area of 1926 um^2 and a power consumption of 0.72 mW. The critical paths of ANS_C are P1 and P2 in Fig. 15(a), and the path delay is 1.01 ns.

We further evaluate the DOP of ANS modules for three commercial accelerators, Ascend310, FSD HW3 and TPU v1. The area, power and MAC number of the three accelerator chips are listed in Table 5 [18], [50], [51]. Here the utilization of MAC array is assumed to be 100%.

The number of cycles is equal to multiplications divided by the number of MAC. AlexNet layer 1 has a total of 105 M multiplications. Thus, three accelerators respectively require 6409, 5697 and 1602 cycles to compute these multiplications when MACs are fully utilized, as shown in Table 5.

AlexNet layer 1 has 283 K output activation values. If these values are output in each computing cycle on average, that means output for each cycle is equal to activations divided by the cycles. Thus, the three accelerators will respectively output 43, 51 and 181 activation values in each cycle, which is also the data that the ANS modules needs to compress in parallel in one cycle.

For 2×2 SB, the three accelerators will require 11(43/4), 13(51/4), and 46(181/4) ANS modules of DOP respectively. For 3×3 SB, the DOP are 5(43/9), 6(51/9), 21(181/9). Then the area/power of multiple ANS modules are equal to the product of DOP and area/power of one single ANS. Our calculations indicate that the area/power overburden of multiple ANS modules in the three accelerators are all less than 0.01%, which could be ignorable in comparison with the total consumption of accelerators.

3) IMPACT OF ANS ON PERFORMANCE

Our evaluation shows that the timing delay could be eliminated through transparent implementation strategy of ANS modules. The timing schematic is shown in Fig. 20.

For original situation without ANS (see Fig. 20(a)), a Conv for a layer starts from reading weights (RW) and input image (RI) or activations (RA) since "t1". Next, convolution computation (Comp1) by PEs starts since "t2". At "t3", the output activations begin to be written back (WA), which lasts until "t4". Then the convolution for next layer is carried out by repeating above operations.

For situation with ANS (see Fig. 20(b)), the ANS_C starts to work at "t3" that the output activations for first SSB judgement come out, and then the compressed activations begin to write back (WAC1). Due to the delay of WAC1 after ANS_C, a tiny time burden is generated. RAC1, the time span for reading the compressed activations of the next layer, is shorter than RA1 thanks to the smaller activation data size. This can compensate the time burdens generated by ANS_C and ANS_D. So, ANS modules could cause no additional time delay under reasonable orchestration of execution flow.

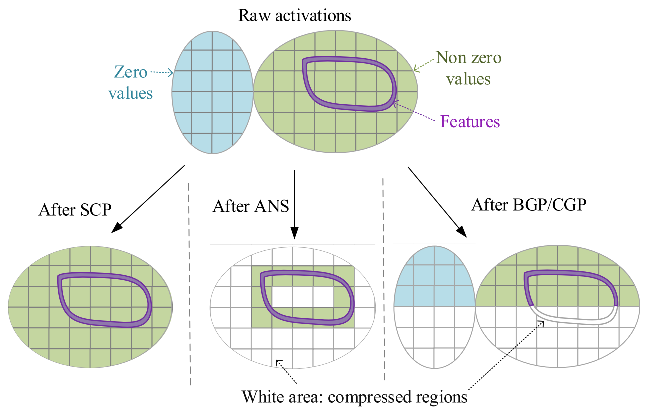


FIGURE 23. **ANS achieves high precision and compression ratio by preserving feature regions and compressing non-feature regions. And a comparison of ANS with SCP, BGP and CGP.**

4) ENERGY EFFICIENCY

For five wide-used networks, the energy consumption of Acc-A, Acc-B and Acc-C (introduced in Section V), including off-chip memory, on-chip memory and the computing unit, is statistically analyzed.

The results at batch size of 1 is shown in Fig. 21. The Acti_Buffer can store activations for most of layers in five networks, and activations are only read from DRAM once and then reused repeatedly on the chip. Therefore, SRA consumes most of the energy for on-chip storage, while DRW consumes most of the energy for off-chip storage. The GM of energy consumption of Acc-B and Acc-C is reduced by 9% and 37% respectively in comparison to Acc-A. The energy consumption advantage of the Acc-C over Acc-B comes from the amount reduction of Acti_Buffer read and multiplication, which corresponds to consumption parts of SRA and MAC. For the network, the greater the proportion of activations in the total (see Fig. 1 for details), the more energy consumption reduces from Acc-A to Acc-C, because ANS compresses more data and transfers less data. Taking MobileNet_v2 and AlexNet for examples, the former activations account for 81%, thereby reducing energy consumption by 54%, while the latter activations account for only 3% of the total data, reducing energy consumption by 11%.

Fig. 22 shows the results at the batch size of 64. Since the 512 KB Acti_Buffer cannot store the 64x activations, the activations have to be read more than once from off-chip DRAM. Therefore, DRA dominates the energy consumption of each network. So the benefits from the reduction of data size by ANS are significantly demonstrated. As a result, the GM of energy consumption of Acc-B is reduced by 38% in comparison to Acc-A due to DRA reduction. Similar to batch size of 1, the energy consumption reduction of Acc-C mainly depends on MAC and SRA, and the final GM of energy consumption is reduced by 56% in comparison to Acc-A.

VII. DISCUSSION

ANS achieves high compression ratio with less accuracy loss, which can be attributed to two root causes: 1st one, NZVs dominate the activations, while there are large percentage of SSB in region of NZVs. ANS handles both regions of NZVs and ZVs, and almost all similar data are compressed. That is why ANS has high compression ratio. This is very different from SCP, which can only handle the ZVs. 2nd one, features only exist within dis-similar NZVs. That is because there is numerical jump at location of features, such as edge/textures/pose, which is convoluted by filter and transferred among activations layer by layer [52]. ANS retains all values in DSSB, the numerical jump of region. In other words, ANS doesn't sacrifice any tiny feature values while ignore all non-feature values. That is why ANS has both less accuracy and high compression ratio. This is very different from BGP/CGP, in which some feature values are cut off indiscriminately, leading to accuracy loss and additional retrain penalty. Fig. 23 evidently depicts the above two points.

VIII. CONCLUSION

We propose ANS compressing method, which can significantly reduce data size of activation/image, and energy consumption. ANS can be combined with low precision data in inference without accuracy loss. We train ResNet18 with ANS in two TAIA and TAINA scenarios, and the accuracy loss was less than 0.286% and 0.892%, respectively, while achieving a compression ratio of 49.4%. The amount of multiplication reduces greatly as the ANS are deeply coupled in accelerator. Comparing to three state-of-art compressed methods, ANS provides compression ratio improvement of 3.2x over RLC5, 1.9x over GRLC and 1.7x over ZVC. We further evaluated the energy consumption of ANS combined into the accelerator system. (a) For accelerator with ANS modules attached at the interface of the chip, energy consumption is reduced by 38% averagely among five popular models at batch size 64, which comes from decrease of data size. (b) For accelerator with ANS modules deeply coupled with changed data-path MAC array, 56% reduction averagely is achieved in energy consumption, of which the additional benefits come from the decrease of multiplications.

REFERENCES

[1] N. P. Jouppi, D. H. Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. Patterson, "Ten lessons from three generations shaped Google's TPUv4i: Industrial product," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2021, pp. 1–14.

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

[3] 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.

[4] M. Paulius, "Fundamentals of scaling out DL training," in *Proc. IEEE Hot Chips 32 Symp.*, Aug. 2020, pp. 1–53.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.

[6] K.-H.-L. Loh, "Fertilizing AIoT from roots to leaves," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 15–21.

[7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[8] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[9] V. Rangharajan, "Basic design approaches to accelerating deep neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 1–93.

[10] N. Samuel, "Architecting chiplet solutions for high volume products," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2021, pp. 1–42.

[11] T.-Y.-J. Chang, Y.-H. Chen, W.-M. Chan, H. Cheng, P.-S. Wang, Y. Lin, H. Fujiwara, R. Lee, H.-J. Liao, P.-W. Wang, G. Yeap, and Q. Li, "A 5-nm 135-mb SRAM in EUV and high-mobility channel FinFET technology with metal coupling and charge-sharing write-assist circuitry schemes for high-density and low-V_{MIN} applications," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 179–187, Jan. 2021.

[12] S. Lee, S.-H. Kang, J. Lee, H. Kim, E. Lee, S. Seo, H. Yoon, S. Lee, K. Lim, H. Shin, J. Kim, O. Seongil, A. Iyer, D. Wang, K. Sohn, and N. S. Kim, "Hardware architecture and software stack for PIM based on commercial DRAM technology: Industrial product," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2021, pp. 43–56.

[13] S. Lee et al., "A 1y nm 1.25 V 8 Gb, 16 Gb/s/pin GDDR6-based accelerator-in-memory supporting 1TFLOPS MAC operation and various activation functions for deep-learning applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 1–3.

[14] G. Van der Plas and E. Beyne, "Design and technology solutions for 3D integrated high performance systems," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2021, pp. 1–2.

[15] R.-L. Bruce, "Designing materials systems and algorithms for analog computing," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2020, pp. 1–40.

[16] H. Kim, T. Yoo, T. T.-H. Kim, and B. Kim, "Colonnade: A reconfigurable SRAM-based digital bit-serial compute-in-memory macro for processing neural networks," *IEEE J. Solid-State Circuits*, vol. 56, no. 7, pp. 2221–2233, Jul. 2021.

[17] H. Fujiwara, H. Mori, W.-C. Zhao, M.-C. Chuang, R. Naous, C.-K. Chuang, T. Hashizume, D. Sun, C.-F. Lee, K. Akarvardar, S. Adham, T.-L. Chou, M. E. Sinangil, Y. Wang, Y.-D. Chih, Y.-H. Chen, H.-J. Liao, and T.-Y.-J. Chang, "A 5-nm 254-TOPS/W 221-TOPS/mm^2 fully-digital computing-in-memory macro supporting wide-range dynamic-voltage-frequency scaling and simultaneous MAC and write operations," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2022, pp. 1–3.

[18] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, and A. Borchers, "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.

[19] P. Jain, A. Jain, A. Nrusimha, A. Gholami, P. Abbeel, J. Gonzalez, K. Keutzer, and I. Stoica, "Checkmate: Breaking the memory wall with optimal tensor rematerialization," *Proc. Mach. Learn. Syst.*, vol. 2, pp. 497–511, Mar. 2020.

[20] A. Chakrabarti and B. Moseley, "Backprop with approximate activations for memory-efficient network training," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2429–2438.

[21] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," 2017, *arXiv:1705.08922*.

[22] J.-H. Luo and J. Wu, "An entropy-based pruning method for CNN compression," 2017, *arXiv:1706.05791*.

[23] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*.

[24] M. Rhu, M. O'Connor, N. Chatterjee, J. Pool, Y. Kwon, and S. W. Keckler, "Compressing DMA engine: Leveraging activation sparsity for training deep neural networks," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2018, pp. 78–91.

[25] Y. Park, Y. Kang, S. Kim, E. Kwon, and S. Kang, "GRLC: Grid-based run-length compression for energy-efficient CNN accelerator," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2020, pp. 91–96.

[26] A. Jain, A. Phanishayee, J. Mars, L. Tang, and G. Pekhimenko, "Gist: Efficient data encoding for deep neural network training," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 776–789.

[27] J. Choquette and W. Gandhi, "NVIDIA A100 GPU: Performance & innovation for GPU computing," in *Proc. IEEE Hot Chips 32 Symp. (HCS)*, Aug. 2020, pp. 1–43.

[28] D. T. Vooturi, D. Mudigere, and S. Avancha, "Hierarchical block sparse neural networks," 2018, *arXiv:1808.03420*.

[29] S. Chou, F. Kjolstad, and S. Amarasinghe, "Format abstraction for sparse tensor algebra compilers," *Proc. ACM Program. Lang.*, vol. 2, pp. 1–30, Oct. 2018.

[30] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 243–254, 2016.

[31] V. Akhlaghi, A. Yazdanbakhsh, K. Samadi, R. K. Gupta, and H. Esmaeilzadeh, "SnaPEA: Predictive early activation for reducing computation in deep convolutional neural networks," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 662–673.

[32] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An accelerator for compressed-sparse convolutional neural networks," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, pp. 27–40, 2017.

[33] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[34] A. R. Alameldeen and D. A. Wood, "Adaptive cache compression for high-performance processors," in *Proc. 31st Annu. Int. Symp. Comput. Archit. (ISCA)*, 2004, pp. 212–223.

[35] Z. Song, B. Fu, F. Wu, Z. Jiang, L. Jiang, N. Jing, and X. Liang, "DRQ: Dynamic region-based quantization for deep neural network acceleration," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit. (ISCA)*, May 2020, pp. 1010–1021.

[36] K. Huang, S. Chen, B. Li, L. Claesen, H. Yao, J. Chen, X. Jiang, Z. Liu, and D. Xiong, "Structured precision skipping: Accelerating convolutional neural networks with budget-aware dynamic precision selection," *J. Syst. Archit.*, vol. 124, Mar. 2022, Art. no. 102403.

[37] J. S. Miguel, J. Albericio, A. Moshovos, and N. E. Jerger, "Doppelgänger: A cache for approximate computing," in *Proc. ACM/IEEE 49th Int. Symp. Microarchitecture (MICRO)*, Dec. 2015, pp. 50–61.

[38] A. Fuchs and D. Wentzlaff, "Scaling datacenter accelerators with compute-reuse architectures," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 353–366.

[39] M. Riera, J.-M. Arnau, and A. González, "Computation reuse in DNNs by exploiting input similarity," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 57–68.

[40] M. Buckler, P. Bedoukian, S. Jayasuriya, and A. Sampson, "EVA^2: Exploiting temporal redundancy in live computer vision," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 533–546.

[41] Z. Yuan, Y. Yang, J. Yue, R. Liu, X. Feng, Z. Lin, X. Wu, X. Li, H. Yang, and Y. Liu, "A 65 nm 24.7 μJ/frame 12.3 mW activation-similarity-aware convolutional neural network video processor using hybrid precision, inter-frame data reuse and mixed-bit-width difference-frame data codec," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 232–234.

[42] N. M. Cicek, L. Ning, O. Ozturk, and X. Shen, "General reuse-centric CNN accelerator," *IEEE Trans. Comput.*, vol. 71, no. 4, pp. 880–891, Apr. 2022.

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

[44] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[45] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson, "A domain-specific supercomputer for training deep neural networks," *Commun. ACM*, vol. 63, no. 7, pp. 67–78, Jun. 2020.

[46] (2017). *NVDLA Deep Learning Accelerator*. [Online]. Available: http://nvdla.org/

[47] B. Zimmer, R. Venkatesan, Y. S. Shao, J. Clemons, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. S. Emer, C. T. Gray, S. W. Keckler, and B. Khailany, "A 0.11 pJ/op, 0.32–128 TOPS, scalable multi-chip-module-based deep neural network accelerator with ground-reference signaling in 16 nm," in *Proc. Symp. VLSI Circuits*, Jun. 2019, pp. C300–C301.

[48] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[49] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7 nm," *Integration*, vol. 58, pp. 74–81, Jun. 2017.

[50] H. Liao, J. Tu, J. Xia, and X. Zhou, "DaVinci: A scalable architecture for neural network computing," in *Proc. IEEE Hot Chips 31 Symp. (HCS)*, Aug. 2019, pp. 1–44.

[51] P. Bannon, G. Venkataramanan, D. D. Sarma, and E. Talpes, "Computer and redundancy solution for the full self-driving computer," in *Proc. IEEE Hot Chips 31 Symp. (HCS)*, Aug. 2019, pp. 1–22.

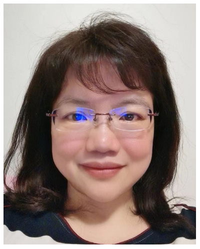
[52] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 818–833.

WANG WANG (Graduate Student Member, IEEE) received the B.S. degree from the School of Microelectronics, Xidian University, Xi'an, China, in 2014, and the M.S. degree from the ASIC and System State Key Laboratory, School of Microelectronics, Fudan University, Shanghai, China, in 2018, where he is currently pursuing the Ph.D. degree. His current research interests include deep-learning accelerator architecture and system design.

XIN ZHONG received the M.S. degree in material engineering from Wuhan University of Technology, in 2018. He is currently pursuing the Ph.D. degree with the ASIC and System State Key Laboratory, School of Microelectronics, Fudan University, Shanghai, China. His current research interests include non-volatile memories, big data, machine learning, and FPGA design.

MANNI LI received the B.S. degree from the School of Microelectronics, Xidian University, Xi'an, China, in 2016. She is currently pursuing the M.S. degree with the ASIC and System State Key Laboratory, School of Microelectronics, Fudan University, Shanghai, China, in 2020. Her current research interests include AI memory architecture and DL accelerator design.

ZIXU LI received the B.S. degree from the School of Microelectronics, Xidian University, Xi'an, China, in 2017. She is currently pursuing the Ph.D. degree with the ASIC and System State Key Laboratory, School of Microelectronics, Fudan University, Shanghai, China, in 2021. Her research interest includes near-memory-processing-based DRAM systems.



YINYIN LIN (Member, IEEE) received the M.S. degree in microelectronics and solid state electronics from Xi'dian University, in 1995, and the Ph.D. degree in microelectronics and solid state electronics from Xi'an Jiaotong University, in 1998. She has been with Fudan University, since 1999, where she is currently a Professor with the School of Microelectronics. Over the past years, she has contributed over 90 papers in journals and conferences. She holds over 50 patents. Her recent research interests include the co-optimization of memory architecture and chip design for AI and low-power systems under PVTA variation.