Quantitative Analysis of Various 2D CNN Structures based on Dataflow

2023 International Conference on Electronics, Information, and Communication (ICEIC) | 979-8-3503-2021-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICEIC57457.2023.10049910

Sangwon Lee
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
swlee.cas@gmail.com

Jiho Park
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
jhpark.cas@gmail.com

Jeongho Kim
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
jhkim.cas@gmail.com

Yongtaek Hwang
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
ythwang.cas@gmail.com

Soyeon Choi
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
sychoi.cas@gmail.com

Hoyoung Yoo
Dept. of Electronics Engineering
Chungnam National University
Daejeon, Korea
hyyoo@cnu.ac.kr

***Abstract*—Convolutional Neural Networks (CNNs) are used in a wide range of fields due to their excellent accuracy. Previous researchers have proposed convolution architectures for the typical convolutional layer in CNN whose input is larger than the kernel size. However, neural networks are continuously evolving and developing, and there is a deep convolutional layer unlike classic CNNs demands a larger kernel size than the input. In this paper, we conduct a quantitative analysis based on dataflow for various CNN structures. A total of eight 2D CNN structures are described and compared in terms of processing time, total area, and energy efficiency, based on different dataflow graphs. As a result, the comparison provides advantages and disadvantages of the different CNN structures and aids in determining the optimal hardware structure solution for various neural network types.**

Keywords— convolutional neural networks, convolutional architecture, dataflow

I. INTRODUCTION

Due to their high precision, artificial neural networks that imitate the behavior of the human brain have been widely used in many applications, including computer vision, speech recognition, and language translation. Among the various types of neural networks (NNs), convolutional neural networks (CNNs) outperform traditional signal processing techniques, especially in the field of image recognition [1]. Convolution comprises about 90 percent of CNN's overall computations and conducts matrix multiplication [2]. In general CNN structure, multiple processing elements (PEs) are often run in parallel to accelerate the CNN hardware accelerator's computations. The growth of PEs enables the simultaneous processing of complicated matrix products. One of the problems of implementing CNN hardware accelerators is the significant energy consumption caused by memory access to store the output and give the input necessary for the matrix product.

Various studies have been conducted to optimize memory structure and data flow to reduce energy consumption due to excessive memory access [3-5]. [3] utilizes a hierarchical memory structure and [4] proposes the application of FIFO memory to efficiently manage input/output data and minimize unnecessary memory access. While [3-4] focus on memory structures, [5] presents a novel approach to optimizing PE and memory configurations through data flow optimization. [5] has analyzed the access pattern of memory by focusing on dataflow and proposed a hardware structure capable of minimizing memory access. More precisely speaking, three data flows are presented for input, kernel, and output, which are the basic components of convolution, and a total of 27 combinations are presented for a single convolution configuration. Of the total 27 combinations, four promising candidates were determined, and the final structure was proposed through analytic comparison. Research [5] has great significance in analyzing data flows and providing a framework for convolution configuration, but unlike the recent development of neural networks, it identifies a promising candidate based on the classic neural networks, which have limitations in final structure selection. Recently, there have been neural networks such as deep convolution where the kernel size is larger than the input size, and in this case, the promising candidate suggested in [5] might be changed. Therefore, this paper presents a total of eight candidates by adding four new promising candidates to the four proposed candidates determined in [5] and analyzes the advantages and disadvantages of each candidate. By comparing and analyzing inputs and kernels of various sizes, it is possible to provide an optimal hardware structure solution for various types of neural networks as well as the classic neural networks

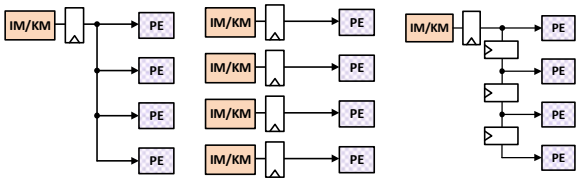
II. BACKGROUND

A CNN is a composition of multi-layered convolution to perform a given neural network operation. A single CNN is scalable from 1-dimension to 3-dimension, and we focus on the 2-dimension (2D). As shown in Equation (1), the 2D convolution operation takes $f_{(y+b)(x+a)}$ as an input, performs a convolution operation with kernel w_{ba}, and outputs o_{yx}.

$$o_{yx} = \sum_{b=0}^{K-1} \sum_{a=0}^{K-1} w_{ba} \times f_{(y+b)(x+a)} \,, \qquad (1)$$

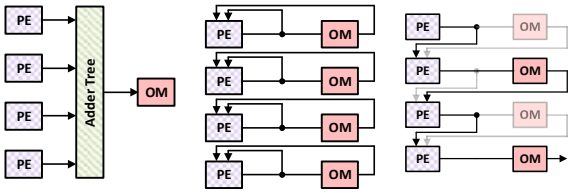
Where x, y have a range of output indexes ($0 \le x \le L\text{-}K$) and a, b have a range of kernel indexes ($0 \le a, b \le K$). Note that L and K indicate the size of the input and the kernel, respectively, and the indices satisfy $y+b$, $x+a = L$.

To analyze the operation of (1) as data flow, [5] describes the flow of data transmission between PE and memory in three ways: broadcast, stay, and forward. Broadcast refers to a data flow in which data stored in memory is connected to the entire PE and transmitted at once. Next, stay refers to a data flow in



(a) Broadcast (b) Stay (c) Forward

Figure 1. Input and kernel load scheme.



(a) Broadcast (b) Stay (c) Forward

Figure 2. Output store scheme.

which data stored in the memory is allocated to each PE and used as a dedicated memory. Lastly, forward refers to a data flow in which data stored in a memory is sequentially transmitted from a PE to a neighboring PE. Based on the three basic data flows, Fig. 1 shows the hardware structure when input data and kernel weight are loaded, and Fig. 2 shows the hardware structure when storing output data, respectively. Broadcast in Fig. 1(a) and Fig. 2(a) implements an architecture in a form in which one input memory (IM), kernel memory (KM), or output memory (OM) is connected to all the PEs. Stay in Fig. 1(b) and Fig. 2(b) has a form in which IM, KM, or OM memory allocated independently to one PE is attributed. Lastly, Fig. 1(c) and Fig. 2(c) have the data flow of forward in which data from IM, KM, or OM memory is sequentially transferred to several PEs. In summary, since the data flow of broadcast, stay, and forward can be independently obtained for input, kernel, and output, there are a total of 27 architectures to perform (1). In this paper, the architecture is named after the first letter of the data flow assigned to the input, kernel, and output. For example, BSF represents a structure that operates as broadcast for input, stay for the kernel, and forward for output.

III. Analysis of 2D CNN Structure

In this paper, the analysis was conducted on a total of 27 components by reflecting the three basic data flows that input, kernel, and output can have, and 8 promiscuous candidates were identified from a practical point of view. Previous research [5] has great significance in analyzing the data flow and providing the basis for the convolution configuration. However, the promising candidates are determined based only on the typical neural network, which limits the final structure selection. In other words, through the recent innovative development of neural networks, the criteria for the most promising candidate can vary in addition to the general criteria. We present a total of eight structures by adding four candidates considering advanced neural networks in addition to the four promising candidates suggested in [5] and compare their advantages and disadvantages. Before presenting the selected eight candidates, the excluded configuration from a total of 27 combinations and the reason will be described. First, it is the case that additional add trees are needed to match the data flow. This includes BBX and SSX or XXB. Where X means don't care, and broadcast, stay, and forward are all applicable. Second, it is the case where a combination logic

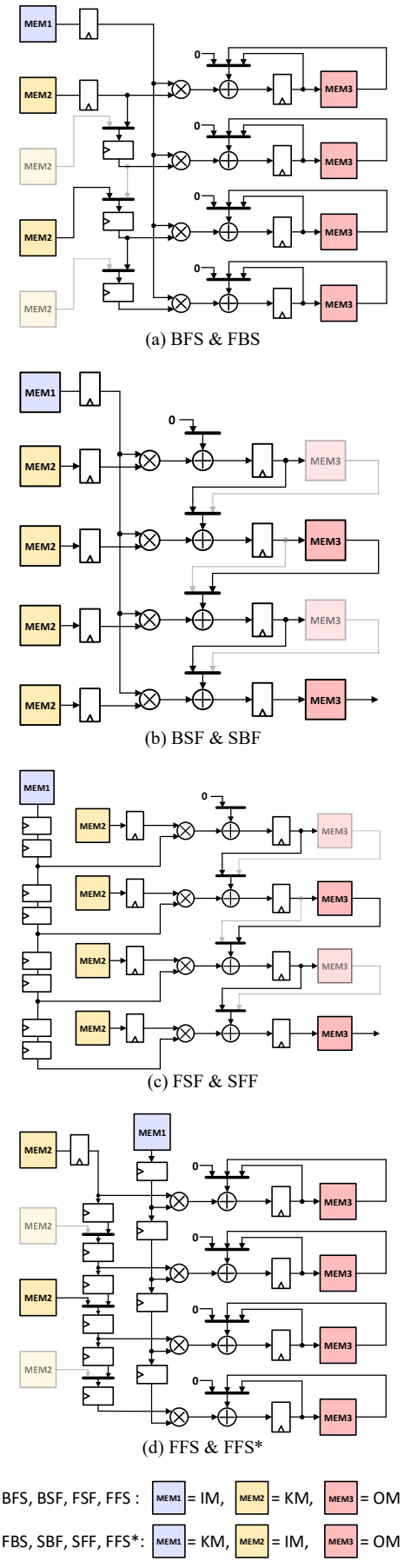


Figure 3. 2D convolutional architectures for eight structure.

TABLE I. HARDWARE RESOURCES REQUIRED FOR 2D CONVOLUTION.

Architecture	Processing Time	Minimum # of PE*	# of Registers			Memory Size			Memory Access		
			Input	Kernel	Output	IM	KM	OM	IM	KM	OM
BFS[5]	L^2	K^2	L^2	KL^3	K^2L^2	L	K^3	KL	L^2	KL	KL^2
BSF[5]	L^2	K^2	L^2	K^2	K^2L^2	L	K^2	K^2L	L^2	K^2	KL^2
FSF[5]	L^2	K^2	K^2L^2	K^2	K^2L^2	L	K^2	K^2L	L^2	K^2	KL^2
FFS[5]	L^2	K^2	K^2L^2	KL^3	K^2L^2	L	K^3	KL	L^2	KL	KL^2
FBS	K^2	L^2	K^3L	K^2	K^2L^2	L^3	K	KL	KL	K^2	K^2L
SBF	K^2	L^2	L^2	K^2	K^2L^2	L^2	K	KL^2	L^2	K^2	K^2L
SFF	K^2	L^2	L^2	K^2L^2	K^2L^2	L^2	K	KL^2	L^2	K^2	K^2L
FFS*	K^2	L^2	K^3L	K^2L^2	K^2L^2	L^3	K	KL	KL	K^2	K^2L

Minimum # of PE*: minimum number of PEs to maximize data reuse

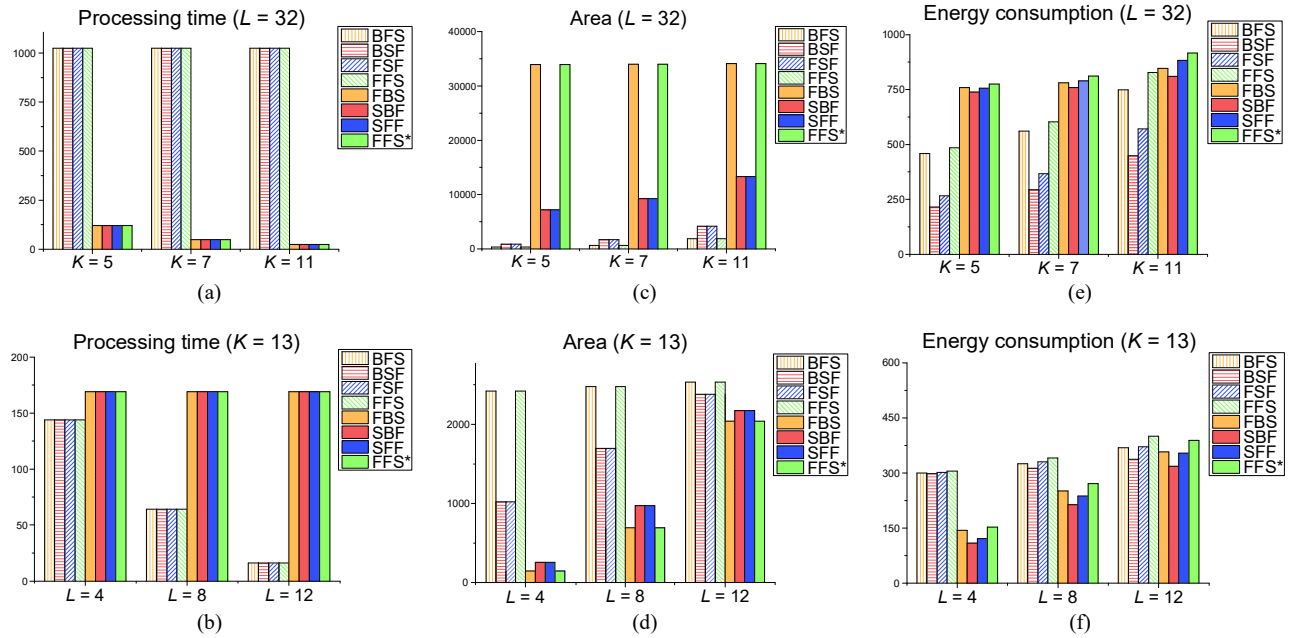


Figure 4. Comparison of hardware performance for eight structures.

circuit must be added to complete convolution. These include BFF, FBF, and FFF. As a result, 19 out of 27 configurations are ruled out because they require substantial hardware resources.

This paper describes and compares the hardware configurations of the BFS, BSF, FSF, and FFS structures presented in [5] and the FBS, SBF, SFF, and FFS* structures added in this paper. FFS* refers to a structure that differs fractionally from the FFS presented in [5]. Note that the four previously presented combinations in [5] and the four presented structures in this paper correspond in different forms.

This means that modifying the input and kernel data flow results in the corresponding configuration, while the overall structure is maintained. Based on Fig. 1 and 2, which depict the fundamental data flow, Fig. 3 illustrates the four structures presented in [5] and the four newly presented structures. Due to space limitations, the basic structure of [5] and the presented structure have been integrated and shown together, and it can be confirmed that only IM and KM memory modifications can be made to the pair's corresponding data flow. In Fig. 3(a), if assigned as IM, KM, and OM of MEM1, MEM2, and MEM3, it operates as BFS, whereas if assigned as KM, IM, and OM of MEM1, MEM2, and MEM3, it operates as FBS. (BSF, SBF), (FSF, SFF), (FFS, FFS*) pairs can all be described in the same way. Lastly, we summarize the hardware requirements for each structure's 2D convolutional architecture based on Fig. 3. Table 1 presents the processing time, which is the time required to perform the convolution of Equation (1), and the necessary hardware resources, PE, registers, and memories. Note that the memory access is included for energy consumption analysis.

IV. ANALYSIS RESULTS OF 2D CNN STRUCTURE

In order to determine the advantages and disadvantages of each candidate, Fig. 4 shows the performance of various L and K combinations. This is to consider the wide range of networks with various L and K combinations, including general convolutional layers and sophisticated convolutional layers. We denote that a classic convolutional layer is a case where $L > K$ and a deep convolutional layer is a case where $L \leq K$. In Fig. 4, K was changed from 5 to 11 with the input size L fixed to 32, and L was changed from 4 to 12 with the kernel size K fixed to 13. First, we examine the processing time, which is the amount of time required to perform the convolution of (1). According to Fig. 4(a) and (b), FBS, SBF, SFF, and FFS* are superior for the general convolutional layer,

and BFS, BSF, FSF, and FFS have a short delay time for the deep convolutional layer. Second, the memory size and number of PE were used to calculate the total hardware area necessary to perform the convolution of (1). Comparing Fig. 4(c) and (d), BFS and FFS use a smaller area for a general convolutional layer, whereas FBS and FFS* use a smaller area for a deep convolutional layer. Finally, the energy used to perform the convolution of (1) was compared using the activity of registers and memories. In Fig. 4 (e) and (f), it can be seen that BSF consumes the least energy in the case of a general convolutional layer, and SBF consumes the least amount of energy for a deep convolutional layer.

V. Experimental Results

This paper describes and analyzes eight structures by including four structures from [5] and the proposed four candidates. According to analysis, FBS, SBF, SFF, and FFS* are the quickest in processing time, BFS and FFS have the lowest size, and BSF spends the least amount of energy for a conventional convolution layer. Whereas BFS, BSF, FSF, and FFS have the shortest processing time, FBS and FFS* have the lowest area, and SBF spends the least energy for a deep convolutional layer. This demonstrates that the optimal structure varies based on the shape of the neural network and the user's desired performance orientation. Using the findings of this study, we can provide a solution for the optimal hardware structure aimed at the future development of various types of neural networks in addition to the classic ones.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2022R1A5A8026986), supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (2022-0-01170), and the EDA tool was supported by the IC Design Education Center (IDEC), Korea.

References

[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton, "Imagenet classification with deep convolutional neural networks," in Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.

[2] J. Cong and B. Xiao, “Minimizing computation in convolutional neural networks,” in Proc. Int. Conf. Artif. Neural Network, pp. 281-290, 2014.

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

[4] B. Moons, R. Uytterhdoeven, W. Dehaene, and M. Verhelst, “ENVISION: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage- accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI,” in IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers, pp. 246–248, February. 2017.

[5] Jo, Jihyuck, Suchang Kim, and In-Cheol Park. "Energy-efficient convolution architecture based on the rescheduled dataflow," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 65, no. 12, pp. 4196-4207, 2018