JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지 Vol. 23, No. 1: 55~62, Jan. 2019

# 가속 회로에 적합한 CNN의 Conv-XP 가지치기

우용근[1] · 강형주[2*]

# Conv-XP Pruning of CNN Suitable for Accelerator

**Yonggeun Woo[1] · Hyeong-Ju Kang[2*]**

[1]Graduate Student, School of Computer Science and Engineering, Korea University of Technology and Education, Cheonan, 31253 Korea
[2*]Associate Professor, School of Computer Science and Engineering, Korea University of Technology and Education, Cheonan, 31253 Korea

## 요 약

CNN은 컴퓨터 영상 인식 부분에서 높은 성능을 보여주고 있으나 많은 연산양을 요구하는 단점으로 인해 전력이나 연산 능력에 제한이 있는 임베디드 환경에서는 사용하기 어렵다. 이러한 단점을 극복하기 위해 CNN을 위한 가속 회로나 가지치기 기법에 대한 연구가 많이 이루어지고 있다. 기존의 가지치기 기법은 가속 회로의 구조를 고려하지 않아서, 가지치기된 CNN을 위한 가속 회로는 비효율적인 구조를 가지게 된다. 이 논문에서는 가속 회로의 구조를 고려한 새로운 가지치기 기법인 Conv-XP 가지치기를 제안한다. Conv-XP 가지치기에서는 'X'와 '+' 모양의 두 가지 패턴으로만 가지치기함으로써, 이 기법으로 가지치기된 CNN을 위한 가속 회로의 구조를 단순하게 설계할 수 있도록 하였다. 실험 결과에 따르면, Conv-XP와 같이 가지치기 패턴을 제한하여도 CNN의 성능이 악화되지 않으며, 가속 회로의 면적은 12.8%을 감소시킬 수 있다.

## ABSTRACT

Convolutional neural networks (CNNs) show high performance in the computer vision, but they require an enormous amount of operations, making them unsuitable for some resource- or energy-starving environments like the embedded environments. To overcome this problem, there have been much research on accelerators or pruning of CNNs. The previous pruning schemes have not considered the architecture of CNN accelerators, so the accelerators for the pruned CNNs have some inefficiency. This paper proposes a new pruning scheme, Conv-XP, which considers the architecture of CNN accelerators. In Conv-XP, the pruning is performed following the 'X' or '+' shape. The Conv-XP scheme induces a simple architecture of the CNN accelerators. The experimental results show that the Conv-XP scheme does not degrade the accuracy of CNNs, and that the accelerator area can be reduced by 12.8%.

**키워드** : 신경망, CNN, 가속 회로, 가지치기

Keywords : Neural networks, Convolutional neural networks, Accelerator, Pruning

**Received** 6 November 2018, **Revised** 16 November 2018, **Accepted** 24 November 2018

*** Corresponding Author** Hyeong-Ju Kang (E-mail: hjkang@koreatech.ac.kr, Tel:+82-41-560-1420)
Associate Professor, School of Computer Science and Engineering, Korea University of Technology and Education, Cheonan, 31253 Korea

Open Access http://doi.org/10.6109/jkiice.2019.23.1.55 print ISSN: 2234-4772 online ISSN: 2288-4165

## Ⅰ. 서 론

지난 몇 년간 CNN(convolutional neural network)은 컴퓨터 영상 인식 분야에서 높은 성능을 보이며 많은 관심을 받고 있다[1-6]. CNN은 여러 층의 합성곱 층(convolutional layer)과 완전 연결 층(fully-connected layer)으로 구성되어 있으며 이로 인해 많은 양의 계수와 연산양을 요구하고 있다. 이러한 높은 연산양으로 인해 연산 자원이 많지 않은 임베디드 환경에서 사용하기에 쉽지 않은 문제가 있다. 이를 해결하기 위해 가지치기(pruning)과 전용 가속 회로(accelerator)에 대한 많은 연구들이 이루어져 왔다.

가지치기는 CNN의 계수들 중 일부를 0으로 만드는 것이다[7-14]. CNN에 있어서 대부분의 연산은 데이터와 계수를 곱한 뒤 누적하는 것으로써, 계수를 0으로 만들면 곱하고 누적하는 것이 의미가 없으므로 그 만큼 연산양을 줄일 수 있다. 그리고 0인 계수는 저장할 필요가 없으므로 계수 저장 공간도 줄일 수 있다.

CNN 가속 회로는 범용 프로세서에 비해 유연성은 낮으나 연산 효율성이 높고 전력 소모가 낮은 장점이 있다[13-21]. 이로 인해 저전력을 요구하는 환경에서 많이 사용되며 CNN에 적합한 가속 회로의 구조에 대해 많이 연구되고 있다. CNN은 보통 많은 수의 곱셈기와 덧셈기로 구성되어 있으며 여러 개의 데이터와 계수들을 한 번에 읽은 뒤 연산자들을 이용하여 계산한다. 이 때 한 번에 읽는 데이터와 계수들을 어떻게 묶느냐에 따라 다양한 구조를 도출할 수 있다[15-23].

가지치기와 가속 회로는 모두 낮은 연산양과 저전력을 요구하는 임베디드 환경에 적합한 방법이나, 서로를 고려한 연구는 많지 않다. 가지치기된 CNN을 처리할 수 있는 가속 회로는 몇 가지가 제안되었으나[24-27], 가속 회로의 구조를 고려한 가지치기 기법에 대해서는 아직 연구가 많이 이루어지지 않고 있다.

이 논문에서는 가속 회로의 구조를 고려한 가지치기 기법인 Conv-XP 가지치기를 제안한다. 가속 회로 중에서 [21]에서 제안된 구조에서는 9개의 계수와 데이터를 동시에 읽어서 9개의 곱셈기로 곱셈을 한다. Conv-XP에서는 이 9개의 계수 중에서 4개를 0으로 가지치기한다. 9개의 계수 중 5개가 남는 모든 가지치기 경우를 고려하게 되면 가속 회로의 구조가 복잡해지므로 이 논문에서는 'X' 글자 모양과 '+' 기호 모양의 계수들이 남는 경우만을 고려할 것이다. 이 논문에서는 이와 같이 가지치기한 신경망을 위한 가속 회로에서는 곱셈기의 개수를 9개에서 5개로 줄임으로써 가속 회로의 면적이 감소할 수 있고, 기존의 방법으로 가지치기했을 때에 비해 0이 아닌 계수의 패턴이 한정되므로 선택 회로 등이 단순화될 수 있음을 보일 것이다. 더불어 가지치기 패턴을 'X'와 '+' 모양으로 제한하더라도 CNN의 성능이 저하되지 않음을 실험 결과로 제시할 것이다.

이 논문은 다음과 같이 구성되어 있다. Ⅱ장에서 CNN과 기존의 가지치기 기법 및 가속 회로를 소개하고, Ⅲ 장에서 Conv-XP 기법을 제안하며 이를 위한 가속 회로의 구조를 설명할 것이다. Ⅳ장에서 실험 결과를 제시하고, Ⅴ장에서 결론을 기술한다.

## Ⅱ. CNN

이 장에서는 CNN을 소개하고, 가지치기 기법과 가속 회로에서 대해 설명할 것이다.

### 2.1. CNN

CNN은 많은 수의 합성곱 층과 몇 개의 완전 연결 층으로 구성된다. 각 층에서는 입력 데이터에 계수들을 곱한 뒤 누적하는 연산을 수행한다. 합성곱 층에서 하나의 출력 데이터를 계산하는 식은 다음과 같다.

$$fo(m,r,c)=\sum_{n=0}^{N-1}\sum_{i=0}^{K-1}\sum_{j=0}^{K-1}w(m,n,i,j)\times fi(n,r+i,c+j)+b(m) \quad (1)$$

위 식에서 $fi$는 입력 데이터이고 $fo$는 출력 데이터이며 $w$는 계수이다. 마지막에 더하는 $b$는 바이어스 값이다. 하나의 $m$에 대한 계수 $w$는 $K\times K\times N$의 3차원적인 구조를 가지고 입력 데이터에 이 계수들을 적용하면 크기가 $R\times C$인 출력 데이터가 생성되며 이를 출력 피처맵이라고 한다. 한 개의 합성곱 층에는 이러한 출력 피처맵이 $M$장 출력된다. 따라서 합성곱 층의 계수 $w$는 $K\times K\times N\times M$의 4차원적인 구조를 가지는데 이 중 $K\times K$의 2차원 구조 계수 그룹들을 $K\times K$ 커널이라고 부른다.

CNN에는 이 외에도 합성곱 층과 완전 연결 층들의 사이에 활성화 층이나 풀링 층을 넣는다. 활성화 층에는 여러 종류가 있으며 이중 많이 사용되는 것은 ReLU로

써, 입력 데이터 rk 음수이면 0을 출력하고 양수이면 그대로 출력한다. 풀링 층은 일정한 영역의 입력 데이터들의 평균이나 최댓값을 출력한다. 이런 여러 가지의 층들 중에 합성곱 층이 가장 많은 연산양을 차지한다고 알려져 있다.

### 2.2. CNN의 가지치기

신경망에서 일부 계수들을 0으로 만드는 가지치기 기법에 대해서는 1990년대부터 연구되어 왔으나, 논문 [7]에서 현대적인 가지치기 기법이 제시되었다. 논문 [7]에서는 크기가 작은 계수들을 0으로 만든 뒤 남은 계수들을 가지고 다시 학습을 진행함으로써 원래의 성능을 복구하는 기법을 제시하였다. 이 이후로 다양한 가지치기 기법이 제안되었다.

채널 단위 가지치기는 입력 데이터 중 특정 채널들에 해당하는 계수들을 0으로 만드는 기법이다[9-14]. 입력 데이터의 채널은 (1)에서 입력 데이터 $fi(n,x,y)$ 중 $n$이 같은 데이터들의 집합으로써, 이에 해당하는 계수들도 $w(m,n,i,j)$중 $n$이 같은 계수들의 집합이다. 기존의 연구에서는 CNN의 채널들 중에서 가지치기할 채널을 어떻게 선택할 지에 대해 많이 다루었다.

이 외에도 모양 단위의 가지치기에 대한 연구도 이루어졌다[28,29]. 모양 단위의 가지치기에서는 $w(m,n,i,j)$ 중 $m, i, j$의 값이 같은 계수들을 묶은 단위로 0으로 만들지 결정하는 기법이다. 이 방법을 사용하여 가지치기된 CNN은, General-Purpose Graphic Processing Unit (범용 GPU)에서 구현했을 때 성능을 향상시킬 수 있다.

### 2.3. CNN의 가속 회로

CNN은 다양한 방법으로 구현될 수 있으며, 이 중 가장 연산의 효율성이 높고 전력이 낮은 것은 전용 가속 회로로 구현하는 방법이다[15-23]. 이러한 특성으로 가속 회로는 저전력으로 많은 연산을 처리해야 하는 환경에서 많이 사용된다. CNN에 있어서 대부분의 연산은 합성곱 층에서 이루어지므로 가속 회로들은 주로 합성곱 층을 다룬다.

CNN의 가속 회로는 대부분 많은 양의 곱셈기와 덧셈기로 이루어져 있으며, 이들 연산회로들의 피연산자들을 가져오는 방식에 따라 분류할 수 있다. DianNao 계열의 가속 회로들은 채널 방향의 데이터와 계수들을 가져와서 연산을 하며[15,16,26], 논문 [21]에서 제시된 가속 회로에서는 3×3 합성곱 층(식(1)에서 $K$=3인 경우)을 처리하기 위해 $w(m,n,i,j)$중 $m,n$의 값이 같은 계수들(즉, 한 개의 커널)과 그에 해당하는 입력 데이터들을 가져와서 연산을 한다.

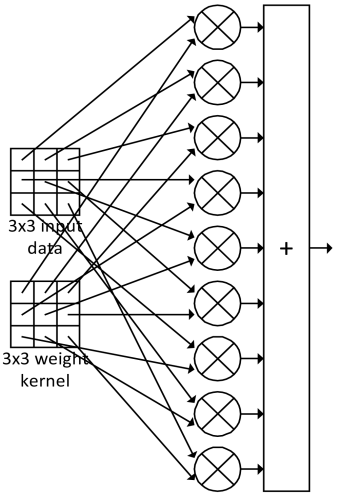


**Fig. 1** Operation architecture of the accelerator in [21]

논문 [21]에서 제시된 가속 회로의 연산 부분을 그린 것이 그림 1이다. 한 개의 커널에 속하는 계수 9개를 계수 버퍼에서 읽고, 입력 데이터 버퍼에서는 그에 해당하는 입력 데이터 9개를 동시에 읽어서 각각을 곱한다. 이러한 곱셈을 위해서는 9개의 곱셈기가 요구된다. 곱셈 결과를 모두 더해진 뒤 누적된다.

가지치기된 CNN을 처리할 수 있는 가속 회로는 sparse 가속 회로라 불린다. Cambricon-X는 대표적인 sparse 가속 회로 중 하나로써 DianNao 계열에 속한다[26]. Sparse 가속 회로에 대한 연구가 진행되고는 있으나 가지치기의 과정에서 가속 회로를 고려하지 않으므로 sparse 가속 회로는 동작할 때 낭비가 발생할 수 있다. 예를 들어, Cambricon-X의 경우 계수와 입력 데이터를 정렬하기 위해 padding zero를 넣어야 한다.

논문 [21]와 같은 구조를 가지는 sparse 가속 회로는 아직 발표되고 있지 않으나 그림 1의 구조로부터 쉽게 유도할 수 있다. 0으로 가지치기된 계수들과의 곱셈은 의미가 없으므로 곱셈기의 개수를 줄일 수 있으며, 그림 2는 그림 1의 구조를 변형하여 5개의 곱셈기를 가지는 sparse 가속 회로 구조로 바꾼 것이다. 5개의 곱셈기를 이용해서 하나의 3×3 커널에 속하는 0이 아닌 계수

를 한 싸이클에 5개까지 처리할 수 있다.

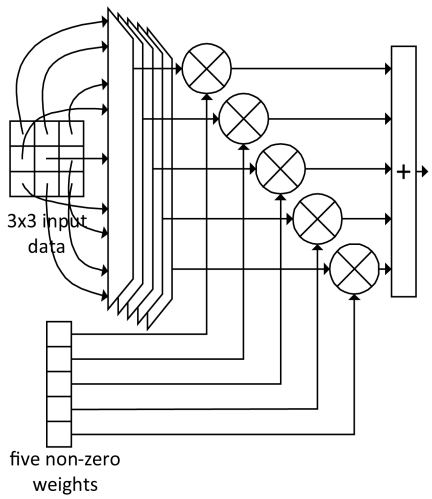


**Fig. 2** Multiplier connection of a sparse accelerator for the conventionally pruned networks

그림 2의 구조에서 곱셈기 앞에 배치된 MUX들은 3×3 영역의 입력 데이터에서 적절한 것들을 선택하는 MUX들이다. 기존의 가지치기 기법에서는 가지치기되는 계수의 패턴이 정해져 있지 않으므로, 3×3 커널 내의 9개 계수 중에서 어느 것이 0이 아닌 것으로 남을 지 미리 알 수 없다. 그러므로 어느 0이 아닌 계수와 같이 곱해야 할 입력 데이터는 9개의 입력 데이터 중 어느 것도 될 수 있다. 이에 대처하기 위해 각각의 곱셈기 앞에는 9-to-1 MUX를 배치해서 적절한 입력 데이터를 선택해야 한다.

## Ⅲ. Conv-XP

이 논문에서 주장하는 Conv-XP는 그림 1과 그림 2와 같은 논문 [21]에서 제안된 가속 회로의 구조에 적합한 가지치기 기법이다. Conv-XP에서는 그림 1과 그림 2에서 동시에 처리하는 단위인 한 커널 내의 계수 9개 중에서 일부 계수를 가지치기한다. 기존 가지치기 기법에서는 9개의 계수 중에서 몇 개의 계수가 가지치기될 지, 그리고 어떤 위치의 계수들이 0이 될 지가 정해져 있지 않다. 즉, 프루닝되고 남은 0이 아닌 계수들이 어떤 패턴을 가질지 미리 알 수 없었다. 이에 반해 Conv-XP에서는 그림 3과 같은 두 가지 패턴으로만 가지치기를 한다. 그림에서 흰색 칸은 0으로 가지치기되는 계수를, 회색은 가지치기 후에 0인 아닌 값으로 남는 계수를 의미한다. 두 개의 패턴은 각각 글자 'X'와 기호 '+'(plus)를 닮아서 Conv-XP라고 작명하였다.

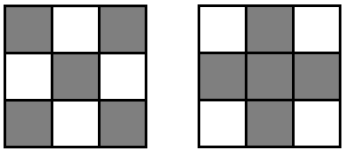


**Fig. 3** Conv-XP pruning pattern: (a) 'X' pattern and (b) '+' pattern

Conv-XP 가지치기는 각각의 3×3 커널에 대해 적용된다. 각각의 3×3 커널에 대해 'X' 패턴과 '+' 패턴에 있는 계수들의 절댓값의 합을 구한 뒤, 그 합이 더 큰 패턴을 취하고 패턴에 따라 계수들을 0으로 만든다. 모든 3×3 커널에 대해 가지치기를 수행한 뒤, 0인 계수들을 0으로 유지하면서 다시 학습한다.

### 3.1. Conv-XP와 가속 회로 복잡도

Conv-XP가 적용된 CNN을 가정하게 되면 그림 1이나 그림 2와 같은 구조의 가속 회로에서 면적을 많이 줄일 수 있다. 가속 회로에 있어서 곱셈기는 많은 면적을 차지하는 연산회로인데, 논문 [21]에서는 3×3 커널을 한 싸이클에 처리하기 위해 9개의 곱셈기를 사용한다. Conv-XP를 사용할 경우, 9개의 계수 중에서 4개가 0이 되므로 이들에 대한 곱셈은 의미가 없어져서 곱셈기의 개수를 9개에서 5개로 줄일 수 있다.

기존의 가지치기 기법이 적용된 CNN을 가정하더라도 그림 2와 같이 곱셈기의 개수 감소를 고려할 수 있으나, 기존의 기법에서는 가속 회로의 구조에 대한 고려가 없어서 가속 회로에서 낭비되는 동작이 생겨난다. 예를 들어 논문 [7]에서 제안된 가지치기에서는 각 레이어의 전체 계수 중에서 크기가 작은 계수들을 0으로 만듦으로, 어느 한 3×3 커널에서는 몇 개의 계수가 0으로 될지 예측할 수 없다. 9개의 모든 계수가 0으로 될 수도 있고 9개 모두 0이 아닌 계수로 남을 수도 있다. 만일 논문 [21]에서와 같이 3×3 커널을 한 싸이클 안에 처리하려고 한다면, 이러한 모든 경우에 대처하기 위해서 곱셈기를

9개로 유지해야 한다. 그렇지 않고 그림 2와 같이 곱셈기의 개수를 5개로 줄인다면 어떤 경우에는 하나의 커널을 두 싸이클에 걸쳐 처리해야 한다.

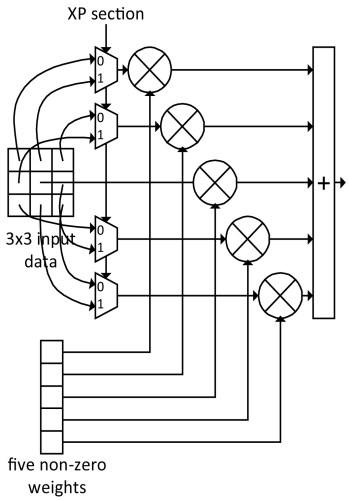


**Fig. 4** Multiplier connection of a sparse accelerator for the networks pruned by Conv-XP

기존의 가지치기 기법이 적용된 CNN을 그림 2와 같은 가속 회로에서 처리하려 할 때의 또 다른 문제점은 입력 데이터 선택이다. 그림 2의 구조에서 각각의 곱셈기 앞에는 0이 아닌 계수와 곱할 입력 데이터를 선택하기 위해 9-to-1 MUX를 배치하고 있다. 그림에서는 한 개의 곱셈기에 한 개의 9-to-1 MUX가 필요한 것으로 그려져 있으나, 실제로는 데이터의 비트 폭과 같은 개수의 9-to-1 MUX가 각 곱셈기마다 필요하다.

이러한 기존의 가지치기 기법에 비해 Conv-XP 기법은 CNN을 가속 회로에 적합하도록 가지치기한다. Conv-XP에서는 가속 회로에서 동시에 처리되는 각각의 3×3 커널에 대해 항상 4개의 계수가 0이 되도록 가지치기한다. 그러므로 5개의 곱셈기로도 항상 3x3 커널을 한 싸이클 안에 처리할 수 있음이 보장된다. 그리고 'X'와 '+', 두 가지 패턴만 사용하므로 한 곱셈기에는 두 개의 입력 데이터만이 피연산자가 될 가능성이 있다. 이를 이용하면 2-to-1 MUX 만으로도 곱셈기의 입력을 선택할 수 있다. 이러한 사항들을 고려한 회로의 구조가 그림 4이다.

### 3.2. Conv-XP와 가속 회로 메모리

Conv-XP를 사용했을 때 또 하나의 이점은 메모리 면적을 줄일 수 있다는 것이다. 기존의 가지치기 기법에서는 각각의 0이 아닌 계수들이 곱해질 입력 데이터를 선택하기 위해, 각 곱셈기 앞에 위치하는 9-to-1 MUX의 선택신호를 같이 저장해야 한다. 한 개의 MUX에 4bit의 선택신호가 필요하고, 5개의 곱셈기를 가정하면 총 20bit의 선택신호가 필요하다.

그러나 Conv-XP를 사용하게 되면 'X' 패턴과 '+' 패턴 사이에만 선택하므로 각 곱셈기 앞에는 2-to-1 MUX만 있으며 되고 이 MUX의 선택 신호 길이는 1bit로 충분하다. 그리고 5개의 곱셈기 앞에 있는 MUX들이 같은 선택신호를 공유하므로 5개의 0이 아닌 계수가 하나의 선택 신호를 공유하게 된다. 그러므로 5개의 곱셈기에 대해 20bit의 선택신호가 1bit로 줄어들게 되고, 따라서 이 선택 신호들을 저장하는 메모리의 크기가 줄어들게 된다.

## Ⅳ. 실험 결과

제안된 Conv-XP 기법을 적용해도 CNN의 성능이 악화되지 않음을 보이기 위해, ImageNet 2012 validation dataset을 이용하여 classification top-5 정확도를 비교하였다. VGG16[4]과 ResNet-50[6]의 학습된 모델에 대해 Conv-XP를 적용한 뒤 재학습했을 경우의 정확도를 측정하였다. VGG16과 ResNet-50은 많은 곳에서 사용되는 대표적인 CNN이면서 3x3 커널을 많이 가지고 있어서 Conv-XP를 적용하기에 적합한 CNN이다. ResNet-50의 경우 3x3 합성곱 층에만 Conv-XP를 적용하였다.

**Table. 1** Classification accuracy comparison on VGG16 and ResNet-50

| | VGG16 | ResNet-50 |
|---|---|---|
| baseline | 88.44% | 91.14% |
| Conv-XP | 89.74% | 91.30% |

표1에서 VGG16과 ResNet-50에 대해 Conv-XP를 적용하기 전의 정확도('baseline')와 Conv-XP를 적용한 후 재학습했을 때의 정확도('Conv-XP')를 비교하였다. 재학습은 학습 속도를 0.001로 하고 12 epoch 동안 학습하였다. 표1의 결과에 따르면 Conv-XP를 적용하더라도

**Table. 2** Area comparison ($\mu m^2$)

| | 16 PEs | Weight Buffer | Index Buffer | Activation MUX | Miscellaneous | Total |
|---|---|---|---|---|---|---|
| No sparsity | 642826 | 507042 | - | - | 3199 | 1153067 |
| Conventional sparsity | 357640 | 281691 | 81895 | 25078 | 56 | 746360 |
| Conv-XP | 357364 | 281691 | 5916 | 6157 | 47 | 651175 |

CNN의 성능이 저하되지 않음을 알 수 있다.

Conv-XP를 사용했을 때 가속 회로의 복잡도 개선을 비교하기 위해서 세 가지의 회로를 RTL(register transfer level)에서 설계한 뒤 합성하여 비교하였다. 첫 번째는 가지치기 되지 않은 CNN을 가정한 그림 1과 같은 기존의 구조이고, 두 번째는 기존의 가지치기 기법이 적용된 CNN을 위한 sparse 가속 회로인 그림 2와 같은 구조이며, 세 번째는 Conv-XP로 가지치기된 CNN을 위한 sparse 가속 회로인 그림 4와 같은 구조이다. 표 2에서는 각각을 'No sparsity', 'Conventional sparsity', 'Conv-XP'로 표기하였다. 합성 라이브러리는 Global Foundry사의 65nm 공정을 이용하였다. 표 2에서 PE(processing element)는 9개 또는 5개의 곱셈기로 이루어져서 하나의 3x3 커널을 동시에 처리할 수 있는 블록을 의미하며, 이러한 PE가 16개 있는 것을 가정하였다.

기존의 가지치기 기법을 위한 sparse 가속 회로(표2에서 'Conventional sparsity')는 가지치기를 고려하지 않는 일반적인 가속 회로(표 2에서 'No sparsity')에 비해 곱셈기의 개수가 9개에서 5개로 줄고 필요한 계수의 개수도 9개에서 5개로 감소하므로, PE와 계수 버퍼(Weight Buffer)의 면적이 감소하였다. 그러나 그림 2에 그려져 있는 9-to-1 MUX들을 모아 놓은 Activation MUX가 추가되고, MUX 선택신호를 저장하는 Index Buffer도 역시 추가된다.

CNN을 Conv-XP로 가지치기하면 sparse 가속 회로에서 Activation MUX와 Index Buffer의 크기를 줄일 수 있다. 기존의 가지치기 기법에서는 가지치기의 패턴이 없으므로 9개의 입력 데이터 모두로부터 1개의 데이터를 선택하는 9-to-1 MUX가 필요했지만, Conv-XP에서는 두 가지의 패턴 중 하나로 가지치기하므로 2-to-1 MUX로 충분하다. 그러므로 Activation MUX 블록의 크기가 약 25%로 감소된다. 그리고 MUX 선택 신호의 비트 폭도 4 비트에서 1 비트로 줄어들고, 5개의 계수들이 하나의 선택신호를 공유하므로 결국 20 비트에서 1 비트로 줄어든다. 그 결과 Index Buffer의 크기가 약 7.2%로 감소하였다. 이러한 효과들로 인해 Conv-XP를 가정하면 sparse 가속 회로의 면적이 기존의 sparse 가속 회로에 비해 약 12.8% 감소한다.

## Ⅴ. 결 론

이 논문에서는 CNN의 Conv-XP 가지치기 기법을 제안하였다. 기존의 가지치기 기법들이 가속 회로의 구조를 고려하지 않아서 생겨나는 가속 회로의 비효율성을 극복하기 위해 일정한 패턴만을 이용하여 가지치기를 하였다. 제안된 방식으로 가지치기된 CNN을 가정한 가속 회로는 기존의 가속 회로들에 비해 낮은 복잡도를 가진다. 가지치기 패턴에 제한을 가했음에도 대표적인 CNN들에서 성능이 악화되지 않음도 보였다. 이 연구에 이어서 가속 회로의 복잡도를 더 줄이기 위한 다양한 가지치기 패턴에 대해 더 연구할 수 있을 것이다.


## ACKNOWLEDGEMENT

This work was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2015R1D1A1A01058768). This work was also supported by the 2017 Professor Education and Research Promotion Program of KOREATECH and also supported by IDEC (EDA Tool).


## References

[ 1 ] Y. -J. Kim and E. -G. Kim, "Image based Fire Detection

using Convolutional Neural Network," *Journal of the Korea Institute of Information and Communication Engineering*, vol. 20, no. 9, pp. 1649-1656, Sep. 2016.

[ 2 ] S. -H. Kwon, K. -W. Park, and B. -H. Chang, "A Comparison of Predicting Movie Success between Artificial Neural Network and Decision Tree," *Asia-pacific Journal of Multimedia Services Convergent with Art, Humanities, and Sociology*, vol. 7, no. 4, pp. 593-601, Apr. 2017.

[ 3 ] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.

[ 4 ] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of International Conference on Learning Representations*, pp. 1-14, 2015.

[ 5 ] 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 Computer Vision and Pattern Recognition*, pp. 1-9, 2015.

[ 6 ] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 770-778, 2016.

[ 7 ] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 1135-1143, 2015.

[ 8 ] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proceedings of International Conference on Learning Representations*, pp. 1-14, 2016.

[ 9 ] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 2074-2082, 2016.

[10] N. Yu, S. Qiu, X. Hu, and J. Li, "Accelerating convolutional neural networks by group-wise 2D-filter pruning," in *Proceedings of International Joint Conference on Neural Networks*, pp. 2502-2509, 2017.

[11] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *Proceedings of International Conference on Learning Representations*, pp. 1-13, 2017.

[12] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of International Conference on Computer Vision*, pp. 1398-1406, 2017.

[13] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das, and S. Mahlke, "Scalpel: Customizing DNN pruning to the underlying hardware parallelism," in *Proceedings of International Symposium on Computer Architecture*, pp. 548-560, 2017.

[14] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proceedings of International Conference on Learning Representations*, pp. 1-17, 2017.

[15] 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 *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 269-283, 2014.

[16] 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 *Proceedings of International Symposium on Micro-architecture*, pp. 609-622, 2014.

[17] S. Liu, Z. Du, J. Tao, D. Han, T. Luo, Y. Zie, Y. Chen, and T. Chen, "Cambricon: An instruction set architecture for neural networks," in *Proceedings of International Symposium on Computer Architecture*, pp. 393-405, 2016.

[18] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 161-170, 2015.

[19] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *Proceedings of International Symposium on Computer Architecture*, pp. 1-13, 2016.

[20] L. Song, Y. Wang, Y. Han, X. Zhao, B. Liu, and X. Li, "C-Brain: A deep learning accelerator that tames the diversity of CNNs through adaptive data-level parallelization," in *Proceedings of Design Automation Conference*, pp. 123:1-123:6, 2016.

[21] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going deeper with embedded FPGA platform for convolutional neural network," in *Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 26-35, 2016.

[22] M. Motamedi, P. Gysel, V. Akella, and S. Ghiasi, "Design

space exploration of FPGA-based deep convolutional neural networks," in *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 575-580, 2016.

[23] J. Jo, S. Cha, D. Rho, and I.-C. Park, "DSIP: A scalable inference accelerator for convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 605-618, Feb. 2018.

[24] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," in *Proceedings of International Symposium on Computer Architecture*, pp. 243-254, 2016.

[25] S. Han, J. Kang, H. Mao, Y. Hu, X. Li, Y. Li, D. Xie, H. Luo, S. Yao, Y. Wang, H. Yang, and W. J. Dally, "ESE: Efficient speech recognition engine with sparse LSTM on FPGA," in *Proceedings of ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 75-84, 2017.

[26] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chyen, "Cambricon-X: An accelerator for sparse neural networks," in *Proceedings of International Symposium on Micro-architecture*, pp. 20:1-20:12, 2016.

[27] D. Kim, J. Ahn, and S. Yoo, "A novel zero weight/ activation-aware hardware architecture of convolutional neural network," in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition*, pp. 1462-1467, 2017.

[28] V. Lebedev and V. Lempitsky, "Fast ConvNets using group-wise brain damage," in *Proceedings of Computer Vision and Pattern Recognition*, pp. 2554-2564, 2016.

[29] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing System*, vol. 13, no. 3, pp. 32:1-32:18, Feb. 2017.

**우용근(Yonggeun Woo)**

2017년 한국기술교육대학교 컴퓨터공학부 학사
2018년 ~ 한국기술교육대학교 컴퓨터공학과 석사과정
※관심분야 : 딥러닝 프로세서 설계

**강형주(Hyeong-Ju Kang)**

1998년 한국과학기술원 전기및전자공학과 학사
2000년 한국과학기술원 전기및전자공학과 석사
2005년 한국과학기술원 전자전산학과 박사
2005년~2006년 (주)매그나칩반도체 선임연구원
2006년~2009년 (주)지씨티리써치 선임연구원
2009년~현재 한국기술교육대학교 컴퓨터공학부 전임강사/조교수
※관심분야 : VLSI설계 및 CAD, 마이크로프로세서 설계, 통신 모뎀 설계