Sagitta: An Energy-Efficient Sparse 3D-CNN Accelerator for Real-Time 3D Understanding

Changchun Zhou, Min Liu, Siyuan Qiu, Xugang Cao, Yuzhe Fu, Yifan He, *Member, IEEE,*
Hailong Jiao, *Member, IEEE*

***Abstract*—Three-dimensional (3D) understanding or inference has received increasing attention, where 3D convolutional neural networks (3D-CNNs) have demonstrated superior performance compared to two-dimensional CNNs (2D-CNNs), since 3D-CNNs learn features from all three dimensions. However, 3D-CNNs suffer from intensive computation and data movement. In this paper, Sagitta, an energy-efficient low-latency on-chip 3D-CNN accelerator, is proposed for edge devices. Locality and small differential value dropout are leveraged to increase the sparsity of activations. A full-zero-skipping convolutional microarchitecture is proposed to fully utilize the sparsity of weights and activations. A hierarchical load-balancing scheme is also introduced to increase the hardware utilization. Specialized architecture and computation flow are proposed to enhance the effectiveness of the proposed techniques. Fabricated in a 55-nm CMOS technology, Sagitta achieves 3.8 TOPS/W for C3D at a latency of 0.1 s and 4.5 TOPS/W for 3D U-Net at a latency of 0.9 s at 100 MHz and 0.91 V supply voltage. Compared to the state-of-the-art 3D-CNN and 2D-CNN accelerators, Sagitta enhances the energy efficiency by up to 379.6× and 11×, respectively.**

***Index Terms*—Three-dimensional inference, three-dimensional convolutional neural networks, edge computing, sparsity, speedup, energy efficiency.**

I. Introduction

In recent years, deep neural networks (DNNs) are widely applied for various inference tasks, such as image classification [1], natural language processing [2], and pattern recognition [3]. As one of the most popular DNNs, convolutional neural networks (CNNs) have achieved great success in two-dimensional (2D) image related applications. Meanwhile, three-dimensional (3D) deep learning [4] has received increasing attention owing to its wide range of applications, such as action recognition [5], [6], air quality measurement [7], virtual reality (VR) [8], augmented reality (AR) [9], autonomous driving [10], stereo matching [11], and physiological signal processing [12], [13]. These applications are precisely the edge computing tasks performed by wearable, mobile, and Internet of Things (IoT) devices, including but not limited to robots [14], drones, smart watches, smart phones, environment monitoring instruments, VR/AR glasses, and health monitoring devices. The inference tasks involved in these applications include video classification [5], medical image analysis [15], and volumetric segmentation [16]. The targets in these tasks consist of 3D voxels which are regularly arranged in a 3D grid [16], [17]. These tasks require feature extraction from the entire 3D voxel set rather than each individual 2D pixel slice. 3D-CNN [18] has been proven to be more effective in these tasks compared to the conventional 2D-CNN, because 3D-CNN convolves across all three dimensions, fully exploiting the global structure and local information simultaneously [18]. 3D-CNN is therefore highly desirable for mobile, wearable, and IoT devices. Various neural architectures for 3D-CNN have been proposed, e.g. C3D [18], 3D U-Net [19], VoxNet [16], and 3D ShapeNets [20]. From the perspective of network architecture, 3D-CNN is inflated from 2D-CNN with an extra convolutional dimension across depth. Therefore, the amount of computation and parameters that are required in 3D-CNN increases significantly compared to 2D-CNN, resulting in considerable energy consumption and latency. These could be serious concerns in edge computing for wearable, mobile, and IoT devices. These devices need to sense the environment and interact with people in real time and therefore require low latency, yet suffer from small batteries. Therefore, on-chip 3D-CNN accelerators with high energy efficiency and low latency are highly desirable at the edge.

Exploring the sparsity in CNNs is a popular way for acceleration while boosting the energy efficiency. Studies show that there are numerous zeros in the feature maps and weights of CNNs [21]. The sparsity in a layer of CNN is defined as the ratio of zeros in the weights (weight sparsity) or activations (activation sparsity) of this layer. The sparsity in certain convolution layers of popular CNNs is even up to 90%. If these zeros are skipped in the calculations, the accuracy is not affected at all. Involving those zeros in the calculations only increases the data storage burden, and wastes significant energy and computation time. If these zeros can be effectively skipped, the inference time and energy consumption of the entire network would be reduced significantly.

For 2D-CNN, a variety of CNN accelerators have explored skipping zeros in activations and/or weights. Cnvlutin investigates the sparsity in activations [22], reducing the energy consumption without any loss in accuracy. Although Cnvlutin gates the multiply-accumulate (MAC) elements for zero

This work was supported in part by the National Natural Science Foundation of China 62074005 and Shenzhen Municipal Scientific Program JCYJ20200109140601691. (*Corresponding author: Hailong Jiao*.)

Changchun Zhou, Min Liu, Siyuan Qiu, Xugang Cao, Yuzhe Fu, and Hailong Jiao are with the School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University, Shenzhen, 518055, China (e-mail: jiaohailong@pku.edu.cn). Yifan He is with Reconova Technologies Co., Ltd., Xiamen, 361015, China.

activations, the computation cycles are unfortunately not skipped [23]. In NullHop [24] and [25], the computation cycles are successfully skipped by fetching only non-zero activations from on-chip memory into MAC array to be convolved with the weights corresponding to these non-zero activations. Furthermore, those zero activations are not required to be stored on chip, saving the storage space. However, zero weights are still not skipped in NullHop and [25]. In [21], SCNN further exploits the sparsity in both weights and activations by using the sparse planar-tiled input-stationary Cartesian product dataflow to maximize the reuse of weights and activations within a set of distributed Processing Elements (PEs). SCNN uses an index-based encoding which however introduces storage overhead for the indexes of non-zero weights and activations. In [26], STICKER explores the sparsity of both activations and weights to improve the computation and storage efficiency. The non-zero activation-weight pairs are multiplied, the addresses of which are calculated from the indexes of the non-zero activations and weights. STICKER requires an external Central Processing Unit (CPU) to constantly re-arrange the activations in external dynamic random access memory (DRAM) to reduce memory write collisions for partial sums. Since adopting the channel-last dataflow, both SCNN and STICKER suffer from memory contention and compute stalls, thereby degrading the performance. In [27], the channel-run-length index and intra-kernel weight index of kernels are both used to fetch the corresponding activations to skip empty kernels and zero weights. However, the hardware complexity is considerably high. ZeNA [28] and TensorDash [29] skip the computations with both zero weights and zero activations, while both of them still need to store and transfer zero activations and weights. Meanwhile, when non-zero activations and weights are extremely sparse, they cannot guarantee a valid input pair to MAC in every cycle, thereby limiting the PE utilization ratio substantially. SNAP [30] and TwoNullHop [31] not only skip both zero weights and activations, but also skip the storage and computations of those zero weights and activations. However, to support compressed forms of weights and activations and to balance workloads, SNAP increases the hardware complexity significantly. The performance of SNAP also degrades at low bit widths (e.g. 8-bit fixed-point or lower) since the control logic used for zero skipping occupies the majority of the latency. TwoNullHop is inefficient on small networks. Furthermore, a lot of time is spent on the input image normalization and reordering off chip with TwoNullHop.

Although the above zero-skipping methods dramatically reduce the number of convolution operations by skipping invalid multiplications, load imbalance is caused by different sparsity among MACs and among PEs. To address load imbalance, the two kernels with complementary sparsity are assigned to a PE group with two PEs in [32]. Then, the two kernels are switched between the two PEs periodically to balance the inter-PE load. However, large storage inside the PE group is required to buffer activations. Extended from [32], a scheduler is employed in [33] to balance the workload of the four PEs in a PE cluster. The scheduler assigns not only the two kernels with complementary sparsity but also the two input feature map tiles with complementary sparsity to each PE. However, [33] requires an external host processor to constantly compute the sparsity of kernels and input feature map tiles. The external host processor sorts the kernels and input feature map tiles in the order of the sparsity and then controls the on-chip scheduler. In [34], a scheduler is designed to skip the convolution of similar input feature maps. These input feature maps with the same ID are similar and thus should be skipped for reading and computation. The scheduler compares the IDs of the input feature maps to skip fetching the similar input feature maps. Therefore, the scheduler ensures that valid input feature maps are fed into PE array in each cycle. However, the scheduler is only suitable for the scheduling of input feature maps, yet not able to schedule filters.

As for the hardware implementation of 3D-CNN, uniform templates are used in [35] to build accelerators for 2D-CNN and 3D-CNN based on the Winograd algorithm. Multiple clusters combined with hierarchical reconfigurable buffers are applied in [36]. Therefore, different spatial and temporal tiling strategies are flexibly supported to exploit the in-memory reusability of 3D-CNN. However, such architectures entail the integration of multiple PE arrays, and incur additional data exchanges between PE arrays and on-chip buffers. Thus, considerable area and energy cost are incurred. In [37], a hardware-aware DNN weight pruning algorithm is proposed for 3D-CNN by leveraging ADMM (Alternating Direction Method of Multipliers). In [38], a flexible Winograd-based decomposition method is proposed to reduce multiplication operations. A fusion 2D/3D computing engine is also introduced to support different strides and different filter sizes. In [39], a temporal redundancy elimination mechanism is implemented by skipping the computations with the same frame tiles and filters, while the temporal reusability is disabled in this case. However, these works have not fully exploited the sparsity of weights or activations in 3D-CNN and are only verified on field programmable gate array (FPGA) or by logic synthesis.

Till now, no 3D-CNN accelerator has been validated via silicon on chip. To achieve this purpose for edge applications, an energy-efficient low-latency 3D-CNN accelerator is desired, where exploring the sparsity in 3D-CNN could play a critical role. However, there are still several critical issues to be solved. First, the activations are not sufficiently sparse in various CNNs, especially compared to weights, thereby restricting the effectiveness of sparse acceleration. Second, zero skipping has been barely explored in the previous 3D-CNN hardware implementations. How to skip all the zeros in both activations and weights in 3D-CNNs is still an obstacle towards achieving the maximal energy efficiency. Third, balancing the loads among PEs is also a great challenge after skipping zeros in activations and/or weights.

In this paper that is extended from our work [40], an energy-efficient low-latency on-chip 3D-CNN accelerator, Sagitta, is proposed for real-time 3D inference on edge devices. A threshold-based differential slice sparsity enhancement method is proposed to increase the sparsity of differential slices by dropping out small differential values. Furthermore, to skip both the redundant operations and redundant computation cycles for a wide sparsity range of activations and weights, a full-zero-skipping microarchitecture is proposed. To address the load imbalance issue after skipping zeros, a hierarchical load-balancing scheme is proposed. All the proposed techniques are enabled on the specialized hardware architecture, which is also optimized for efficient data flow of 3D-CNN. With the proposed techniques and architecture, Sagitta reduces the latency by up to 17.2 × and 19.5 ×, compared to the baseline implementation, while running the C3D network classifying the UCF101 dataset [41] and 3D U-Net segmenting BraTS 2020 [42], respectively. Implemented in the UMC 55-nm low-power CMOS technology, Sagitta achieves an energy efficiency of 3.8 TOPS/W for C3D and 4.5 TOPS/W for 3D U-Net at 100 MHz and 0.91 V supply voltage.

The paper is organized as follows. The background of 3D-CNN and the motivations of this work are introduced in Section II. The system architecture innovations are stated in Section III. The proposed techniques that enable low latency and high energy efficiency are presented in Section IV. Experimental results of the proposed 3D-CNN accelerator are discussed in Section V. The paper is concluded in Section VI.

II. Background and Motivations

In this section, the basic information about 3D convolution operation, the classical network architecture of 3D-CNN, and the corresponding datasets are introduced. The motivations of the proposed techniques are also presented.

A. Preliminaries

In a 3D convolution, the filter moves in three directions. 3D convolution is therefore often used in 3D images such as video and volumetric imaging.

The computation process of the convolution layers in 3D-CNN is described as follows, which is also illustrated in Fig. 1.

$$Output[m][g][e][f] = \sum_{c=0}^{C}\sum_{d=0}^{D}\sum_{r=0}^{R}\sum_{s=0}^{S} Filter[m][c][d][r][s] \times Input[c][g * stride + d][e * stride + r][f * stride + s]. \quad (1)$$

C, T, H, and W are the number of channels, the number of slices, the height, and the width of input feature maps, respectively. D, R, and S are the number of slices, the height, and the width of filters, respectively. M, G, E, and F are the number of channels, the number of slices, the height, and the width of output feature maps, respectively.

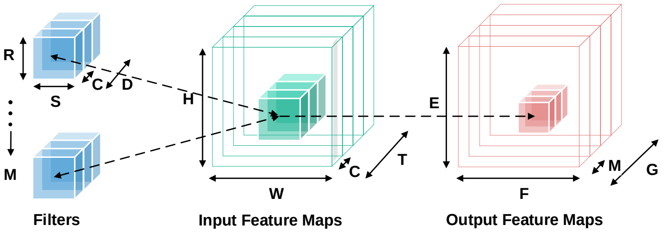


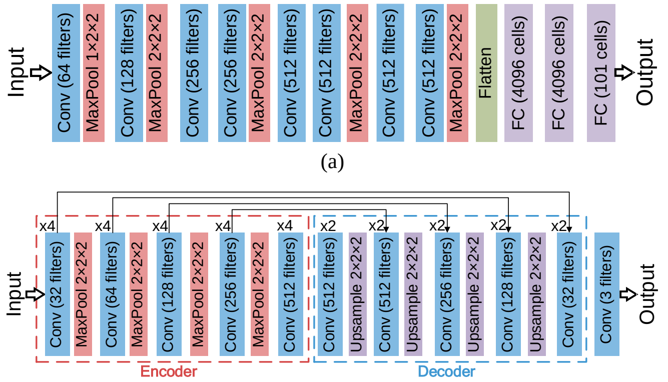
Fig. 1. The computation process of the convolution layers in 3D-CNN.

As shown in Fig. 1, in each layer, a set of C input feature maps of size $T \times H \times W$ are convolved by M sets of filters of size $C \times D \times R \times S$, yielding M output feature maps of size $G \times E \times F$. 2D convolution is the special case of 3D convolution when both T and D are equal to 1. The computation complexity of the convolution layers is $2 \times M \times E \times F \times R \times S \times C$ in 2D-CNN while $2 \times M \times E \times F \times G \times R \times S \times D \times C$ in 3D-CNN.

B. Classical Network Architecture of 3D-CNN

C3D is a classical 3D-CNN architecture for classifying actions in videos [18]. C3D can model appearance and motion information simultaneously and outperforms 2D-CNN on video analysis tasks. C3D can be simply extended from VGG-16 [43] with a homogenous architecture containing 3×3×3 convolutional filters followed by 2×2×2 pooling at each layer. C3D consists of eight convolution layers, five pooling layers, and three fully-connected (FC) layers as shown in Fig. 2a.

Similarly, by replacing all 2D operations of U-Net [44] with their 3D counterparts, 3D U-Net is proposed in [19] for volumetric segmentation of 3D images. As an indicator of inference accuracy, dice score represents the similarity between the labelled image and the predicted image with a range from 0 to 1 and is used for the evaluation of 3D U-Net in this paper. The network architecture of 3D U-Net is illustrated in Fig. 2b. Similar to C3D, 3D U-Net contains 30 3×3×3 convolutional filters followed by 2×2×2 pooling or 2×2×2 up-sampling.



(a)

(b)

Fig. 2. The network architecture of classical 3D-CNNs. (a) The architecture of the C3D network [18]. (b) The architecture of the 3D U-Net network [19].

C3D is the representative for 3D classification applications, while 3D U-Net is for 3D segmentation applications. To train and compress neural networks, we use Distiller [45], an open-source Python package for neural network compression research that is developed by Intel. For C3D, UCF101 [41], a dataset of 101 human action classes from videos in the wild, is used as the evaluation dataset. For 3D U-Net, BraTS 2020 [42], a state-of-the-art, accurately-annotated, and large-scale dataset of 3D brain tumor segmentation, is used as the benchmark dataset. Prior works have shown that linear quantization to 8 bits does not cause accuracy loss for 2D-CNNs [46]–[48], which is also verified in this work for 3D-CNNs.

C. Motivations

3D-CNNs have more parameters and computations than 2D-CNNs. The data feature of 3D-CNNs, such as sparsity, should

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3306435

be heavily leveraged to enhance the energy efficiency and shorten the inference latency. Quantization and pruning are the primary source of weight sparsity. The activation sparsity mainly results from the activation functions, such as rectified linear unit (ReLU). Quantization also increases the sparsity of activation [49].

The input feature maps of each convolution layer in 3D-CNN are usually a sequence of consecutive slices, the number of which is T, as shown in Fig. 1. There are plenty of similarity between any two consecutive slices for input feature maps, which is the locality. The locality could be potentially used to increase the activation sparsity. If we subtract the back slice from the front slice, the differential slice T_diff is obtained as

$$T_{diff} = In[t+1][C][H][W] - In[t][C][H][W]. \qquad (2)$$

In this way, the original slices are converted into one base slice and a series of differential slices. The distribution of quantized non-zero activations is illustrated in Fig. 3, when activations and weights are quantized to 8 bits and weights are pruned to the sparsity of 97.6% and 96.3% for C3D and 3D U-Net, respectively. Compared to the raw slices across all layers, the sparsity of differential slices in C3D grows slightly from 54.5% to 57.5%, while 3D U-Net even drops from 89.3% to 82.5%. Meanwhile, the sparsity of activations in the differential slices is still significantly lower than the sparsity of weights (e.g. 97.6% in C3D). Direct leveraging the locality of inputs therefore cannot enhance the sparsity of activations substantially. Therefore, a specialized method to increase the sparsity of activations beyond merely using locality is required.

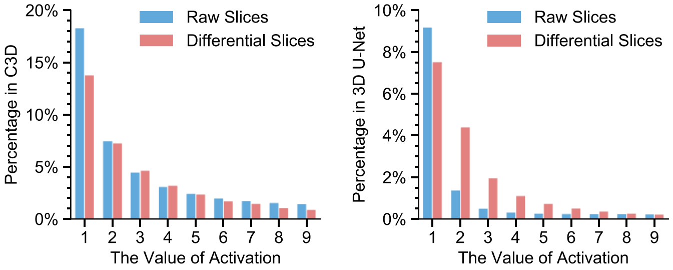


Fig. 3. The distribution of non-zero activations in C3D and 3D U-Net.

To leverage the sparsity in activations and weights, it is critical to keep only non-zero data and exclude zero data from data movement and computations. Furthermore, only non-zero input pairs should be fetched every cycle to achieve high throughput. An efficient zero-skipping architecture is therefore needed to ensure there are sufficient valid input pairs being fed to MAC to maintain high MAC utilization ratio, especially in the case of high sparsity.

The hardware architecture is divided into different computation levels with different computation flows. However, a serious issue in these designs is the mismatch between the granularity of computation levels and the granularity of the load balance schedule. Although a fine-grained schedule leads to high PE utilization ratio, the hardware complexity increases significantly. While a coarse-grained schedule has lower hardware complexity, the load imbalance issue is still severe. Therefore, a design with mixed granularity for different computation levels, which benefits from twofold, is a more promising choice.

III. System Architecture Overview

The system architecture of the proposed 3D-CNN accelerator Sagitta is introduced in this section. The architecture is specially designed to enable the 3D convolution of differential input feature maps. To obtain high energy efficiency and low latency, the computation flow of the proposed PE is optimized for high data reuse, load balance, low hardware complexity, as well as the configurability. The high-level hardware architecture is introduced in Section III-A. The computation flow of the proposed PE is presented in Section III-B.

A. Hardware Architecture

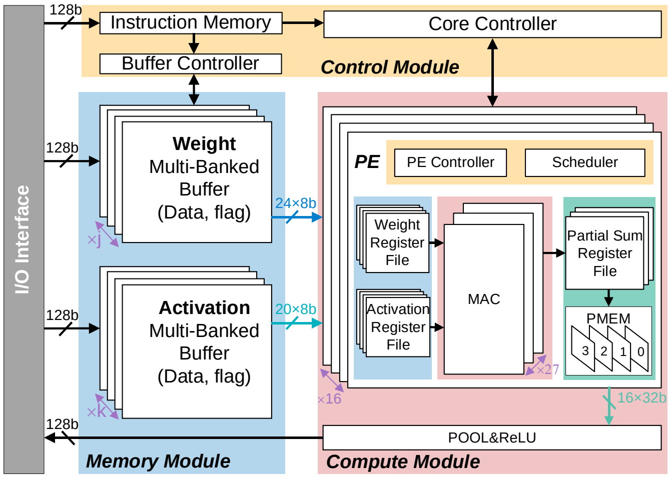


Fig. 4. The hardware architecture of the proposed 3D-CNN accelerator.

The hardware architecture of Sagitta is shown in Fig. 4. The accelerator consists of a compute module, a memory module, a control module, and an I/O interface. The compute module is made up of 16 PEs, each of which contains 27 MACs, a PE controller, a scheduler, weight register files, activation register files, partial sum register files (PRF, implemented by flip-flops), and partial sum memory (PMEM, implemented by static random-access memory (SRAM)). The memory module is composed of 16 SRAM banks of 8 KB, each of which can be dynamically allocated for the weights and activations ($j + k = 16$ and $j, k \in [1, 15]$) as shown in Fig. 4. The control module receives the instructions from the I/O interface and then controls the computation schedule of the compute module and the SRAM allocation of the memory module. Only non-zero weights and activations are fetched from off-chip memory and stored in allocated SRAM banks following the system configuration. The activations flow from PE0 to PE15 in a one-dimensional (1D) systolic fashion to guarantee that the activations of all PEs are the same, which is beneficial for inter-PE load balance [50]. Once the current PE finishes the convolutions, the activations are transferred to the next PE. Activations are reused among PEs. The weights are broadcasted [50] from the multi-banked buffer to all PEs [50]. The 16 PEs are allocated 16 different filters. Each PE executes the convolution of a filter and input feature maps so that 16 output feature map channels are computed in parallel.

Within a PE, the scheduler is responsible for feeding activations and weights from activation and weight register files into each MAC and storing partial sums in PRF. Partial sums are added by propagation to compute differential output feature maps in PMEM. All operations can be executed in differential format except the non-linear ReLU and pooling. Therefore, the raw output feature maps need to be recovered before being sent to the POOL&ReLU module. After pooling and ReLU on the original output feature maps, the POOL&ReLU module converts the raw output feature maps back to differential output feature maps. The hardware to make conversion between the raw feature maps and differential feature maps does not degrade performance and accounts for only 1.2% of the total area and 1.4% of the total power consumption (by simulation) of the accelerator. Finally, the differential output feature maps are compressed into a format of non-zero values and bit-vectors, and then sent back to the external memory via the I/O interface.

B. Computation Flow of PE

In this subsection, the computation flow of the proposed PE is presented in detail. The computation flow is specially designed to enable 3D convolution, as shown in Fig. 6. To perform 1D convolution in the T dimension of the 3D convolution (see Fig. 1), PMEM in PE is designed to accumulate the partial sums of the 2D convolution computed by the MAC array. Furthermore, various schemes are adopted to enhance the computing efficiency of the proposed accelerator. First, to reduce the amount of partial sums and align the addresses of the partial sums, the channel-first dataflow is adopted in the MAC array of PE [30], as shown in Fig. 5. The channel-first dataflow also simplifies task scheduling, thereby resulting in low hardware complexity. Second, the sparsity difference between the first raw input slice and the following differential input slices is large. To reduce the amount of required cache and balance the convolutions of the fixed filter and the different input slices, the input slices are serially fed into PE for computation. Third, to enhance the energy efficiency, the input feature maps are further reused in the D dimension, while the filters are further reused in the T dimension in the MAC array.

In the proposed PE, the allocated filters are stored in the weight register file. The input slices are stored in the activation register file. The convolution process of an input slice with the filter slices is a channel-first dataflow in the proposed MAC array. In the channel-first dataflow, the activations and weights are ordered in the channel dimension first, then in rows, and in columns afterwards according to the 2D pixel-location. As defined in (1), m, c, d, r, and s are the indexes of the filter, the channel, the slice, the row, and the column of an element in filters, respectively. Similarly, c, t, h, and w are the indexes of the channel, the slice, the row, and the column of an element in input feature maps, respectively. A bundle of activations contains data in matrix ($\underline{c}$, t, h, $\underline{w}$), where $\underline{c}$ and $\underline{w}$ represent all elements along the channel dimension and width dimension, respectively. A sub-bundle of activations contains data in matrix ($\underline{c}$, t, h, w), where $\underline{c}$ represents all elements along the channel dimension. A bundle of weights contains data in matrix (m, $\underline{c}$, d, r, s), where $\underline{c}$ also represents all elements along the channel dimension. The convolution of a bundle of weights and a bundle of activations is defined as a task which is allocated to a MAC at a time. Therefore, the convolution of the input slice with the filter slices can be divided into multiple tasks. The convolution of a filter slice with an input slice that is shown in Fig. 5 is in two steps. *Step_i*: A weight bundle of matrix (m, $\underline{c}$, d, r, s) = (1, 32, 1, 1, 1) is convolved with an activation bundle of matrix ($\underline{c}$, t, h, $\underline{w}$) = (32, 1, 1, 16) to compute a bundle of partial sums, as noted by ① and ② in Fig. 5. To compute the whole input slice, the bundle convolution is then performed across the column direction, as noted by ③. Meanwhile, the addresses of the computed partial sums are aligned automatically with themselves because the address of the partial sums computed by one MAC stays the same until the MAC completes the convolution between an activation sub-bundle of matrix ($\underline{c}$, t, h, w) = (32, 1, 1, 1) and a weight bundle, and switches to a new activation sub-bundle (change of w) [30]. Furthermore, the partial sums can be immediately reduced along the channel dimension from $\underline{c}$ to 1 [30]. *Step_ii*: The corresponding 9 (R×S = 9 in Fig. 5) rows of partial sums are accumulated to get the final partial sum matrix ($\underline{m}$, g, e, f) = (1, 1, 1, 16) of the input slice. The final partial sums are stored in the partial sum register file.

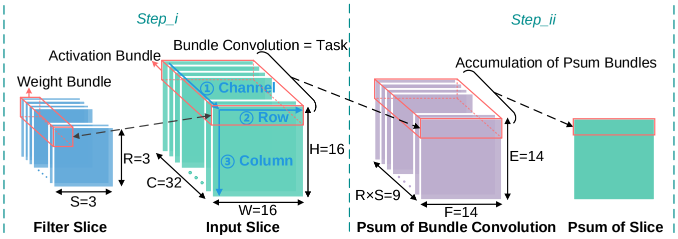


Fig. 5. The convolution process in the MAC array.

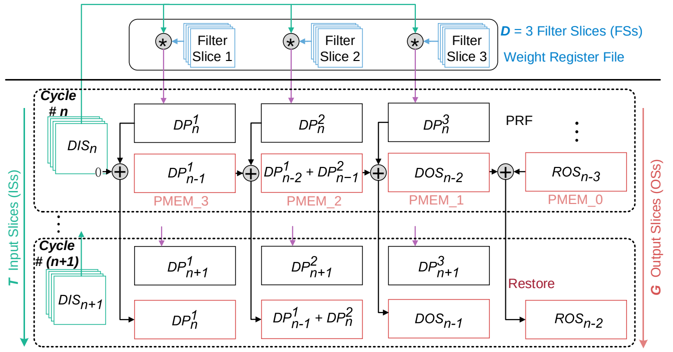


Fig. 6. The computation flow inside one PE.

An example of the computation flow of PE when $D = 3$ is shown in Fig. 6. DP_n^m represents the differential partial sums obtained from the convolution of the n^{th} ($1 \le \mathrm{n} \le T$) differential input slice and the m^{th} ($1 \le m \le D$) filter slice. DIS_n, DOS_n, and ROS_n represent the n^{th} differential input slice (IS), the n^{th} differential output slice (OS), and the n^{th} raw output slice, respectively. As illustrated in Fig. 6, the G output slices of 3D convolution that are shown in Fig. 1 are calculated by a PE in two phases.

Phase_i: From the first input slice to the T^{th} input slice, the T input slices are serially fed to PE and stored in the activation register file. Note that the first input slice has the raw data while the rest have the differential values. At the n^{th} cycle, the MAC array performs a convolution of the n^{th} input slice and the D fixed filter slices to obtain DP_n^1, DP_n^2, and DP_n^3 which are stored in PRF. At the same time, PMEM_3, PMEM_2, PMEM_1, and PMEM_0 store the computed DP_{n-1}^1, ($DP_{n-2}^1 + DP_{n-1}^2$), DOS_{n-2}, and ROS_{n-3}, respectively.

Phase_ii: At the $(n+1)^{th}$ cycle, DP_n^1, DP_n^2, and DP_n^3 in PRF computed in *Phase_i* are added to 0, DP_{n-1}^1 of PMEM_3 (generated at the n^{th} cycle), and (DP_{n-2}^1+DP_{n-1}^2) of PMEM_2 (generated at the n^{th} cycle), to obtain DP_n^1 of PMEM_3, ($DP_{n-1}^1 + DP_n^2$) of PMEM_2, and DOS_{n-1} of PMEM_1, respectively. Meanwhile, DOS_{n-2} in PMEM_1 (generated at the n^{th} cycle) is added to ROS_{n-3} in PMEM_0 (generated at the n^{th} cycle) to restore ROS_{n-2}, which is stored in PMEM_0. ROS_{n-3} (generated at the n^{th} cycle) is sent to the POOL&ReLU module.

In *Phase_i*, the convolution of the three fixed filter slices and the serial input slices is the 2D convolution in the $H \times W$ dimension of 3D convolution (see Fig. 1). In *Phase_ii*, the addition of the data in PMEM_3, PMEM_2, and PMEM_1 with the three differential partial sums in PRF is the 1D convolution in the T dimension of 3D convolution (see Fig. 1). Therefore, a PE computes the complete 3D convolution of three filter slices and T input slices.

In the proposed PE computation flow, the filters are convolved with T input slices and are therefore reused in the $T \times H \times W$ dimension of 3D convolution while the filters are reused only in the $H \times W$ dimension of 2D convolution. Similarly, the input feature maps are convolved with D filter slices and are thus reused in the $D \times R \times S$ dimension of 3D convolution while the input feature maps are used only in the $R \times S$ dimension of 2D convolution. The additional reuse of the filters and input feature maps leads to higher energy efficiency and lower latency by reducing memory access. Since the sparsity of the differential input slices is typically higher than that of the first raw input slice (e.g. 11.2× higher in Conv2 of C3D), all input slices are computed serially rather than in parallel to mitigate load imbalance. Since the POOL&ReLU module needs to convert the raw output slices back to the differential output slices according to (2), the serial computation flow significantly reduces the amount of required cache and saves the chip area.

IV. Strategies Enabling Low Latency and High Energy Efficiency

In this section, a specialized method to increase the sparsity of activations by leveraging the differential value dropout is presented. To fully take advantage of highly sparse activations and weights, an efficient zero-skipping architecture is proposed for acceleration. A hierarchical load balancing method is also introduced to solve the imbalance among both PEs and MACs inside each PE after full zero skipping.

A. Threshold Differential Value Dropout (TDVD)

As discussed in Section II-C (see Fig. 3), the differential slices are not sufficiently sparse in the convolution layers. By counting the number of each absolute value from 1 to 128 (8-bit), it is noticeable that small values, such as 1, 2, and 3, take a large portion. Because the slices are temporally/spatially sequential, the small differential values are the tiny temporal/spatial changes, which could be meaningless noise such as light change or camera jitter. The meaningless noise in the convolution layers does not improve the performance of networks. Therefore, those small differential values below the selected threshold could be dropped out with negligible accuracy loss, as described below.

$$f(x) = \begin{cases} 0, \ for \ |x| \le threshold. \\ x, \ for \ |x| > threshold. \end{cases} \tag{3}$$

$threshold$ is the selected threshold. x is the differential values. This technique is named threshold differential value dropout (TDVD). TDVD is implemented in the POOL&ReLU module (see Fig. 4). To demonstrate the effect of TDVD on the activation sparsity enhancement, the activation sparsity under different thresholds is shown in Fig. 7. The overall activation sparsity of all the convolution layers increases significantly with a higher threshold. Particularly, for both C3D and 3D U-Net, the overall activation sparsity increases to 86.0% and 97.5% when $threshold = 5$.

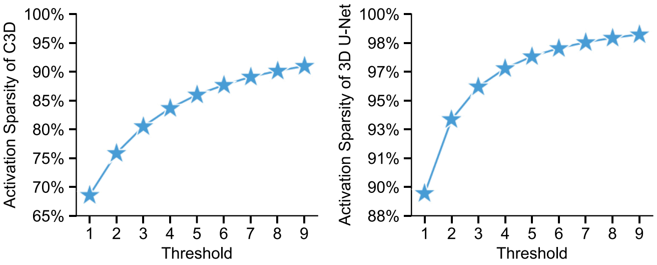


Fig. 7. The activation sparsity at different thresholds in C3D and 3D U-Net.

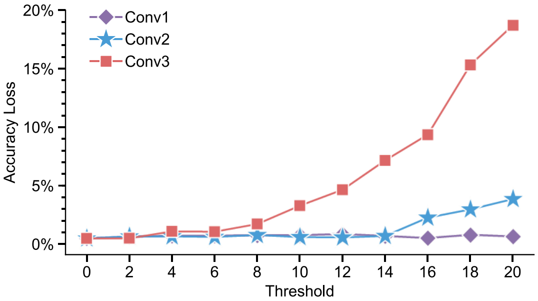


Fig. 8. The sensitivity of the accuracy to the threshold in the first three convolution layers in C3D.

Although a higher threshold benefits the performance of the accelerator because of the increased sparsity, the accuracy may be degraded to an intolerable level. To investigate how the threshold in each layer influences the accuracy of the network, the accuracy loss is scanned with the threshold increased from 0 to 20 in the first three convolution layers, as illustrated in Fig. 8. The benchmark is the quantized and pruned C3D. As shown in Fig. 8, the accuracy tolerance to the threshold varies among layers. Furthermore, although a large threshold adopted by TDVD leads to explosive accuracy loss, a small threshold can ensure negligible accuracy loss

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2023.3306435

while obtaining substantial sparsity improvement.

A suitable threshold is the key to trade off sparsity with accuracy. However, it is difficult to find the optimal threshold combination due to the large depth of neural networks and the wide range of the threshold. A method to automatically learn the threshold for different layers is proposed, where the threshold is a learnable parameter as part of the network loss. The threshold are used to dropout the small values of T_{diff} in (2). We use an activation function called Hard Shrinkage to perform the dropout operation. $\frac{T_{diff}}{threshold}$ is the input of the Hard Shrinkage.

$$T'_{diff} = \text{HardShrinkage}\left(\frac{T_{diff}}{threshold}\right) * |threshold|. \qquad (4)$$

T'_{diff}, the differential values after TDVD, can be used for network inference. $|threshold|$, the norm of $threshold$, is used to recover the range of T'_{diff}. The inference loss function $loss_{inference}$ is maintained as follows

$$loss_{inference} = \text{criterion}(outputs, labels). \qquad (5)$$

Larger $threshold$ is harmful for the accuracy. Therefore, optimizing $loss_{inference}$ suppresses the $threshold$. To increase the activation sparsity, we add a loss function $loss_{threshold}$ to increase the $threshold$

$$loss_{threshold} = \frac{\lambda}{\|threshold\|+\epsilon}, \qquad (6)$$

where $\|threshold\|$ is the L2 norm of $threshold$. λ is a hyperparameter that is used to tune the value of the $threshold$. $\epsilon = 1 \times 10^{-5}$ is a small number for numerical stability. Therefore, the final loss function of the network is

$$loss = loss_{inference} + loss_{threshold}. \qquad (7)$$

As λ increases, the optimizer tends to decrease $loss_{threshold}$ and increase $threshold$ and the activation sparsity. By adjusting λ and then retraining C3D and 3D U-net, the corresponding sparsity and accuracy/dice score are shown in Fig. 9.

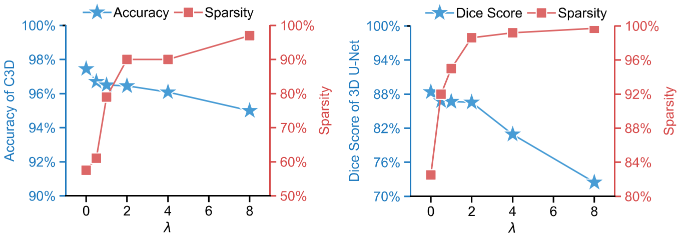
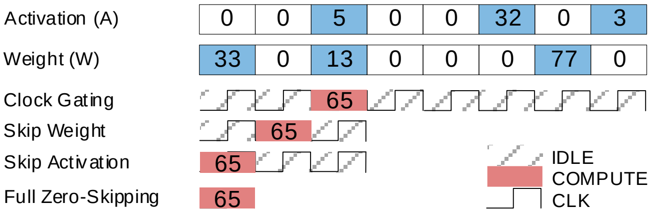


Fig. 9. The accuracy/dice score loss and sparsity improvement with increasing learnable threshold in C3D and 3D U-Net.

When $\lambda = 2$ ($threshold = (10, 7, 5, 5, 5, 5, 5, 2)$) in C3D, 4.6×non-zero activation reduction is obtained with 1.3% accuracy loss. The sparsity of activations is increased from 57.5% to 90.8%. When $\lambda = 2$ ($threshold = (0, 1, 4, 5, 3, 3, 4, 4, 5, 5, 5, 5, 4, 5, 5, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)$) for 30 3×3×3 convolutions mentioned in Section II-B in 3D U-Net, 12.5×non-zero activation reduction is achieved with 1.8% dice score loss. The sparsity of activations is increased from 82.5% to 98.6%. Accordingly, considering the significant sparsity enhancement with negligible accuracy/dice score loss, these two dropout schemes are adopted in the rest of this work for performance evaluation.

B. Flag-based Full Zero Skipping (FFZS)

Clock gating is one of the efficient zero-skipping methods by gating circuits once encountering zero activations or weights [51]. Those skipped zeros are still required to be fetched from SRAM arrays or registers to computation units. Therefore, for the example in Fig. 10, the clock gating method requires eight cycles to process these data. With the previously published activation zero-skipping [25], [24], [52] or weight zero-skipping methods, three cycles are still needed even when either activation or weight is zero in two of the three cycles for the example in Fig. 10. A full-zero-skipping method is proposed in this work, which requires only one cycle, as shown in Fig. 10. Only when both activation and weight are non-zero, the non-zero data are fetched for computation.



Activation (A)	0	0	5	0	0	32	0	3
Weight (W)	33	0	13	0	0	0	77	0

Fig. 10. The comparison of different zero-skipping methods.

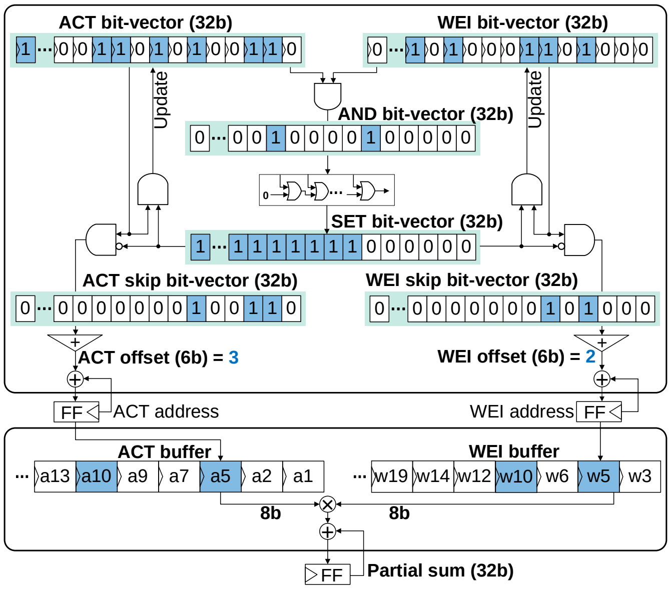


Fig. 11. The proposed microarchitecture of full zero skipping.

The proposed microarchitecture to realize full zero skipping is illustrated in Fig. 11. All the activations and weights are checked whether they are zero or not. The non-zero activations, the non-zero weights, and the zero/non-zero status are stored in the off-chip DRAM array and then fetched into the on-chip memory module (mentioned in Fig. 4 and implemented by SRAM). When the activations and weights are required for computation, the zero/non-zero status of these data is loaded from the memory module into the ACT bit-vector registers and WEI bit-vector registers via the on-chip bus in the accelerator. Meanwhile, only the non-zero activations and weights are loaded from the memory module into the ACT buffer and WEI buffer. A bitwise AND of the ACT bit-vector and WEI bit-vector is performed to get an AND bit-vector, which indicates where the activation and weight are both non-zero so that the multiplication is valid. To fetch the non-zero activation and weight, the ACT/WEI skip bit-vector is obtained by performing a bit-wise AND of the ACT/WEI bit-vector and the SET bit-vector. In this way, these "ones" in the ACT and WEI bit-vectors, which are high bits compared to the first "one" in the AND bit-vector, are dropped out. The number of "ones" that are low bits in the ACT/WET skip bit-vectors indicates the address offset of the activation and weight, respectively, in the buffers. Meanwhile, the ACT/WEI bit-vector is updated by performing a bit-wise AND of the ACT/WEI bit-vector and the SET bit-vector. The computation unit fetches, from the buffers, only the data referenced by these address offsets (3 and 2 in Fig. 11) and performs computations with them (a5 and w5 in Fig. 11). From the ACT/WEI bit-vectors to address generation, only one cycle is needed. To increase the frequency and reduce the dynamic power consumption of the accelerator, computing address and generating partial sums are designed to be pipelined.

The designed MAC consists of a multiplier and an accumulator, the logic for generating the addresses of partial sums, and the flag-based full-zero-skipping microarchitecture. The area of the FFZS module is 37.3% in one MAC, while only 8.7% of one PE. The simulated power consumption of FFZS is 30.7% of one MAC and 9.8% of one PE.

In the microarchitecture of ZeNA [28] which also skips both zero activations and weights, the skipped zeros are still fetched from memory to the activation and weight buffers. Alternatively, in TwoNullHop [31] and our proposed microarchitecture, these skipped zeros are not fetched from memory at all. However, TwoNullHop is inefficient on small networks (such as R = S = 1 and H = W = 32), because of the insufficient bandwidth for the partial sums and I/O interface. Meanwhile, a lot of time is spent on the input image normalization and reordering off chip [31]. In our proposed microarchitecture, the bitwise AND of ACT bit-vector and WEI bit-vector is only used as the address offset 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 ZeNA. The advantage of our proposed microarchitecture is increasingly larger with increased sparsity of activations and weights, because FFZS only consumes one cycle to generate the valid address of non-zero data, while increased cycles are required to obtain the valid address in ZeNA.

In TensorDash [29] that also performs full zero skipping, non-zero activation and weight pairs are moved to four MACs by two types of movement so that four MACs are utilized as much as possible. However, zero activations and weights are still fetched into buffers so that the latency and power consumption are significantly increased compared to the proposed FFZS. Meanwhile, in TensorDash, the 4-deep staging buffer is used so that the maximum performance of each MAC is limited to four MAC operations per cycle. Alternatively, the peak performance of our proposed FFZS reaches up to 32 MAC operations per cycle. Therefore, the utilization of MACs in our accelerator is higher compared to TensorDash, especially in the case of high sparsity.

C. Hierarchical Load Balancing (HLB)

After full zero skipping, load imbalance is caused by different sparsity among PEs and among MACs. As described in Section III-A, the overall hardware architecture and computation flow are divided into two levels. The coarse-grained PE level computes the convolution of input feature maps and 16 filters in parallel to generate 16 output feature map channels. The fine-grained MAC level computes the convolution of activations of an input slice and weights of a filter. A hierarchical load-balancing scheme combining inter-PE load balance at the PE level and intra-PE load balance at the MAC level is proposed to solve the load imbalance issue by integrating scheduler-based task allocation and filter reordering in this accelerator.

The intra-PE load-balancing is to balance loads among the 27 MACs in each PE. For each MAC, due to adopting full zero skipping, the activation sparsity and weight sparsity are random. The computation cycles of MACs may vary in a wide range. The computation time of each PE is determined by the MAC with the heaviest loading, leading to idle hardware and speed declination. As stated in Section III-B, MACs are responsible for completing the divided tasks. As shown in Fig. 12, a scheduler is designed to allocate new tasks to idle MACs. A look-up table (LUT) in the scheduler records the task status of the small tasks ("0" indicates "finished" and "1" indicates "not finished"). Each PE processes 32 channels (C) ×16 heights (H) ×16 widths (W) of the input slice as a basic unit. A filter size of $D = S = 3$ is usually applied in classical 3D-CNNs. Therefore, the LUT in the scheduler is designed to contain 16×27 ($H = 16$ and $D \times R \times S = 27$) bits. Therefore, during the convolution process, the scheduler is aware of which weight bundle should be convolved with a specific activation bundle, according to the internal LUT. The LUT can be initially configured to adapt different convolutions in which the constraint of $D \times R \times S \leq 27$ and $D \leq 3$ is satisfied. Meanwhile, the MAC utilization does not decrease because the scheduler can allocate sufficient workload to each MAC. Once finishing a task, a MAC issues a "Request" signal. Once ready to receive the partial sums, the partial sum register issues a "Ready_PRF" signal. "Ready_PRF" indicates which weights should be selected and sent to the first-stage arbiter (Arbiter_1). Thus, the task status in the LUT needs to be performed AND with "Ready_PRF" one by one. Arbiter_1 generates which weight bundle (ID_WB) needs to be convolved, while the second-stage arbiter (Arbiter_2) generates which activation bundle (ID_AB) needs to be convolved. ID_WB and ID_AB are used to update the LUT by setting the corresponding task status to "0", indicating the task is allocated. Furthermore, by directly arbitrating the "Request" signal, Arbiter_3 generates which MAC (ID_MAC) to receive ID_WB and ID_AB. The scheduler informs the partial sum register to receive the result of ID_MAC. Overall, the scheduling information decides where the inputs of MAC are from and where the outputs of MAC are transferred. This method is with low hardware and energy overheads due to the compatibility with the channel-first dataflow. Only 2.6% of the PE area is occupied by the scheduler.

The inter-PE load-balancing is to balance loads among PEs. In the proposed accelerator, the 16 PEs share the same activations but perform convolution with different filters. The sparsity difference among filters leads to load imbalance among PEs. To understand the inter-PE load-balancing, consider the Conv2 layer with 128 filters as an example as shown in Fig. 13. There is a strong correlation between the sparsity of a filter and the convolution time. The PEs with sparser filters have to wait until the PE with the least sparse filter finishes the task, which degrades the resource utilization of the accelerator.

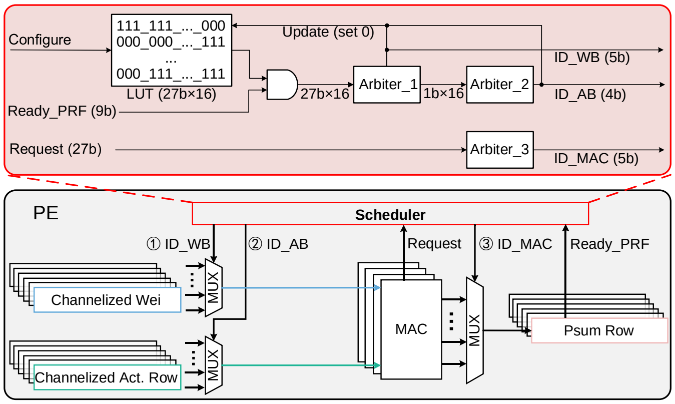


Fig. 12. Illustration of the intra-PE load balancing scheme.

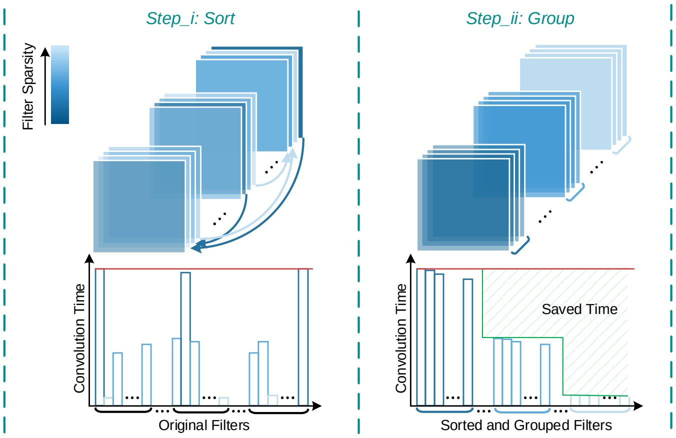


Fig. 13. Filter re-ordering for inter-PE load balancing.

The proposed inter-PE load balancing addresses this issue in two steps. *Step_i*: All filters of each layer are sorted offline in the ascending order of their sparsity as shown in Fig. 13. The adjacent filters after re-ordering therefore have relatively close computation time. *Step_ii*: Since there are 16 PEs in total, the sorted filters are divided into multiple groups, each of which contains 16 filters that are fed into 16 PEs. In this way, the calculation time is close among the filters in each group. The computation time of 16 PEs is therefore close to each other as well. Especially for the deep convolution layers, the number of filters is larger and the channel is deeper (e.g. 512 filters and 512 channels in Layer 5, 6, and 7 of 3D U-Net), compared to the shallow layers. The calculation time is therefore more proportional to the weight sparsity in the filters in statistics, thereby achieving even closer computation time in each group compared to the shallow layers. In Fig. 13, the thick red line represents the convolution time of the filters in the original order, while the thick green stepped line indicates the convolution time of the sorted filters. The green anti-diagonal area represents the convolution time savings with the inter-PE load balancing. Note that STICKER [26] requires an external CPU to constantly re-arrange the activations during every convolution process. However, the substantial overhead of re-arranging activations is not included in the evaluation of STICKER. Alternatively, our proposed inter-PE load-balancing only requires sorting the trained filters offline once, resulting in no overhead in the later convolutions. The power consumption of the sorting is negligible compared to the total power consumption of thousands of inferences.

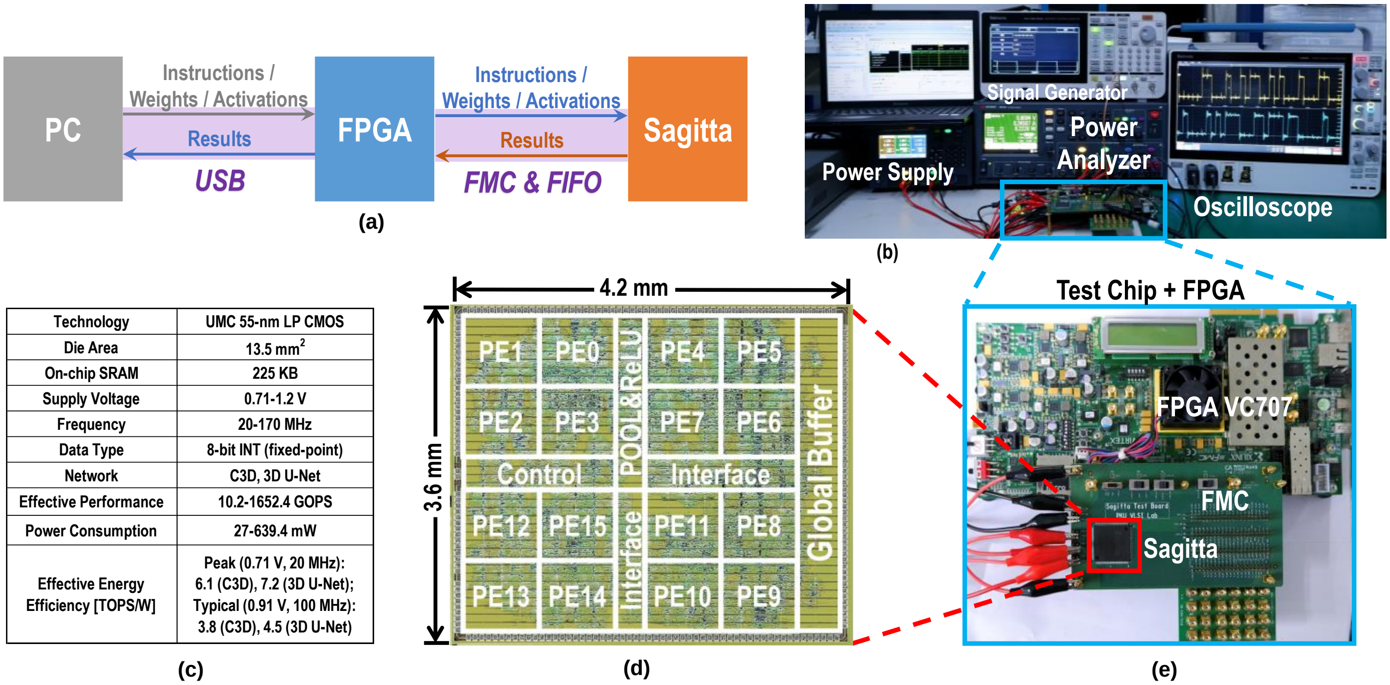
V. Experimental Results

In this section, the simulation and measurement results of the proposed 3D-CNN accelerator Sagitta are presented. The performance of Sagitta is also compared with the state of the art.

A. Experiment Setup

Sagitta is fabricated in the UMC 55-nm low-power CMOS technology and sealed in a LQFP208 package. Three types of standard cells (low threshold voltage (LVT), regular threshold voltage (RVT), and high threshold voltage (HVT)) and seven metal layers are used in Sagitta. The physical design is performed with Cadence Innovus. The total chip area (including I/O pads) is 4.2 mm×3.6 mm, while the core area of Sagitta is 13.5 mm^2, both after shrinking. The die micrograph, test platform, and measurement setup are shown in Fig. 14. The printed circuit board (PCB) with Sagitta is connected to an FPGA board (Xilinx Virtex-7 FPGA VC707 Evaluation board [53]) via an FPGA Mezzanine Card (FMC) connector board. Xilinx Vivado Design Suite is used to program the FPGA board. The power analyzer Keysight N6705C is used to measure the real-time power consumption of Sagitta. The personal computer (PC) loads instructions, weights, and activations into the double data rate synchronous dynamic random access memory (DDR SDRAM) on the FPGA board. Sagitta fetches instructions, weights, and activations from the FPGA board and returns the inference results to FPGA using an asynchronous FIFO (designed on Sagitta) via FMC. FPGA then sends the interference results to PC via the universal serial bus (USB). The signal generator feeds a clock signal to Sagitta as the system clock for the core.

The two benchmarks that are introduced in the previous sections are used for the evaluation of Sagitta: C3D classifying UCF101 and 3D U-Net segmenting BraTS 2020, which are trained in PyTorch. In particular, the open-source tool Distiller is used for pruning, quantizing activations and weights to 8-bit integers (INT8) and partial sums to 32-bit integers (INT32), and retraining C3D and 3D U-Net to recover the accuracy.



Technology	UMC 55-nm LP CMOS
Die Area	13.5 mm^2
On-chip SRAM	225 KB
Supply Voltage	0.71-1.2 V
Frequency	20-170 MHz
Data Type	8-bit INT (fixed-point)
Network	C3D, 3D U-Net
Effective Performance	10.2-1652.4 GOPS
Power Consumption	27-639.4 mW

Effective Energy Efficiency [TOPS/W]	Peak (0.71 V, 20 MHz):
	6.1 (C3D), 7.2 (3D U-Net);
	Typical (0.91 V, 100 MHz):
	3.8 (C3D), 4.5 (3D U-Net)

Fig. 14. (a) The diagram of the chip test system. (b) The measurement platform. (c) Summary of the chip measurement results. (d) The micrograph of Sagitta. (e) The connections between the PCB board with Sagitta, the FMC connector board, and the FPGA evaluation board.

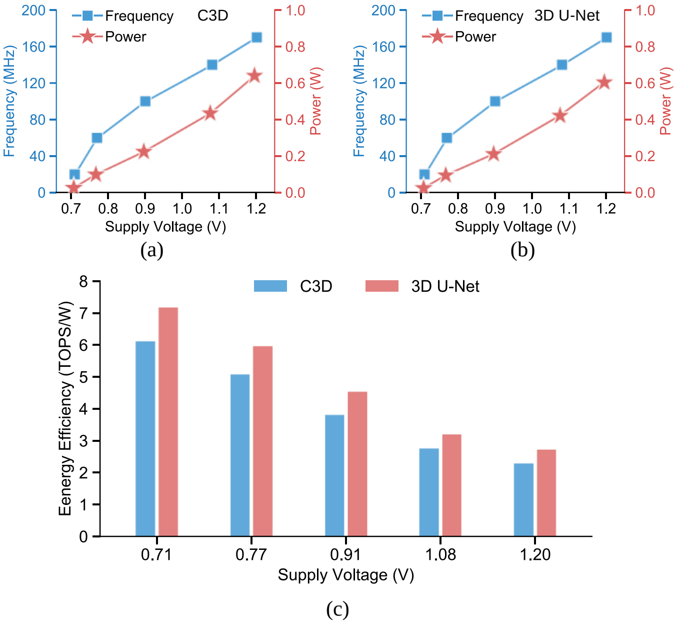


Fig. 15. The measured frequency and power consumption of Sagitta at different supply voltages while running (a) C3D and (b) 3D U-Net. (c) The measured effective energy efficiency of Sagitta at different supply voltages.

B. Performance Evaluation

The performance, power consumption, and energy efficiency of Sagitta for running C3D and 3D U-Net at different supply voltages are shown in Fig. 15. The maximum frequency is 170 MHz at 1.2 V where the maximum power consumption is 639.4 mW for C3D and 605.3 mW for 3D U-Net. The ideal peak performance of Sagitta is 146.88 GOPS @ 170 MHz without considering data sparsity (0% sparsity of activations and weights). At the typical operating point of 0.91 V and 100 MHz, with 0% sparsity of activations and weights, Sagitta achieves a performance of 50 GOPS and an energy efficiency of 0.17 TOPS/W. The minimum operating voltage of Sagitta is 0.71 V, at which the maximum achievable frequency is 20 MHz. The maximum energy efficiency of Sagitta is also achieved at 0.71 V and 20 MHz for both benchmarks, which is 6.1 TOPS/W for C3D and 7.2 TOPS/W for 3D U-Net.

The area breakdown of the Sagitta core is shown in Fig. 16a. The PE array accounts for 69.5% of the total area. The area of the PE array includes all 16 PEs. The area breakdown of each PE is shown in Fig. 16b. The buffers inside each PE take nearly half of the PE area, which is 2.1 × larger than that of the global buffer. Overall, the on-chip storage, including the global buffer and the buffers inside PEs, takes approximately half of the total area while the MACs from all 16 PEs only account for 16.3%.

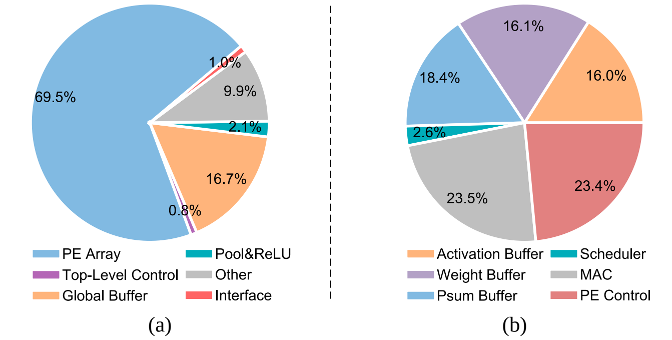
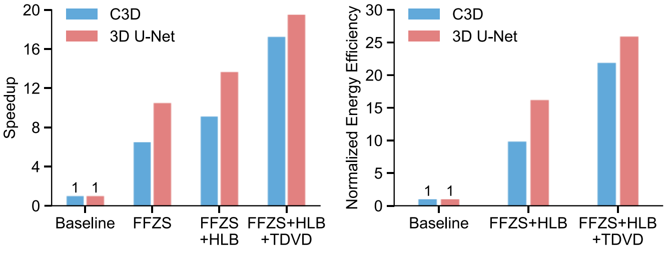


Fig. 16. The area breakdown of (a) the Sagitta core and (b) one PE.

Sagitta can be configured to disable the three proposed techniques (TDVD, FFZS, and HLB) to become a baseline version to evaluate the effectiveness of those techniques in more details. For the baseline evaluation, TDVD is not used for activations and pruning is not used for weights, because the baseline cannot skip zeros in both activations and weights anyway. The improvement by using the three proposed techniques in C3D and 3D U-Net at 0.91 V and 100 MHz is shown in Fig. 17. The speedup is obtained by simulating the register transfer level (RTL) code of Sagitta.

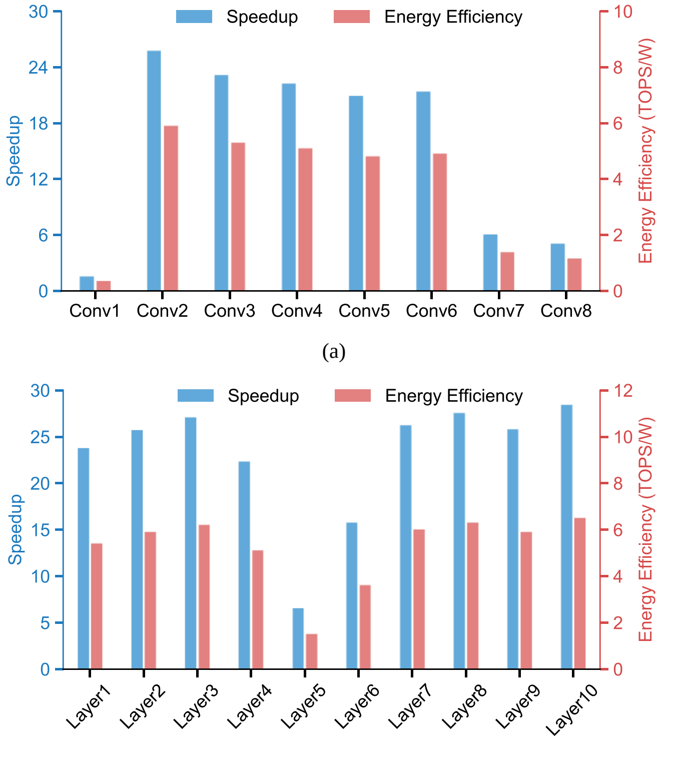


(a) (b)

Fig. 17. The (a) speedup and (b) normalized (measured) effective energy efficiency by using the proposed techniques in C3D and 3D U-Net.

Pruning significantly increases the number of zeros in weights. After pruning, the average sparsity of weights with C3D and 3D U-Net is 97.6% and 96.3%, respectively. To evaluate the effectiveness of FFZS alone, TDVD is not applied. Therefore, the average sparsity of activations with C3D and 3D U-Net is still maintained at 54.5% and 89.3%, respectively. Because the proposed HLB cannot be disabled on the chip, the energy efficiency with FFZS alone is unavailable. The speedup with FFZS alone can be obtained by simulating the RTL code of Sagitta. As shown in Fig. 17a, with the proposed FFZS alone, the speedup of the 3D-CNN accelerator is 6.5× for C3D and 10.5× for 3D U-Net, compared to the baseline. Note that the acceleration effect of FFZS is more significant for 3D U-Net than C3D because the sparsity of activations in 3D U-Net is dramatically higher than in C3D. Then, HLB is used together with FFZS. Compared to using FFZS alone, the speed of the accelerator is boosted by 1.4× for C3D and 1.3× for 3D U-Net. Compared to the baseline, the energy efficiency of the accelerator is enhanced by 9.8× for C3D and 16.2× for 3D U-Net, by using FFZS and HLB together. Afterwards, TDVD is also applied to enhance the sparsity of activations. The sparsity of activation is increased to 90.8% in C3D and 98.6% in 3D U-Net. By combining TDVD, FFZS, and HLB together, the speed is enhanced by 1.9× for C3D and 1.4× for 3D U-Net, compared to only using FFZS and HLB. The energy efficiency of the accelerator is further increased by 2.2× for C3D and 1.6× for 3D U-Net, compared to only using FFZS and HLB. Furthermore, by combining TDVD, FFZS, and HLB together, the speed is accelerated by 17.2× for C3D and 19.5× for 3D U-Net, compared to the baseline. The accelerator takes only 0.1 s to process C3D and 0.9 s to process 3D U-Net. The energy efficiency of the accelerator is increased by 21.8× for C3D and 25.9× for 3D U-Net, compared to the baseline. 3D U-Net is different from C3D in terms of network architecture, input of spatial volumetric, and learning task of semantic segmentation. Even so, benefitting from the specified hardware architecture for 3D convolution and the proposed techniques, the chip test results still show that the proposed work enhances the energy efficiency of 3D U-Net significantly. This validates the generalization capability of the proposed hardware architecture and techniques on 3D-CNN. The comparisons between the baseline and the optimized version of Sagitta for running C3D from 20 MHz to 170 MHz are summarized in Table I.

The speedup and effective energy efficiency of each layer in C3D and 3D U-Net are demonstrated in Fig. 18. The speedup and energy efficiency vary dramatically among layers. It is due to the fact that apart from the sparsity of activations and weights, the size of input feature maps and filters is the key parameter that influences the performance of the chip. In detail, each PE processes 32 channels×16 heights×16 widths as the basic unit. However, the first convolution layer in C3D has only three channels, leading to an 8.7× lower active MAC rate compared to the average active MAC rate of the whole network. Meanwhile, the seventh convolution layer in C3D has only 7 heights×7 widths, resulting in a 2.5× lower active MAC rate compared to the average active MAC rate of the whole network. As the sparsity grows, the read/write collisions in the register files of activations, weights, and partial sums tend to be more severe and the overhead of flag, including storage and movement, becomes heavier. For these two reasons, the active MAC rate decreases, limiting the peak energy efficiency. To alleviate the impact of the second factor, the amount of register files inside each PE can be increased to enhance the performance. To double the peak performance of the accelerator, the total size of register files in the PEs needs to be increased by 2.3×, which would lead to ~44.5% increase of the total chip area.



(a)

(b)

Fig. 18. The speedup and measured effective energy efficiency of each layer in (a) C3D and (b) 3D U-Net.

TABLE I
COMPARISONS BETWEEN THE BASELINE AND THE OPTIMIZED VERSION OF SAGITTA FOR RUNNING C3D FROM 20 MHZ TO 170 MHZ

	Effective Performance [GOPS]	Power Consumption [mW]	Effective Area Efficiency [GOPS/mm^2]	Peak Effective Energy Efficiency [TOPS/W] @ (0.71 V, 20 MHz)	Typical Effective Energy Efficiency [TOPS/W] @ (0.91V, 100 MHz)
Baseline Version	10.2-86.7	31.3-694.5	0.8-6.4	0.3	0.2
Optimized Version	174.3-1481.6	28.4-639.4	12.9-109.7	6.1	3.8

TABLE II
COMPARISONS OF SAGITTA WITH THE STATE OF THE ART

	TwoNullHop [31] 2020	Samsung Butterfly [54] ISSCC 2019	MediaTek [52] ISSCC 2020	STICKER [26] JSSC 2020	[27] ISSCC 2020	SNAP [30] JSSC 2021	[35] TCAD 2020	[37] DAC 2020	**This work**	
Technology	GF 28 nm	Samsung 8 nm	7 nm	TSMC 65 nm	TSMC 65 nm	TSMC 16 nm/65 nm*	FPGA 14 nm VCU 118	FPGA 16 nm ZCU 102	**UMC 55 nm**	
Tapeout	No	Yes	Yes	Yes	Yes	Yes/No*	No	No	**Yes**	
Area [mm^2]	4.0	5.5	3.04	12	12	2.3/9.3	N/A	N/A	**13.5**	
SRAM [KB]	256	1568	2176	170	196	280.6	N/A	N/A	**225**	
MAC	128	1024 (8b)/ 512 (16b)	1024 (8b)/ 512 (16b)	256	64 (8b)/ 128 (4b)	252	3224 (DSP)	N/A	**432**	
Supply Voltage [V]	1.0	0.5-0.8	0.575-0.825	0.67-1.0	0.51-1.0	0.55-0.80/1.0*	N/A	N/A	**0.71-1.2**	
Frequency [MHz]	500	67-933	290-880	20-200	25-100	33-480/250*	200	150	**20-170**	
Data Type	A: 16b; W: 8b	8b/16b	8b/16b	8b	4b/8b	16b/8b*	16b	16b	**8b**	
Network Type	VGG19, Proteins, S-Multires-Edge, ResNet-20, RoshamboNet	Inception-v3	MobileNet-v1, Inception-v3	AlexNet	Mobilenet-16, Mobilenet-27	AlexNet, VGG-16, ResNet-50	C3D, VGG-16	C3D, R(2+1)D	**C3D**	**3D U-Net**
Effective Performance [GOPS]	2431 @500 MHz, VGG19	6937 @8b, 933 MHz‡	3604 @880 MHz, Dense	(102-5638) @200 MHz	680 @100 MHz	347.4 @260MHz/ 370 @250 MHz*	5054	79	**174.3-1481.6**	**194.4-1652.4**
Power Consumption [mW]	238 @500 MHz, VGG19	39 @67 MHz/ 1553 @933 MHz	174-1053	20.5 @20 MHz/ 248.4 @200 MHz	7.3-99	16.3-364/500*	32000	6700	**28.4-639.4**	**27.0-605.3**
Effective Area Efficiency [GOPS/mm^2]	609 @500 MHz, VGG19	1261 @8b, 933 MHz ‡	1185 @880 MHz, Dense	(13.1-722.8) @200 MHz	56.7 @100 MHz	151 @260MHz/ 39.8 @250 MHz*	N/A	N/A	**12.9-109.7**	**14.4-122.4**
Effective Energy Efficiency [TOPS/W]	10.21 @500 MHz, VGG19	3.4 @8b, 67 MHz, Inception-v3	6.22 @290 MHz, Inception-v3/ 6.55 @290 MHz, MobileNet-v1	2.82 @100 MHz, AlexNet	0.33 @100 MHz, Mobilenet-27	3.86 @260 MHz/ 0.74 @250 MHz*, AlexNet	0.158 @ 200MHz, C3D	0.01 @150 MHz, C3D	**Peak: 6.1 @0.71 V, 20 MHz** **Typical: 3.8 @0.91 V, 100 MHz**	**Peak: 7.2 @0.71 V, 20 MHz** **Typical: 4.5 @0.91 V, 100 MHz**
Scaled Effective Energy Efficiency† [TOPS/W]	1.77 @500 MHz, VGG19	N/A	N/A	3.48 @100 MHz, AlexNet	0.41 @100 MHz, Mobilenet-27	0.91 @250 MHz*, AlexNet	N/A	N/A	**Peak: 6.1 @0.71 V, 20 MHz** **Typical: 3.8 @0.91 V, 100 MHz**	**Peak: 7.2 @0.71 V, 20 MHz** **Typical: 4.5 @0.91 V, 100 MHz**

*The logic synthesis results for the same SNAP design with 8-bit fixed-point data width in a 65-nm process.
‡Values for single layers using 5×5 kernel.
†Scaled to 55 nm technology, according to $Energy\ Efficiency \propto \frac{1}{Area*V_{DD}^2}$ derived in [55].

C. Comparisons with the State of the Art

The comparisons of the proposed 3D-CNN accelerator Sagitta with the state-of-the-art 2D-CNN accelerators are detailed in Table II. For Sagitta, the sparsity of activation and weight is 90.8% and 97.6%, respectively, for C3D, and 98.6% and 96.3%, respectively, for 3D U-Net. According to the layout shrinking rules of TSMC and UMC (for both 55 nm and 28 nm), the area is reduced by 0.81× (0.9^2) with the technology scaled from 65 nm to 55 nm, and the area is increased by 4× (2^2) with the technology scaled from 28 nm to 55 nm. The supply voltage is typically maintained with the technology scaled from 65 nm to 55 nm, and increased from 1.0 V to 1.2 V with the technology scaled from 28 nm to 55 nm. With the area and supply voltage scaling, the energy efficiency of TwoNullHop, STICKER, [27], and 65-nm 8-bit fixed-point SNAP can be approximately scaled to the energy efficiency in 55-nm technology as shown in the last row of Table II. The energy efficiency scaling of Samsung Butterfly [54] and MediaTek [52] is inaccurate and thus not given because their technologies are far more advanced than 55 nm. Even when scaled by only 2× (the area scaling factor is 4 from 28 nm to 55 nm and the supply voltage scaling is not even considered), Sagitta has already outperformed Samsung Butterfly [54] and MediaTek [52] running real networks in terms of energy efficiency. While running C3D, Sagitta improves the effective energy efficiency by 9.3×, 4.2×, 2.1×, and 1.1×, compared with [27], 65-nm 8-bit fixed-point SNAP

[30], TwoNullHop [31], and STICKER [26], respectively. While running 3D U-Net, Sagitta improves the effective energy efficiency by 11×, 4.9×, 2.6×, and 1.3×, compared with [27], 65-nm 8-bit fixed-point SNAP, TwoNullHop, and STICKER, respectively. One important reason why Sagitta achieves higher energy efficiency compared to these 2D-CNN accelerators, is that the on-chip weights and activations are efficiently reused over the dimension T and the dimension D , respectively.

The state-of-the-art 3D-CNN accelerators are also compared with Sagitta in Table II. To the best of our knowledge, Sagitta is the first taped-out 3D-CNN accelerator. For running C3D, compared to [35] operating at 200 MHz and [37] operating at 150 MHz, Sagitta achieves 24.1×and 379.6×energy efficiency improvement, respectively, at 100 MHz. In summary, Sagitta outperforms the state-of-the-art 2D-CNN accelerators and 3D-CNN accelerators in terms of energy efficiency.

VI. Conclusions

In this paper, Sagitta, an energy-efficient real-time 3D-CNN accelerator deeply exploiting the data sparsity, is proposed. To benefit from the sparsity of both activations and weights, a full-zero-skipping microarchitecture is proposed to skip both the zero activations and weights for speedup and energy savings. A hierarchical load-balancing scheme is employed to mitigate the load imbalance among and inside PEs after zero skipping. To amplify the effect of zero skipping, a threshold differential value dropout method is proposed to increase the sparsity of activations in the convolution layers. All the proposed techniques are enabled on the system architecture with specialized computation flow. The proposed 3D-CNN accelerator is fabricated in the UMC 55-nm low power CMOS technology. Benchmarked with the C3D network classifying the UCF101 dataset and 3D U-Net segmenting the BraTS 2020 dataset, the proposed accelerator achieves an effective energy efficiency of 3.8 TOPS/W and 4.5 TOPS/W, and a latency of 0.1 s and 0.9 s, respectively, with all the optimization techniques while running at 100 MHz with 0.91 V supply voltage. Compared to the state-of-the-art 3D-CNN and 2D-CNN accelerators, Sagitta enhances the energy efficiency by up to 379.6×and 11×, respectively, thereby providing an attractive option for edge computing.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. of the International Conference on Neural Information Processing Systems*, vol. 25, Dec. 2012, pp. 1097-1105.

[2] Y. Kim, “Convolutional neural networks for sentence classification,” in *Proc. of the Conference on Empirical Methods in Natural Language Processing*, Oct. 2014, pp. 1746-1751.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580-587.

[4] Z. Liu, H. Tang, Y. Lin, and S. Han, “Point-voxel CNN for efficient 3D deep learning,” in *Proc. of the International Conference on Neural Information Processing Systems*, vol. 32, Dec. 2019, pp. 965-975.

[5] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F.-F. Li, “Large-scale video classification with convolutional neural networks.” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1725-1732.

[6] Y. Zhang, L. Shi, Y. Wu, K. Cheng, J. Cheng, and H. Lu, “Gesture recognition based on deep deformable 3D convolutional neural networks,” *Elsevier Pattern Recognition*, vol. 107, pp. 107416-107427, Jun. 2020.

[7] Z. Wang, S. Yue, and C. Song, “Video-based air quality measurement with dual-channel 3-D convolutional network,” *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 14372-14384, Sept. 2021.

[8] G. D. de Dinechin, A. Paljic, “Automatic generation of interactive 3D characters and scenes for virtual reality from a single-viewpoint 360-degree video,” in *Proc. of the IEEE Conference on Virtual Reality and 3D User Interfaces*, Mar. 2019, pp. 908-909.

[9] G. Kim, K. Lee, Y. Kim, S. Park, I. Hong, K. Bong, H.-J. Yoo, “A 1.22 TOPS and 1.52 mW/MHz augmented reality multicore processor with neural network NoC for HMD applications,” *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 113-124, Jan. 2015.

[10] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, “LaserNet: An efficient probabilistic 3D object detector for autonomous driving,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2019, pp. 12677-12686.

[11] Z. Liang, Y. Guo, Y. Feng, W. Chen, L. Qiao, L. Zhou, J. Zhang, and H. Liu, “Stereo matching using multi-level cost volume and multi-scale feature constancy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 300–315, Jan. 2021.

[12] A. R. Ozcan and S. Erturk, “Seizure prediction in scalp EEG using 3D convolutional neural networks with an image-based approach,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 11, pp. 2284-2293, Nov. 2019.

[13] D. Botina-Monsalve, Y. Benezeth, and J. Miteran, “RTrPPG: An ultra light 3DCNN for real-time remote photoplethysmography,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2022, pp. 2146-2154.

[14] L. Kastner, V. C. Frasineanu, and J. Lambrecht, “A 3D-deep-learning-based augmented reality calibration method for robotic environments using depth sensor data,” in *Proc. of the IEEE International Conference on Robotics and Automation*, May 2020, pp. 1135-1141.

[15] S. P. Singh, L. Wang, S. Gupta, H. Goli, P. Padmanabhan, and B. Guly ás, “3D deep learning on medical images: A review,” *MDPI Sensors*, vol. 20, no. 18, Sept. 2020.

[16] D. Maturana and S. Scherer, “VoxNet: A 3D convolutional neural network for real-time object recognition,” in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept. 2015, pp. 922-928.

[17] R. Janovský, D. Sedláček, and J. Žára, “On improving 3D U-net architecture,” in *Proc. of the International Conference on Software Technologies*, Jul. 2019, pp. 649–656.

[18] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3D convolutional networks,” in *Proc. of the IEEE International Conference on Computer Vision*, Dec. 2015, pp. 4489-4497.

[19] O. Cicek, A. Abdulkadir, S.S. Lienkamp, T. Brox, and O. Ronneberger, “3D U-Net: Learning dense volumetric segmentation from sparse annotation,” in *Proc. of the Springer International Conference on Medical Image Computing and Computer-Assisted Intervention*, vol. 9901, pp. 424-432, Oct. 2016.

[20] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3D ShapeNets: A deep representation for volumetric shape modeling,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015, pp. 1912-1920.

[21] 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,” in *Proc. of the ACM/IEEE Annual International Symposium on Computer Architecture*, Jun. 2017, pp. 27-40.

[22] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, “Cnvlutin: Ineffectual-neuron-free deep neural network computing,” in *Proc. of the ACM/IEEE Annual International Symposium on Computer Architecture*, Jun. 2016, pp. 1-13.

[23] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, “Efficient processing of deep neural networks: A tutorial and survey,” *Proc. of the IEEE*, vol. 105, no. 12, pp. 2295-2329, Dec. 2017.

[24] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I.-A. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S.-C. Liu, and T. Delbruck, “A flexible convolutional neural network accelerator based on sparse representations of feature maps,” *IEEE Transactions on*

Neural Networks and Learning Systems, vol. 30, no. 3, pp. 644–656, Mar. 2019.

[25]M. Liu, Y. He, and H. Jiao, "Efficient zero-activation-skipping for on-chip low-energy CNN acceleration," in *Proc. of the IEEE International Conference on Artificial Intelligence Circuits and Systems*, Jun. 2021, pp. 1-4.

[26]Z. Yuan, Y. Liu, J. Yue, Y. Yang, J. Wang, X. Feng, J. Zhao, X. Li, and H. Yang, "STICKER: An energy-efficient multi-sparsity compatible accelerator for convolutional neural networks in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 2, pp. 465-477, Feb. 2020.

[27]Z. Yuan, Y. Yang, J. Yue, R. Liu, X. Feng, Z. Lin, X. Wu, X. Li, H. Yang, and Y. Liu, "A 65nm 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 International Solid-State Circuits Conference (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 232-234.

[28]D. Kim, J. Ahn, and S. Yoo, "ZeNA: Zero-Aware neural network accelerator," *IEEE Design & Test*, vol. 35, no. 1, pp. 39-46, Feb. 2018.

[29]M. Mahmoud, I. Edo, A. H. Zadeh, O. M. Awad, G. Pekhimenko, J. Albericio, and A. Moshovos, "TensorDash: Exploiting sparsity to accelerate deep neural network training," in *Proc. of the IEEE/ACM Annual International Symposium on Microarchitecture*, Oct. 2020, pp. 781-795.

[30]J. F. Zhang, C. E. Lee, C. Liu, Y. S. Shao, S. W. Keckler, and Z. Zhang, "SNAP: An efficient sparse neural acceleration processor for unstructured sparse deep neural network inference," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 2, pp. 636-647, Feb. 2021.

[31]A. Aimar, "Energy-efficient convolutional neural network accelerators for edge intelligence," *Doctoral dissertation*, University of Zurich, Aug. 2021.

[32]S. Yin, P. Ouyang, J. Yang, T. Lu, X. Li, L. Liu, and S. Wei, "An energy-efficient reconfigurable processor for binary-and ternary-weight neural networks with flexible data bit width," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1120–1136, Apr. 2019.

[33]Q. Chen, Y. Huang, R. Sun, W. Song, Z. Liu, Y. Fu, and L. Li, "An efficient accelerator for multiple convolutions from the sparsity perspective," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1540–1544, Jun. 2020.

[34]S. Kim, J. Lee, S. Kang, J. Lee, and H. -J. Yoo, "A power-efficient CNN accelerator with similar feature skipping for face recognition in mobile devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 4, pp. 1181-1193, Apr. 2020.

[35]J. Shen, Y. Huang, Z. Wang, Y. Qiao, M. Wen, and C. Zhang, "Toward an efficient deep pipelined template-based architecture for accelerating the entire 2-D and 3-D CNNs on FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 7, pp. 1442-1455, Jul. 2020.

[36]K. Hegde, R. Agrawal, Y. Yao, and C. W. Fletcher, "Morph: Flexible acceleration for 3D CNN-based video understanding," in *Proc. of the IEEE/ACM Annual International Symposium on Microarchitecture*, Oct. 2018, pp. 933–946.

[37]M. Sun, P. Zhao, M. Gungor, M. Pedram, M. Leeser, and X. Lin, "3D CNN acceleration on FPGA using hardware-aware pruning," in *Proc. of the ACM/IEEE Design Automation Conference*, Jul. 2020, pp. 1-6.

[38]H. Deng, J. Wang, H. Ye, S. Xiao, X. Meng, and Z. Yu, "3D-VNPU: A flexible accelerator for 2D/3D CNNs on FPGA," in *Proc. of the IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, May 2021, pp. 181–185.

[39]Y. Wang, Y. Wang, C. Shi, L. Cheng, H. Li, and X. Li, "An edge 3D CNN accelerator for low-power activity recognition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 5, pp. 918-930, May 2021.

[40]C. Zhou, M. Liu, S. Qiu, Y. He, and H. Jiao, "An energy-efficient low-latency 3D-CNN accelerator leveraging temporal locality, full zero-skipping, and hierarchical load balance," in *Proc. of the ACM/IEEE Design Automation Conference*, Dec. 2021, pp. 241-246.

[41]S. Khurram, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," Dec. 2012, *arXiv:1212.0402*. [Online]. Available: https://arxiv.org/abs/1212.0402.

[42]B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, *et al.*, "The multimodal brain tumor image segmentation benchmark (BRATS)," *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp.1993-2024, Oct. 2015.

[43]K. Simonyan, and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of the International Conference on Learning Representations*, May 2015.

[44]O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. of the Springer International Conference on Medical Image Computing and Computer-Assisted Intervention*, vol. 9351, pp. 234-241, Nov. 2015.

[45]Distiller. Accessed: Apr. 12, 2022. [Online]. Available: https://github.com/IntelLabs/distiller.

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

[47]B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 2704-2713.

[48]B. L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proc. of the IEEE*, vol. 108, no. 4, pp. 485–532, Apr. 2020.

[49]M. B. Milde, D. Neil, A. Aimar, T. Delbruck, and G. Indiveri, "ADaPTION: Toolbox and benchmark for training convolutional neural networks with reduced numerical precision weights and activation," Nov. 2017, *arXiv:1711.04713*. [Online]. Available: https://arxiv.org/abs/1711.04713.

[50]K.-C. Chen, M. Ebrahimi, T.-Y. Wang, and Y. Yang, "NoC-based DNN accelerator: A future design paradigm," in *Proc. of the IEEE/ACM International Symposium on Networks-on-Chip*, Oct. 2019, pp. 1-8.

[51]Y. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE International Solid-State Circuits Conference (ISSCC) Dig. Tech. Papers*, Jan. 2016, pp. 262-263.

[52]C.-H. Lin, C.-C. Cheng, Y.-M. Tsai, *et al.*, "A 3.4-to-13.3TOPS/W 3.6TOPS dual-core deep-learning accelerator for versatile AI applications in 7nm 5G Smartphone SoC," in *IEEE International Solid-State Circuits Conference (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 134-136.

[53]Xilinx. *Xilinx Virtex-7 FPGA VC707 Evaluation Kit*. Accessed: Apr. 9, 2022. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html.

[54]J. Song, Y. Cho, J. S. Park, J. Woo Jang, S. Leev, J. H. Song, J. G. Lee, and I. Kang. "An 11.5TOPS/W 1024-MAC butterfly structure dual-core sparsity-aware neural processing unit in 8nm flagship mobile SoC," in *IEEE International Solid-State Circuits Conference (ISSCC) Dig. Tech. Papers*, Mar. 2019, pp. 130–132.

[55]V. Sze, Y. Chen, T. Yang, and J. S. Emer, "How to understand and evaluate deep learning processors," *IEEE Solid-State Circuits Magazine*, pp. 28–41, Aug. 2020.

Changchun Zhou received the Bachelor degree from Sun Yat-sen University, Guangzhou, China, in 2018. He is currently pursuing his Ph. D. degree in Shenzhen Graduate School, Peking University, Shenzhen, China. His research interest is energy-efficient neural network accelerators for edge computing.

Min Liu received the Bachelor degree in electronic science and technology from the Harbin Institute of Technology, Weihai, China, in 2017. She is currently pursuing the Ph.D. degree in Shenzhen Graduate School, Peking University, Shenzhen, China. Her research interest is energy-efficient on-chip deep learning acceleration.

Siyuan Qiu received the Bachelor degree from Huazhong University of Science and Technology, Wuhan, China, in 2016. He is currently pursuing his Ph. D. degree in Shenzhen Graduate School, Peking University, Shenzhen, China. His research interest is ultra-low power on-chip machine learning for biomedical applications.

Xugang Cao received the Bachelor degree from Dalian University of Technology, Dalian, China, in 2018 and the Master degree from Shenzhen Graduate School, Peking University, Shenzhen, China, in 2021. His research interest is low power error-resilient circuits.

Yuzhe Fu received the Bachelor degree from Southern University of Science and Technology, Shenzhen, China, in 2021. He is currently pursuing his M.S. degree in Shenzhen Graduate School, Peking University, Shenzhen, China. His research interests are algorithm-hardware co-design and energy-efficient neural network acceleration.

YiFan He received his M.S. degree (cum laude) and Ph.D. degree in electrical engineering from the Eindhoven University of Technology (TU/e), The Netherlands, in 2008 and 2013 respectively. He is now the CTO of Reconova Co. Ltd. His current research interests include deep neural networks, low-power computing systems, and computer architectures of AI chips. He has published over 50 articles in international journals and conferences.

Hailong Jiao received the Bachelor degree from Shandong University, Shandong, China, in 2004, the Master degree from the Institute of Microelectronics, Chinese Academy of Sciences, Beijing, China, in 2008, and the Ph. D. degree from the Hong Kong University of Science and Technology, Hong Kong, China, in 2012.

He served as an Assistant Professor in the Electronic Systems group with the Department of Electrical Engineering, Eindhoven University of Technology (TU/e), Eindhoven, The Netherlands from September 2013 to January 2017, and got tenured position in September 2016. He was with IMEC, Leuven, Belgium as a part-time Visiting Researcher from February 2015 to January 2017. He has been an Associate Professor in the School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University since January 2017, and got tenured position in January 2023.

Dr. Jiao's research area is low-power and variations-resilient VLSI circuit and system design. He has interests in ultra-low voltage circuits, error-resilient systems, in-memory computing, and machine learning in VLSI. He also has interests in emerging devices and integration techniques, as well as on-chip bio-signal processing.