CPU 기반 장치에서 효율적인 딥러닝 추론을 위한 컨볼루션 신경망의 가속 방법의 성능 비교

박상수, 정기석
한양대학교

po092000@hanyang.ac.kr, kchung@hanyang.ac.kr

Performance Comparison of Acceleration Methods for Efficient Deep Learning Inference on CPUs

Sang-Soo Park, Ki-Seok Chung
Hanyang University, Seoul, Korea

요 약

Internet of Things (IoT) 디바이스와 같은 저전력 소형 장치에서 CPU 를 효과적으로 활용하여 컨볼루션 신경망의 추론의 실행 시간을 개선하는 연구가 활발히 이뤄지고 있다. 이런 연구들은 연산량을 감소시킨 컨볼루션 신경망 모델의 설계, 신경망 압축 방법 등을 적용하여 제한적인 하드웨어 성능에서 딥러닝의 추론의 실행시간 개선을 위한 방법을 제안하였다. 하지만 이러한 최적화 방법은 컨볼루션 레이어의 특성에 따라 가속화 효과가 클 수도 있지만, 매우 제한적일 때도 많다. 본 논문에서는 IoT 장치에서 몇 가지 컨볼루션 레이어의 가속화 기법을 적용한 후 성능을 측정하고, 분석하였으며, 이를 통하여 컨볼루션 레이어의 특성에 적합한 가속화 방법을 적용하였을 때, 가속화 전후 대비 최대 17.68 배 개선된 실행 시간의 성능 향상을 얻을 수 있다는 것을 보인다.

Ⅰ. 서 론

컨볼루션 신경망은 인간의 뉴런 구조를 모사한 기계 학습 모델로, 최근 이미지 분류, 음성인식, 자연어 처리 등 다양한 분야에 적용되어 우수한 분류 성능을 보여주고 있다 [1,2]. 우수한 분류 성능을 얻기 위해서, 컨볼루션 신경망은 깊은 깊이의 망으로 구성되어 있으며, 이는 연산량의 증가를 야기한다. 현재, 대부분의 연구는 많은 연산이 요구되는 신경망을 가속하기 위해 고성능 Graphic Processing Unit (GPU)나 전용 하드웨어 가속기를 이용한 방법을 제안하고 있다 [3,4].

최근, Internet of Things (IoT) 디바이스와 같은 제한된 연산 성능의 디바이스에서 컨볼루션 신경망을 가속하기 위한 에지 컴퓨팅에 대한 연구가 진행되고 있다 [5-7]. IoT 장치는 제한된 연산능력을 갖추고 있기 때문에, 컨볼루션 신경망의 추론 실행 시간을 단축시키기 위해서 경량화 된 모델의 설계, 네트워크 압축 기법들이 제안되었다. 이러한 연구는 모델의 크기를 줄이거나 곱셈 연산의 수를 줄이는 것에 중점을 두고 있다.

하지만, 이러한 신경망 경량화 방법은 컨볼루션 신경망의 특성에 따라 추론 시간 감소에 효과적일 때도 있지만, 효과가 제한적인 경우도 많이 발생한다. 따라서, 본 논문에서는 컨볼루션 신경망 가속에 사용되는 세 종류의 가속 방법 (Im2col, Im2row, MCMK)을 컨볼루션 레이어에 적용한 후 성능을 평가 및 분석하고, 이를 통하여 컨볼루션 레이어의 특성에 적합한 가속화 방법을 선택하였을 때 효과가 클 수 있다는 것을 보인다.

Ⅱ. 본론

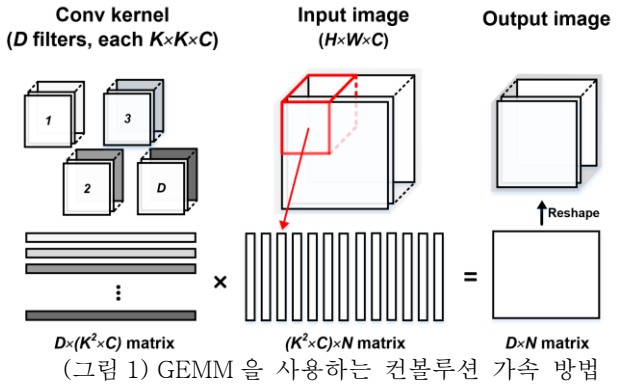
2.1 컨볼루션 신경망 구조

컨볼루션 신경망의 컨볼루션 레이어는 입력 피쳐맵 (Input feature map)과 컨볼루션 커널 (Convolution kernel)의 성분 별 곱셈으로 이뤄진다. 입력 피쳐맵과 컨볼루션 커널 간의 성분 별 곱셈을 계산하고, 바이어스 (Bias)를 더한 후 활성함수를 적용하여 출력 피쳐맵 (Output feature map)을 계산한다.

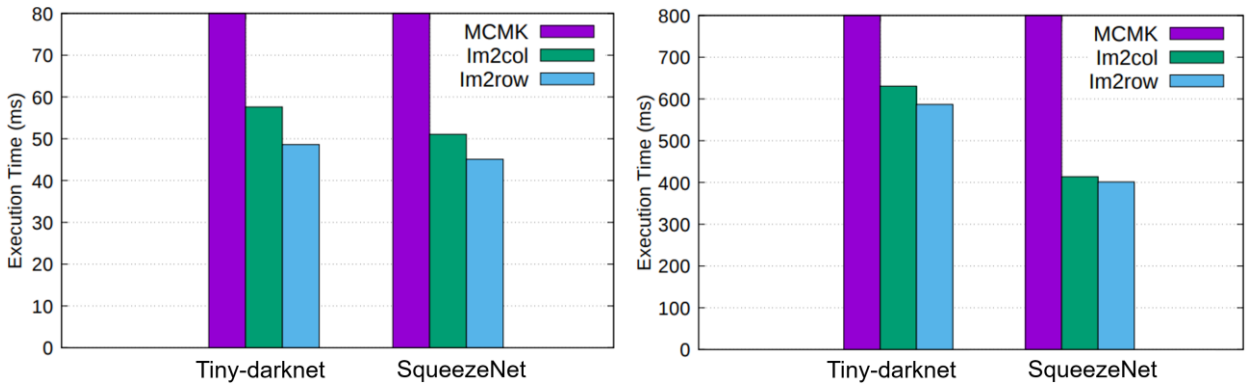
일반적으로 컨볼루션 레이어는 많은 연산량으로 인하여 IoT 디바이스에서 처리하기에는 적합하지 않으며, 이를 해결하기 위해 제한적인 연산 능력의 장치에서도 동작 가능한 SqueezeNet 과 Tiny-darknet 이 제안되었다 [6,7]. SqueezeNet 은 기존 3×3 컨볼루션의 일부를 1×1 컨볼루션으로 변환한 Fire 모듈을, Tiny-darknet 은 1×1 컨볼루션을 사용하여 연산량을 줄였다. 본 논문에서는 두 컨볼루션 신경망 모델에 몇 가지 가속화 기법을 적용하여 실행시간을 측정한 후, 적합한 가속 방법의 효과가 매우 큰 차이가 나는 것을 보인다.

2.2 컨볼루션 신경망의 가속 방법

컨볼루션 레이어를 가속하는 방법에는 행렬 연산을 기반으로 하는 General matrix multiplication (GEMM)과 Multi-channel Multi-kernel (MCMK) 방법이 있다.



(그림 1) GEMM 을 사용하는 컨볼루션 가속 방법



(a) Ryzen 1800x 결과 (b) Odroid XU4 결과

(그림 2) GEMM 을 사용하는 컨볼루션 가속 방법

행렬 연산을 기반으로 하는 GEMM 방법은 입력 피처맵과 컨볼루션 커널을 행렬 곱셈에서 사용 가능한 형태로 변환하여 계산을 하는 방식으로, 그림 1 과 같이 행렬로 변환하는 과정, 행렬 곱셈을 하는 과정으로 구성되어 있다.

GEMM 의 대표적인 방법인 Image to column (Im2col)은 H×W×C 형태의 입력 피처맵을 (K×K×C)×N 형태로 변환하여 행렬 곱셈을 하며 H, W, C 는 각각 입력 영상의 높이, 폭, 깊이, K 와 D 는 각각 커널 크기, 커널의 종류를 의미한다. 이외에도 입력 피처맵을 열 단위로 저장하는 Image to row (Im2row)이 있으며, Im2col 에 비해 캐시의 로컬리티를 더 효과적으로 활용하기 때문에 실행 시간이 더 빠르다.

MCMK 는 컨볼루션 레이어를 가속하는 가장 보편적인 방법으로 6 단계의 중첩된 for 문으로 구성되어 있다. MCMK 에서 입력, 출력 피처맵은 3 차원 형태 (H×W×C)로, 컨볼루션 커널은 4 차원 형태 (K×K×C×D)로 저장되어 연산에 사용된다.

2.3 실험 결과 및 분석

본 논문에서는 AMD 社의 Ryzen 1800x 와 Odroid XU4 에서 실험을 진행하였다. 두 플랫폼은 각각 4.0GHz, 2.0GHz 로 동작하는 8 개의 CPU 를 포함하고 있으며, 32GB, 2GB 메모리를 사용하였다.

세 종류의 컨볼루션 가속 방법을 Darknet 에 구현하여 실험을 진행하였으며, MCMK 는 OpenMP 를, GEMM 기반의 방법은 행렬 곱셈 연산에 OpenBLAS 를 적용하였다 [8,9].

그림 2 는 Ryzen 과 Odroid 에서 추론의 실행시간 결과를 나타내며, MCMK 보다 GEMM 을 사용한 방법이 일반적으로 더 빠르다. 이는 MCMK 는 다중 for 문으로 동작하기 때문에 GEMM 을 사용하는 방법보다 더 많은 시간이 소요되기 때문이다. GEMM 에서는 Im2col 이 Im2row 보다 더 많은 실행시간이 소요되었는데, 이는 Im2row 는 입력 피처맵을 열 단위로 저장하기 때문에 캐시의 hit ratio 가 더 좋기 때문이다. 표 1 은 MCMK, Im2col, Im2row 의 실행시간의 결과를 나타낸 것으로, 적합한 가속 방법의 사용을 통해 MCMK 대비 최대 17.68 배 (Ryzen), 13.15 배 (Odroid) 실행시간이 개선되었다.

(표 1) MCMK 대비 가속방법의 실행시간 개선

장치	신경망 모델	MCMK (ms)	Im2col (ms)	Im2row (ms)
Ryzen	Tiny- darknet	154.43	57.62	48.61
	SqueezeNet	797.42	51.06	45.1
Odroid	Tiny- darknet	7719.04	630.7	586.9
	SqueezeNet	3766.36	413.74	401.41

Ⅲ. 결론

IoT 디바이스와 같은 제한된 연산 능력을 가진 장치에서 컨볼루션 신경망을 가속하기 위한 연구가 진행되고 있다. 제한된 연산 능력으로도 동작 할 수 있도록 다양한 모델 경량화 방법이 개발되었지만, 신경망을 가속하는 방법에 따라 효과 차이가 크다. 본 논문에서는 신경망을 가속하는 방법을 분석하고, 최적의 가속 방법을 찾는 방법에 대해 소개하였다. 실험을 통해, 컨볼루션 신경망 가속에 사용되는 가속 방법에 따라 성능 차이가 매우 크며, 적합한 가속 방법을 적용할 때, 17.68 배까지 실행 시간 차이가 남을 보였다.

ACKNOWLEDGMENT

본 연구는 IDEC 에서 EDA Tool 를 지원받아 수행하였습니다.

참 고 문 헌

[1] Szegedy, Christian, et al. "Going deeper with convolutions," IEEE CVPR, 2015.

[2] Simonyan, et al., "Very deep convolutional networks for large-scale image recognition," arXiv, 2014.

[3] Chen, et al., "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE JSSC, 2017.

[4] Jia, Zhe, et al. "Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking," arXiv, 2018.

[5] Han, Song, et al., "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv, 2015.

[6] Iandola, Forrest N., et al. "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," arXiv, 2016.

[7] https://pjreddie.com/darknet/tiny-darknet/, 2016.

[8] https://pjreddie.com/darknet/, 2013.

[9] Xianyi, Zhang, et al., "OpenBLAS," http://xianyi.
github. io/OpenBLAS, 2014.