Approach to Improve the Performance Using Bit-level Sparsity in Neural Networks

Yesung Kang, Eunji Kwon, Seunggyu Lee, Younghoon Byun, Youngjoo Lee, and Seokhyeong Kang*
EE Department, POSTECH, South Korea
*shkang@postech.ac.kr

***Abstract*—This paper presents a convolutional neural network (CNN) accelerator that can skip zero weights and handle outliers, which are few but have a significant impact on the accuracy of CNNs, to achieve speedup and increase the energy efficiency of CNN. We propose an offline weight-scheduling algorithm which can skip zero weights and combine two non-outlier weights simultaneously using bit-level sparsity of CNNs. We use a reconfigurable multiplier-and-accumulator (MAC) unit for two purposes; usually used to compute combined two non-outliers and sometimes to compute outliers. We further improve the speedup of our accelerator by clipping some of the outliers with negligible accuracy loss. Compared to DaDianNao [7] and Bit-Tactical [16] architectures, our CNN accelerator can improve the speed by 3.34 and 2.31 times higher and reduce energy consumption by 29.3% and 30.2%, respectively.**

I. Introduction

Convolutional neural networks (CNNs) have achieved higher accuracy than rule-based algorithms on image classification, object detection, and segmentation applications [1]. However, this accuracy improvement is accompanied by a large increase in computational complexity. The increased computational complexity impedes deployment of CNNs on mobile devices or embedded hardware that has a limited power budget.

A viable approach to alleviate the increased computational complexity is to reduce the precision of weights and activations of CNNs. Although the precision and the accuracy of CNNs are usually trade-off, CNNs can be quantized up to 8-bit with negligible accuracy loss [2]–[6]. The quantization of CNNs can lower the memory bandwidth, the on-chip storage requirements, the power consumption, and the computation time. Precision-scalable accelerators [8]–[10] can increase the energy-efficiency of the quantized CNNs by adjusting the computational precision to the requirements, which vary across CNNs and CNN layers.

However, the accelerators still perform numerous ineffectual computations. When we quantize VGG-16 [11] to 8-bit fixed-point precision, a large number of weights are zero as shown in Figure 1. We define non-zero weights that can be expressed with 4-bit precision as non-outliers, and the other non-zero weights as outliers. On average, 34.2% of weights have zero magnitude, although the proportion varies among layers. Multiplications of zero weights do not affect output activations, so these computations waste computing resources.

To reduce waste of computing resources from ineffectual computations, CNN accelerators that can skip zero weights have been proposed [12]–[16]. However, these zero-skip accelerators require high-precision multipliers to compute outlier weights, and waste resources for non-outlier weights. Considering that the number of non-outlier weights is also comparable to the number of zero weights, energy consumed by the computation of non-outlier weights is not negligible. Using of 4×8 multipliers instead of 8×8 multipliers can roughly halve the energy used to compute non-outlier weights. The

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A5A1027055).

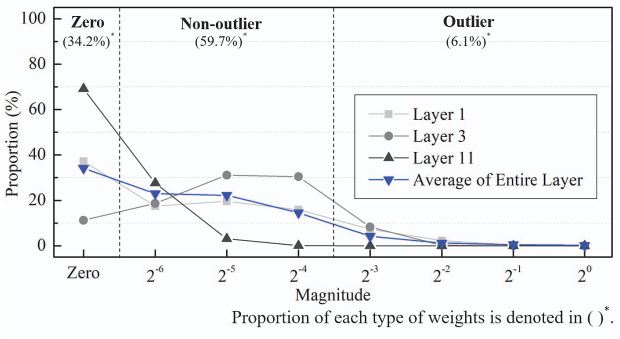


Fig. 1: The proportions of zero, non-outlier and outlier weights in layer 1, 3, 11 and the average of the 8-bit quantized VGG-16 [11].

data proportions in Figure 1 suggest that low-precision multipliers for non-outlier weights can theoretically reduce energy consumption by 29.9%. Accordingly, outlier-aware accelerators have been proposed, which can reduce energy consumption wasted for non-outlier weights [17]–[19]. However, the outlier-aware accelerators require additional logics to index outliers, which cause large power and area overheads.

In this paper, we propose a weight scheduling algorithm that skips both zero weights and activations, and combines two non-outlier weights. Additionally, we propose energy-efficient CNN accelerators using a reconfigurable multiplier-and-accumulator (MAC) unit which can be reconstructed according to runtime for two purposes; one for computing combined two non-outliers simultaneously and the other for an outlier. Therefore, we can reduce the energy consumption in computing both the non-outlier and zero weights with a small area and power overhead.

The main contributions of our work are:

- To reduce the computational cost of non-outlier weights and zero weights, we propose an approach of the reconfigurable MAC unit.
- To minimize hardware burden for precision scaling, we propose a weight-scheduling algorithm that removes zero weights and combines pairs of non-outlier weights.
- For the case study, we investigate the trade-off between additional speedup and the accuracy loss when some outliers are converted into non-outliers.
- To exploit the sparsity of activations, we propose an online scheduler which is compatible with the weight-scheduling algorithm.

The rest of the paper is organized as follows: Section II introduces previous works on zero-skip, and the accelerator specialized for handling outliers. Section III presents the proposed weight-scheduling algorithm, evaluation of speedup of the algorithm, and a case study to investigate the trade-off between additional speedup and accuracy drop by clipping some outliers. Section IV presents the structure of the proposed CNN accelerators, zero skip method of activations,

TABLE I: Comparison of CNN accelerators; DPRed [17], OLAccel [18], Overwrite Quantization [19], Bit-Tactical [16] and a proposed accelerator.

	DPRed [17]	**OLAccel [18]**	**Overwrite Quantization [19]**	**Bit-Tactical [16]**	**Proposed**
Software Processing	Dynamic precision reduction	Outlier-aware quantization	Overwrite quantization, channel reordering	Weight scheduling	Weight scheduling, outlier clipping
Zero Skip	X	Activation	X	Weight/Activation	Weight/Activation
Precision-Scaling	Inter-layer	Intra-layer	Intra-layer	X	Intra-layer
MAC	Serial inner-product unit	Sparse outlier PE, Dense PE	Specialized MAC for overwrite quantization	Conventional MAC	Reconfigurable MAC
Hardware complexity	Additional registers	Additional dedicated MAC for outliers	Additional muxing, shifters	Additional muxing, shifters	Additional muxing, shifters

and an evaluation of the energy consumption of the proposed CNN accelerators. Section V concludes the paper.

II. Previous Works

A. Zero-skip accelerators

Weights and activations in CNNs include many zero values. The ineffectual computations from zero values account for a significant portion of total computations. Accordingly, many researchers have proposed various algorithms to skip unnecessary computations.

Zhang et al. [12] have proposed an accelerator which skips zero weights by using indices of weights. With the indices, the accelerator finds the locations of effectual weights, then uses an indexing module to deliver appropriate activations to each processing element (PE). Similarly, Albericio et al. [13] use the indices of activations to skip zero in activations. Although these approaches can improve the performance, it requires additional computations to encode indices. Also, if the sparsity of data assigned to each PE is different, it causes load imbalance in which some PEs are working hard, but others are not. Kim et al. [14] have proposed an accelerator that can relieve load imbalance. However, this method requires precise investigations and allocations of weights and activations across channels.

Parashar et al. [15] have proposed zero-skip architectures which eliminates delivery of zero weights and activations. The method fetches only non-zero weights and activations from the storage and distributes the computed results to the corresponding accumulators according to the output index. For the delivery of the products of weights and activations, the method uses a crossbar. The crossbar induces large areas and energy overheads. To minimize the overheads of the crossbar, Lascorz et al. [16] have proposed an accelerator which constrains the movement of weights using a software scheduling algorithm. The method can efficiently alleviate load imbalance between PEs. However, the method cannot fully exploit the bit-level sparsity of weights; the detailed comparison will be described in Section III.

B. Outlier-aware accelerators

Most weights can be expressed as low bit-width data. However, the existence of few outliers makes it hard to quantize the weights. The outliers have a significant effect on the accuracy of overall calculations, to process outliers, several CNN accelerators handling outliers have been proposed.

Delmas et al. [17] group weights and activations, then adjust the precision of each group to increase execution speed and energy efficiency. Kim et al. [18] employed dedicated multipliers in an accelerator to process outliers. The method computes low-precision multiplications for a majority of data and computes outlier data by dedicated high-precision multipliers. However, logics to transfer outlier data induce large power and area overheads. Besides, since the method should minimize the number of multipliers dedicated to outlier data, an increase in the proportion of outlier data significantly degrades the performance of CNN accelerators. The method requires elaborate management for outliers which are amounting to 3% of total data. Zhao et al. [19] proposed to overwrite quantization; outliers can overwrite unimportant values opportunistically if the data are non-critical. However, overwriting can decreases accuracy. To relieve this problem, additional software processing such as channel reordering is required to lessen the accuracy drop.

Table I shows a comparison among other outlier-aware accelerators, Bit-Tactical architecture [16]–[19], and our proposed approach. Each accelerator of previous approaches has limitations such as accuracy drop, separate software processing, hardware complexity, area overhead, and so on. The proposed accelerator does not need to make additional MAC deal with outliers. Additionally, our accelerator supports intra-layer precision scalability which means we do not have to take any extra hard work to find the optimal precision per layer. Our accelerator is designed to skip both zero weights and activations and to handle outliers simultaneously.

III. Proposed Weight-scheduling Approach

A. Proposed Weight-scheduling Algorithm

To exploit the sparsity, we rearrange weight in an offline manner. Figure 2 (a) shows the proposed weight-scheduling approach which skips the zero weights and combines two non-outliers to exploit bit-level sparsity of CNNs. In this figure, twelve weights are scheduled to be multiplied by three multipliers during four cycles. We classify weights into outliers, non-outliers, and zeros. Outlier weight, OL, is a weight that has a large magnitude which cannot be represented by $N/2$-bit width where N is the number of 8. Then, we define non-outliers, NOL, as a weight that can be represented with $N/2$-bit. The proposed scheduling algorithm searches zero weights and substitutes and replaces the zero weight with another adjacent non-zero weight. The proposed algorithm can move a weight within a lane and across lanes. The movements of a weight within a lane is called a “lookahead”. If we lookahead a non-zero weight, the multiplier in the lane can compute the non-zero weight in advance. The proposed algorithm can move weights across lanes as well, and this movement is called a “lookaside”. If we lookaside a non-zero weight, the multiplier can steal a weight value from another lane. Our proposed approach not only skips zero weight but also supports the multiplications of two non-outliers by combining them.

For the combined non-outliers, corresponding activations should be transferred to the multiplier. Since we assumed that there are three candidate weights behind the target weight, we use 4:2 multiplexer. Activation selector signals (AS) pair the activations (A_3 and A_4) and each non-outlier weights (NOL_3 and NOL_4). As a result, with the proposed weight scheduling, we can reduce two computation cycles of the shaded part.

In our proposed approach, we need to perform two types of multiplications; (i) outlier case, and (ii) combined non-outlier case.

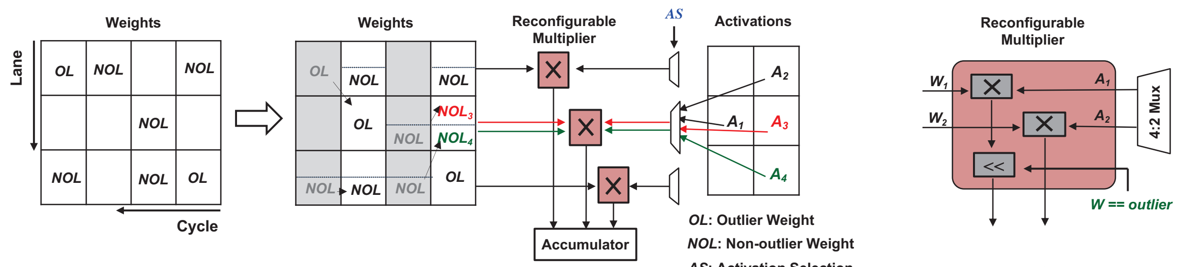


Fig. 2: Our proposed weight-scheduling procedure by using reconfigurable multipliers. OL stands for an ouliter weight, NOL stands for non-outlier weight and AS represents activation selection signal.

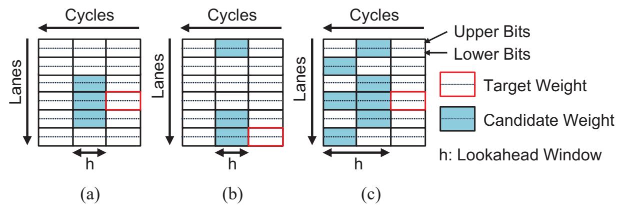
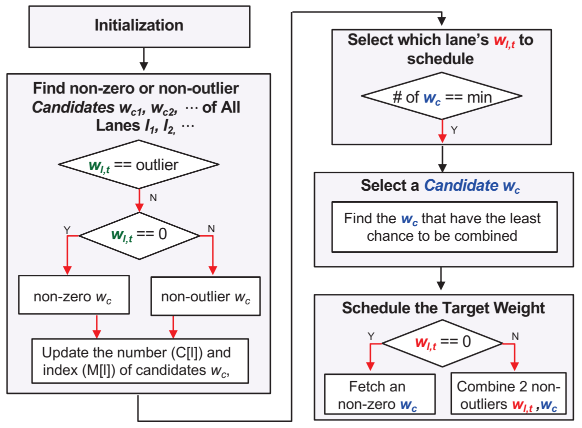


Fig. 3: Examples of the constraints for weight movements: (a) and (b) require 4:2 multiplexer and (c) requires 8:2 multiplexer for two transfer activations to each multiplier. Only weights in blue regions can move to red regions.



Initialization
Find non-zero or non-outlier Candidates w_{c1}, w_{c2}, ⋯ of All Lanes l_1, l_2, ⋯
$w_{l,t}$ == outlier
N
$w_{l,t}$ == 0
Y
N
non-zero w_c
non-outlier w_c
Update the number (C[l]) and index (M[l]) of candidates w_c,
Select which lane's $w_{l,t}$ to schedule
# of w_c == min
Y
Select a Candidate w_c
Find the w_c that have the least chance to be combined
Schedule the Target Weight
$w_{l,t}$ == 0
Y
N
Fetch an non-zero w_c
Combine 2 non-outliers $w_{l,t}$,w_c

Fig. 4: Algorithm Flow of the proposed weight-scheduling algorithm.

To support each type of multiplication, we propose a reconfigurable multiplier that can handle both types of multiplications during runtime. To be more specific, in Figure 2 (b), if W_1 and W_2 are non-outliers, each multiplication of W_1 and W_2 with A_1 and A_2 is done and accumulated independently in the accumulator. On the other hand, in the case of outliers, we shift the result from MSB parts by N/2-bit before the accumulation.

However, the replacements and the combinations of weights require additional movements of activations. If activations are allowed to move a very long distance, the power, and energy overhead of the logic to transfer activations increase dramatically; therefore, we should limit the movements of weights. We constraint the movements of weights as shown in Figure 3. If a target weight in the red region is non-outlier or zero, the software scheduler moves weights in blue regions to the red region. The weights located in the blue region correspond to the candidate. Then, multiplexer supplies an activation that corresponds to the weights to be moved. The range of lookahead and lookaside is determined by the size of the multiplexer. To maximize movements of lookahead and lookaside of weights, we set the trident form like blue regions in Figure 3 [16]. If the red region is at the top or bottom of the lane, the blue region is crossed over to the opposite lane to prevent scheduling bottlenecks as shown in Figure 3 (b) because of two reasons; (i) to use the same size of multiplexer and (ii) to avoid scheduling bottleneck. If we take fewer candidates for the top or bottom lanes than others and the scheduler fails to substitute the target weights, the probability of scheduling bottleneck increases.

Figure 4 shows the flow of weight scheduling and Algorithm 1 1 provides a detailed scheduling procedure. After initialization, we find all candidate weights w_c which are movable to target weight $w_{l,t}$ for lane l and cycle t. If the target weight $w_{l,t}$ is non-outlier weight, we only count non-outlier candidate weights. If the $w_{l,t}$ is zero, we count both non-outlier and outlier candidate weights. And then, we can update the indices and number of candidate weights to $M[l]$ and

Algorithm 1 Weight Scheduling

```
1:  Initialize $AS[i, j, k]$
2:  for t = 0 to T - 1 do
3:      Initialize $C[0, ..., L-1]$ and $M[0, ..., L-1]$
4:      if $w_{x,t}$ for all $x \in 0, .., L-1 == 0$ then
5:          $B[t] \leftarrow 0$
6:          Continue
7:      for $l = 0$ to $L-1$ do
8:          $C[l] \leftarrow$ the number of candidate weights which can be combined to $w_{l,t}$
9:          $M[l] \leftarrow \bigcup$ indices of candidate weights which can be combined to $w_{l,t}$
10:     $C_{min} \leftarrow$ Find the minimum positive number in $C[0, ..., L-1]$
11:     while $C_{min} > 0$ do
12:         for $l = 0$ to $L-1$ do
13:             if $C[l] == C_{min}$ then
14:                 for $w_{i,j}$ in $M[l]$ which has priority on movement do
15:                     if both $w_{i,j}$ and $w_{l,t}$ are non-outlier then
16:                         Append $w_{i,j}$ into $w_{l,t}$
17:                         $AS[l, t, 1] \leftarrow$ relative index for $w_{i,j}$
18:                         Update $C[0, ..., L-1]$ and $C_{min}$
19:                         Break
20:                     else if $w_{l,t}$ is zero then
21:                         Swap $w_{i,j}$ and $w_{l,t}$
22:                         $AS[l, t, 0] \leftarrow$ relative index for $w_{i,j}$
23:                         if $w_{i,j}$ is Outlier then
24:                             $AS[l, t, 1] \leftarrow$ relative index for $w_{i,j}$
25:                         Update $C[0, ..., L-1]$ and $C_{min}$
26:                         Break
27: return $W$, $AS$
```

$C[l]$ respectively. (Lines 8-9) For the lane which has a minimum number of candidate weights, we give a priority on movement to the target weight that has the least target weight to move. (Lines 11-14) Depending on whether the selected $w_{l,t}$ is zero or non-outlier, it is decided to fetch or combine. (Lines 15-26) Algorithm 1 includes whole weight scheduling, operation of activation selector $AS_{l,t,0|1}$ bit vector $B[t]$ changes during combining.

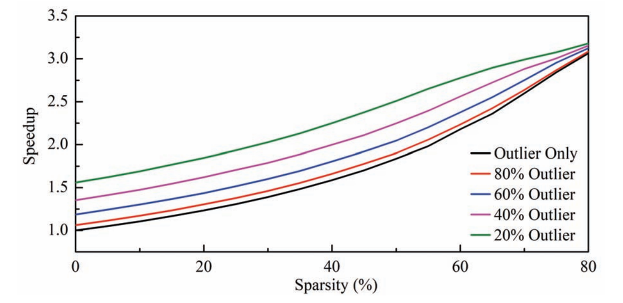


Fig. 5: The relationship between speedup and weight sparsity. Each point is the average value of one-hundred times iterations and the number in parentheses represents the ratio of non-ouliters in weights.

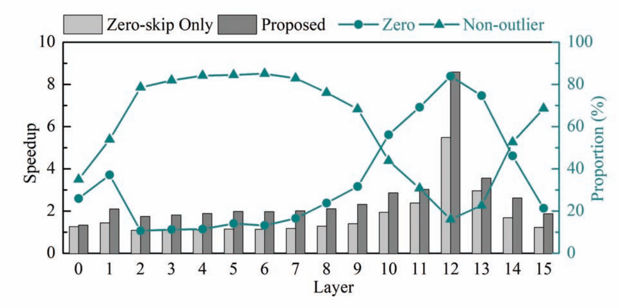


Fig. 6: Speedup of the zero-skip [16] and proposed architecture and proportion of zero and non-oultier weights of VGG-16.

B. Evaluation of Speedup of Weight-scheduling Algorithm

According to our observation, the proportions of zero and non-outlier weights vary greatly from layer to layer in a CNN. The performance improvement can vary according to the sparsity and the proportion of outlier weights among non-zero weights. If we use the zero-skip approach, we can greatly improve performance on relatively sparse layers but expect little performance improvement on dense layers. In contrast, the use of the proposed method can improve the performance sufficiently if the ratio of non-outlier is high even in sparse layers.

To evaluate speedups obtained using the zero-skip approach and proposed method in the various density of weights, we implement a cycle-accurate simulator in Python. For the experiment, we randomly generated one hundred $256 \times 3 \times 3$ filters for each sparsity and non-outlier ratios. The performance is normalized against a previous high-performance accelerator [7] with eight lanes. As the sparsity of filters increased, the speedup of both the zero-skip approach and the proposed method increased (Figure 5). In the absence of non-outlier weights, the speedup of the proposed method was identical to that of the zero-skip approach. As the proportion of the non-outlier weights increased, the proposed method achieved additional performance improvements by combining non-outlier weights. The speedup only using the zero-skip approach was imperceptible with a little sparsity. In contrast, the proposed method can still improve the performance when the non-outlier proportion is high, even if the sparsity is insignificant. Although the proposed method cannot increase performance when both proportions of zeroes and non-outliers are low, this is an exceptional case that rarely occurs on CNNs, so we can ignore it.

We conducted some experiments with VGG-16 [11] to obtain how much performance improvement can be achieved. The weights of VGG-16 were trained using the cifar-100 training set. The weights and activations were quantized into 8-bit fixed point and 16-bit fixed point, respectively. After the quantization, the weights were fine-tuned so that the accuracy drop due to the quantization was negligible. The sparsity and proportions of non-outliers varied greatly among layers (Figure 6); the sparsity increased, and the proportions of non-outliers decreases in deeper layers. The proposed method achieved greater speedup than the zero-skip approach in all layers.

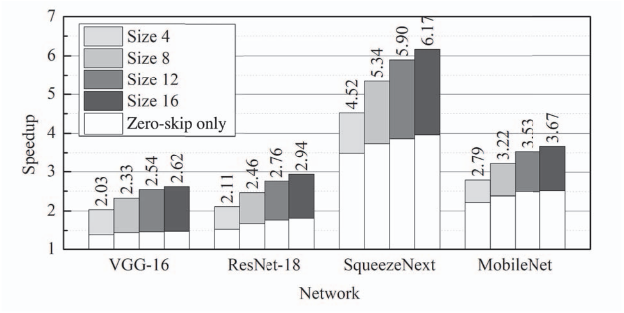


Fig. 7: Performance improvement with different size of multiplexer. Bottom (white): zero skip only, and Top (gray): additional speedup from the proposed method.

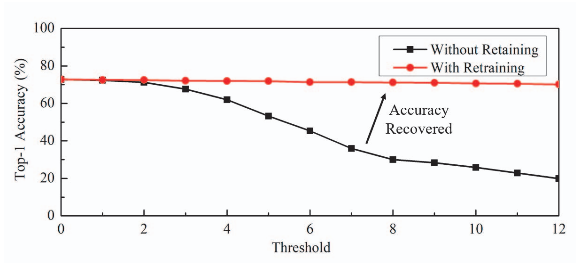
The proposed approach achieved greater speedup than the zero-skip approach on four CNNs (VGG-16, ResNet-18, SqueezeNext, MobileNet) [11], [20]–[22] and four sizes (4, 8, 12, 16) of multiplexers (Figure 7). Both approaches achieved increased speedup as the size of multiplexer increases, and the improvement by the proposed method is overwhelming. Using size 8:2 multiplexer, the proposed approach improved performance by 3.34 times, whereas the zero-skip approach only improved performance by 2.31 times, on average.

C. Case Study: Clipping Weights

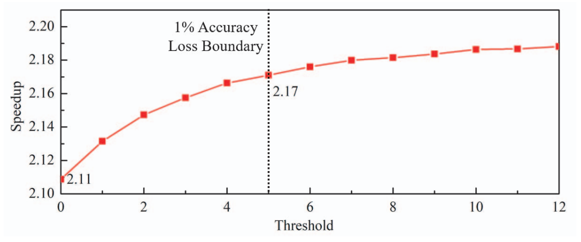
Since there are weights whose magnitudes are slightly outside the boundary of the non-outlier, conversion of those weights to non-outliers could further improve the performance. We convert an outlier weight W that is within threshold value to non-outlier W_{clip} as:

$$W_{clip} = \begin{cases} -2^3 & \text{if } (-2^3 - threshold \leq W < -2^3) \\ 2^3 - 1 & \text{if } (2^3 \leq W < 2^3 + threshold) \\ W, & \text{otherwise} \end{cases} \quad (1)$$

In VGG-16, the clipping of weights revealed a trade-off between accuracy and speedup (Figure 8). As the threshold increased, the accuracy dramatically decreased (see the black line in Figure 8 (a)). To recover accuracy, we retrained VGG-16 with the clipped weights. After retraining, the accuracy was recovered as in the red line in Figure 8 (b). An increase in the threshold increases the number of non-outlier weights, so the degree of increase in accuracy increased as the threshold was increased. If the accuracy-loss boundary is set as 1%, we can improve an additional 6% of the performance for VGG-16. We also evaluated the clipping method for different CNNs (Table II). At an accuracy reduction of 1%, clipping outlier weights can improve the performance by an additional 10% for ResNet-18, 12% for SqueezeNext, and 15% for MobileNet. The clipping method improved performance very little because the proportion of outlier weights within the threshold value is not so high. However, if we quantize CNNs more aggressively, the proportion of outlier weights within the threshold value would increase so that the clipping method would be more effective.



(a)



(b)

Fig. 8: (a) Accuracy reduction and (b) additional performance improvement by clipping outlier weights of VGG-16.

TABLE II: Accuracy reduction and speedup by clipping outliers.

Accuracy Drop

Network	Lossless	Threshold 1	2	3	4	5	6
VGG-16	72.8	-0.3	-0.4	-0.7	-0.9	-0.9	-1.5
ResNet-18	70.6	-0.1	-0.3	-0.3	-0.6	-0.9	-0.9
SqueezeNext	71.2	0.0	-0.1	-0.2	-0.4	-0.6	-0.9
MobileNet	73.0	0.0	-0.6	-0.5	-1.0	-1.8	-2.2

Additional Speedup

Network	Lossless	Threshold 1	2	3	4	5	6
VGG-16	2.11	2.13	2.15	2.16	2.17	2.17	2.18
ResNet-18	2.13	2.16	2.19	2.2	2.21	2.22	2.23
SqueezeNext	3.39	3.42	3.45	3.47	3.49	3.5	3.51
MobileNet	2.71	2.76	2.79	2.83	2.86	2.9	2.92

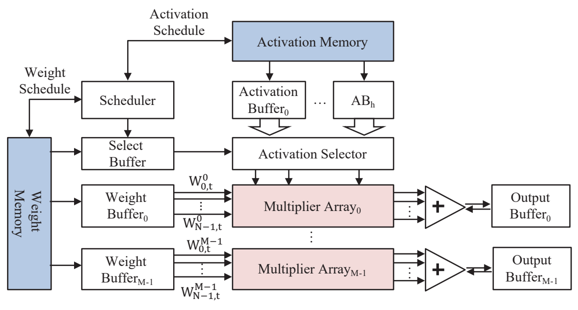
IV. Micro-Architecture and Evaluation

A. *Micro-Architecture*

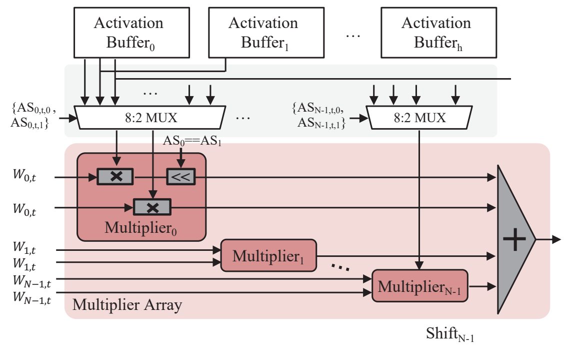
To increase the performance with scheduled weights, we propose a dedicated CNN accelerator. Figure 9 (a) shows the micro-architecture of the PE that is designed by referencing a previous accelerator [16]. The PE consists of the scheduler, activation selector, weight buffer, M multiplier arrays, M accumulators, activation memory, and weight memory. The weight memory delivers weights to buffer which are scheduled by the scheduling-algorithm. The activation selector transfers activations that correspond to scheduled weights (see Figure 9 (b)). With the activation select signal, AS, from the select signal buffer, $2N$ multiplexers select appropriate activations for rescheduled weights. Each multiplier array consists of N multipliers. To adjust the precision of weights, we deploy two sub-multipliers and a shifter to each multiplier. The two sub-multipliers compute two different non-outlier weights in most of the run-time so that the throughputs of the multipliers are doubled. Meanwhile, a few outlier weights are computed with two sub-multiplier together and the result of the upper sub-multiplier is shifted.

B. *Zero Skip of Activation*

So far we have only focused on exploiting the sparsity and bit-level sparsity of weight, but the sparsity of activation is also high.



(a)



(b)

Fig. 9: Micro-architecture of (a) proposed deep learning accelerator and (b) the activation selector.

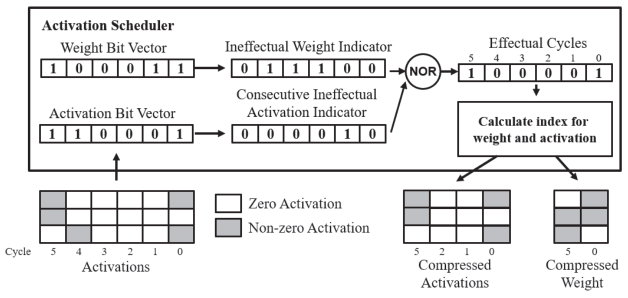


Fig. 10: The operating principle of the online scheduler.

To exploit the sparsity of activation, we deploy an online scheduler (see Figure 10). A simple way to exploit the sparsity of activation is to examine the schedule of activations and eliminate the cycles in which activations of all lanes are zero.

However, activation is controlled by activation selector signal without scheduling relocation. Thus we cannot skip all zero columns of activation and we need to check whether the following activation columns are also zero as long as the lookahead size h of the candidate window. For candidate window with lookahead size h, we check h+1 columns of activation. If all activation columns are zero, then we can eliminate the leading column.

After performing "NOR" operation of two indicators, the online scheduler generates a final bit vector. Since the weights and activations are compressed, the scheduler produces schedules for the compressed activations and weight. With the proposed scheduler, we can achieve additional speedups, which are summarized in Table III.

TABLE III: Speedup by skipping zero activations.

	VGG-16	ResNet-18	SqueezeNext	MobileNet
Weight Scheduling Only	2.33	2.46	5.34	3.22
+ Activation Zero Skip	2.50	2.76	7.29	5.33

C. Evaluation of Micro-architecture

For the evaluation of the area and power overhead of the proposed methodology, we design a CNN accelerator with Verilog HDL. The design parameters are summarized in Table IV. RTLs were synthesized to a TSMC 65-nm GP cell library at $1GHz$ clock frequency with activity factor 0.1 using *Synopsys Design Compiler*. The total power consumption is the sum of dynamic power and leakage power. Table V shows the area breakdowns for the same comparison group. The overhead for configuring the activation select signal, AS, is included in the peripheral circuit. Our proposed approach has only 1.16× more area as compared to the zero-skip architecture.

TABLE IV: Hardware configurations of the baseline accelerator, the zero-skip accelerator and the proposed accelerator.

Parameters			
Filters	8	Lanes	8
Lookahead Buffers	2	Size of multiplexer	8
Precision of Weight	8 bits	Precision of Activation	16 bits
Global Buffer			
Weight Memory	2MB	Activation Memory	4MB
Local Buffer			
Baseline	3KB	Zero-skip & Proposed	3KB

TABLE V: Area breakdowns of the previous baseline architecture [7], the zero-skip architecture [16] and the proposed accelerator. The results of the area is normalized based on the baseline architecture.

	Local Buffer	MAC	Peripheral Circuit	Total Area
Baseline	0.50	0.37	0.12	1.00
Zero-skip	1.01	0.37	0.14	1.52
Proposed	1.16	0.39	0.22	1.77

Figure 11 shows the energy breakdowns of the previous baseline architecture [7], the zero-skip accelerator [16], and the proposed CNN accelerator. The buffer accounts for most portion of the energy consumption, so the overhead of power consumption of activation selector and multipliers are negligible.

The zero-skip approach reduces energy consumptions for SqueezeNext and MobileNet, while the approaches increase the energy consumptions for VGG-16 and ResNet-18. This is because the method requires a bigger size of the local buffer whereas the performance improvement is not significant. In contrast, the proposed approach can save energy consumption, because the performance improvement of the proposed method is overwhelming. On average, the proposed approach can achieve 29.3% and 30.2% of energy reduction compared to the baseline [7] and zero-skip architecture [16] by adjusting the precision of multipliers.

V. Conclusion

In this paper, we propose an approach to double MAC to exploit bit-level sparsity of CNNs. With the weight scheduling process, we can skip zero weights and combine non-zero weights. Compared to the previous zero-skip accelerator, our proposed method is efficient regardless of the sparsity of layers. Because the sparsity of CNN dramatically changes according to layers, it is very important for CNN accelerator to improve the performance in sparse layers and dense layers. The proposed architecture also skips zero activations. We have implemented our proposed architecture, and it achieves significant speedup by 3.34× and 2.31×, and reduces 29.3% and 30.2% of energy consumption compared to the previous CNN architectures, [7] and [16] respectively.

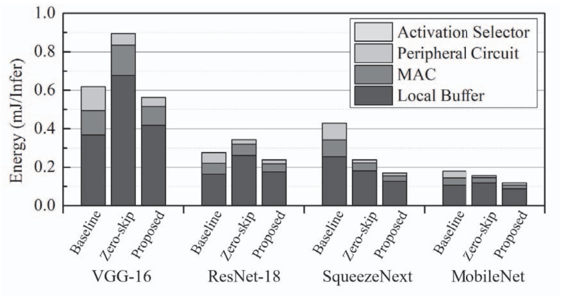


Fig. 11: Energy breakdowns of the previous baseline architecture [7], the zero-skip architecture [16] and the proposed accelerator.

References

[1] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Proc. NIPS*, 2012, pp.1097-1105.
[2] D. Lin, S. Talathi and V. Annapureddy, "Fixed Point Quantization of Deep Convolutional Networks", *Proc. ICML*, 2016, pp.2849-2858.
[3] S. Migacz, "8-bit Inference with TensorRT", *GPU Technology Conference*, 2017.
[4] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen and Y. Zou, "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients", *arXiv preprint arXiv:1606.06160*, 2016.
[5] G. Philipp, "Ristretto: Hardware-oriented Approximation of Convolutional Neural Networks", *arXiv preprint arXiv:1605.06402*, 2016.
[6] E. Park, S. Yoo and P. Vajda, "Value-aware Quantization for Training and Inference of Neural Networks", *ECCV*, 2018, pp.580-595.
[7] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun and O. Temam, "Dadiannao: A Machine-learning Supercomputer." *MICRO*, 2014, pp.609-622.
[8] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10TOPS/W Subword-parallel Dynamic-voltage-accuracy-frequency-scalable Convolutional Neural Network Processor in 28nm FDSOI", *Proc. ISSCC*, 2017, pp.246-247.
[9] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, J. K. Kim, V. Chandra, and H. Esmaeilzadeh, "Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks", *Proc. ISCA*, 2018, pp.764-775.
[10] S. Ryu, H. Kim, W. Yi and J. J. Kim, "BitBlade: Area and Energy-Efficient Precision-Scalable Neural Network Accelerator with Bitwise Summation", *Proc. DAC*, 2019, pp.1-6.
[11] K. Simonyan and. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.
[12] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen and Y. Chen, "Cambricon-x: An Accelerator for Sparse Neural Networks", *MICRO*, 2016, pp.1-12.
[13] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free Deep Neural Network Computing", *ACM SIGARCH Computer Architecture News*, 44(3), (2016), pp.1-13.
[14] D. Kim, J. Ahn, and S. Yoo, "Zena: Zero-aware Neural Network Accelerator", *IEEE Design & Test*, 35(1), (2017) pp.39-46.
[15] 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", *ISCA*, 2017, pp.27-40.
[16] A. D. Lascorz, P. Judd, D. M. Stuart, Z. Poulos, M. Mahmoud, S. Sharify, M. Nikolic, K. Siu and A. Moshovos, "Bit-tactical: A Software/Hardware Approach to Exploiting Value and Bit Sparsity in Neural Networks", *ASPLOS*, 2019, pp.749-763.
[17] A. Delmas, S. Sharify, P. Judd, K. Siu, M. Nikolic and A. Moshovos, "DPRed: Making Typical Activation and Weight Values Matter In Deep Learning Computing", *arXiv preprint arXiv:1804.06732*, 2018.
[18] E. Park, D. Kim, and S. Yoo, "Energy-Efficient Neural Network Accelerator Based on Outlier-aware Low-Precision Computation", *Proc. ISCA*, 2018, pp.688-698.
[19] R., C. D. Sa, Z. Zhang, "Overwrite Quantization: Opportunistic Outlier Handling for Neural Network Accelerators", *arXiv preprint arXiv:1910.06909*, 2019.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *Proc. CVPR*, 2016, pp.770-778.
[21] A. Gholami, K. Kwon, B. Wu, Z. Tai, X. Yue, P. Jin, S. Zhao, and K. Keutzer, "SqueezeNext: Hardware-Aware Neural Network Design", *Proc. CVPR*, 2018, pp.1638-1647.
[22] Y. Li, H. Huang, Q. Xie, L. Yao, and Q. Chen, "Research on a surface defect detection algorithm based on MobileNet-SSD." *Applied Sciences*, 8(9),(2018) pp.1678.