Dataflow에 따른 Systolic Array의 연산 성능 분석

(Analysis of Computing Performance of Systolic Arrays depending on Dataflows)

위 대 은, 박 상 수, 정 기 석*
한양대학교
(Dae-Eun Wi, Sang-Soo Park, Ki-Seok Chung)
(Hanyang Univ., Seoul)

Abstract : Today, Deep Neural Networks (DNNs) have been widely used for various applications. Because the DNNs require a large amount of computation, hardware accelerators are commonly used to speed up the inference processing. In the systolic array architecture, a common hardware structure for neural network accelerators, the type of dataflow defines how data is stored in a processing element (PE) and exchanged among adjacent PEs. The computing performance of the systolic array differs depending on the type of the dataflows. Therefore, data flow analysis is crucial to maximize the inference performance. In this work, the computing performance depending on the data flows is evaluated using an open-source systolic array simulator called SCALE-Sim, Experimental results show that the inference latency differs up to 3.2 times depending on the type of the dataflow.

Keywords : Convolution neural network, Systolic array, Dataflow, Weight stationary, Output stationary, Compute cycles, etc.

I. 서 론

컴퓨터 비전, 음성 인식, 자연어 처리 등 다양한 분야에 걸쳐 심층 신경망 (Deep Neural Network, DNN)에 대한 관심이 급증하고 있다 [1]. 생물학의 신경망에서 영감을 받은 심층신경망은 사용되는 어플리케이션에 따라 컨볼루션 신경망 (Convolutional Neural Network, CNN), 재귀 신경망 (Recurrent Neural Network, RNN) 등으로 분류된다 [2].

최근, Internet of Things (IoT) 디바이스에서 다양한 신경망을 사용한 어플리케이션이 적용되고 있다 [3]. 신경망의 추론에는 많은 연산이 수행되며 제한적인 연산 성능을 가진 IoT 디바이스에서 실시간으로 고속 연산하는 것은 어렵다 [4]. 이러한 문제를 해결하기 위해 IoT 디바이스 내부에 신경망 가속기 (Neural Processing Unit, NPU)를 포함하여 연산 성능을 가속하고 있는 추세이다 [5-6].

신경망 가속기에서 가장 널리 사용되고 있는 대표적인 하드웨어 구조 중 하나인 systolic array는 multiply-accumulate (MAC) 연산을 수행하는 processing element (PE)를 다수 포함하는 PE 배열로 구성되어있다. PE 내부에는 신경망의 weight 또는 activation을 저장하고 재사용하여, 메모리의 불필요한 접근 횟수를 줄이는 것이 가능하다 [7]. 이러한 방식을 dataflow라 하며, 저장되는 데이터의 종류에 따라 input/weight/output stationary (IS/WS/OS)로 분류된다. 이러한 systolic array는 신경망의 특성에 따라 데이터 흐름간의 큰 실행시간 차이를 보이는데, 이는 신경망의 레이어 종류에 따라 입력과 weight의 높이 (height)와 넓이 (width), 깊이 (depth), 채널 (channel) 등이 다양하기 때문이다 [5]. 본 연구에서는 systolic array의 성능을 분석하는 오픈 소스 NPU 시뮬레이터인 SCALE-Sim[8]을 사용하여 데이터 흐름에 따른 추론 과정에서의 소요 시간을 비교 및 분석하였으며,

*Corresponding Author (kchung@hanyang.ac.kr)
위대은, 박상수, 정기석: 한양대학교
※ 이 논문은 2020년도 정부 (과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2020-0-01304, 모바일 자가 학습 가능 재귀 뉴럴 네트워크 프로세서 기술 개발).

MobileNet을 사용하여 컨볼루션 신경망에서 어떤 데이터 흐름이 가장 높은 성능을 얻을 수 있는지 확인하였다.

Ⅱ. 본 론

1. 컨볼루션 신경망

컨볼루션 신경망은 심층 신경망의 대표적인 신경망으로 컴퓨터 비전, 자연어 처리 등 다양한 분야에서 사용되고 있다. 컨볼루션 신경망의 핵심적인 연산에 해당하는 컨볼루션 레이어는 아래의 그림 1과 같이 입력 (input feature map)과 컨볼루션 커널 (convolution kernel)의 성분 별 곱셈으로 이뤄진다. 입력과 컨볼루션 커널 간의 성분 별 곱셈을 계산하고, 바이어스 (bias)를 더하고, 활성화 함수 (activation function)을 적용하여 출력 (output feature map)을 계산한다.

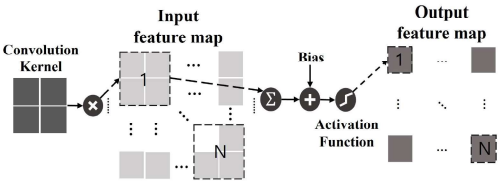

그림 1. 컨볼루션 레이어 연산 과정
Fig. 1. The process of convolution layer

이러한 컨볼루션 레이어는 신경망의 종류와 컨볼루션 레이어의 위치 등에 따라 다양한 크기 및 형태의 채널로 구성되어 있다.

2. Systolic array의 dataflow

신호처리, 영상처리 및 행렬 연산과 같이 연산량이 많은 응용의 고속 처리에 적합한 systolic array를 다양한 신경망에 적용하여, 추론의 효율을 개선하려는 연구가 활발하게 진행되고 있다 [6]. Systolic array의 PE 내부에는 입력, 컨볼루션 커널, 출력의 부분 합 (partial sum)이 저장되며, 컨볼루션 커널을 저장하는 weight stationary (WS), 또는 부분 합을 저장하는 output stationary (OS) 방식이 dataflow에서 가장 많이 사용되고 있다 [8].

그림 2는 데이터 흐름의 각 방식마다 PE의 맵핑과 연산 과정을 보여준다. (a)는 Global Buffer에 컨볼루션 커널 (W0~W3)을 각 PE에 미리 저장하고 입력 값과의 곱셈을 진행한다. 이때 발생된 부분 합은 아래 PE에 넘겨주면서 최종 부분 합을 Global Buffer에 저장한다. (b)는 입력값을 차례대로 아래 PE에 넘겨주면서 각 PE에 들어오는 컨볼루션 커널 값들과 곱셈을 진행한다. 곱셈 후 생기는 부분 합들은 각 PE에 저장하고 다음에 발생한 부분 합을 누적시킨다.

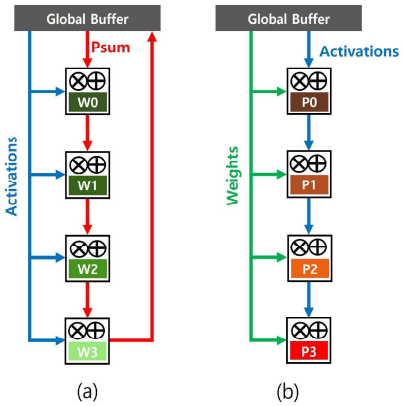

그림 2. dataflow의 맵핑 (a)WS (b)OS
Fig. 2. Mapping process of dataflow (a)WS (b)OS

Ⅲ. 시뮬레이션 결과 및 분석

1. 시뮬레이션 방법

본 연구에서는 다양한 신경망을 레이어 단위로 계산하는 systolic array 구조의 시뮬레이션이 가능한 오픈소스 NPU 시뮬레이터인 SCALE-Sim을 사용하여 OS와 WS로 동작하는 경우의 각 연산 시간을 총 소요 clock cycle 수로 분석하였다. 실험에서 systolic array는 32×32 크기의 PE로 구성하였으며, bandwidth는 10byte/cycle, SRAM 크기 (입력, 출력, filter)는 64KB로 구성하여 하여 실험을 진행하였다. Systolic array의 성능을 확인하기 위해, 컨볼루션 레이어의 종류에 따라 커널의 크기와 깊이가 상이한 MobileNet v1.0을 성능 평가의 벤치마크로 사용하였다 [8-9].

2. 실험 결과 및 분석

연산 시간은 NPU 내부의 Global Buffer에서 데이터를 로드, systolic array 내부에서 컨볼루션 레이어 연산, 연산의 결과가 Global Buffer에 저장되기까지의 총 cycle 수를 측정하였으며, 전체 실행시간은 그림 4와 같고, 각 레이어별로 측정한 결과는 그림 3과 같다. MobileNet의 컨볼루션 레이어

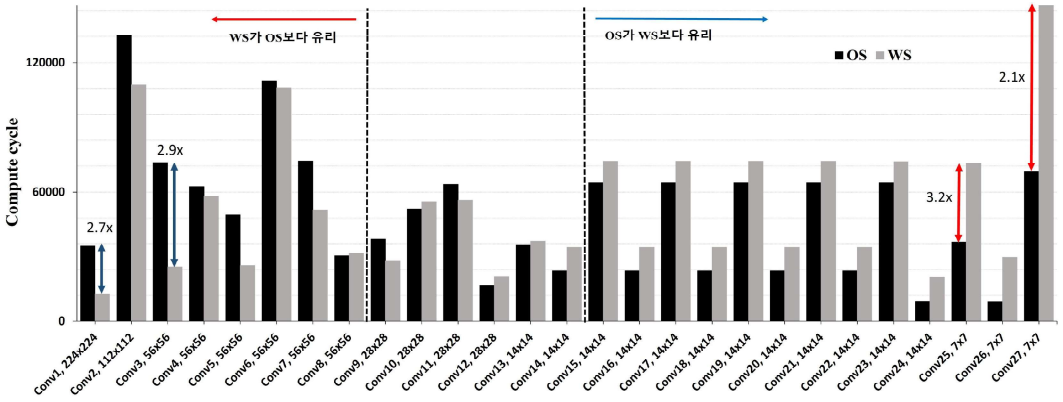

그림 3. 컨볼루션 레이어마다 OS, WS 연산 속도
Fig. 3. Compute cycle of WS, OS by convolution layer

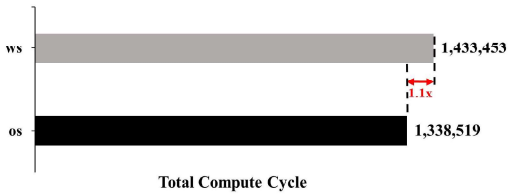

그림 4. OS와 WS의 총 계산 싸이클 수
Fig. 4. The total number of cycles of WS and OS

입력크기 및 출력 크기는 표 1과 같다 [9].

그림 4에서 전체 연산 시간은 WS가 OS에 비해 1.1배 많은 시간이 소요되었지만, 차이는 근소하다고 할 수 있다. 하지만, 그림 3에서 알 수 있듯이 신경망의 레이어 특성에 따라 각 dataflow 타입에 따른 성능은 상이하였다. 경향성을 살펴보면, Conv9 이전의 레이어에서는 WS가 더 좋은 성능을 보이고, Conv14 이후의 레이어에서는 OS가 더 높은 성능을 얻는 경향이 보였다. 그 중 가장 큰 연산 시간에 차이를 보인 레이어는 Conv3, Conv25 레이어이고 WS가 OS에 비해 각각 2.9배 빠르거나 3.2배 느린 것을 확인 하였다.

위와 같은 결과를 보이는 이유는 dataflow의 맵핑 방법에 따른 차이에서 발생한 것으로 크게 두 가지 관점으로 분석할 수 있다.

첫 번째로 입력의 크기가 클수록 WS가 유리하다. 그림 2에서 볼 수 있듯이 입력은 (a)에서 PE에 동시에 맵핑되어 빠르게 연산을 진행할 수 있지만 (b)에서는 PE를 통하여 아래 방향으로 차례대로 전달되어 비교적 오랜 시간이 소요된다.

표 1 Mobilenet 컨볼루션 레이어
Table 1. MobileNet convolution layers

Type	Filter Shape	Input Size
Conv1	3×3×3×32	224×224×3
Conv2 dw	3×3×32 dw	112×112×32
Conv3 pw	1×1×32×64	112×112×32
Conv4 dw	3×3×64 dw	112×112×64
Conv5 pw	1×1×64×128	56×56×64
Conv6 dw	3×3×128 dw	56×56×128
Conv7 pw	1×1×128×128	56×56×128
Conv8 dw	3×3×128 dw	56×56×128
Conv9 pw	1×1×128×256	28×28×128
Conv10 dw	3×3×256 dw	28×28×256
Conv11 pw	1×1×256×256	28×28×256
Conv12 dw	3×3×256 dw	28×28×256
Conv13 pw	1×1×256×512	14×14×256
Conv14 dw	3×3×512 dw	14×14×512
Conv15 pw	1×1×512×512	14×14×512
Conv16 dw	3×3×512 dw	14×14×512
Conv17 pw	1×1×512×512	14×14×512
Conv18 dw	3×3×512 dw	14×14×512
Conv19 pw	1×1×512×512	14×14×512
Conv20 dw	3×3×512 dw	14×14×512
Conv21 pw	1×1×512×512	14×14×512
Conv22 dw	3×3×512 dw	14×14×512
Conv23 pw	1×1×512×512	14×14×512
Conv24 dw	3×3×512 dw	14×14×512
Conv25 pw	1×1×512×1024	7×7×512
Conv26 dw	3×3×1024 dw	7×7×1024
Conv27 pw	1×1×1024×1024	7×7×1024

*Depth-wise/Point-wise convolution (pw/dw)

두 번째로 출력의 채널이 클수록 OS가 유리하다. 출력의 채널 값은 표 1의 Input Size의 마지막 숫자이다. (b)에서는 출력 채널의 크기와 상관없이 차례대로 PE에 맵핑되어 연산을 멈추지 않고 진행하는 반면, (a) WS 에서는 출력 채널이 PE의 개수를 넘을 경우 새로운 weight값을 저장해야하므로 추가 딜레이가 발생한다.

따라서 위 두 가지 관점으로 경향성을 분석하면 그림 3과 표1과 같이 입력 값이 크고 출력 채널이 작은 앞부분에선 WS가 유리한 것이고 반대 경향을 보이는 뒷부분에서는 OS가 유리하다.

Ⅳ. 결 론

심층 신경망이 발전함에 따라 가속기의 있어서 가속기의 성능 향상에 대한 연구는 앞으로도 큰 이슈일 것이다. 본 연구에서는 systolic array 상에서 동작하는 dataflow의 타입에 따른 연산 시간을 측정 및 분석하였다. 분석 결과 dataflow와 PE의 맵핑 방법에 따라 성능 차이를 보이는 것을 발견하였다. 본 연구의 실험 결과는 그 성능 차이가 최대 3.2배까지 될 수 있는 것을 보였다. 이러한 분석 결과를 토대로, systolic array가 가장 효과적인 성능을 얻을 수 있도록, 회로 내에서 dataflow를 선택할 수 있는 systolic array을 설계하는 것을 목표로 추후 연구가 진행될 예정이다.

References

[1] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, 2015, pp. 1-9.

[2] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick “Haffner. 1998. Gradient-based learning applied to document recognition.” Proceedings of the IEEE 86(11):2278 - 2324

[3] Anbarasan, M., et al. "Detection of flood disaster system based on IoT, big data and convolutional deep neural network." Computer Communications 150 (2020): 150-157.

[4] Chen, Xing, et al. "DNNOff: offloading DNN-based intelligent IoT applications in mobile edge computing." IEEE transactions on industrial informatics 18.4 (2021): 2820-2829.

[5] Kwon, Hyoukjun, Ananda Samajdar, and Tushar Krishna. "Maeri: Enabling flexible dataflow mapping over dnn accelerators via reconfigurable interconnects." ACM SIGPLAN Notices 53.2 (2018): 461-475.

[6] Chen, Yu-Hsin, et al. "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks." IEEE journal of solid-state circuits 52.1 (2016): 127-138.

[7] Kung, Hsiang-Tsung. "Why systolic architectures?." Computer 15.01 (1982): 37-46.

[8] Samajdar, A., Zhu, Y., Whatmough, P., Mattina, M., & Krishna, T.; "Scale-sim: Systolic cnn accelerator simulator." arXiv preprint arXiv:1811.02883 (2018).

[9] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).