프루닝된 컨볼루션 신경망의 고효율 연산 방법

이원혁, 박상수, 정기석*
한양대학교

louislee111@naver.com, po092000@hanyang.ac.kr, kchung@hanyang.ac.kr

Highly-efficient Computation Method for Pruned CNN

Won-Hyuk Lee, Sang-Soo Park, Ki-Seok Chung*
Hanyang University, Seoul, Korea.

요 약

최근, 컨볼루션 신경망이 깊어짐에 따라 연산량과 파라미터의 양이 커지는 것이 큰 문제로 대두되고 있다. 특히, 메모리 용량과 연산 능력이 제한된 임베디드 환경에서는 기존의 딥러닝 모델을 경량화하는 것이 필수적이며, 이를 위해서 프루닝 기법이 널리 사용되고 있다. 프루닝을 통해 간략화된 컨볼루션 신경망은 0 값이 많은 sparse 한 구조를 갖지만, 기존의 신경망 연산 방법은 피연산자가 0 값인 경우에도 그대로 연산을 하기 때문에 불필요한 연산이 많이 수행된다. 본 논문에서는 프루닝된 컨볼루션 신경망에서의 효율적인 연산 방법으로 외적을 이용한 Sparse Matrix-Matrix Multiplication (SpMM)을 제안한다. 실험을 통해, 임베디드 환경에서 기존 연산 방법 대비 속도가 최대 7.1 배 개선된 것을 확인하였다.

Ⅰ. 서 론

컨볼루션 신경망은 이미지 분류, 음성 인식, 번역 시스템 등 다양한 분야에서 문제 해결을 위한 방법으로 큰 주목을 받고 있다. 하지만 높은 정확도를 얻기 위해서 컨볼루션 신경망의 깊이가 점점 깊어지고 있다 [1]. 따라서, 많은 연산 량과 많은 수의 파라미터를 필요로 하며, 이를 하드웨어 자원이 제한적인 임베디드 환경에서 있는 그대로 실행하는 것은 쉽지 않다.

이러한 문제를 해결하기 위해, 정확성에 미치는 영향이 적은 연결을 제거하고, 정확성을 회복하기 위해 재 학습하는 프루닝 기법이 제안되었다 [2]. 프루닝 기법을 통해 가중치 연결 중 특정 임계 값을 넘지 못하는 값을 0 으로 만들어서 신경망의 중요하지 않은 가중치들을 제거한다. 그 결과로, 신경망의 모델 크기를 정확도의 큰 손실 없이 줄이는 것이 가능하다. 프루닝 이후의 가중치들은 많게는 80%까지 0 값을 갖는 경우가 존재한다. 또한 ReLU (Rectified Linear Unit) 함수를 통해 얻은 뉴런의 값도 0 값을 갖는 경우가 많다 [3].

뉴런과 가중치에 0 값이 많다면, 그에 해당되는 부분을 계산하는 것은 불필요한 연산에 해당된다. 기존의 연산 방법인 Multiple Channel Multiple Kernel (MCMK)과 Image to Column (im2col) 등의 방법은 0 에 해당되는 값도 계산에 포함하기 때문에 불필요한 연산이 많게는 80% 이상 존재한다 [3]. 본 논문에서는 이러한 비효율적인 연산을 개선하기 위하여, 외적을 이용한 Sparse Matrix-Matrix Multiplication (SpMM)을 프루닝 된 컨볼루션 신경망의 연산에 적용하여, 연산 효율을 높이는 방법을 제시한다.

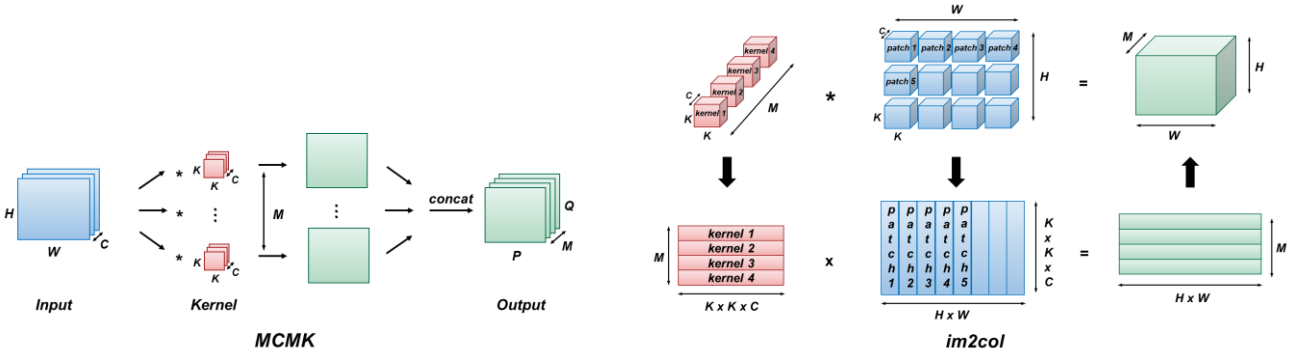
Ⅱ. 본론

2.1 기존 연산 방법 분석

컨볼루션 신경망의 연산에는 MCMK 와 im2col 이 널리 사용되고 있다 [4]. MCMK 는 그림 1 과 같이 높이, 폭, 깊이 성분을 갖는 3 차원 입력 피쳐맵과 높이, 폭, 깊이, 개수 성분을 갖는 4 차원 가중치 커널의 요소별 곱셈을 통하여 3 차원 출력 피쳐맵 이 나오는 연산 방법이다. Im2col 은 3 차원 입력 피쳐맵과 4 차원 가중치 커널을 2 차원 행렬로 변환한 후, 행렬 곱셈을 통하여 2 차원 출력 행렬을 얻는 연산 방법이다. Im2col 에서 결과로 얻은 출력 행렬을 변형 과정을 통하여 MCMK 의 결과와 같은 3 차원 출력 피쳐맵을 얻을 수 있다. 하지만 기존 컨볼루션 신경망에서 사용하는 두 연산 방법 모두 0 인 값을 연산에서 제외하지 않기 때문에 0 인 값이 많다면 다수의 불필요한 연산이 이뤄지게 된다.

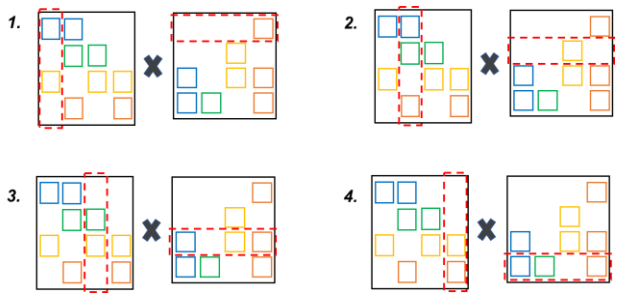
2.2 Sparse Matrix-Matrix Multiplication

비효율적 연산을 개선하기 위하여 본 논문에서는 SpMM 을 제시한다. SpMM 은 입력 행렬이 0 이 많은 행렬일 때, 0 인 값은 제외하고 0 이 아닌 값들에 대해서만 연산을 수행하여, 불필요한 연산을 하지 않는 행렬 곱셈 알고리즘이다. SpMM 을 하기 위해선 행렬 중 0 인 값들을 제외하고, 0 이 아닌 값들로 이루어진 배열로 재구성하여 저장해야 한다. 이러한 저장 방법은 Compressed Sparse Column (CSC) 그리고 Compressed Sparse Row (CSR) 등의 방법들이 존재하며 [5], 본 논문에서는 입력 피쳐맵과 가중치 커널에 해당되는 값들을 저장하기 위해 CSC 와 CSR 을 같이 사용하였다.



(그림 1) MCMK 요소별 곱셈 (왼쪽)과 im2col 행렬 곱셈 (오른쪽)

변환한 CSC 와 CSR 값은 그림 2 와 같이 외적 (Outer Product)을 사용하여 연산을 수행한다 [6]. 외적은 행렬의 열과 행의 곱셈을 통하여 얻어진 행렬을 부분 합하여 결과를 얻어내는 방식이다. SpMM 에서 내적 (Inner Product)은 CSC 와 CSR 의 0 이 아닌 값들의 인덱스가 같은지를 확인한 후 곱셈을 하여야 하기 때문에 비효율적이다. 하지만 외적은 이러한 확인이 필요하지 않기 때문에 불필요한 연산이 줄어들고 병렬처리에 용이해진다. 또한 외적을 사용하면 0 이 아닌 값들을 최대로 재사용 할 수 있고, CSC 의 열과 CSR 의 행을 최소한으로 읽을 수 있어서 효율적이다.



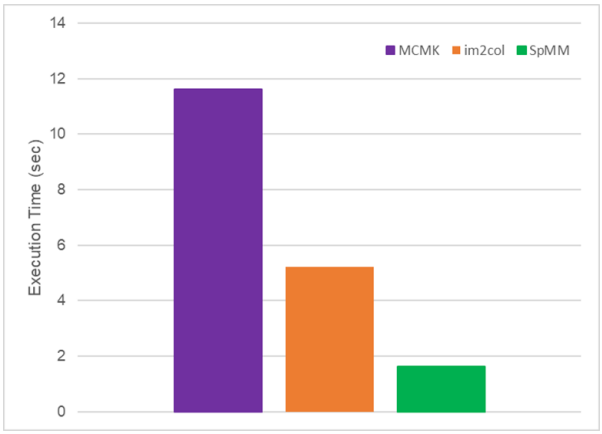
(그림 2) 외적을 이용한 SpMM 연산 방법

2.3 실험 결과 및 분석

본 논문의 실험은 임베디드 환경인 Odroid-N2 에서 진행하였다. Odroid-N2 의 프로세서는 ARM 社의 Cortex-A73 과 Cortex-A53 으로 구성되어 있으며, 메모리는 4GB 를 포함하고 있다. 실험에서는 기존 연산 방법인 MCMK, im2col 과 본 논문에서 제시한 외적을 이용한 SpMM 연산방법을 Pruned AlexNet [2] 모델에서 추론 실행 시간을 측정하였다. 그림 3 은 추론 실행 시간을 그래프로 나타낸 것으로, MCMK 는 11.6 초, im2col 은 5.22 초, 그리고 SpMM 은 1.63 초를 기록하였다. 그 결과, SpMM 의 사용을 통해 기존 연산 방법 대비 7.1 배 (MCMK), 3.2 배 (im2col) 실행 시간이 개선되었다.

Ⅲ. 결론

컨볼루션 신경망이 깊어짐에 따라 신경망에 프루닝 기법을 적용하는 연구가 진행되고 있다. 하지만 기존 신경망의 연산 방법은 프루닝된 네트워크에서 불필요한 연산을 포함하고 있다. 본 논문에서는 외적을 이용한 SpMM 을 프루닝된 신경망에서의 효율적인 연산 방법으로 제시하였다. 실험을 통해, 임베디드 환경인 Odroid-N2 에서 외적을 이용한 SpMM 이 기존 연산 방법 대비 실행 시간을 최대 7.1 배 개선할 수 있는 것을 확인하였다.



(그림 3) MCMK, im2col, SpMM 의 추론 실행시간 비교

ACKNOWLEDGMENT

본 연구는 IDEC 에서 EDA Tool 을 지원받아 수행하였습니다.

참 고 문 헌

[1] Christian Szegedy, Wei Liu, et al. “Going deeper with convolutions,” IEEE CVPR, 2015.

[2] Song Han, Jeff Pool, et al. “Learning both Weights and Connections for Efficient Neural Networks”, NIPS, 2015.

[3] Angshuman Parshar, Minsoo Rhu, et al. “SCNN: An Accelerator for Compressed-sparse Convolutional Neural Networks”, ISCA, 2017.

[4] Aravind Vasudevan, Andrew Anderson, et al. “Parallel Multi Channel Convolution using General Matrix Multiplication”, IEEE ASAP, 2017.

[5] Fred G. Gustavson, “Some basic techniques for solving sparse systems of linear equations”, in Sparse matrices and their applications, pp. 41- 52, Springer, 1972.

[6] Subhankar Pal, Jonathan Beaumont, et al. “OuterSPACE: An Outer Product based Sparse Matrix Multiplication Accelerator”, IEEE HPCA, 2018.