희소 행렬 곱셈을 효율적으로 수행하기 위한 유동적 시스톨릭 어레이 구조 설계

서주원, 공준호
경북대학교 전자전기공학부
mscjos97@knu.ac.kr, joonho.kong@knu.ac.kr

Design of the Adaptive Systolic Array Architecture for Efficient Sparse Matrix Multiplication

Juwon Seo, Joonho Kong
School of Electronic and Electrical Engineering, Kyungpook National University

요 약

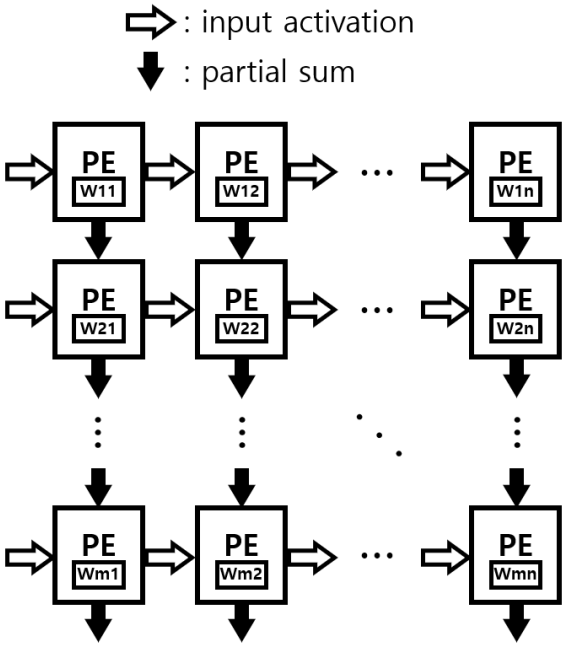
시스톨릭 어레이는 DNN training 등 인공지능 연산의 대부분을 차지하는 행렬 곱셈을 수행하기 위한 하드웨어 구조로 많이 사용되지만, sparsity 가 높은 행렬을 연산할 때 불필요한 동작으로 인해 효율성이 크게 떨어진다. 본 논문에서 제안된 유동적 시스톨릭 어레이는 matrix condensing, weight switching, 그리고 direct output path 의 방법과 구조를 통해 sparsity 가 높은 행렬 곱셈의 수행 사이클을 줄일 수 있다. 시뮬레이션을 통해 기존 시스톨릭 어레이와 유동적 시스톨릭 어레이의 성능을 비교하였으며 8X8, 16X16, 32X32 의 크기를 가진 행렬을 동일 크기의 시스톨릭 어레이로 연산하였을 때 필요 사이클 수를 최대 12 사이클 절감할 수 있는 것을 확인하였다.

1. 서론

최근 인공지능 분야에서 Deep Neural Network(DNN) training 등의 연산을 위해 행렬 곱셈이 많이 사용된다. 행렬 곱셈을 수행하기 위한 구조인 시스톨릭 어레이(Systolic Array) [1]는 multiply and accumulation(MAC) 연산을 수행하는 processing element(PE)들의 배열로 구성되며, 데이터 재사용을 위해 보통 두 input 행렬 중 하나 혹은 output 행렬의 value 들을 PE 내의 레지스터에 저장하여 데이터 로딩 시간을 줄인다. 그러나, 많은 양의 정보를 담기 위해 행렬의 크기가 점점 커지고 있고, ReLU activation [2]과 pruning [3]에 의해서 0 값이 많이 존재하기 때문에 시스톨릭 어레이는 이를 연산하는 과정에서 불필요한 동작이 많이 발생한다. 따라서, 본 논문에서는 sparsity 에 따라 다르게 동작하도록 하는 유동적 시스톨릭 어레이를 제안하여 행렬의 곱셈을 빠르고 효율적으로 수행하고자 한다.

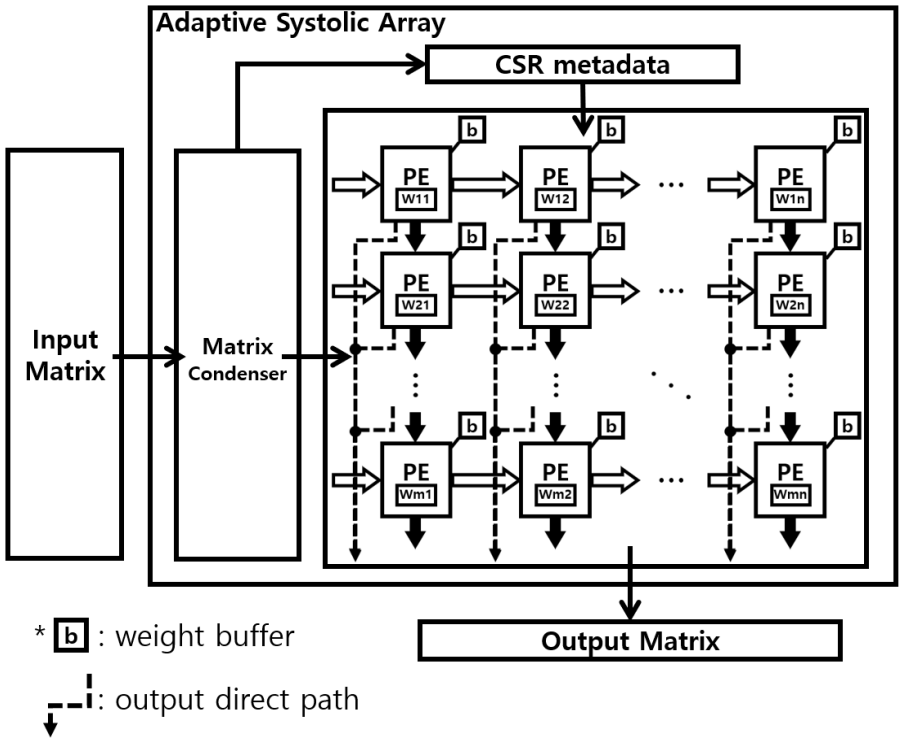
2. 유동적 시스톨릭 어레이의 구조 및 연산과정

Weight stationary 기반의 시스톨릭 어레이의 구조는 <그림 1>과 같다. Input value 는 input activation path 를 따라서 매 사이클마다 이동하며, PE 에 미리 저장된 weight value 와 곱해진다. 곱한 결과를 아래의 partial sum path 로 내보내는데 이를 아래의 PE 가 받아서 자신의 연산(input X weight) 결과와 합친 후 아래의 partial sum path 로 내보낸다. 따라서, 가장 윗줄의 PE 들은 partial sum 입력이 없고, 가장 아랫줄의 PE 들의 partial sum 출력이 최종 output 이 된다. 이 output 값들을 재정렬하면 input 행렬과 weight 행렬을 곱한 output 행렬이 완성된다.



<그림 1> 시스톨릭 어레이의 구조 (m X n)

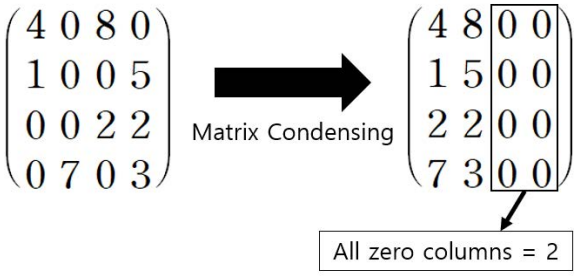
본 논문에서 제안하는 유동적 시스톨릭 어레이의 구조는 <그림 2>와 같으며 다음과 같은 구조 및 연산 과정을 가지고 있다.



<그림 2> 유동적 시스톨릭 어레이 구조

가) Matrix condenser

희소 행렬은 0 값과 0 이 아닌 값들이 섞여 있는데 이를 그대로 시스톨릭 어레이에 넣게 되면 중간중간에 등장하는 0 값으로 인해 불필요한 이동 및 연산이 일어난다. 이때 matrix condensing [4]을 사용하면 <그림 3>에서처럼 0 이 아닌 값들을 따로 분리할 수 있고 해당 값들을 먼저 연산한 후, 뒤에 등장하는 0 값들을 무시하도록 할 수 있다. 또한, 나중에 condensing 이전과 이후 value 들의 위치를 비교하기 위해 CSR format [5]의 metadata 를 따로 저장한다.



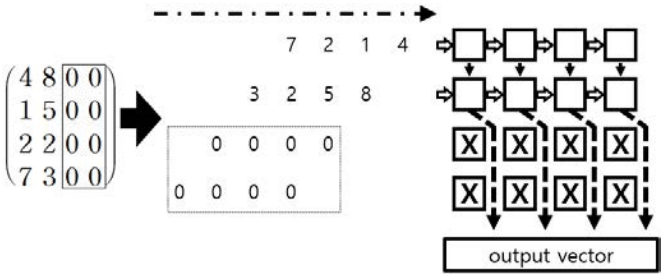
<그림 3> Matrix condensing 과 All zero columns

나) Weight buffer

Condensing 을 진행한 행렬을 그대로 연산에 사용할 경우 일부 value 의 위치가 변경되기 때문에 잘못된 weight value 가 곱해진다. 이를 위해서 metadata 를 참조해 위치가 변경된 input value 에 올바르게 대응하는 weight value 를 해당 PE 에 불러와서 곱해지도록 하는 weight switching 을 사용하며, 본래 PE 에 저장되어 있던 weight value 는 weight buffer 에 잠시 저장해 놓았다가 위치가 변경되지 않아 본래 자리에 대응하는 input value 가 들어올 때 다시 가져와서 곱셈에 사용한다.

다) Direct path to output

<그림 3>에서 알 수 있듯이, 희소 행렬의 경우 matrix condensing 이후에 All zero columns 가 발생할 수 있다. <그림 4>에서처럼 All zero columns 는 시스톨릭 어레이에서 하나의 행에 전부 0 값인 벡터로 들어가기 때문에 해당 행에서는 곱셈과 덧셈이 일어날 필요가 없다. 따라서 해당 행들을 건너뛰도록 하면 불필요한 사이클을 줄일 수 있으며 더 빠르게 output 을 얻을 수 있다. 이를 위하여 별도의 path 를 가지며 이를 output direct path 라고 한다. 새롭게 제안된 유동적 시스톨릭 어레이에서는 이러한 output direct path 를 매 행마다 가지고 있으며 이를 통해 더 빠르게 연산 결과를 얻을 수 있다.



<그림 4> All zero columns 에 의한 output direct path

3. 실험 및 결과

본 논문에서 제안된 유동적 시스톨릭 어레이 구조 시뮬레이션을 위해 HPDLA [6]를 이용하여 기존의 시스톨릭 어레이와 유동적 시스톨릭 어레이를 비교하였다. 비슷한 sparsity(약 40%)를 가진 8X8, 16X16, 32X32

희소 행렬을 input 으로 사용하여 사이클이 얼마나 줄어들었는지를 확인하였다. 시스톨릭 어레이의 크기도 행렬의 크기와 동일하게 8X8, 16X16, 32X32 를 각각 사용하였다.

Matrix size 및 SA size	Original SA	Adaptive SA
8X8	22 cycles	19 cycles
16X16	46 cycles	40 cycles
32X32	94 cycles	82 cycles

<표 1> 기존 시스톨릭 어레이와 유동적 시스톨릭 어레이에서 연산에 소모되는 사이클 비교

All zero columns 의 수는 행렬의 크기가 8X8 일 때 3 개, 16X16 일 때 6 개, 32X32 일 때 12 개였으며, 이는 거의 sparsity 에 비례하게 나오는 것을 확인할 수 있다. 유동적 시스톨릭 어레이에서 해당 수만큼의 행들을 건너뛰었기 때문에 그만큼 수행 사이클이 줄어든 것을 확인할 수 있으며, 이는 잠재적인 성능 향상에 기여할 수 있다. 또한, weight switching 을 통해서 변형된 행렬임에도 올바른 weight 가 곱해지도록 하여 결과가 기존 시스톨릭 어레이의 output 행렬과 동일함을 시뮬레이션을 통해 검증하였다.

4. 결론

기존의 시스톨릭 어레이는 sparsity 가 높은 행렬을 연산할 경우 0 값의 불필요한 덧셈, 곱셈, 그리고 이동으로 인해 비효율성이 발생한다. 본 논문의 유동적 시스톨릭 어레이를 사용하면 matrix condensing, weight switching, output direct path 등의 방법을 통해 0 값을 가지는 행의 연산을 건너뛰어 효율적인 연산이 가능하며, 소모되는 사이클을 32X32 시스톨릭 어레이 및 행렬 크기 기준 12 사이클을 줄일 수 있어 희소 행렬 곱셈의 성능 향상에 기여할 수 있다.

사사

이 논문은 2022 년도 정부(교육부)의 재원으로 한국연구재단 기초연구사업의 지원을 받아 수행된 연구임 (No. 2021R1I1A3A04037455).

참고문헌

[1] Kung, H. T., and Charles E. Leiserson. "Systolic arrays (for VLSI)." Sparse Matrix Proceedings 1978. Vol. 1. Philadelphia, PA, USA: Society for industrial and applied mathematics, 1979.

[2] Sze, Vivienne, et al. "Efficient processing of deep neural networks: A tutorial and survey." Proceedings of the IEEE 105.12 (2017): 2295-2329.

[3] Han, Song, et al. "Learning both weights and connections for efficient neural network." Advances in neural information processing systems 28 (2015).

[4] Zhang, Zhekai, et al. "Sparch: Efficient architecture for sparse matrix multiplication." 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2020.

[5] Lee, Jong Hun, et al. "Row-Wise Product-Based Sparse Matrix Multiplication Hardware Accelerator With Optimal Load Balancing." IEEE Access 10 (2022): 64547-64559.

[6] https://github.com/maao666/HPDLA