디지털산업정보학회 논문지 제18권 제2호-2022년 6월
http://dx.doi.org/10.17662/ksdim.2022.18.2.017

# 1-D PE 어레이로 컨볼루션 연산을 수행하는 저전력 DCNN 가속기

이 정 혁*·한 상 욱**·최 승 원***

# *Power-Efficient DCNN Accelerator Mapping Convolutional Operation with 1-D PE Array*

Lee Jeonghyeok·Han Sangwook·Choi Seungwon

〈Abstract〉

In this paper, we propose a novel method of performing convolutional operations on a 2-D Processing Element(PE) array. The conventional method [1] of mapping the convolutional operation using the 2-D PE array lacks flexibility and provides low utilization of PEs. However, by mapping a convolutional operation from a 2-D PE array to a 1-D PE array, the proposed method can increase the number and utilization of active PEs. Consequently, the throughput of the proposed Deep Convolutional Neural Network(DCNN) accelerator can be increased significantly. Furthermore, the power consumption for the transmission of weights between PEs can be saved. Based on the simulation results, the performance of the proposed method provides approximately 4.55%, 13.7%, and 2.27% throughput gains for each of the convolutional layers of AlexNet, VGG16, and ResNet50 using the DCNN accelerator with a (weights size) x (output data size) 2-D PE array compared to the conventional method. Additionally the proposed method provides approximately 63.21%, 52.46%, and 39.23% power savings.

Key Words : FPGA, Deep Convolutional Neural Network, Accelerator, Processing Element, Data Reuse

## Ⅰ. 서론

Convolutional Neural Networks (CNNs)은 이미지 분류[1], 문자 인식[2] 그리고 통신[3] 등 많이 분야에서 사용되고 있다. CNNs에 대한 연구가 진행됨에 따라 더 나은 성능을 얻기 위해 Neural Network (NN) Layer 수가 증가해야 했고, 이러한 이유로 AlexNet[4], VGGNet[5], 그리고 ResNet[6]와 같은 Deep Convolutional Neural Networks (DCNNs)가 제안되었다. AlexNet와 ResNet의 NN Layer의 수는 각각 3개와 53개이고 Top-5 error는 각각 15.3%와 5.3%로 Layer 수에 비례하여 성능이 향상된 것을 알 수 있다.

* 한양대학교 융합전자공학과 석사과정(제1저자)
** 한양대학교 전자컴퓨터통신공학과 박사과정
*** 한양대학교 전자컴퓨터통신공학과 교수(교신저자)

이처럼 성능을 높이기 위해 DCNNs의 NN Layer 수는 점점 더 커졌고, 이로인해 DCNNs은 높은 연산 복잡성, 많은 양의 연산 데이터, 그리고 많은 메모리 액세스와 같은 문제들을 직면하게 되었다. 이 문제들 중 많은 양의 연산 데이터와 많은 메모리 액세스는 많은 양의 전력 소모를 발생시키기 때문에 이를 해결하기 위한 많은 DCNN 가속기가 제안되었다[7-12]. 제안된 DCNN 가속기들은 칩 내부에 메모리와 Processing Elements (PEs)를 함께 구현하는 방식을 사용하여 DRAM에서 Global Buffer (GLB)로 전달된 weights와 input data 같은 연산 데이터가 PEs로 전송되면 동일한 데이터를 전송받기 위해서 DRAM와 GLB 모두에 다시 액세스하지 않도록 데이터 재사용 방법을 사용한다.

데이터 재사용 방법으로 weights에 대한 전송에 소모되는 전력을 최소화하는 Weight Stationary (WS) dataflow[8, 9]와 input data에 대한 전송에 소모되는 전력을 최소화하는 Input Stationary (IS) dataflow[10] 그리고 partial sums에 대한 전송에 소모되는 전력을 최소화하는 Output Stationary (OS) dataflow[11, 12]가 제시되었다. 그리고 최근 weights, input data, 그리고 partial sums 모두를 재사용하여 소모되는 전력을 최소화하는 Row Stationary (RS) dataflow가 제시되었다[7]. 이 연구에서는 RS dataflow를 사용하여 기존 WS, IS, 그리고 OS dataflow 대비 메모리 액세스와 전력 소모를 획기적으로 낮출 수 있다는 것을 증명하였다. RS dataflow를 사용하여 고정된 PE 어레이에 컨볼루션 연산을 더 효율적으로 매핑하기 위하여 두 가지 매핑전략(Replication와 Folding)이 제안되었지만, 여전히 낮은 utilization of PEs가 문제점으로 남아있다.

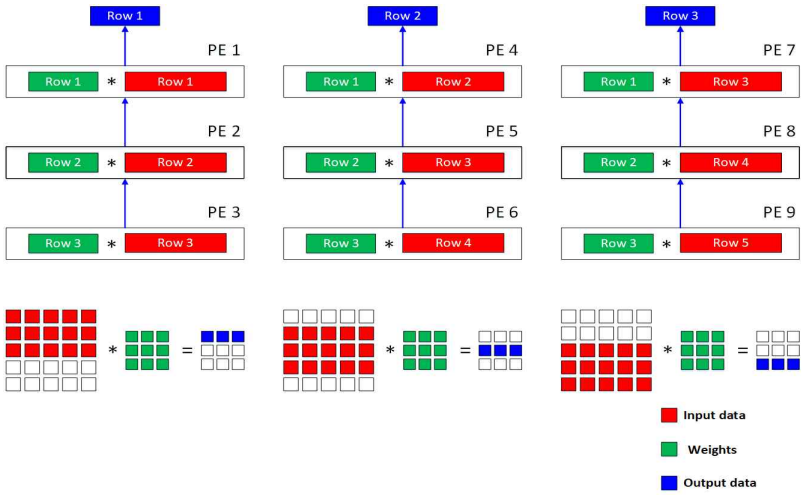
해당 논문에서는 새로운 매핑 방법을 제안함으로써 낮은 utilization of PEs 문제를 해결하여 throughput을 높인다. 또한, weights에 대한 PE들 간의 전송을 하지 않도록 하여 전력소모 또한 감소시킨다.

논문의 구성은 2장에서 2-D PE 어레이에서 컨볼루션 연산에 사용할 PE를 정하는 새로운 매핑 방법을 소개하고 기존의 매핑 방법과 비교한다. 3장에서는 기존 매핑 방법대비 새롭게 제안된 매핑 방법의 우월성을 증명하기 위하여 수식적으로 분석한다. 마지막으로, 4장에서 결론을 맺는다.

## Ⅱ. 기존 매핑 방법과 제안하는 매핑 방법

이 장에서는 컨볼루션 연산을 2-D PE 어레이에 매핑하던 기존 매핑 방법대신 1-D PE 어레이로 매핑하여 throughput을 향상시키고 전력 소모를 감소시킬 수 있는 새로운 매핑 방법을 제안한다. 즉, [7]에서 제시한 $W_s \times O_s$의 2-D PE 어레이 매핑 방법대신 $W_s \times 1$의 1-D PE 어레이를 이용하는 1-D 매핑 방법을 제안한다. ($W_s$= weights size, $O_s$=output data size)

<그림 1>은 5 x 5 input data(input data size, $I_s = 5$)와 3 x 3 weights(weights size, $W_s = 3$)에 대한 컨볼루션 연산을 2-D PE 어레이에 매핑하는 기존 방법을 보여준다. 해당 연산의 output data는 3 x 3의 크기(output data size, $O_s = 3$)를 가지기 때문에 기존 방법에서는 3 x 3의 2-D PE 어레이가 필요하다. 따라서 연산에 사용되는 PE의 수는 9이다. 또한, 기존 방법은 2-D PE 어레이에서 데이터를 재사용하기 위해 weights의 각 행은 수평으로 재사용되고, input data의 각 행은 대각선으로 재사용되며, partial sums의 각 행은 수직으로 누적되고 output data가 된다. 이 때문에 2-D PE 어레이의 첫 번째 열을 제외한 나머지 열의 PE들은 컨볼루션 연산에 필요한 데이터를 이전 열의 PE들로부터 전달받아야 한다. input data와 weights의 각 행끼리의 컨볼루션 연산을 수행하는데 1 cycle이 소모된다고 가정하면, <그림 1>의 컨볼

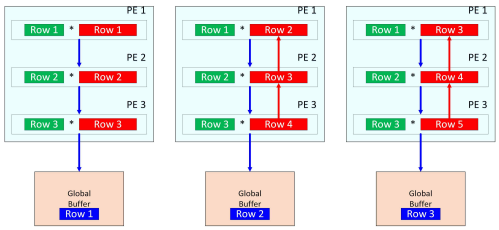



<그림 1> 2-D PE 어레이에 컨볼루션 연산을 매핑하는 기존 방법[7]

루션 연산을 3 cycles동안 수행하고, 각 cycles에서 연산을 할당 받아 수행하는 PE의 수는 3이다. 따라서 평균 utilization of active PEs는 $\frac{1}{3}$이다.

<그림 2>는 5 x 5 input data와 3 x 3 weights에 대한 컨볼루션 연산을 1-D PE 어레이에 매핑하는 새로운 방법을 보여준다. 제안하는 방법은 output data의 크기와는 상관없이 3 x 1의 1-D PE 어레이가 필요하다. 따라서, 같은 연산을 기존 방법보다 적은 수의 PE로 수행할 수 있고 그 수는 3이다. 또한, 1-D PE 어레이에서 데이터를 재사용하기 위해 weights의 각 행은 PE안에 저장되어 재사용되고, input data의 각 행은 수직으로 재사용되며, partial sums의 각 행은 수직으로 누적되고 output data가 된다. 컨볼루션 연산을 수행하는데 소모되는 cycles은 3으로 동일하고 각 cycles에서 연산을 할당 받아 수행하는 PE의 수는 3이므로 평균 utilization of active PEs는 1이다.

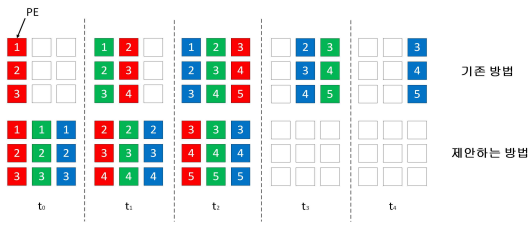
<그림 3>은 고정된 3 x 3 2-D PE 어레이에서 3-channel 5 x 5 input data(the channel of input data, $I_c = 3$)와 3-channel 3 x 3 weights에 대한 컨볼




<그림 2> 1-D PE 어레이에 컨볼루션 연산을 매핑하는 제안하는 방법

루션 연산을 두 가지 매핑 방법에서 수행하는 방법을 보여준다. Input data와 weights의 각 channel을 서로 다른 색으로 나타내었고, PE안의 숫자는 input data의 행을 의미하며 weights에 대한 정보는 생략하였다. 앞서 말한 것처럼, 기존 방법은 첫 번째 열을 제외한 나머지 열의 PE들은 컨볼루션 연산에 필요한 데이터를 이전 열의 PE들로부터 전달받아야하기 때문에 utilization of active PEs가 각 cycle에서 $\frac{1}{3}$, $\frac{2}{3}$, 1, $\frac{2}{3}$, 그리고 $\frac{1}{3}$ 이므로 평균 utilization of active PEs는 $\frac{3}{5}$이다. 그리고 컨볼루션 연산을 5 cycles동안

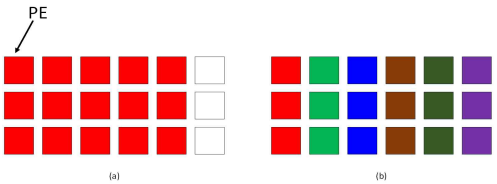
수행하게 된다. 반면에 제안하는 방법에서는 평균 utilization of active PEs는 1이며 컨볼루션 연산을 3 cycles동안 수행해 기존 방법대비 2 cycles 먼저 끝나 throughput이 증가하게 된다. 이것을 $O_s$의 관점에서 살펴보면, 기존 방법은 컨볼루션 연산을 $W_s \times O_s$의 2-D PE 어레이에 매핑하기 때문에 모든 PE가 연산을 할당 받아 수행하는 상태인 완전 활성화 상태(utilization of active PEs가 1인 경우)가 되기 위해 모든 PE가 연산을 할당 받아 수행하지 않는 상태인 부분 활성화 상태(utilization of active PEs가 1이 아닌 경우)가 $O_s - 1$ cycles동안 필요하고 완전 활성화 상태가 끝난 이후에도 동일한 cycles동안의 부분 활성화 상태가 필요하다. 따라서, $I_s$가 커지거나 $W_s$가 작아져 $O_s$가 커지면 부분 활성화 상태가 전체 cycles에서 차지하는 비중이 커지므로 제안하는 방법에서 얻을 수 있는 throughput gain은 커지게 된다. 반면에, $I_c$가 커지게 되면 완전 활성화 상태가 전체 cycles에서 차지하는 비중이 커져 제안하는 방법에서 얻을 수 있는 throughput gain은 작아지게 된다. 또한, throughput gain은 $I_c$의 개수가 2-D PE 어레이의 열의 개수, 즉 $O_s$의 배수인 경우 가장 높은 값인 max throughput gain을 가지고 그 값은 배수가 커질수록 작아진다. 그리고 $I_c$의 개수가 $O_s$의 배수가 아닌 경우에 throughput gain은 감소하는데 그 이유는 제안하는 방법에도 부분 활성화 상태가 생기기 때문이다. <그림 3>의 경우를 예로 들면 $I_c$의 개수가 $O_s$의 배수인 3인 경우에 제안하는 방법에는 부분 활성화 상태가 발생하지 않아 컨볼루션 연산이 2 cycles 먼저 끝나 max throughput gain을 가지지만 $I_c$의 개수가 $O_s$의 배수가 아닌 4와 5인 경우에는 제안하는 방법에서도 부분 활성화 상태가 생겨서 기존 방법대비 각각 0 그리고 1 cycles의 차이를 가지게 되며 throughput gain이 아예 없거나 3일 때 보다 작아진다. 그리고 다시 $I_c$의 개수가 6이 되어 $O_s$의 배수가 되면 다시 max throughput gain을 가지지만 그 값은 $I_c$의 개수가 3일 때 가지는 값보다는 작아진다. 제안하는 방법이 이러한 throughput gain을 가지기 위해서는 2-D PE 어레이의 각 열에 있는 PE들이 GLB에 동시 액세스가 가능하여야 하는데 그러기 위해서는 특별한 메모리 구조가 필요하다. 그리고 <그림 5>에서 특별한 메모리 구조에 대하여 설명한다.




<그림 3> 고정된 3 x 3 PE 어레이에서 컨볼루션 연산 수행 방법

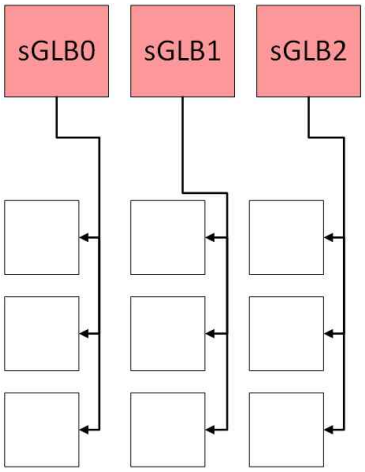
<그림 4>는 고정된 3 x 6 2-D PE 어레이에서 7 x 7 input data와 3 x 3 weights에 대한 컨볼루션 연산을 두 가지 매핑 방법으로 수행하는 방법을 보여준다. 기존 방법에서는 컨볼루션 연산을 위해서 $W_s \times O_s$의 2-D PE 어레이가 필요하므로 3 x 5의 2-D PE 어레이가 필요하다. 따라서 주어진 3 x 6 2-D PE 어레이에서 마지막 열의 PE들은 컨볼루션 연산에 사용되지 않게 된다. 그리고 이것을 주어진 PE개수 중 연산에 사용되는 PE(active PE)개수의 비로 나타내면 $\frac{15}{18}$이다. 반면에, 제안하는 방법은 $W_s \times 1$의 1-D PE 어레이가 필요하므로 3 x 1의 1-D PE 어레이 6개가 컨볼루션 연산에 사용될 수 있다. 6개의 1-D PE 어레이는 특별한 메모리 구조로 인해 각각의 GLB로 동시 액세스가 가능하고, 이를 통해 서로 다른 channel에 대한 연산을 동시에 시작할 수 있다. 따라서 제안하는 방법에서 주어진 PE개수 중 연산에 사

용되는 PE개수의 비는 1이다. 앞서 말한 것처럼, $I_c$의 개수가 $O_s$의 배수인 값(예를 들어, 6, 12, 18)에서 max throughput gain을 가지고 $O_s$의 배수가 아닌 값(예를 들어, 1, 2, 3)에서는 제안하는 방법에서도 연산에 사용되는 PE개수가 작아지기 때문에 throughput gain이 아예 없거나 매우 작다. 그리고 utilization of active PEs와 active PE개수의 의미가 다른 것을 명심하자. Active PE개수는 주어진 2-D PE 어레이에서 컨볼루션 연산을 수행하는 PE들의 개수이고 utilization of active PEs는 컨볼루션 연산이 수행되는 각 cycle에서 실제 연산을 수행하고 있는 active PE개수이다. 예를 들어, <그림 3>와 <그림 4(a)>에서 active PE개수는 각각 9와 15이고, $t_1$에서의 utilization of active PEs는 각각 6와 8이다.




<그림 4> 고정된 3 x 6 PE 어레이에서 컨볼루션 연산 수행 방법 (a) 기존 방법의 매핑 (b) 제안하는 방법의 매핑

<그림 5>는 제안하는 방법이 사용가능하도록 하는 메모리 구조를 보여준다. 앞서 말한 것처럼, 제안하는 방법은 2-D PE 어레이의 각 열의 PE들이 동시에 연산을 시작하여야 하므로 각 열의 PE들은 동시에 GLB 액세스를 할 수 있어야한다. 예를 들어, 고정된 3 x 3의 2-D PE 어레이가 주어진 경우에, 3개의 열에 있는 PE들이 동시에 GLB 액세스할 수 있도록 하나의 large GLB(lGLB)를 3개의 small GLB (sGLB)로 나누어 구현하여야한다. 제안하는 방법은 각 열마다 서로 다른 channel에 대한 연산을 하므로, 각각의 sGLB에는 서로 다른 channel에 해당하는 데이터가 저장되어 있어야한다.




<그림 5> 제안하는 방법을 사용하기 위해 필요한 메모리 구조

## Ⅲ. 시뮬레이션

이 장에서는 제안하는 방법으로 얻을 수 있는 throughput gain와 power saving을 관련 수식 설명 후 컴퓨터 시뮬레이션을 통해 나타낸다.

첫 번째로 제안하는 방법에서 얻을 수 있는 throughput gain을 수식적으로 나타낸다. 이 논문에서는 throughput을 구하기 위해 아래의 식(1,2)을 사용한다. 올바른 비교를 위하여 두 방법 모두에서 식(1)의 첫 번째 항과 두 번째 항은 같다고 가정한다. 따라서 utilization of PEs가 throughput의 차이를 만들어내고, 식(2)로 나타낼 수 있다.

$$throughput = \left(\frac{operation}{cycles} \times \frac{cycles}{\text{second}}\right) \times number\ of\ PEs \times utiliztaion\ of\ PEs \quad (1)\ [13]$$

$$utilization\ of\ PEs = \frac{number\ of\ active\ PEs}{number\ of\ PEs} \times utilization\ of\ active\ PEs \quad (2)\ [13]$$

식(2)를 분석하면, 주어진 PE개수 중 실제 연산에

사용되는 active PE개수의 비와 연산에 사용되는 active PE개수 중 각 cycle에서 실제 연산을 수행하고 있는 active PE개수의 곱이다. 따라서 이 두식을 이용하여 throughput gain에 관한 식(3)을 아래와 같이 나타낼 수 있다.

$$throughput\,gain = 100\times(\frac{throughput_{proposed}}{throughput_{conventional}}-1)\% \quad (3)$$

두 번째로 제안하는 방법에서 얻을 수 있는 power saving을 수식적으로 나타낸다. 데이터 전송으로 발생하는 전력은 sGLB와 PE사이에 데이터 전송으로 발생하는 전력과 PE들 간의 데이터 전송으로 발생하는 전력이 있다. 데이터 전송에는 input data, weights, 그리고 partial sums 세 가지가 있으므로 총 6가지의 데이터 전송으로 전력이 소모되게 된다. 앞 장에서 설명한 것처럼, 제안하는 방법에서는 weights의 각 행은 각 PE안에 저장되고 해당 PE에서만 재사용되기 때문에 PE들 간의 전송이 발생하지 않아 그만큼의 전력 소모량을 감소시킬 수 있다. 따라서, 제안하는 방법에서 얻을 수 있는 power saving은 아래의 식(4)와 같이 나타낼 수 있다. ($p_{gi}, p_{gw}, p_{gp}$은 각각 sGLB와 PE사이에서 input data, weights, partial sums를 전송할 때 발생하는 전력을 나타내고 $p_{pi}, p_{pw}, p_{pp}$은 각각 PE들 간의 input data, weights, partial sums를 전송할 때 발생하는 전력을 나타낸다.)

$$power\,saving = 100\times(1-\frac{p_{gi}+p_{gw}+p_{gp}+p_{pi}+p_{pp}}{p_{gi}+p_{gw}+p_{gp}+p_{pi}+p_{pw}+p_{pp}})\% \quad (4)$$

<그림 6>은 식(3)를 토대로 $W_s \times O_s$의 2-D PE 어레이에서 10 x 10, 20 x 20, 그리고 30 x 30 input data와 3 x 3 weights를 컨볼루션 할 때, 제안하는 방법에서 얻을 수 있는 throughput gain을 변하는 $I_c$에 따라 나타낸다. 10 x 10 input data는 빨간색, 20 x 20 input data는 파란색, 그리고 30 x 30 input data는 초록색으로 나타내었다. 앞서 말한 것처럼, 기존 방법은 부분 활성화 상태가 연산을 시작하는 초반과 마지막 부분에 각각 $O_s-1$ cycles 씩 총 $2\times(O_s-1)$ cycles 동안 발생하게 된다. 그리고 이 부분 활성화 상태로 발생하는 cycles이 전체 연산에 사용되는 cycles 중 차지하는 비중은 $I_c$에 반비례한다. 제안하는 방법에서는 $W_s \times O_s$의 2-D PE 어레이의 환경에서 $I_c$가 $O_s$의 정수배가 아닌 경우에 부분 활성화 상태가 $O_s$ cycles 동안 발생하게 된다. <그림 6>의 파란색의 20 x 20 input data의 경우를 보면 $O_s$가 18이기 때문에 $I_c$가 18의 정수배인 경우에 제안하는 방법에서는 부분 활성화 상태가 존재하지 않아 max throughput gain을 가지는 것을 볼 수 있다. 그리고 $I_c$가 점차 증가함에 따라 기존 방법에서 부분 활성화 상태가 차지하는 비중이 작아져 max throughput gain이 작아지는 것을 볼 수 있다. 또한 $I_c$가 $O_s$의 정수배가 아닌 경우 제안하는 방법에서도 부분 활성화 상태가 발생하기 때문에 throughput gain이 감소하는 것을 알 수 있다. $I_c$가 36인 경우 기존 방법의 경우 연산에 사용되는 총 cycles이 53이고 그 중 부분 활성화 상태가 34 cycles이다. 그리고 연산에 사용되는 PE의 개수는 초반 부분 활성화 상태에서 3개부터 51개 까지 3개씩 증가하다가 완전 활성화 상태에서 54개를 사용하고 다시 마지막 부분 활성화 상태에서 51개부터 3개까지 감소한다. 따라서 부분 활성화 상태에서의 utilization of active PEs는 $\frac{2\times(3+6+\cdots+48+51)}{54}=17$와 같이 나타낼 수 있고 완전 활성화 상태에서의 utilization of active PEs

는 $\frac{19\times54}{54}=19$로 나타낼 수 있으므로 전체 utilization of active PEs는 36이다. 반면에 제안하는 방법에서는 부분 활성화 상태가 존재하지 않고 연산에 사용되는 총 cycles은 36이다. 따라서 utilization of active PEs는 $\frac{36\times54}{54}=36$으로 기존 방법과 동일하지만, 연산에 걸린 cycles이 53과 36으로 다르다. 따라서 throughput gain은 $100\times(\frac{36/36}{36/53}-1)=47.22\%$이다. 이와 같은 방식으로 $I_c$가 54, 72, 그리고 90인 경우 각각의 throughput gain은 31.48%, 23.61%, 그리고 18.89%이다. 하지만, $I_c$가 $O_s$의 정수배보다 1이 큰 값인 경우에는 제안하는 방법에서도 부분 활성화 상태가 기존 방법과 동등하게 나타나므로 throughput gain은 0이 된다.

<그림 6> $I_c$에 따른 제안하는 방법에서 얻을 수 있는 throughput gain

<표 1>은 위의 내용을 토대로 $W_s\times O_s$의 2-D PE 어레이에서 AlexNet, VGG16, ResNet50의 컨볼루션 레이어에서 기존 방법대비 제안하는 방법으로 얻을 수 있는 throughput gain과 식(4)를 토대로 power saving 까지 나타낸다. AlexNet의 컨볼루션 레이어2에서 $I_s$, $O_s$, 그리고 $I_c$가 각각 27, 27, 그리고 96이므로 기존 방법에서 부분 활성화 상태가 52 cycles 그리고 완전 활성화 상태가 70 cycles로 연산에 총 122 cycles 소요된다. 그리고 제안하는 방법은 $I_c$가 $O_s$의 정수배가 아니므로 완전 활성화 상태 81 cycles, 부분 활성화 상태가 27 cycles로 연산에 총 108 cycles 소요된다. 그리고 utilization of active PEs는 기존 방법이 $\frac{2\times(3+\cdots+78)}{81}+70=96$이고 제안하는 방법이 $\frac{15\times3}{81}\times27+81=96$이다. 연산에 소요되는 총 cycles을 반영하면 $100\times(\frac{96/108}{96/122}-1)=12.96\%$의 throughput gain을 얻을 수 있다. AlexNet의 컨볼루션 레이어 1-5에서 각각 3.64%, 12.96%, 3.07%, 1.54%. 그리고 1.54%의 throughput gain을 가지기 때문에 AlexNet의 컨볼루션 레이어에서 평균 4.55%의 throughput gain을 가지고 power saving은 63.21%이다. 마찬가지로, VGG16의 컨볼루션 레이어4에서 $I_s$, $O_s$, 그리고 $I_c$가 각각 28, 28, 그리고 256이므로 기존 방법에서 부분 활성화 상태가 54 cycles 그리고 완전 활성화 상태가 229 cycles로 연산에 총 283 cycles 소요된다. 그리고 제안하는 방법은 $I_c$가 $O_s$의 정수배가 아니므로 완전 활성화 상태 252 cycles, 부분 활성화 상태가 28 cycles로 연산에 총 280 cycles 소요된다. 그리고 utilization of active PEs는 기존 방법이 $\frac{2\times(3+\cdots+81)}{84}+229=256$이고 제안하는 방법이 $\frac{4\times3}{84}\times28+252=256$이다. 여기에 연산에 소요되는 cycles을 반영하면 1.07%의 throughput gain을 얻을 수 있다. VGG16의 컨볼루션 레이어에서 각각 0.89%, 56.25%, 8.93%, 1.07%. 그리고 1.35%의 throughput gain을 가지기 때문에 VGG16의 컨볼루션 레이어에서 평균 13.7%의 throughput gain을 가지고 power saving은 52.46%이다. 마지막으로 ResNet50의 컨볼루

션 레이어에서 평균 2.27%의 throughput gain을 가지고 power saving은 39.23%이다.

<표 1> 제안하는 방법으로 얻을 수 있는 throughput gain과 power saving

| 모델 | throughput gain (%) | power saving (%) |
|---|---|---|
| AlexNet | 4.55 | 63.21 |
| VGG16 | 13.7 | 52.46 |
| ResNet50 | 2.27 | 39.23 |

## Ⅳ. 결론

이 논문에서는 [7]에서 제시한 두 가지 매핑전략(Replication and Folding)에도 불구하고 발생하는 낮은 utilization of PEs 문제를 해결하고 weights에 대한 PE들 간의 전송으로 발생하는 전력 소모를 줄이기 위하여 새로운 매핑 방법을 제시한다. 컨볼루션 연산을 $W_s \times O_s$의 2-D PE 어레이에 매핑하는 기존 방법대신 제안하는 방법은 $W_s \times 1$의 1-D PE 어레이에 매핑하여 동일한 연산을하기위해서 필요한 PE의 개수가 감소하였고, 2-D PE 어레이에서는 더 효율적인 매핑으로 utilization of active PEs를 높여 throughput을 높일 수 있었다. 그리고 $I_c$의 개수가 $O_s$의 배수인 경우, max throughput gain을 얻을 수 있었다. 하지만 1-D PE 어레이 매핑을 이용하여 throughput을 높이기 위해서는 하나의 lGLB를 2-D PE 어레이의 column 수만큼의 sGLB로 나누어 구현하여야 했다. 이로 인해 총 용량은 동일하지만 컨트롤러를 sGLB 수만큼 설계해야하므로 H/W 리소스 측면에서 오버헤드가 발생하게 된다. 하지만, lGLB에 필요한 컨트롤러보다 sGLB에 필요한 컨트롤러가 단순하므로 더 작게 설계가 가능하고 따라서 오버헤드 또한 column 수만큼 배로 증가하지는 않는다. 또한, 제안하는 방법에서는 weights가 각 PE안에 저장된 후 재사용되기 때문에 PE들 간의 전송이 발생하지 않아 기존 방법대비 전력소모량 또한 감소시킬 수 있었다. $W_s \times O_s$의 2-D PE 어레이에서 제안하는 방법은 기존 방법대비 AlexNet, VGG16, 그리고 ResNet50의 컨볼루션 레이어에서 각각 약 4.55%, 13.7%, 그리고 2.27%의 throughput gain과 약 63.21%, 52.46%, 그리고 39.23%의 power saving을 이룰 수 있었다.

## 참고문헌


[1] Tripathi, Milan. "Analysis of convolutional neural network based image classification techniques," Journal of Innovative Image Processing (JIIP), Vol.3, No.2, 2021, pp.100-117.

[2] 김진호, “딥러닝 신경망을 이용한 문자 및 단어 단위의 영문 차량 번호판 인식,” 디지털산업정보학회 논문지, 제16권, 제4호, 2020, pp.19-28.

[3] 김진호·안흥섭·최승원, “CNN 기반의 IEEE 802.11 WLAN 프레임 포맷 검출,” 디지털산업정보학회 논문지, 제16권, 제 2호, 2020, pp.27-33.

[4] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton, “Imagenet classification with deep convolutional neural networks,” In Advances in neural information processing systems, 2012, pp.1097-1105.

[5] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[6] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition,” In Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp.770-778.

[7] Chen, Y.-H., Emer, J., Sze, V. Eyeriss, A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks. ACM SIGARCH computer architecture news 2016, 44.3: pp.367-379.

[8] Nvidia, NVDLA Open Source Project, 2017. http://nvdla.org/ 69, 76, 92, 94, 96, 97, 113, 114.

[9] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers et al., In-datacenter performance analysis of a tensor processing unit, in International Symposium on Computer Architecture (ISCA), 2017.

[10] 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 International Symposium on Computer Architecture (ISCA), 2017.

[11] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, DaDianNao: A machine-learning supercomputer, in International Symposium on Microarchitecture (MICRO), 2014.

[12] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning, in Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2014.

[13] V. Sze, Y.-H.Chen, T.-J. Yang, J. S. Emer, "Efficient Processing of Deep Neural Networks," Synthesis Lectures on Computer Architecture, Morgan & Claypool Publishers, 2020, pp.44-46.

## ■ 저자소개 ■

이 정 혁
Lee, JeongHyeok

2020년 9월~현재 한양대학교 융합전자공학과 석사과정
2020년 8월 동국대학교 전자전기공학부 (공학학사)

관심분야 : FPGA, Deep Learning, NPU, etc
E-mail : jeonghyeok.lee@dsplab.hanyang.ac.kr

한 상 욱
Han, SangWook

2013년 9월~현재 한양대학교 전자통신컴퓨터공학과 박사과정
2013년 8월 한양대학교 전자통신컴퓨터공학과 석사과정
2011년 8월 한양대학교 전자통신컴퓨터공학부 (공학학사)

관심분야 : wireless communication, 5G, LTE-A, MU-MIMO, Deep Learning
E-mail : ssssang3234@dsplab.hanyang.ac.kr

최 승 원
Choi Seungwon

2012년 3월~현재 HY-MC 연구센터 센터장
2002년~2011년 HY-SDR 연구센터 센터장
1992년~현재 한양대학교 융합전자공학부 교수
1990년~1992년 일본 우정성 통신연구소 선임 연구원
1989년~1990년 ETRI 선임 연구원
1988년~1989년 미국 Syracuse대학 전지 및 전산과 교수
1988년 12월 미국 Syracuse대학 전기공학 (공학박사)
1985년 12월 미국 Syracuse대학 전기공학 (공학석사)
1982년 2월 서울대학교 전자공학 (공학석사)
1982년 2월 한양대학교 전자공학 (공학학사)

관심분야 : SDR, 이동통신, 신호처리
E-mail : choi@dsplab.hanyang.ac.kr

논문접수일 : 2022년 4월 24일
수 정 일 : 2022년 5월 14일
게재확정일 : 2022년 5월 25일