(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2024년11월26일
(11) 등록번호 10-2733737
(24) 등록일자 2024년11월20일

(51) 국제특허분류(Int. Cl.)
G06F 7/544 (2017.01) *G06F 17/15* (2006.01)
G06F 17/16 (2006.01) *G06N 3/063* (2023.01)
G06N 3/08 (2023.01)
(52) CPC특허분류
G06F 7/5443 (2013.01)
G06F 17/153 (2013.01)
(21) 출원번호 10-2023-0062180
(22) 출원일자 2023년05월15일
심사청구일자 2023년05월15일
(65) 공개번호 10-2024-0001028
(43) 공개일자 2024년01월03일
(30) 우선권주장
1020220077387 2022년06월24일 대한민국(KR)
(56) 선행기술조사문헌
KR1020070085755 A
KR1020050089067 A
KR1020190062225 A
KR1020210059659 A

(73) 특허권자
서울대학교산학협력단
서울특별시 관악구 관악로 1 (신림동)
(72) 발명자
이경한
서울특별시 관악구 관악로 1, 301동 1006호 (신림동, 서울대학교)
빈경민
서울특별시 관악구 관악로 1, 301동 1006호 (신림동, 서울대학교)
박종석
서울특별시 관악구 관악로 1, 301동 1006호 (신림동, 서울대학교)
(74) 대리인
특허법인엠에이피에스

전체 청구항 수 : 총 6 항 심사관 : 지정훈

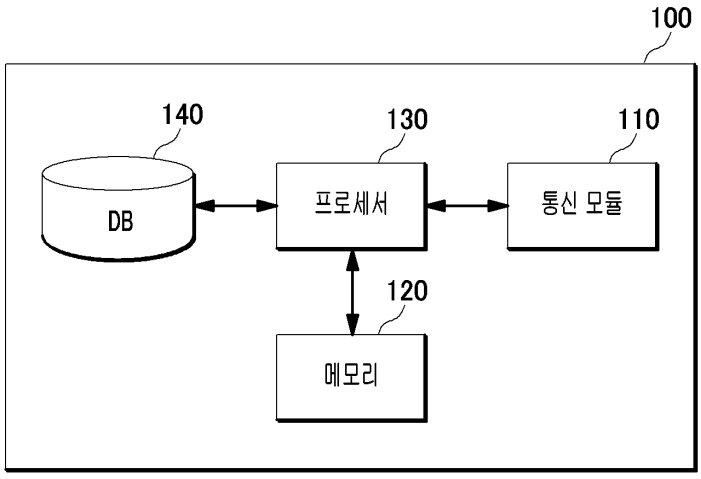
(54) 발명의 명칭 **모바일 환경을 위한 저지연 합성곱 연산 장치 및 방법**

(57) 요 약

본 발명의 일 실시예에 따른 저지연 합성곱 연산 장치는 행렬 연산 프로그램이 저장된 메모리; 및 메모리에 저장된 프로그램을 실행하는 프로세서를 포함하며, 행렬 연산 프로그램은, 입력 매트릭스와 필터 매트릭스의 각 요소별 행렬 연산에 따라 출력 매트릭스를 생성하는 행렬 연산 유닛을 포함하며, 행렬 연산 유닛은 스트라이드

(뒷면에 계속)

대 표 도 - 도3



(stride) 및 딜레이션(dilation)을 포함한 컨볼루션 파라미터를 고려하여 입력 매트릭스의 행과 필터 매트릭스의 열을 매칭하는 (a) 단계, 필터 텐서의 너비 차원에 대하여 깊이 방향으로 이동하는 필터 블록과 가로 방향으로 이동하는 입력 블록을 연산하여 출력 블록을 생성하는 (b) 단계, 및 (a) 단계와 (b) 단계를 반복하여 산출된 모든 출력 블록을 하나의 행렬로 합산하여 출력 매트릭스를 생성하는 (c) 단계를 포함하며, 입력 매트릭스는 너비 차원과 채널 차원이 각각 가로 방향과 세로 방향으로 나열된 입력 블록을 포함하는 것이고, 필터 매트릭스는 채널 차원이 가로 방향 및 세로 방향으로 나열되며, 너비 차원이 깊이 방향으로 나열된 필터 블록을 포함한다.

(52) CPC특허분류
G06F 17/16 (2013.01)
G06N 3/063 (2013.01)
G06N 3/08 (2023.01)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711152944
과제번호 IITP-2021-0-02048
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 대학ICT연구센터육성지원사업
연구과제명 UAM (Urban Air Mobility) 고밀도 운항을 위한 URAN (Ultra Reliable Aerial Network) 설계 및 원천기술 개발
기 여 율 1/2
과제수행기관명 서울대학교 산학협력단
연구기간 2021.07.01 ~ 2028.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1711152891
과제번호 2021-0-02094-002
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신융합국제공동연구
연구과제명 6G 네트워크 아키텍처 및 핵심 요소기술 국제 협력 연구
기 여 율 1/2
과제수행기관명 한국전자통신연구원
연구기간 2021.07.01 ~ 2024.06.30

명 세 서

청구범위

청구항 1

저지연 합성곱 연산 장치에 있어서,

행렬 연산 프로그램이 저장된 메모리; 및

상기 메모리에 저장된 프로그램을 실행하는 프로세서를 포함하며,

상기 행렬 연산 프로그램은,

입력 매트릭스와 필터 매트릭스의 각 요소별 행렬 연산에 따라 출력 매트릭스를 생성하는 행렬 연산 유닛을 포함하며,

상기 행렬 연산 유닛은 스트라이드(stride) 및 딜레이션(dilation)을 포함한 컨볼루션 파라미터를 고려하여 상기 입력 매트릭스의 행과 상기 필터 매트릭스의 열을 매칭하는 (a) 단계, 필터 텐서의 너비 차원에 대하여 깊이 방향으로 이동하는 필터 블록과 가로 방향으로 이동하는 입력 블록을 연산하여 출력 블록을 생성하는 (b) 단계, 및 상기 (a) 단계와 상기 (b) 단계를 반복하여 산출된 모든 출력 블록을 하나의 행렬로 합산하여 상기 출력 매트릭스를 생성하는 (c) 단계를 포함하며,

상기 입력 매트릭스는 너비 차원과 채널 차원이 각각 가로 방향과 세로 방향으로 나열된 상기 입력 블록을 포함하는 것이고,

상기 필터 매트릭스는 채널 차원이 가로 방향 및 세로 방향으로 나열되며, 너비 차원이 깊이 방향으로 나열된 상기 필터 블록을 포함하는 것인, 저지연 합성곱 연산 장치.

청구항 2

제1항에 있어서,

상기 행렬 연산 유닛은

대상 계산에 대한 합성곱 연산 차원을 GEMM 연산 차원에 매핑하고, 상기 입력 매트릭스 및 필터 매트릭스를 소정 크기의 복수의 블록으로 타일링하는 것인, 저지연 합성곱 연산 장치.

청구항 3

제1항에 있어서,

상기 행렬 연산 유닛은

수식1에 의해, 상기 입력 블록의 가로 방향 위치와 상기 필터 블록의 깊이 방향 위치를 기초로 상기 출력 블록을 산출하는 것인, 저지연 합성곱 연산 장치.

<수식1>

$Wo+Wf-1$

여기서, Wo는 출력 너비 차원, Wf는 필터 너비 차원이다.

청구항 4

저지연 합성곱 연산 장치에 의해 수행되는 저지연 합성곱 연산 방법에 있어서,

행렬 연산 유닛이 입력 매트릭스와 필터 매트릭스의 각 요소별 행렬 연산에 따라 출력 매트릭스를 생성하는 단계를 포함하되,

상기 생성 단계는

(a) 상기 행렬 연산 유닛은 스트라이드(stride) 및 딜레이션(dilation)을 포함한 컨볼루션 파라미터를 고려하여 상기 입력 매트릭스의 행과 상기 필터 매트릭스의 열을 매칭하는 단계,

(b) 필터 텐서의 너비 차원에 대하여 깊이 방향으로 이동하는 필터 블록과 가로 방향으로 이동하는 입력 블록을 연산하여 출력 블록을 생성하는 단계, 및

(c) 상기 (a) 단계와 상기 (a) 단계를 반복하여 산출된 모든 출력 블록을 하나의 행렬로 합산하여 상기 출력 매트릭스를 생성하는 단계를 포함하며,

상기 입력 매트릭스는 너비 차원과 채널 차원이 각각 가로 방향과 세로 방향으로 나열된 상기 입력 블록을 포함하는 것이고,

상기 필터 매트릭스는 채널 차원이 가로 방향 및 세로 방향으로 나열되며, 너비 차원이 깊이 방향으로 나열된 상기 필터 블록을 포함하는 것인, 저지연 합성곱 연산 방법.

청구항 5

제4항에 있어서,

상기 (a) 단계 전에

상기 행렬 연산 유닛은 대상 계산에 대한 합성곱 연산 차원을 GEMM 연산 차원에 매핑하고, 상기 입력 매트릭스 및 필터 매트릭스를 소정 크기의 복수의 블록으로 타일링하는 단계를 더 포함하는 것인, 저지연 합성곱 연산 방법.

청구항 6

제4항에 있어서,

상기 행렬 연산 유닛은

수식1에 의해, 상기 입력 블록의 가로 방향 위치와 상기 필터 블록의 깊이 방향 위치를 기초로 상기 출력 블록을 산출하는 것인, 저지연 합성곱 연산 방법.

<수식1>

Wo+Wf-1

여기서, Wo는 출력 너비 차원, Wf는 필터 너비 차원이다.

발명의 설명

기 술 분 야

[0001] 본 발명은 모바일 환경을 위한 저지연 합성곱 연산 장치 및 방법에 관한 것이다.

배 경 기 술

[0002] 현재 사물 인식, 이미지 처리와 같은 많은 인공 신경망 기반 서비스들은 합성곱(Convolution) 연산을 그 핵심 구성 요소로 사용하고 있다. 대부분의 기존 인공 신경망 구현 솔루션들은 합성곱 연산의 처리를 위해 합성곱 연산을 행렬곱(GEMM, General Matrix Multiplication) 연산으로 변환하여 처리하는 기법을 사용하고 있으며, 이는 모바일 환경에서도 마찬가지이다.

[0003] 그러나 합성곱 연산을 GEMM 연산으로 변환하는 과정은 데이터의 복제 및 이동 오버헤드를 발생시키고, 데이터

재사용률을 낮춰 연산 성능의 하락을 야기한다. 기존 서버나 데스크탑 환경에서는 다량의 데이터를 기초로 네트워크를 훈련하거나, 신경망 서비스에 대한 다양한 요청을 동시에 처리하는 경우가 많다. 이와 같은 환경에서는 한 번의 인공신경망 연산을 통해 다량의 데이터를 한 번에 취합하여 처리하는 전략을 취한다. 전체 연산 중 데이터 복제 및 이동 오버헤드가 차지하는 시간 비중을 낮추고, 다량의 데이터에 대해 같은 연산을 반복하여 데이터 재사용률을 높인다.

[0004] 하지만 모바일 환경에서는 서버나 데스크탑 환경과 다르게, 대다수가 사용자 개인의 즉각적인 요청에 따라 단일한 데이터에 대해 신경망 서비스를 제공한다. 즉 다량의 데이터를 한 번에 취합하여 처리하는 전략이 불가능하다. 따라서, 기존 GEMM 방식에서 발생하는 데이터 복제 및 이동 오버헤드와 데이터 재사용률이 낮은 문제로 인한 성능 하락이 그대로 드러나게 된다.

[0005] 또한 모바일 환경에서 제공되는 인공 신경망 서비스는 개개인의 요구에 대한 신경망 결과를 최대한 빠른 속도로 제공함으로써 사용자 만족도를 극대화하는 것이 중요하다. 이를 위해서는 적은 양의 데이터라도 사용자 요청부터 결과 제공까지의 시간을 최소화하는 것이 필수적이다.

[0006] 따라서 기존 GEMM 방식은 다량의 데이터를 한 번에 처리하여 데이터당 평균 연산 속도(throughput)는 빨라지지만, 각 데이터가 처리되어 결과가 나오기까지의 지연시간(latency)이 늘어난다. 즉 모바일 환경에서는 사용자 요청으로부터 결과를 최대한 빠르게 제공하기 위해 기존의 GEMM 방식은 적절하지 않다.

[0007] 이와 관련하여 도 1은 일반적인 GEMM 연산에서 N 차원을 반복 진행(iterate)하는 것을 도식화한 것이다. N차원의 반복 진행(N Iteration)은 행렬A의 데이터를 재사용한다. 도 2는 일반적인 2D CNN 연산에서 출력 너비(Wout) 차원을 반복 진행하는 것을 도식화한 것이다. 출력 너비 차원(Wout)의 반복 진행(Wout Iteration)은 필터 텐서의 데이터를 재사용한다.

[0008] 예시적으로 도 1에 도시된 것처럼 행렬곱(GEMM) 연산은 3개의 행렬 A, B, C에 대해 3개의 M, N, K의 연산 차원을 중첩 루프(nested loop) 형태로 반복 진행(iterate)하며 연산한다. 이때 M, N, K의 각 연산 차원에 대한 루프 진행(loop iteration)은 다음 연산에서 각각의 행렬 B, A, C에 해당하는 데이터를 재사용 한다. 반면 2D CNN 연산은 4차원 텐서인 입력(Input), 필터(Filter), 출력(Output)에 대해 총 7개의 차원을 중첩 루프 형태로 반복하여 연산한다. 이때 연산 차원은 배치(Batch), 입력 채널(Input channels(Cin)), 출력 채널(Output Channels(Cout)), 출력 높이(Output Height(Hout)), 출력 너비(Output Width(Wout)), 필터 높이(Filter Height(Hfil)), 필터 너비(Filter Width(Wfil))이다.

[0009] 기존의 CNN을 GEMM으로 변환하는 방식은 주로 입력, 필터, 출력 텐서를 각각 행렬 B, A, C에 매핑하고, 7개의 CNN 연산 차원들을 3개의 GEMM 연산 차원에 매핑한다. 이 때, 필터의 공간(spatial) 차원(Hfil, Wfil)을 제외한 CNN의 연산 차원들은 직접적으로 GEMM 연산 차원에 매핑될 수 있다. 하지만, 필터의 공간 차원은 GEMM의 연산 차원 M, N, K와 다른 데이터 엑세스 및 재사용 패턴을 가지므로 직접적인 매핑이 불가능하다. 이를 해결하기 위해 Im2col 등의 추가 처리가 필요하며, 이는 데이터 복제 및 재배열 등의 오버헤드로 인한 연산 시간이 증가한다.

[0010] 따라서 필터의 공간 차원이 GEMM의 연산 차원으로 표현이 불가능하다는 점에서, 기존의 CNN을 GEMM으로 변환하는 방식은 비효율적이다.

발명의 내용

해결하려는 과제

[0011] 본 발명은 전술한 문제점을 해결하기 위한 것으로, 기존 GEMM방식에 비해 모바일 환경에서 적은 양의 데이터에 대해 빠른 처리 성능을 제공하는 모바일 환경을 위한 저지연 합성곱 연산 장치 및 방법을 제시하고자 한다.

[0012] 다만, 본 실시예가 이루고자 하는 기술적 과제는 상기된 바와 같은 기술적 과제로 한정되지 않으며, 또 다른 기술적 과제들이 존재할 수 있다.

과제의 해결 수단

[0013] 상술한 기술적 과제를 해결하기 위한 기술적 수단으로서, 본 발명의 일 실시예에 따른 저지연 합성곱 연산 장치는 행렬 연산 프로그램이 저장된 메모리; 및 메모리에 저장된 프로그램을 실행하는 프로세서를 포함하며, 행렬 연산 프로그램은, 입력 매트릭스와 필터 매트릭스의 각 요소별 행렬 연산에 따라 출력 매트릭스를 생성하는 행

렬 연산 유닛을 포함하며, 행렬 연산 유닛은 스트라이드(stride) 및 딜레이션(dilation)을 포함한 컨볼루션 파라미터를 고려하여 입력 매트릭스의 행과 필터 매트릭스의 열을 매칭하는 (a) 단계, 필터 텐서의 너비 차원에 대하여 깊이 방향으로 이동하는 필터 블록과 가로 방향으로 이동하는 입력 블록을 연산하여 출력 블록을 생성하는 (b) 단계, 및 (a) 단계와 (b) 단계를 반복하여 산출된 모든 출력 블록을 하나의 행렬로 합산하여 출력 매트릭스를 생성하는 (c) 단계를 포함하며, 입력 매트릭스는 너비 차원과 채널 차원이 각각 가로 방향과 세로 방향으로 나열된 입력 블록을 포함하는 것이고, 필터 매트릭스는 채널 차원이 가로 방향 및 세로 방향으로 나열되며, 너비 차원이 깊이 방향으로 나열된 필터 블록을 포함한다.

[0014] 본 발명의 다른 실시예에 따른 저지연 합성곱 연산 장치에 의해 수행되는 저지연 합성곱 연산 방법은 행렬 연산 유닛이 입력 매트릭스와 필터 매트릭스의 각 요소별 행렬 연산에 따라 출력 매트릭스를 생성하는 단계를 포함하되, 생성 단계는 (a) 행렬 연산 유닛은 스트라이드(stride) 및 딜레이션(dilation)을 포함한 컨볼루션 파라미터를 고려하여 입력 매트릭스의 행과 필터 매트릭스의 열을 매칭하는 단계, (b) 필터 텐서의 너비 차원에 대하여 깊이 방향으로 이동하는 필터 블록과 가로 방향으로 이동하는 입력 블록을 연산하여 출력 블록을 생성하는 단계, 및 (c) (a) 단계와 (a) 단계를 반복하여 산출된 모든 출력 블록을 하나의 행렬로 합산하여 출력 매트릭스를 생성하는 단계를 포함하며, 입력 매트릭스는 너비 차원과 채널 차원이 각각 가로 방향과 세로 방향으로 나열된 입력 블록을 포함하는 것이고, 필터 매트릭스는 채널 차원이 가로 방향 및 세로 방향으로 나열되며, 너비 차원이 깊이 방향으로 나열된 필터 블록을 포함한다.

발명의 효과

[0015] 전술한 본원의 과제 해결 수단 중 어느 하나에 의하면, 기존 방식에 비해 인공 신경망 처리 시 메모리 사용과 연산 지연 시간(latency)을 줄일 수 있다. 이는 모바일 환경에서 자원이 한정된 메모리 자원, 하드웨어 성능, 배터리 등을 이용하는 인공 신경망 처리를 더 효율적으로 가능케 한다.

[0016] 또한 메모리 사용이 감소하면, 처리 성능 면에서 이점을 제공한다. 뿐만 아니라, CPU, GPU, DSP등 다양한 연산 자원이 공통으로 사용되는 메인 메모리와 버스를 포함한 모바일 시스템에서 메모리 엑세스에 따른 병목 현상이 감소된다. 이로 인해 사용자는 원활한 인공 신경망 서비스를 제공받을 수 있다.

[0017] 더불어 추가적인 오버헤드의 제거, 연산 효율성 증가에 따라 지연 시간이 감소된다. 이는 인공 신경망 가동에 필요한 에너지의 양을 감소시키고, 한정된 배터리 용량으로 작동하는 모바일 시스템이 인공 신경망 기반 서비스에 좀 더 쉽게 활용할 수 있도록 한다.

도면의 간단한 설명

[0018] 도 1은 일반적인 GEMM 연산에서 N 차원을 반복 진행(iterate)하는 것을 도식화한 것이다.

도 2는 일반적인 2D CNN 연산에서 출력 너비(Wout) 차원을 반복 진행하는 것을 도식화한 것이다.

도 3은 본 발명의 일 실시예에 따른 모바일 환경을 위한 저지연 합성곱 연산 장치의 구조를 설명하는 도면이다.

도 4는 본 발명의 일 실시예에 따른 행렬 연산 유닛의 구조를를 설명하는 도면이다.

도 5는 도 4의 행렬 연산 유닛의 연산 방법을 설명하는 순서도이다.

도 6은 기존의 CNN을 GEMM으로 변환하는 방식 중 Im2Col과 GEMM이 조합된 구조의 연산 방법을 설명하는 도면이다.

도 7은 도 4의 행렬 연산 유닛의 연산 방법을 설명하기 위한 도면이다.

도 8은 본 발명의 일 실시예에 따른 저지연 합성곱 신경망 연산 방법의 알고리즘을 도시한 것이다.

도 9는 본 발명의 일 실시예에 따른 저지연 합성곱 연산 방법과 기존의 방식을 비교한 결과를 도시한 것이다.

발명을 실시하기 위한 구체적인 내용

[0019] 아래에서는 첨부한 도면을 참조하여 본원이 속하는 기술 분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 본원의 실시예를 상세히 설명한다. 그러나 본원은 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 실시예에 한정되지 않는다. 그리고 도면에서 본원을 명확하게 설명하기 위해서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 유사한 부분에 대해서는 유사한 도면 부호를 붙였다.

[0020] 명세서 전체에서, 어떤 부분이 다른 부분과 "연결"되어 있다고 할 때, 이는 "직접적으로 연결"되어 있는 경우뿐 아니라, 그 중간에 다른 소자를 사이에 두고 "전기적으로 연결"되어 있는 경우도 포함한다. 또한 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다.

[0021] 본 명세서에 있어서 '부(部)'란, 하드웨어에 의해 실현되는 유닛(unit), 소프트웨어에 의해 실현되는 유닛, 양방을 이용하여 실현되는 유닛을 포함한다. 또한, 1 개의 유닛이 2 개 이상의 하드웨어를 이용하여 실현되어도 되고, 2 개 이상의 유닛이 1 개의 하드웨어에 의해 실현되어도 된다. 한편, '~부'는 소프트웨어 또는 하드웨어에 한정되는 의미는 아니며, '~부'는 어드레싱할 수 있는 저장 매체에 있도록 구성될 수도 있고 하나 또는 그 이상의 프로세서들을 재생시키도록 구성될 수도 있다. 따라서, 일 예로서 '~부'는 소프트웨어 구성요소들, 객체지향 소프트웨어 구성요소들, 클래스 구성요소들 및 태스크 구성요소들과 같은 구성요소들과, 프로세스들, 함수들, 속성들, 프로시저들, 서브루틴들, 프로그램 코드의 세그먼트들, 드라이버들, 펌웨어, 마이크로코드, 회로, 데이터, 데이터베이스, 데이터 구조들, 테이블들, 어레이들 및 변수들을 포함한다. 구성요소들과 '~부'들 안에서 제공되는 기능은 더 작은 수의 구성요소들 및 '~부'들로 결합되거나 추가적인 구성요소들과 '~부'들로 더 분리될 수 있다. 뿐만 아니라, 구성요소들 및 '~부'들은 디바이스 내의 하나 또는 그 이상의 CPU들을 재생시키도록 구현될 수도 있다.

[0022] 네트워크는 단말들 및 서버들과 같은 각각의 노드 상호 간에 정보 교환이 가능한 연결 구조를 의미하는 것으로, 근거리 통신망(LAN: Local Area Network), 광역 통신망(WAN: Wide Area Network), 인터넷 (WWW: World Wide Web), 유무선 데이터 통신망, 전화망, 유무선 텔레비전 통신망 등을 포함한다. 무선 데이터 통신망의 일례에는 3G, 4G, 5G, 3GPP(3rd Generation Partnership Project), LTE(Long Term Evolution), WIMAX(World Interoperability for Microwave Access), 와이파이(Wi-Fi), 블루투스 통신, 적외선 통신, 초음파 통신, 가시광 통신(VLC: Visible Light Communication), 라이파이(LiFi) 등이 포함되나 이에 한정되지는 않는다.

[0023] 이하, 첨부된 도면을 참고하여 본 발명의 일 실시예를 상세히 설명하기로 한다.

[0024] 먼저 기존 모바일 환경에서는 합성곱 연산을 처리하기 위해 행렬곱(GEMM) 연산이 주로 사용된다. 하지만, 도 1 및 도 2를 참조하여 상술한 바와 같이 GEMM 연산은 데이터 복제 및 재배치의 오버헤드로 인해 처리 속도가 느리고, 합성곱 연산과 비교하여 데이터 재사용률이 떨어지는 등의 비효율성 문제가 있다. 특히, 이와 같은 비효율성 문제는 모바일 환경의 인공지능 신경망에서 더욱 큰 성능 하락을 야기한다.

[0025] 전술한 문제점을 해결하기 위해, 본 발명은 모바일 환경의 특성을 고려하여 인공 신경망 처리 시의 핵심 구성요소인 합성곱(Convolution) 신경망 처리를 가속하기 위한 저지연 합성곱 연산 장치 및 방법에 관한 것이다.

[0026] 도 3은 본 발명의 일 실시예에 따른 모바일 환경을 위한 저지연 합성곱 연산 장치의 구조를 설명하는 도면이다.

[0027] 도 3을 참조하면 합성곱 연산 장치(100)는 통신 모듈(110), 메모리(120), 프로세서(130) 및 데이터베이스(140)를 포함할 수 있다.

[0028] 통신 모듈(110)은 입력 데이터를 수신하여 프로세서(130)로 전송할 수 있다. 여기서 통신 모듈(110)은 다른 네트워크 장치와 유무선 연결을 통해 제어 신호 또는 데이터 신호와 같은 신호를 송수신하기 위해 필요한 하드웨어 및 소프트웨어를 포함하는 장치일 수 있다.

[0029] 메모리(120)는 행렬 연산 프로그램이 기록된 것일 수 있다. 행렬 연산 프로그램은 입력 매트릭스와 필터 매트릭스의 각 요소별 행렬 연산에 따라 출력 매트릭스를 생성하는 행렬 연산 유닛을 포함한다. 이때 행렬 연산 유닛은 스트라이드(stride) 및 딜레이션(dilation)을 포함한 컨볼루션 파라미터를 고려하여 입력 매트릭스의 행과 필터 매트릭스의 열을 매칭하는 단계(S110), 필터 텐서의 너비 차원에 대하여 깊이 방향으로 이동하는 필터 블록과 가로 방향으로 이동하는 입력 블록을 연산하여 출력 블록을 생성하는 단계(S120), 및 S110 단계와 S120 단계를 반복하여 산출된 모든 출력 블록을 하나의 행렬로 합산하여 출력 매트릭스를 생성하는 단계(S130)를 포함한다.

[0030] 메모리(120)는 저장된 정보를 유지하기 위하여 전력이 필요한 휘발성 저장장치 외에 자기 저장 매체(magnetic storage media) 또는 플래시 저장 매체(flash storage media)를 포함할 수 있으나, 본 발명의 범위가 이에 한정되는 것은 아니다.

[0031] 메모리(120)는 프로세서(130)의 처리 및 제어를 위한 운영체제 등 별도의 프로그램이 저장될 수도 있고, 입력되거나 출력되는 데이터들의 임시 저장을 위한 기능을 수행할 수도 있다.

[0032] 프로세서(130)는 메모리(120)에 저장된 행렬 연산 프로그램(이하, 프로그램)을 실행하고, 프로그램의 실행에 따라 합성곱 연산 장치(100)의 하드웨어를 제어하는 기능을 제공한다. 즉, 프로세서(130)는 프로그램을 실행함에 따라 필요한 파일 시스템, 메모리 할당, 네트워크, 기본 라이브러리, 타이머, 디바이스 제어(디스플레이, 미디어, 입력장치, 3D 등), 기타 유틸리티 등의 하드웨어 제어 기능을 수행할 수 있다.

[0033] 또한, 프로그램의 실행에 따른 행렬 연산 과정의 구체적인 각 단계에 대해서는 도 4 내지 도 7을 참조하여 후술하도록 한다.

[0034] 프로세서(130)는 데이터를 처리할 수 있는 모든 종류의 장치를 포함할 수 있다. 예를 들어 프로그램 내에 포함된 코드 또는 명령으로 표현된 기능을 수행하기 위해 물리적으로 구조화된 회로를 갖는, 하드웨어에 내장된 데이터 처리 장치를 의미할 수 있다. 이와 같이 하드웨어에 내장된 데이터 처리 장치의 일 예로써, 마이크로프로세서(microprocessor), 중앙처리장치(central processing unit: CPU), 프로세서 코어(processor core), 멀티프로세서(multiprocessor), ASIC(application-specific integrated circuit), FPGA(field programmable gate array) 등의 처리 장치를 망라할 수 있으나, 본 발명의 범위가 이에 한정되는 것은 아니다.

[0035] 데이터베이스(140)는 프로세서(130)의 제어에 따라, 합성곱 연산 장치(100)에 필요한 데이터를 저장 또는 제공한다. 이러한 데이터베이스(140)는 메모리(120)와는 별도의 구성 요소로서 포함되거나, 또는 메모리(120)의 일부 영역에 구축될 수도 있다.

[0036] 도 4는 본 발명의 일 실시예에 따른 행렬 연산 유닛의 구조를를 설명하는 도면이고, 도 5는 도 4의 행렬 연산 유닛의 연산 방법을 설명하는 순서도이다.

[0037] 도 4를 참조하면 프로세서(130)는 입력 매트릭스(200)와 필터 매트릭스(300)의 각 요소별 행렬 연산에 따라 출력 매트릭스(400)를 생성하는 행렬 연산 유닛(150)을 포함한다.

[0038] 도 5를 참조하면 행렬 연산 유닛(150)은 스트라이드(stride) 및 딜레이션(dilation)을 포함한 컨볼루션 파라미터를 고려하여 입력 매트릭스(200)의 행과 필터 매트릭스(300)의 열을 매칭하는 단계(S110), 필터 텐서의 너비 차원에 대하여 깊이 방향으로 이동하는 필터 블록과 가로 방향으로 이동하는 입력 블록을 연산하여 출력 블록을 생성하는 단계(S120) 및 S110단계와 S120단계를 반복하여 산출된 모든 출력 블록을 하나의 행렬로 합산하여 출력 매트릭스(400)를 생성하는 단계(S130)를 포함한다. 이때 컨볼루션 파라미터는 스트라이드 및 딜레이션을 포함하나 이에 한정되는 것은 아니다.

[0039] S110단계 이전에 행렬 연산 유닛(150)은 대상 계산에 대한 합성곱 연산 차원을 GEMM 연산 차원에 매핑하고, 입력 매트릭스(200) 및 필터 매트릭스(300)를 소정 크기의 복수의 블록으로 타일링할 수 있다. 이와 같은 타일링 과정은 기존의 GEMM 연산 방법을 이용하며, 이에 대한 상세한 설명은 생략한다. 여기서, 블록은 각 매트릭스를 소정의 크기로 분할한 부분 행렬을 의미하며, 부분 행렬은 서로 인접한 요소들을 포함한다.

[0040] 여기서 입력 매트릭스(200)는 너비 차원과 채널 차원이 각각 가로 방향과 세로 방향으로 나열된 블록을 포함한다. 또한 필터 매트릭스(300)는 채널 차원이 가로 방향 및 세로 방향으로 나열되며, 너비 차원이 깊이 방향으로 나열된 필터 블록을 포함한다. 출력 매트릭스(300)는 너비 차원과 채널 차원이 각각 가로 방향과 세로 방향으로 나열된 블록을 포함한다.

[0041] 따라서, 본 발명은 기존 GEMM 방식에서 발생되는 비효율성을 제거한 새로운 알고리즘을 제시한다. 이를 통하여 모바일 환경에서의 인공 신경망 처리 시간을 획기적으로 단축 가능하다.

[0042] 구체적으로 본 발명은 기존의 CNN 연산을 GEMM 연산으로 변환하는 방식에서 발생하는 문제들의 원인이 GEMM 연산이 CNN 연산의 모든 차원을 표현할 수 없다는 한계점에 착안하여 새로운 알고리즘 제시한 것이다. 즉 본 발명의 일 실시예에 따른 저지연 합성곱 신경망 연산 장치는 기존 GEMM 알고리즘에서 표현이 불가능한 CNN 연산의 필터 공간 차원과 동일한 특징을 갖는 연산 차원을 추가하여 기존의 방식을 확장한 것이다.

[0043] 여기서, 추가된 연산 차원은 CNN 연산의 필터(커널)의 공간 차원(Spatial dimension(Width, Height))을 표현하는 차원이다. 해당 차원은 입력 (피처맵) 데이터와 출력 데이터를 재사용하고, 필터 데이터를 새로이 액세스 한다는 특징이 있다. 한편, 본 발명은 CNN 연산의 차원 중 GEMM 알고리즘으로 처리가 가능한 부분은 기존 GEMM 알고리즘을 그대로 사용하여 연산을 처리할 수 있다. 하지만, 본 발명은 새로이 추가된 연산 차원을 이용하여, 기존 GEMM 알고리즘으로 처리가 불가능한 부분(필터의 공간 차원)을 처리할 수 있다. 즉, 본 발명은 행렬 연산 유닛(150)을 통해 기존 GEMM 알고리즘으로 처리할 수 없었던 필터의 공간 차원을 포함한 모든 CNN 연산을 효율적으로 처리할 수 있다. 따라서 모바일 환경에서의 합성곱 신경망 처리를 빠르고 효율적으로 수행할 수 있다.

[0044] 도 6은 기존의 CNN을 GEMM으로 변환하는 방식 중 Im2Col과 GEMM이 조합된 구조의 연산 방법을 설명하는 도면이고, 도 7은 도 4의 행렬 연산 유닛의 연산 방법을 설명하기 위한 도면이다.

[0045] 먼저, 기존의 조합 구조(Im2Col+GEMM)에 따른 연산 수행시 발생하는 문제점을 도 6을 참조하여 설명하도록 한다.

[0046] 도 6은 CNN을 GEMM으로 변환하는 방식 중 가장 일반적으로 사용되는 기존의 조합 구조(Im2Col+GEMM)를 이용하여 CNN 연산을 수행하는 과정이다.

[0047] 기존의 조합 구조(Im2Col+GEMM)에서 필터 너비 차원(Wf)은 입력 채널 차원(Ci)을 사용하여 K 행렬 차원으로 평탄화된다. 이에 따라, 도 6(a)에 도시된 것처럼 필터 너비 차원(Wf)에서 입력 블록의 재사용은 입력 블록의 복제(duplication) 및 이동(shift)으로 표현된다. 즉, 기존의 조합 구조는 표현 불가능한 CNN의 필터 공간 차원 연산을 GEMM 연산의 틀에 맞추어 처리하기 위해 입력 데이터를 복제하고 재배열하는 과정(Im2Col 등)을 거친다. 이 과정에서 CNN의 필터 공간 차원이 제공하던 데이터 재사용 기회가 제한된다.

[0048] 그러나, 본 발명의 저지연 합성곱 연산 장치(100)는 필터 매트릭스(300)에 추가된 필터 너비 차원(Wf)에 대한 반복 진행(iteration)을 통해 기존의 조합 구조와 달리 데이터의 복제 및 재배열과 같은 추가 처리 없이 연산을 처리할 수 있다.

[0049] 이하, 도 7을 참조하여 본 발명의 행렬 연산 유닛(150)에 따른 연산 과정을 설명한다. 이때 도 7(a) 내지 7(e)는 도 6(a) 내지 6(e)에 도시된 연산 과정과 각각 수학적으로 동일한 연산 과정을 나타낸 것이다.

[0050] 도 4 및 도 7(a)를 참조하면 행렬 연산 유닛(150)은 입력 매트릭스(200)와 필터 매트릭스(300)의 각 요소별 행렬 연산에 따라 출력 매트릭스(400)를 생성할 수 있다.

[0051] 예시적으로 행렬 연산 유닛(150)은 수식1에 의해, 입력 블록의 가로 방향 위치와 필터 블록의 깊이 방향 위치를 기초로 출력 블록을 산출할 수 있다.

[0052] <수식1>

[0053] Wo+Wf-1

[0054] 여기서, Wo는 출력 너비 차원, Wf는 필터 너비 차원이다.

[0055] 행렬 연산 유닛(150)은 입력 채널 차원(Ci)에 대하여 입력 매트릭스(200)의 행과 필터 매트릭스(300)의 열을 매칭할 수 있다. 예를 들어 도 7(a)에 도시된 것처럼, 행렬 연산 유닛(150)은 입력 채널 차원이 첫 번째 열(Ci=0)에 위치한 필터 블록(301-303)과 입력 채널 차원이 첫 번째 행(Ci=0)에 위치한 입력 블록들(201-207) 간의 연산을 수행할 수 있다.

[0056] 또한, 행렬 연산 유닛(150)은 필터 너비 차원(Wf)에 대하여 깊이 방향으로 이동하는 필터 블록(301-303, 311-313)과 가로 방향으로 이동하는 입력 블록(201-207, 211-217)을 연산하여 출력 블록(401-405)을 생성할 수 있다. 예를 들어, 도 7(b) 내지 7(d)에 도시된 것처럼, 입력 채널 차원이 필터 매트릭스의 첫 번째 열과 입력 매트릭스의 첫 번째 행(Ci=0)으로 설정될 수 있다. 이때, 필터 너비 차원의 0깊이(Wf=0), 1깊이(Wf=1), 2깊이(Wf=2)로 필터 블록(301, 302, 303)이 이동하면, 입력 블록들(201-205, 202-206, 203-207)은 가로 방향으로 한 블록씩 이동하면서 연산을 수행할 수 있다. 이때 하나의 필터 블록과 연산되는 입력 블록의 열의 개수는 5개이며, 이는 출력 블록의 열의 개수로 결정될 수 있다.

[0057] 다음으로, 도 7(e)를 참조하면 행렬 연산 유닛(150)은 입력 채널 차원의Ci=1(필터 매트릭스의 두 번째 열과 입력 매트릭스의 두 번째 행)에서도, 필터 너비 차원(Wf)에 대하여 한 블록씩 이동하며 필터 블록(311-313)과 입력 블록(211-217)의 연산을 수행할 수 있다. 즉 입력 블록(201-207, 211-217)과 필터 블록(301-303, 311-313)의 요소 간 곱셈 연산이수행되며, 연산된 결과값은 모두 합산되어, 출력 블록(401-405)을 생성할 수 있다.

[0058] 이때, 도 7(a) 내지 7(e)에 도시된 연산 과정에서 불투명한 블록은 레지스터에 로드된 데이터를 나타내며, 입력에서 출력까지 도시된 화살표는 로드된 데이터에서 MAC 연산이 수행되는 것을 의미한다.

[0059] 예시적으로, 도 7(b) 내지 도 7(d)를 참조하여 입력 채널 차원이 Ci=0인 경우, 행렬 연산 유닛(150)이 필터 너비 차원(Wf)에 대하여 한 블록씩 이동하며 연산을 수행하는 과정을 설명한다.

[0060] 도 7(b)에 도시된 것처럼, 0깊이(Wf=0)에 위치한 제1 필터 블록(301)은 제1 그룹의 입력 블록들(201-205)과 연산을 수행하고, 연산된 결과값은 제1 출력 블록(401) 내지 제5 출력 블록(405)의 위치에 저장될 수 있다.

[0061] 도 7(c)에 도시된 것처럼, 1깊이(wf=1)에 위치한 제2필터 블록(302)은 제2 그룹의 입력 블록들(202-206)과 연산을 수행하고, 연산된 결과값은 제1 출력 블록(401) 내지 제5 출력 블록(405)의 위치에 저장될 수 있다.

[0062] 도 7(d)에 도시된 것처럼, 2깊이(wf=2)에 위치한 제3필터 블록(303)은 제3 그룹의 입력 블록(203-207)과 연산을 수행하고, 연산된 결과값은 제1 출력 블록(401) 내지 제5 출력 블록(405)의 위치에 저장될 수 있다.

[0063] 또한, 도 7(e)에 도시된 것처럼, 입력 채널 차원이 Ci=1인 경우에도 행렬 연산 유닛(150)은 필터 너비 차원에 대하여 한 블록씩 이동하며 연산을 수행할 수 있다. 마찬가지로, 각 필터 블록(311, 312, 313)과 이에 해당하는 그룹 별 입력 블록들(211-215, 212-216, 213-217)의 연산 결과값은 제1 출력 블록(401) 내지 제5 출력 블록(405)의 위치에 저장될 수 있다. 최종적으로 누적 저장된 연산 결과값은 합산되고, 제1 출력 블록(401) 내지 제5 출력 블록(405)이 생성될 수 있다.

[0064] 또한, 각 입력 그룹(5개의 열) 별로 일괄 처리되는 연산 결과값은 제1 출력 블록(401) 내지 제5 출력 블록(405)을 포함한 출력 그룹(5개의 열)에 한번에 저장될 수 있다. 예를 들어 출력 그룹(401-405)은 출력 너비 차원의 첫 번째 행(Wo=0)에 위치할 수 있다.

[0065] 예시적으로, 화살표(MAC연산)의 방향은 필터 너비 차원(Wf)에 대한 각 그룹 별 입력 블록의 연산 결과값이 어느 출력 블록의 위치에 저장되는지 나타낼 수 있다.

[0066] 예를 들어, 수식1에 따르면 제1그룹의 경우, Wo(0)+Wf(0)-1=-1이고, 제2 그룹의 경우, Wo(0)+Wf(1)-1=0이고, 제3그룹의 경우, Wo(0)+Wf(2)-1=1이 도출될 수 있다. 즉, 제1 그룹의 입력 블록(201-205)과 출력 그룹(401-405)을 연결하는 화살표 방향(↘)은 우측(-1)을 향할 수 있다. 제2 그룹의 입력 블록(202-205)과 출력 그룹(401-405)을 연결하는 화살표 방향(↓)은 제자리(0)를 유지할 수 있다. 제3그룹의 입력 블록(203-207)과 출력 그룹(401-405)을 연결하는 화살표 방향(↙)은 좌측(1)을 향할 수 있다.

[0067] 즉, 본 발명은 필터 매트릭스(300)의 필터 너비 차원(Wf)을 이용하여 입력 매트릭스(200)의 입력 블록의 복제 및 재배열 과정을 생략할 수 있다. 또한, CNN의 필터 공간 차원의 데이터 재사용 기회를 극대화할 수 있다.이처럼 본 발명은 기존의 조합 구조와 수학적으로 동일한 연산을 수행하면서 데이터 복제 과정을 생략할 수 있다. 즉, 같은 데이터(입력 블록)를 더 많은 연산에 재사용하며 연산을 진행할 수 있다. 이로 인해 기존의 조합 구조의 연산에 비해 더 적은 메모리를 이용하고, 데이터 엑세스 횟수를 줄여 더 빠른 연산이 가능하다.

[0068] 따라서 본 발명은 기존의 GEMM 연산이 표현할 수 없는 CNN의 필터 공간 차원을 표현한 연산 차원(필터 너비 차원(Wf))을 추가한 새로운 알고리즘이다. 즉 필터 너비 차원(Wf)을 통해 기존의 CNN을 GEMM연산에 매핑하여 처리하던 방식이 갖던 비효율성 문제를 해결하는 효과를 제공할 수 있다.

[0069] 도 8은 본 발명의 일 실시예에 따른 저지연 합성곱 신경망 연산 방법의 알고리즘을 도시한 것이다.

[0070] 본 발명은 기존 행렬곱(GEMM) 기반의 인공신경망 합성곱 연산 처리를 개선하는 방법이며, 기존 GEMM 방식의 알고리즘에 새로운 연산 차원을 추가하여, 모바일 환경의 특성에 최적화한 합성곱 연산 알고리즘을 제시하고자 한다. 이를 실증하기 위해 대부분의 모바일 시스템이 사용하고 있는 ARMv8 아키텍처의 프로세서 위에서, 도 8에 도시된 바와 같은 방식으로 알고리즘을 구현하였다. 내부 커널(Inner kernel)은 합성곱 연산을 구현하기 위한 핵심 구성 요소이며, 도 8과 같은 연산을 수행하기 위한 ARMv8-A 어셈블리어 코드이다. 이 코드는 ARM NEON, prefetching등 다양한 하드웨어 가속 기술을 활용하여 고정된 출력 너비(wo), 출력 채널(co) 값에 따라 가장 효율적인 방법으로 연산을 수행한다.

[0071] 도 8의 알고리즘에서 줄 7부터 13이 내부 커널(inner kernel)에 해당하며, 전체 연산은 타일링(tiling) 방식으로 구현하였다. 워크로드의 입력 파라미터들을 고려하여 최적의 효율을 갖는 wo, co값 조합의 inner kernel을 선택하고, 이를 반복적으로 수행하여 전체 연산을 구성한다.

[0072] 이를 효율적으로 실현하기 위해 줄 1~6은 입력 파라미터를 적절히 쪼개고, 순서를 재배열해 주어진 하드웨어의 메모리 계층 구조(hierarchy)에서 최대한의 데이터가 재사용 될 수 있도록 구성하였다. Cout 파라미터를 co뿐만 아니라 coo로 추가적으로 나누고, 레지스터 레벨에서 출력 텐서 값들이 스트리밍되고, L1 캐시에서 필터 값들이 스트리밍되고, L2/Last level 캐시에서 입력 값들이 스트리밍되며, 다양한 코어에서 공유되며 재사용 될 수 있도록 배치하였다.

[0073] 이에 추가적으로 wo, co값에 따라 데이터의 엑세스가 순차적으로 일어날 수 있도록, 입력과 출력 데이터를 N-(C/co)-H-W-co 순으로, 필터 데이터를 (K/co)-R-C-S-co 순으로 재배열하였다.

[0074] 도 9는 본 발명의 일 실시예에 따른 저지연 합성곱 연산 방법과 기존의 방식을 비교한 결과를 도시한 것이다.

[0075] 도 9를 참조하면 본 발명의 일 실시예에 따른 방식으로 구현된 mGEMM의 연산 코드를 기존 방식들과 비교해 본 결과, YoloV3-Tiny Object Detection task에 대해 다양한 모바일 디바이스에서 성능 향상이 나타났다.

[0076] 즉 본 발명은 mGEMM을 C 기반의 소프트웨어로 구현하여, ARMv8 기반의 모바일 환경에서 그 성능을 기존 방법들과 비교한 결과 기존 방법들에 비해 1.29배에서 1.58배의 성능 향상이 있음을 확인하였다.

[0077] 본 발명의 일 실시예는 컴퓨터에 의해 실행되는 프로그램 모듈과 같은 컴퓨터에 의해 실행가능한 명령어를 포함하는 기록 매체의 형태로도 구현될 수 있다. 컴퓨터 판독 가능 매체는 컴퓨터에 의해 액세스될 수 있는 임의의 가용 매체일 수 있고, 휘발성 및 비휘발성 매체, 분리형 및 비분리형 매체를 모두 포함한다. 또한, 컴퓨터 판독 가능 매체는 컴퓨터 저장 매체를 포함할 수 있다. 컴퓨터 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보의 저장을 위한 임의의 방법 또는 기술로 구현된 휘발성 및 비휘발성, 분리형 및 비분리형 매체를 모두 포함한다.

[0078] 본 발명의 방법 및 장치는 특정 실시예와 관련하여 설명되었지만, 그것들의 구성 요소 또는 동작의 일부 또는 전부는 범용 하드웨어 아키텍쳐를 갖는 컴퓨터 시스템을 사용하여 구현될 수 있다.

[0079] 전술한 본원의 설명은 예시를 위한 것이며, 본원이 속하는 기술분야의 통상의 지식을 가진 자는 본원의 기술적 사상이나 필수적인 특징을 변경하지 않고서 다른 구체적인 형태로 쉽게 변형이 가능하다는 것을 이해할 수 있을 것이다. 그러므로 이상에서 기술한 실시예들은 모든 면에서 예시적인 것이며 한정적이 아닌 것으로 이해해야만 한다. 예를 들어, 단일형으로 설명되어 있는 각 구성 요소는 분산되어 실시될 수도 있으며, 마찬가지로 분산된 것으로 설명되어 있는 구성 요소들도 결합된 형태로 실시될 수 있다.

[0080] 본원의 범위는 상기 상세한 설명보다는 후술하는 특허청구범위에 의하여 나타내어지며, 특허청구범위의 의미 및 범위 그리고 그 균등 개념으로부터 도출되는 모든 변경 또는 변형된 형태가 본원의 범위에 포함되는 것으로 해석되어야 한다.

부호의 설명

[0081] 100: 저지연 합성곱 연산 장치

110: 행렬 연산 유닛

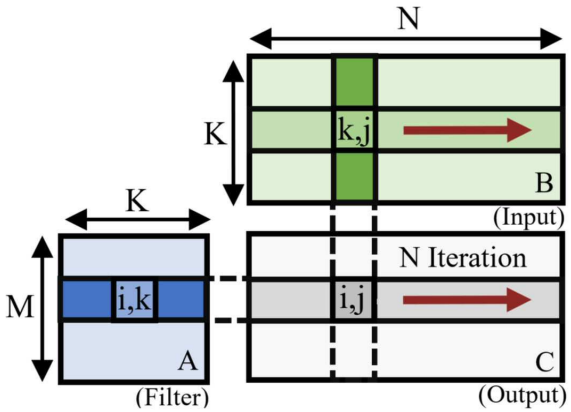
200: 입력 매트릭스

300: 필터 매트릭스

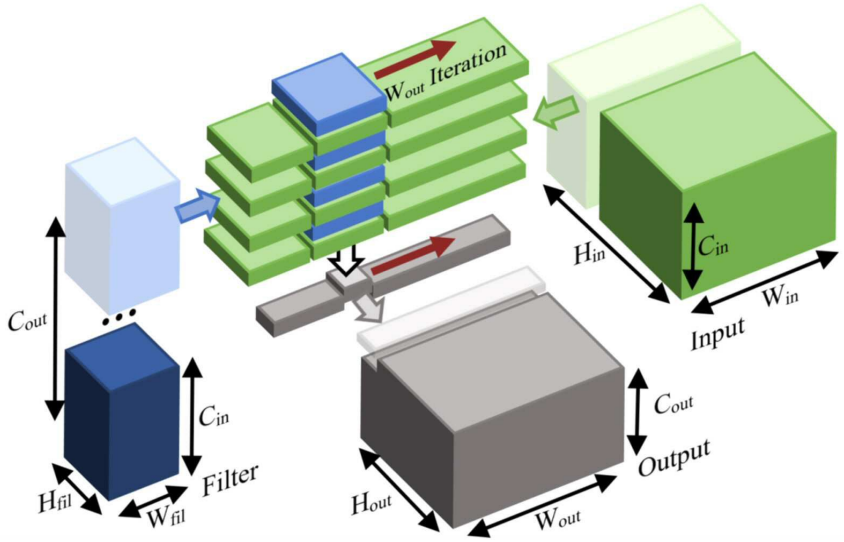
400: 출력 매트릭스

도면

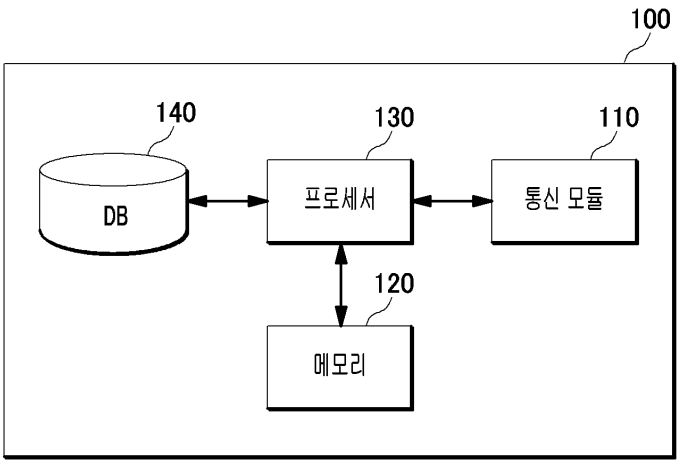
도면1



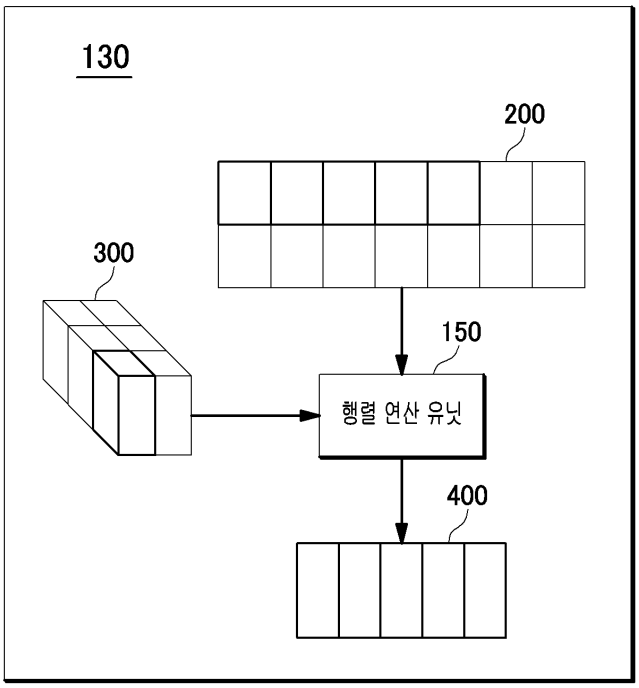
도면2


Wout Iteration
Hin
Cin
Win
Input
Cout
Cin
Filter
Hfil
Wfil
Cout
Output
Hout
Wout

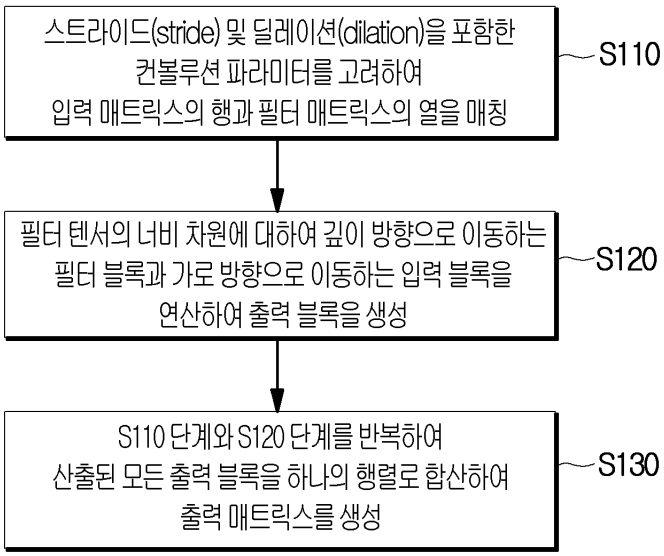
도면3


100
140
130
110
DB
프로세서
통신 모듈
120
메모리

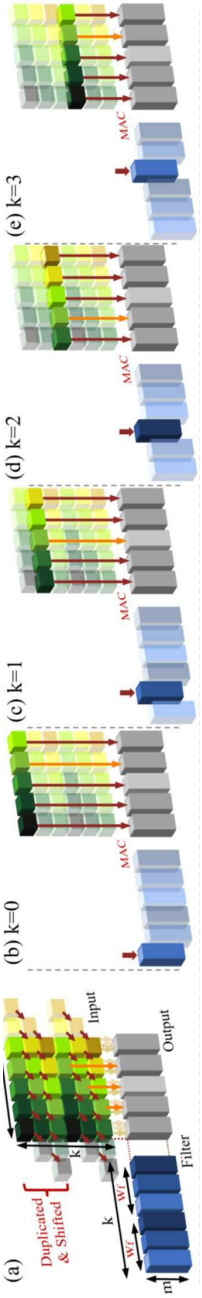
도면4


130
200
300
150
행렬 연산 유닛
400

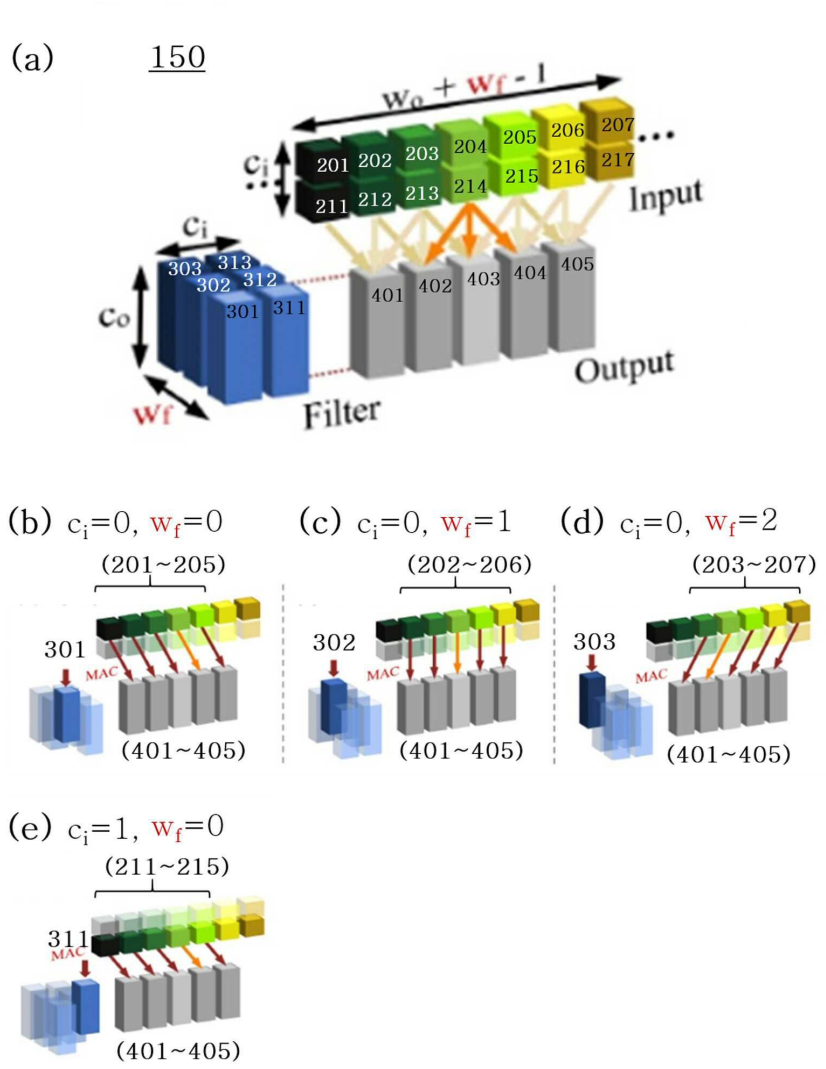
도면5


스트라이드(stride) 및 딜레이션(dilation)을 포함한
컨볼루션 파라미터를 고려하여
입력 매트릭스의 행과 필터 매트릭스의 열을 매칭
S110
필터 텐서의 너비 차원에 대하여 깊이 방향으로 이동하는
필터 블록과 가로 방향으로 이동하는 입력 블록을
연산하여 출력 블록을 생성
S120
S110 단계와 S120 단계를 반복하여
산출된 모든 출력 블록을 하나의 행렬로 합산하여
출력 매트릭스를 생성
S130

도면6


(a)
Duplicated & Shifted
Input
Output
Filter
k
k
W1
W1
m1
(b) k=0
(c) k=1
(d) k=2
(e) k=3
MAC

도면7



도면8

Algorithm mGEMM Algorithm

Input: Input tensor I, Filter tensor W, Stride s
Output: Output tensor O

for $i = 1, \ldots, B-1$ **do**
  for $j_1 = 1, \ldots, (C_{out} \div (cc_o \times c_o)) - 1$ **do**
    for $k = 1, \ldots, H_{out} - 1$ **do**
      for $l_1 = 1, \ldots, (W_{out} \div w_o) - 1$ **do**
        for $n = 1, \ldots, H_{fil} - 1$ **do**
          for $j_2 = 1, \ldots, (cc_o \div c_o) - 1$ **do**
            for $m = 1, \ldots, C_{in} - 1$ **do**
              for $o = 1, \ldots, W_{fil} - 1$ **do**
                for $l_2 = 1, \ldots, w_o - 1$ **do**
                  $l = l_1 \times w_o + l_2$
                  for $j_3 = 1, \ldots, c_o - 1$ **do**
                    $j = j_1 \times cc_o \times c_o + j_2 \times c_o + j_3$
                    $O_{i,j,k,l} += I_{i,m,(l+n),(m+o)} \times W_{j,m,n,o}$

도면9

Convolution Backend	Raspberry Pi 4			Odroid N2+			Pixel 4			Max Heap Usage (MB)
	Latency (s)	Power (W)	Energy (J)	Latency (s)	Power (W)	Energy (J)	Latency (s)	Power (W)	Energy (J)	
None	0.1926	3.23	0.62	0.0979	3.65	0.36	0.0686	1.29	0.09	111.717
Darknet Original	2.5543	4.53	11.57	1.2666	5.71	7.23	0.7309	2.54	1.86	135.494
OpenBLAS	0.9059	5.04	4.57	0.6075	5.66	3.44	0.3064	1.96	0.6	135.483
ARMNN	0.4951	4.17	2.06	0.3187	4.9	1.56	0.3674	0.77	0.28	173.366
XNNPACK	0.5488	5.03	2.76	0.2698	6.5	1.75	0.161	2.2	0.35	119.252
mGEMM	0.3838	4.53	1.74	0.2184	5.34	1.17	0.1317	1.8	0.24	111.735