Specific Responses to the Reviewers' Comments

Reviewer 1

Comments to the Author

Section 4.2 introduces a few numbers but it is hard to extract insights from the insufficient discussions. More specifically, it would be great if the authors clarify how the provided data is led to the conclusion of "highest energy efficiency and fastest speed achievement".

Answer: Thank you for pointing out the unclear points in our manuscript. Due to the page limit of the DAC paper, we have not been able to put all the useful information of our design in the submitted manuscript. Hope the following explanations can make the reviewer understand our design and methods in a clearer way.

1. In addition, TOPS/W does not deliver much insight without having the context. How much utilization of the TOPS do you achieve? Which layers did you actually run? Does the energy efficiency consider the memory access power, which is considered the most power-hungry logic on the CNN accelerators?

Answer: Our reported effective energy efficiency 4.66 TOPS/W is not a theoretical number, but obtained by running all the convolution layers of the C3D network with realistic data from the UCF101 dataset. In detail, the total number of multiplication and addition operations is 0.0697 T for all the convolution layers of the C3D network. The proposed accelerator achieves a latency of 0.174 s and an average power consumption of 0.086 W with all the proposed techniques, while running at 100 MHz with 1.08 V supply voltage. We have considered the on-chip SRAM access power and the off-chip communication power during power and energy efficiency analysis.

$$0.0697\ \text{T} / 0.174\ \text{s} / 0.086\ \text{W} = 4.66\ \text{TOPS/W}. \tag{1.1}$$

By considering all the multiplication and addition operations in all the convolution layers of the C3D network, the effective performance of our design is 0.4 TOPS (0.0697 T / 0.174 s). By using the proposed TDVD technique, all the differential activations equal to or smaller than 1 are dropped out (become 0). Then by using the proposed FFZS technique, all the zeros in activations and weights are skipped. The actual number of multiplication and addition operations that are performed on the accelerator is 0.003173 T, leading to the actual performance of our design, 18.2 GOPS. The effective energy efficiency 4.66 TOPS/W can be achieved just because of using our proposed TDVD, FFZS, and HLB techniques to skip those meaningless operations with relatively balanced PE loading.

2. What does it mean to have the threshold 1? If the condition means, the value is dropped when it is smaller than 1.0, wouldn't such mechanism drop too many values? What is the distributions of data you observed and how much percentage of data is dropped?

Answer: "Threshold 1" indeed means those values equal to or smaller than 1 are dropped.

We randomly feed 64 samples from UCF101 dataset into the C3D network and separate all quantized differential values when threshold ≤ 1, $1 <$ threshold ≤ 4 , $4 <$ threshold ≤ 8, $8 <$ threshold ≤ 16, and threshold > 16. In the first layer, the distribution for the five groups is 21.6%, 29.7%, 12.1%, 9.4%, and 27.0%. In the second layer, the distribution is 51.8%, 42.7%, 2.7%, 2.3%, and 0.4%. In the third layer, the distribution is 48.7%, 44.5%, 4.8%, 1.6%, and 0.3%.

The TDVD technique is applied to the first three convolution layers of C3D network. The dropped differential values account for 46.3% of all differential values of the first three layers when threshold = 1. The accuracy only decreases from 95.3% to 95.1%.

Reviewer 2

Comments to the Author

1. It is interesting to provide the accuracy loss of corresponding weight/activation sparsity in Fig.9 because the improvement of energy efficiency leads to the accuracy lost for sure, especially the sparsity tends to 1.

Answer: The weights and activations in Fig. 9 are generated randomly with different sparsity to show where the energy efficiency upper limit of the proposed design is. The TDVD technique is actually not considered when calculating energy efficiency in Fig. 9. Therefore, there is no accuracy loss issue in this analysis.

2. Also, the design concept of intra-PE balancing is not clear for me.

Answer: The function of a PE is a convolution of 16 rows of input feature maps and 27 32-channel weights. The whole function can be divided into 16×27 small tasks, each of which is defined as the convolution of 32-channel weights with a row of 16 32-channel activations. A scheduler is designed to allocate tasks to idle MACs. During the bootup of the accelerator, the scheduler is configured to be aware of which 32-channel weights are convolved with a specific row of 16 32-channel activations, according to the parameters (e.g. stride) of the networks running on the accelerator. A look-up table (LUT) in the scheduler is used to record which tasks among the 16×27 small tasks are not finished. Once finishing a task, a MAC issues an "Empty" signal. Then, the scheduler generates the index of weights (1~27 among the 27 32-channel weights) and the base address of the specific row of 16 32-channel activations, and send them to the idle MAC. Furthermore, the scheduler informs the partial sum register to receive the result of which MAC.

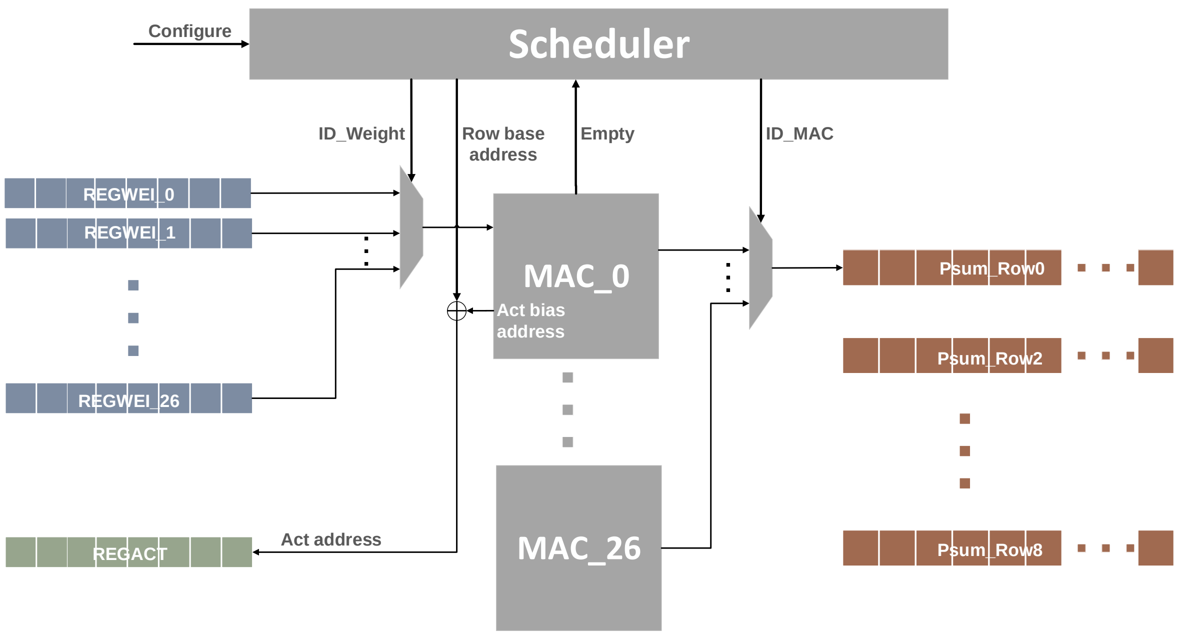

Fig. 2.1 Illustration of the intra-PE load balancing scheme.

3. Besides, it seems that the method of flag-based full-zero skipping has been proposed in [R1]. Could you discuss the proposed flag-based fully-skipping design in detail and compare it to [R1]?

[R1] Kim, Dongyoung, Soobeom Kim, and Sungjoo Yoo. "FPGA Prototyping of Low-Precision Zero-Skipping Accelerator for Neural Networks." 2018 International Symposium on Rapid System Prototyping (RSP). IEEE, 2018.

Answer: In the microarchitecture of PE in [R1], these skipped zeros are still fetched from memory to the act buffer and weight buffer. The non-zero address is generated by a zero-skip controller with a bitwise AND of act zero bit-vector and weight zero bit-vector. This non-zero address is used to find out the non-zero data without further address processing because both zero and non-zero data are in the buffers.

Alternatively, in our proposed PE, these skipped zeros are not fetched from memory at all. All the fetched data are non-zeros. The bitwise AND of ACT bit-vector and WEI bit-vector is only used as the bias address to find the corresponding non-zero data. Since the zeros are not fetched from memory, the latency and power consumption of the accelerator are both reduced compared to [R1]. The advantage of our proposed microarchitecture is increasingly larger with increased sparsity of activation and weight.

Reviewer 3

Comments to the Author

1. Why the technique is good for 3D CNN only? or can it be applied other CNN or even other neural networks?

Answer: The proposed TDVD technique is suitable for those layers where the input feature maps and output feature maps are stream data and the transformation from input feature maps to output feature maps is linear. 3D CNN in general fits the two conditions very well. Furthermore, the adjacent feature maps are highly similar. TDVD therefore performs well for 3D CNN. For 2D CNN and other neural networks, we have not found a suitable network for TDVD, and keep exploring. Alternatively, FFZS and HLB are applicable for other CNNs, but not any neural networks.

2. Can you compare FFZS to any previous zero skipping architectures in the domain of deep learning accelerators? (e.g. cnvlutin, TensorDash, etc..)

Answer: In Cnvlutin [3A], only zero activations are skipped. Our proposed FFZS skips the zeros in both activations and weights.

In TensorDash [3B], the zeros in both activations and weights are skipped. The non-zero pairs of activation and weight are moved to 4 MACs by two movement schemes to improve the utilization of the 4 MACs. However, zeros are still fetched into buffers to quickly recognize which pairs of activation and weight are non-zero. thereby increasing the latency and power consumption significantly. Furthermore, in TensorDash, a 4-deep staging buffer is used so that the maximum performance of each MAC is 4 multiplications and additions per cycle. Alternatively, the peak performance of our proposed FFZS is 32 multiplications and additions per cycle. Therefore, the utilization of MACs in our accelerator is higher compared to TensorDash, especially with increased sparsity of activation and weight.

[3A] J. Albericio *et al.*, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," *ISCA*, 2016.
[3B] M. Mahmoud *et al.*, "Tensordash: Exploiting sparsity to accelerate deep neural network training," *MICRO*, 2020.

3. TDVD is the only part that the idea is designed for 3D CNN since it is the only part discussing dropping out differential values in layers. The remaining two techniques (FFZS and HLB) are general methods to improve performance of accelerators. However, the description on TDVD is too simple and the accuracy analysis is not enough. Can you describe the details of the C3D networks and how it can be generalized for other 3D CNN?

Answer: In this work, dropping out the differential values in the first three convolution layers with threshold = 1 increases the activation sparsity from 68% to 80%. We agree with the reviewer that there are still a lot to explore for TDVD, including: 1) the threshold for dropout: what small values should be dropped out; 2) the layer depth for dropout: how many layers should be performed dropout. We are currently working on this topic as well.

Dropping out the differential values in the first three convolution layers with threshold = 1 results in a marginal accuracy loss from 95.3% to 95.1%. Dropping out the differential values of all the convolution layers with threshold = 1 makes the classification accuracy drop to 92.5% because those dropped differential values contribute to classification.

The architecture of the C3D network is illustrated in Fig. 3.1. The C3D network has 8 convolution layers and 5 pooling layers, 2 fully-connected layers, and a softmax loss layer to predict action labels. The size of filters of convolution layers is all 3×3×3. The number of filters for 8 convolution layers from 1 to 8 is 64, 128, 256, 256, 512, 512, 512, 512, respectively. The number of channels, the temporal dimension, the height, and width of input feature maps of each convolution layer are (3, 16, 112, 112), (64, 16, 56, 56), (128, 8, 28, 28), (256, 8, 28, 28), (256, 4, 14, 14), (512, 4, 14, 14), (512, 2, 7, 7), and (512, 2, 7, 7). All the pooling layers are max pooling with kernel size 2×2×2 (except for the first layer) with stride = 1.

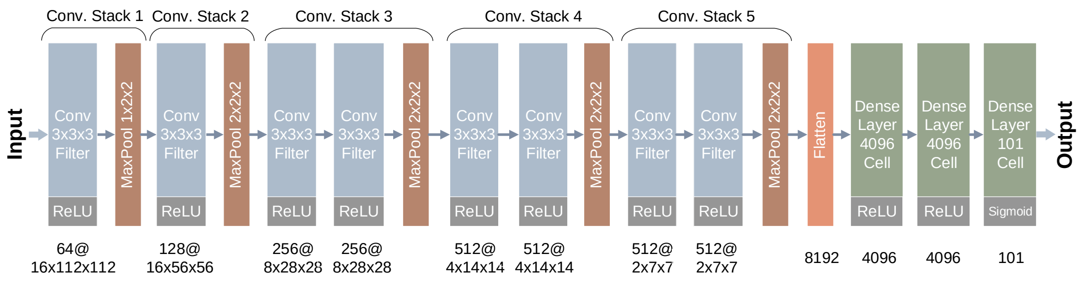

Fig. 3.1 The architecture of the C3D network.

TDVD is suitable for these layers where the input feature maps and output feature maps are stream data and the transformation from input feature maps to output feature maps is linear. 3D CNN in general fits the two conditions very well.

4. The accelerators compared in Figure 10 use different workload. How can we compare hardwares running different softwares?

Answer: In Fig. 10, we compare the proposed 3D-CNN accelerator with the state-of-the-art accelerators which run 2D-CNN (e.g. AlexNet). All the accelerators run at 100 MHz. The normalized energy efficiency is the ratio of the energy efficiency of the other three accelerators to that of Eyeriss. For fair comparison, we calculate the total number of multiplications and additions divided by the number of cycles used for those computations, and divided by the power consumption.

Reviewer 4

Comments to the Author

1. Also I don't understand paragraph 2 in Section 3.4 which explains how to dispatch small tasks to the MAC units. Where 9 MACs come from? Previous text says 27 MACs, are they grouped into 3 9-MAC groups?

Answer: Sorry that the expression of the sentence "the 2-dimension convolution is executed by accumulating the result of 9 MACs" in the original manuscript is not accurate. In the proposed PE, each of the 27 MACs is allocated with a new task by a specially designed scheduler. The 27 MACs are not grouped. In the C3D network, the 2-dimension kernel size of 3-dimension filters is 3×3, where the "9" comes from (but not "9 MACs"). To compute a row of partial sums, the 2-dimension convolution is executed by accumulating the 1×1 convolutions of "9" 32-channel weights and 3 rows of 16 32-channel activations.