(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2024년04월15일
(11) 등록번호 10-2657104
(24) 등록일자 2024년04월08일

(51) 국제특허분류(Int. Cl.)
G06F 7/544 (2017.01) *G06F 17/16* (2006.01)
G06F 9/30 (2018.01) *G06N 3/063* (2023.01)
(52) CPC특허분류
G06F 7/5443 (2013.01)
G06F 17/16 (2013.01)
(21) 출원번호 10-2022-0063051
(22) 출원일자 2022년05월23일
심사청구일자 2022년05월23일
(65) 공개번호 10-2022-0158639
(43) 공개일자 2022년12월01일
(30) 우선권주장
1020210066176 2021년05월24일 대한민국(KR)
(56) 선행기술조사문헌
KR1020180034853 A
KR1020190113007 A

(73) 특허권자
연세대학교 산학협력단
서울특별시 서대문구 연세로 50 (신촌동, 연세대학교)
(72) 발명자
송진호
서울특별시 용산구 원효로 216, 101동 1901호(신계동, 용산e편한세상)
노원우
서울특별시 강남구 삼성로51길 35, 201동 1202호 (대치동, 래미안 대치 팰리스)
(뒷면에 계속)
(74) 대리인
리앤목특허법인

전체 청구항 수 : 총 19 항 심사관 : 지정훈

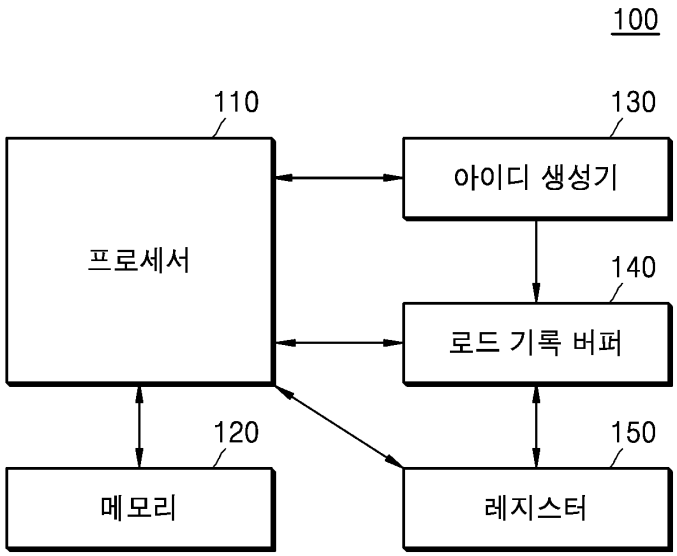
(54) 발명의 명칭 **합성곱 연산 장치, 합성곱 연산 방법 및 합성곱 연산 방법을 실행시키도록 기록매체에 저장된 컴퓨터 프로그램**

(57) 요 약

개시된 발명의 일 측면에 의하면, 입력 데이터 행렬의 성분마다 엘레멘트 아이디를 부여하고, 이러한 엘레멘트 아이디를 기초로 서로 중복되는 중복 성분들에 대하여 하나의 레지스터를 할당할 수 있어서, 일반적인 합성곱 연산 방법보다 연산 속도가 빠르고, 에너지 효율이 좋은 합성곱 연산 방법을 제공할 수 있다.

(뒷면에 계속)

대 표 도 - 도1



일 실시예에 따른 입력 데이터 행렬을 설정된 필터 행렬과 일반 행렬 곱(GEneral Matrix Multiplication) 연산하여 출력 데이터 행렬에 대응하는 특징 데이터 행렬을 생성하는 합성곱 연산 방법은, 적어도 하나의 프로세서에 의해, 상기 입력 데이터 행렬의 복수의 성분들 중 서로 중복되는 데이터를 나타내는 중복 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블을 갱신하는 단계; 및 적어도 하나의 프로세서에 의해, 레지스터 매핑 테이블에 기초하여 중복 성분들에 대해 동일한 제2 목적 레지스터 주소를 가지는 레지스터를 재사용하여 합성곱 연산을 수행하는 단계;를 포함할 수 있다.

(52) CPC특허분류
G06F 9/3012 (2013.01)
G06N 3/063 (2013.01)

(72) 발명자
김현진
서울특별시 서대문구 연희로5길 54-11, 405호(연희동)
안성우
서울특별시 서대문구 성산로17길 20, 201호(연희동)
오윤호
서울특별시 성북구 삼선교로4길 118-7(삼선동1가)
김보길
서울특별시 서대문구 신촌로1안길 16, 301호(창천동)

이 발명을 지원한 국가연구개발사업
과제고유번호 1711158682
과제번호 2021R1A2C1095162
부처명 과학기술정보통신부
과제관리(전문)기관명 한국연구재단
연구사업명 개인기초연구(과기정통부)
연구과제명 온-디바이스 개인화 추천시스템을 위한 가속기 프로세서 설계 및 최적화
기 여 율 1/2
과제수행기관명 연세대학교
연구기간 2022.03.01 ~ 2023.02.28
이 발명을 지원한 국가연구개발사업
과제고유번호 1711120090
과제번호 2020-0-01847-001
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 정보통신방송혁신인재양성(R&D)
연구과제명 비대면·인공지능 사회를 위한 반도체 시스템 융합혁신기술 개발
기 여 율 1/2
과제수행기관명 한국과학기술원
연구기간 2020.07.01 ~ 2020.12.31
공지예외적용 : 있음

명 세 서

청구범위

청구항 1

입력 데이터 행렬을 설정된 필터 행렬과 일반 행렬 곱(GEneral Matrix Multiplication) 연산하여 출력 데이터 행렬에 대응하는 특징 데이터 행렬을 생성하는 합성곱 연산 방법에 있어서,

적어도 하나의 프로세서에 의해, 상기 입력 데이터 행렬의 복수의 성분들 중 서로 중복되는 데이터를 나타내는 중복 성분들의 제1 목적 레지스터들 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블을 갱신하는 단계; 및

상기 적어도 하나의 프로세서에 의해, 상기 레지스터 매핑 테이블에 기초하여 상기 중복 성분들에 대해 상기 동일한 제2 목적 레지스터 주소를 가지는 레지스터를 재사용하여 합성곱 연산을 수행하는 단계;를 포함하는, 합성곱 연산 방법.

청구항 2

제1항에 있어서,

상기 레지스터 매핑 테이블을 갱신하는 단계는,

상기 복수의 성분들의 식별자를 생성하는 단계; 및

상기 복수의 성분들 중 동일한 식별자가 생성된 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하는 단계;를 포함하는, 합성곱 연산 방법.

청구항 3

제2항에 있어서,

상기 식별자는, 엘레멘트 아이디를 포함하고,

상기 식별자를 생성하는 단계는,

상기 복수의 성분들의 어레이 인덱스, 상기 필터 행렬의 행의 개수 및 열의 개수 및 상기 출력 데이터 행렬의 열의 개수에 기초하여 상기 복수의 성분들의 패치 아이디를 생성하는 단계; 및

상기 패치 아이디 및 상기 복수의 성분들의 오프셋에 기초하여 상기 복수의 성분들의 엘레멘트 아이디를 생성하는 단계;를 포함하며,

상기 오프셋은,

상기 패치 아이디, 상기 입력 데이터 행렬의 열의 개수 및 채널의 개수에 기초하여 결정되는 값이고,

상기 어레이 인덱스는, 상기 복수의 성분들이 단일 차원 어레이로 배열되었을 때 각 성분들의 위치를 나타내는 값인, 합성곱 연산방법.

청구항 4

제3항에 있어서,

상기 식별자는, 배치 아이디를 더 포함하고,

상기 식별자를 생성하는 단계는,

상기 복수의 성분들의 어레이 인덱스, 상기 필터 행렬의 행의 개수 및 열의 개수 및 출력 데이터 행렬의 행의 개수 및 열의 개수에 기초하여 상기 복수의 성분들의 배치 아이디를 생성하는 단계;를 더 포함하는, 합성곱 연산 방법.

청구항 5

제3항에 있어서,

상기 패치 아이디를 생성하는 단계는,

상기 복수의 성분들의 제1 패치 요소 및 제2 패치 요소를 산출하는 단계; 및

상기 제1 패치 요소에 상기 필터 행렬의 스트라이드를 곱한 값에 상기 제2 패치 요소를 더하여 상기 패치 아이디를 생성하는 단계;를 포함하고,

상기 제1 패치 요소는,

상기 복수의 성분들의 행 요소를 상기 출력 데이터 행렬의 열의 개수로 나누었을 때 출력되는 몫이고,

상기 제2 패치 요소는,

상기 복수의 성분들의 열 요소를 상기 필터 행렬의 열의 개수로 나누었을 때 출력되는 몫이며,

상기 행 요소는, 상기 어레이 인덱스를 상기 필터 행렬의 크기로 나누었을때 출력되는 몫이고,

상기 열 요소는, 상기 어레이 인덱스를 상기 필터 행렬의 크기로 나누었을때, 출력되는 나머지인, 합성곱 연산 방법.

청구항 6

제5항에 있어서,

상기 엘레멘트 아이디를 생성하는 단계는,

상기 오프셋에 상기 행 요소를 상기 출력 데이터 행렬의 열의 개수에 상기 채널의 개수 및 상기 스트라이드를 곱한 값으로 나누었을 때 출력되는 나머지 값과, 상기 열 요소를 상기 필터 행렬의 열의 개수에 상기 채널의 개수를 곱한 값으로 나누었을 때 출력되는 나머지 값을 더하여 상기 엘레멘트 아이디를 생성하고,

상기 입력 데이터 행렬의 오프셋은,

상기 패치 아이디에 상기 입력 데이터 행렬의 열의 개수 및 채널의 개수를 곱한 값인, 합성곱 연산 방법.

청구항 7

제1항에 있어서,

상기 적어도 하나의 프로세서에 의해, 원본 입력 데이터 행렬의 크기, 복수의 성분들의 개수 및 순서를 워크스페이스에 해당하는 메모리 영역으로 변경하여 상기 입력 데이터 행렬을 생성하는 단계;를 더 포함하고,

상기 입력 데이터 행렬은,

상기 원본 입력 데이터 행렬이 필터 행렬과 GEMM 연산으로 상기 특징 데이터 행렬을 출력할 수 있도록, 상기 원본 입력 데이터 행렬의 복수의 성분들이 규칙을 가지고 재조합되어 배열된 행렬인, 합성곱 연산 방법.

청구항 8

제7항에 있어서,

상기 입력 데이터 행렬을 생성하는 단계는,

상기 원본 입력 데이터 행렬을 특징 데이터 행렬의 행의 개수와 동일한 행의 개수를 가지고, 상기 필터 행렬의 크기와 동일한 열의 개수를 가지는 워크스페이스로 변환하여 상기 입력 데이터 행렬을 생성하는 단계;를 포함하는, 합성곱 연산 방법.

청구항 9

제1항에 있어서,

상기 레지스터 매핑 테이블을 갱신하는 단계는,

텐서 코어 로드 데이터를 입력받는 단계;

상기 텐서 코어 로드 데이터에 포함된 제1 목적 레지스터 주소를 가지는 성분의 식별자 및 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있는지 식별하는 단계;

상기 식별자 및 상기 제2 목적 레지스터 주소가 상기 로드 기록 버퍼에 기록되어 있지 않은 경우, 메모리에 접근하여 메모리 계층으로부터 상기 성분의 데이터를 패치하고, 상기 식별자 및 상기 패치된 데이터가 저장된 레지스터의 제2 목적 레지스터 주소를 상기 로드 기록 버퍼에 기록하고, 상기 로드 기록 버퍼에 기록된 제2 목적 레지스터 주소가 상기 제1 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하는 단계; 및

상기 식별자 및 상기 제2 목적 레지스터 주소가 상기 로드 기록 버퍼에 기록되어 있는 경우, 메모리에 접근하지 않고 상기 로드 기록 버퍼에 기록된 제2 목적 레지스터 주소가 상기 제1 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하는 단계;를 포함하는, 합성곱 연산 방법.

청구항 10

제1항 내지 제9항 중 어느 한 항의 합성곱 연산 방법을 실행시키도록 컴퓨터로 판독 가능한 기록매체에 저장된 컴퓨터 프로그램.

청구항 11

합성곱 연산 장치에 있어서,

레지스터 매핑 테이블이 저장된 메모리; 및

입력 데이터 행렬을 설정된 필터 행렬과 일반 행렬 곱(GEneral Matrix Multiplication) 연산하여 출력 데이터 행렬에 대응하는 특징 데이터 행렬을 생성하는 합성곱 연산을 수행하는 프로세서;를 포함하고,

상기 프로세서는,

상기 입력 데이터 행렬의 복수의 성분들 중 서로 중복되는 데이터를 나타내는 중복 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하고,

상기 레지스터 매핑 테이블에 기초하여 상기 중복 성분들에 대해 상기 동일한 제2 목적 레지스터 주소를 가지는 레지스터를 재사용하여 합성곱 연산을 수행하는, 합성곱 연산 장치.

청구항 12

제11항에 있어서,

상기 프로세서는,

상기 복수의 성분들의 식별자를 생성하고,

상기 복수의 성분들 중 동일한 식별자가 생성된 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하는, 합성곱 연산 장치.

청구항 13

제12항에 있어서,

상기 식별자는, 엘레멘트 아이디를 포함하고,

상기 프로세서는,

상기 복수의 성분들의 어레이 인덱스, 상기 필터 행렬의 행의 개수 및 열의 개수 및 상기 출력 데이터 행렬의 열의 개수에 기초하여 상기 복수의 성분들의 패치 아이디를 생성하고,

상기 패치 아이디 및 상기 복수의 성분들의 오프셋에 기초하여 상기 복수의 성분들의 엘레멘트 아이디를 생성하며,

상기 오프셋은,

상기 패치 아이디, 상기 입력 데이터 행렬의 열의 개수 및 채널의 개수에 기초하여 결정되는 값이고,

상기 어레이 인덱스는, 상기 복수의 성분들이 단일 차원 어레이로 배열되었을 때 각 성분들의 위치를 나타내는 값인, 합성곱 연산장치.

청구항 14

제13항에 있어서,

상기 식별자는, 배치 아이디를 더 포함하고,

상기 프로세서는,

상기 복수의 성분들의 어레이 인덱스, 상기 필터 행렬의 행의 개수 및 열의 개수 및 출력 데이터 행렬의 행의 개수 및 열의 개수에 기초하여 상기 복수의 성분들의 배치 아이디를 생성하는, 합성곱 연산 장치.

청구항 15

제13항에 있어서,

상기 프로세서는,

상기 복수의 성분들의 제1 패치 요소 및 제2 패치 요소를 산출하고,

상기 제1 패치 요소에 상기 필터 행렬의 스트라이드를 곱한 값에 상기 제2 패치 요소를 더하여 상기 패치 아이디를 생성하고,

상기 제1 패치 요소는,

상기 복수의 성분들의 행 요소를 상기 출력 데이터 행렬의 열의 개수로 나누었을 때 출력되는 몫이고,

상기 제2 패치 요소는,

상기 복수의 성분들의 열 요소를 상기 필터 행렬의 열의 개수로 나누었을 때 출력되는 몫이며,

상기 행 요소는, 상기 어레이 인덱스를 상기 필터 행렬의 크기로 나누었을때 출력되는 몫이고,

상기 열 요소는, 상기 어레이 인덱스를 상기 필터 행렬의 크기로 나누었을때, 출력되는 나머지인, 합성곱 연산 장치.

청구항 16

제15항에 있어서,

상기 프로세서는,

상기 오프셋에 상기 행 요소를 상기 출력 데이터 행렬의 열의 개수에 상기 채널의 개수 및 상기 스트라이드를 곱한 값으로 나누었을 때 출력되는 나머지 값과, 상기 열 요소를 상기 필터 행렬의 열의 개수에 상기 채널의 개수를 곱한 값으로 나누었을 때 출력되는 나머지 값을 더하여 상기 엘레멘트 아이디를 생성하고,

상기 입력 데이터 행렬의 오프셋은,

상기 패치 아이디에 상기 입력 데이터 행렬의 열의 개수 및 채널의 개수를 곱한 값인, 합성곱 연산 장치.

청구항 17

제11항에 있어서,

상기 프로세서는,

원본 입력 데이터 행렬의 크기, 복수의 성분들의 개수 및 순서를 워크스페이스에 해당하는 메모리 영역으로 변경하여 상기 입력 데이터 행렬을 생성하고,

상기 입력 데이터 행렬은,

상기 원본 입력 데이터 행렬이 필터 행렬과 GEMM 연산으로 상기 특징 데이터 행렬을 출력할 수 있도록, 상기 원본 입력 데이터 행렬의 복수의 성분들이 규칙을 가지고 재조합되어 배열된 행렬인, 합성곱 연산 장치.

청구항 18

제17항에 있어서,

상기 프로세서는,

상기 원본 입력 데이터 행렬을 특징 데이터 행렬의 행의 개수와 동일한 행의 개수를 가지고, 상기 필터 행렬의 크기와 동일한 열의 개수를 가지는 워크스페이스로 변환하여 상기 입력 데이터 행렬을 생성하는, 합성곱 연산 장치.

청구항 19

제11항에 있어서,

상기 프로세서는,

텐서 코어 로드 데이터를 입력받고,

상기 텐서 코어 로드 데이터에 포함된 제1 목적 레지스터 주소를 가지는 성분의 식별자 및 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있는지 식별하고,

상기 식별자 및 상기 제2 목적 레지스터 주소가 상기 로드 기록 버퍼에 기록되어 있지 않은 경우, 상기 성분의 데이터를 패치하고, 상기 식별자 및 상기 패치된 데이터가 저장된 레지스터의 제2 목적 레지스터 주소를 상기 로드 기록 버퍼에 기록하고, 상기 로드 기록 버퍼에 기록된 제2 목적 레지스터 주소가 상기 제1 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하고,

상기 식별자 및 상기 제2 목적 레지스터 주소가 상기 로드 기록 버퍼에 기록되어 있는 경우, 상기 로드 기록 버퍼에 기록된 제2 목적 레지스터 주소가 상기 제1 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하는, 합성곱 연산 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 인공신경망의 합성곱 연산 중 발생하는 중복 데이터에 대한 불필요한 메모리 접근을 제거하고, 레지스터 파일에 저장된 중복 데이터를 재사용함으로써 메모리 효율을 개선할 수 있는 합성곱 연산 장치 및 합성곱 연산 방법에 관한 것이다.

배 경 기 술

[0002] 합성곱 연산은 심층 인공신경망의 핵심 연산 중 하나로 사물 인식(object detection), 영상 분할(semantic segmentation), 이미지 생성(image generation) 등 많은 컴퓨터 분야에서 범용적으로 사용된다. 이러한 합성곱 연산은 자율주행, VR/AR 등 인공지능 및 애플리케이션에 널리 활용될 수 있는 연산 방식이다.

[0003] 심층 인공신경망의 많은 데이터와 연산량으로 인하여 범용 그래픽스 처리장치가 가속 하드웨어로서 사용되고 있다. 합성곱 연산은 심층 인공신경망의 전체 처리시간 중 대부분을 차지할 정도로 연산량이 많다.

[0004] 한편, 합성곱 연산 과정에 포함되어 있는 행렬 연산 과정에서는 워크스페이스 내에 다수의 데이터를 복제하여 이용하기 때문에 메모리 사용량과 접근횟수가 늘어나서 연산 속도가 느려지고, 에너지 효율이 낮아질 수 있다는 문제가 있다.

발명의 내용

해결하려는 과제

[0005] 개시된 발명의 일 측면에 의하면, 입력 데이터 행렬의 성분마다 엘레멘트 아이디를 부여하고, 이러한 엘레멘트 아이디를 기초로 서로 중복되는 중복 성분들에 대하여 하나의 레지스터를 할당할 수 있어서, 일반적인 합성곱 연산 방법보다 연산 속도가 빠르고, 에너지 효율이 좋은 합성곱 연산 방법을 제공할 수 있다.

과제의 해결 수단

[0006] 개시된 발명의 일 측면에 따른 합성곱 연산 방법은, 입력 데이터 행렬을 설정된 필터 행렬과 일반 행렬 곱(GEneral Matrix Multiplication) 연산하여 출력 데이터 행렬에 대응하는 특징 데이터 행렬을 생성하는 합성곱 연산 방법에 있어서, 적어도 하나의 프로세서에 의해, 상기 입력 데이터 행렬의 복수의 성분들 중 서로 중복되는 데이터를 나타내는 중복 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블을 갱신하는 단계; 및 상기 적어도 하나의 프로세서에 의해, 상기 레지스터 매핑 테이블에 기초하여 상기 중복 성분들에 대해 상기 동일한 제2 목적 레지스터 주소를 가지는 레지스터를 재사용하여 합성곱 연산을 수행하는 단계;를 포함한다.

[0007] 또한, 상기 레지스터 매핑 테이블을 갱신하는 단계는, 상기 복수의 성분들의 식별자를 생성하는 단계; 및 상기 복수의 성분들 중 동일한 식별자가 생성된 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하는 단계;를 포함할 수 있다.

[0008] 또한, 상기 식별자는, 엘레멘트 아이디를 포함하고, 상기 식별자를 생성하는 단계는, 상기 복수의 성분들의 어레이 인덱스, 상기 필터 행렬의 행의 개수 및 열의 개수 및 상기 출력 데이터 행렬의 열의 개수에 기초하여 상기 복수의 성분들의 패치 아이디를 생성하는 단계; 및 상기 패치 아이디 및 상기 복수의 성분들의 오프셋에 기초하여 상기 복수의 성분들의 엘레멘트 아이디를 생성하는 단계;를 포함하며, 상기 오프셋은, 상기 패치 아이디, 상기 입력 데이터 행렬의 열의 개수 및 채널의 개수에 기초하여 결정되는 값이고, 상기 어레이 인덱스는, 상기 복수의 성분들이 단일 차원 어레이로 배열되었을 때 각 성분들의 위치를 나타내는 값일 수 있다.

[0009] 또한, 상기 식별자는, 배치 아이디를 더 포함하고, 상기 식별자를 생성하는 단계는, 상기 복수의 성분들의 어레이 인덱스, 상기 필터 행렬의 행의 개수 및 열의 개수 및 출력 데이터 행렬의 행의 개수 및 열의 개수에 기초하여 상기 복수의 성분들의 배치 아이디를 생성하는 단계;를 더 포함할 수 있다.

[0010] 또한, 상기 패치 아이디를 생성하는 단계는, 상기 복수의 성분들의 제1 패치 요소 및 제2 패치 요소를 산출하는 단계; 및 상기 제1 패치 요소에 상기 필터 행렬의 스트라이드를 곱한 값에 상기 제2 패치 요소를 더하여 상기 패치 아이디를 생성하는 단계;를 포함하고, 상기 제1 패치 요소는, 상기 복수의 성분들의 행 요소를 상기 출력 데이터 행렬의 열의 개수로 나누었을 때 출력되는 몫이고, 상기 제2 패치 요소는, 상기 복수의 성분들의 열 요소를 상기 필터 행렬의 열의 개수로 나누었을 때 출력되는 몫이며, 상기 행 요소는, 상기 어레이 인덱스를 상기 필터 행렬의 크기로 나누었을때 출력되는 몫이고, 상기 열 요소는, 상기 어레이 인덱스를 상기 필터 행렬의 크기로 나누었을때, 출력되는 나머지일 수 있다.

[0011] 또한, 상기 엘레멘트 아이디를 생성하는 단계는, 상기 오프셋에 상기 행 요소를 상기 출력 데이터 행렬의 열의 개수에 상기 채널의 개수 및 상기 스트라이드를 곱한 값으로 나누었을 때 출력되는 나머지 값과, 상기 열 요소를 상기 필터 행렬의 열의 개수에 상기 채널의 개수를 곱한 값으로 나누었을 때 출력되는 나머지 값을 더하여 상기 엘레멘트 아이디를 생성하고, 상기 입력 데이터 행렬의 오프셋은 상기 패치 아이디에 상기 입력 데이터 행렬의 열의 개수 및 채널의 개수를 곱한 값일 수 있다.

[0012] 또한, 상기 적어도 하나의 프로세서에 의해, 원본 입력 데이터 행렬의 크기, 복수의 성분들의 개수 및 순서를 워크스페이스에 해당하는 메모리 영역으로 변경하여 상기 입력 데이터 행렬을 생성하는 단계;를 더 포함하고, 상기 입력 데이터 행렬은, 상기 원본 입력 데이터 행렬이 필터 행렬과 GEMM 연산으로 상기 특징 데이터 행렬을 출력할 수 있도록, 상기 원본 입력 데이터 행렬의 복수의 성분들이 규칙을 가지고 재조합되어 배열된 행렬일 수 있다.

[0013] 또한, 상기 입력 데이터 행렬을 생성하는 단계는, 상기 원본 입력 데이터 행렬을 특징 데이터 행렬의 행의 개수와 동일한 행의 개수를 가지고, 상기 필터 행렬의 크기와 동일한 열의 개수를 가지는 워크스페이스로 변환하여 상기 입력 데이터 행렬을 생성하는 단계;를 포함할 수 있다.

[0014] 또한, 상기 레지스터 매핑 테이블을 갱신하는 단계는, 텐서 코어 로드 데이터를 입력받는 단계; 상기 텐서 코어 로드 데이터에 포함된 제1 목적 레지스터 주소를 가지는 성분의 식별자 및 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있는지 식별하는 단계; 상기 식별자 및 상기 제2 목적 레지스터 주소가 상기 로드 기록 버퍼에 기록되어 있지 않은 경우, 메모리에 접근하여 메모리 계층으로부터 상기 성분의 데이터를 패치하고, 상기 식별자 및 상기 패치된 데이터가 저장된 레지스터의 제2 목적 레지스터 주소를 상기 로드 기록 버퍼에 기록하고, 상기 로드 기록 버퍼에 기록된 제2 목적 레지스터 주소가 상기 제1 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하는 단계; 및 상기 식별자 및 상기 제2 목적 레지스터 주소가 상기 로드 기록 버퍼에 기록되어 있는 경우, 메모리에 접근하지 않고 상기 로드 기록 버퍼에 기록된 제2 목적 레지스터 주소가 상기 제1

목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하는 단계;를 포함할 수 있다.

[0015] 개시된 발명의 일 측면에 따른 컴퓨터 프로그램은, 상기 합성곱 연산 방법을 실행시키도록 컴퓨터로 판독 가능한 기록매체에 저장된다.

[0016] 개시된 발명의 일 측면에 따른 합성곱 연산 장치는, 합성곱 연산 장치에 있어서, 레지스터 매핑 테이블이 저장된 메모리; 및 입력 데이터 행렬을 설정된 필터 행렬과 일반 행렬 곱(GEneral Matrix Multiplication) 연산하여 출력 데이터 행렬에 대응하는 특징 데이터 행렬을 생성하는 합성곱 연산을 수행하는 프로세서;를 포함하고, 상기 프로세서는, 상기 입력 데이터 행렬의 복수의 성분들 중 서로 중복되는 데이터를 나타내는 중복 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하고, 상기 레지스터 매핑 테이블에 기초하여 상기 중복 성분들에 대해 상기 동일한 제2 목적 레지스터 주소를 가지는 레지스터를 재사용하여 합성곱 연산을 수행한다.

[0017] 또한, 상기 프로세서는, 상기 복수의 성분들의 식별자를 생성하고, 상기 복수의 성분들 중 동일한 식별자가 생성된 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신할 수 있다.

[0018] 또한, 상기 식별자는, 엘레멘트 아이디를 포함하고, 상기 프로세서는, 상기 복수의 성분들의 어레이 인덱스, 상기 필터 행렬의 행의 개수 및 열의 개수 및 상기 출력 데이터 행렬의 열의 개수에 기초하여 상기 복수의 성분들의 패치 아이디를 생성하고, 상기 패치 아이디 및 상기 복수의 성분들의 오프셋에 기초하여 상기 복수의 성분들의 엘레멘트 아이디를 생성하며, 상기 오프셋은, 상기 패치 아이디, 상기 입력 데이터 행렬의 열의 개수 및 채널의 개수에 기초하여 결정되는 값이고, 상기 어레이 인덱스는, 상기 복수의 성분들이 단일 차원 어레이로 배열되었을 때 각 성분들의 위치를 나타내는 값일 수 있다.

[0019] 또한, 상기 식별자는, 배치 아이디를 더 포함하고, 상기 프로세서는, 상기 복수의 성분들의 어레이 인덱스, 상기 필터 행렬의 행의 개수 및 열의 개수 및 출력 데이터 행렬의 행의 개수 및 열의 개수에 기초하여 상기 복수의 성분들의 배치 아이디를 생성할 수 있다.

[0020] 또한, 상기 프로세서는, 상기 복수의 성분들의 제1 패치 요소 및 제2 패치 요소를 산출하고, 상기 제1 패치 요소에 상기 필터 행렬의 스트라이드를 곱한 값에 상기 제2 패치 요소를 더하여 상기 패치 아이디를 생성하고, 상기 제1 패치 요소는, 상기 복수의 성분들의 행 요소를 상기 출력 데이터 행렬의 열의 개수로 나누었을 때 출력되는 몫이고, 상기 제2 패치 요소는, 상기 복수의 성분들의 열 요소를 상기 필터 행렬의 열의 개수로 나누었을 때 출력되는 몫이며, 상기 행 요소는, 상기 어레이 인덱스를 상기 필터 행렬의 크기로 나누었을때 출력되는 몫이고, 상기 열 요소는, 상기 어레이 인덱스를 상기 필터 행렬의 크기로 나누었을때, 출력되는 나머지일 수 있다.

[0021] 또한, 상기 프로세서는, 상기 오프셋에 상기 행 요소를 상기 출력 데이터 행렬의 열의 개수에 상기 채널의 개수 및 상기 스트라이드를 곱한 값으로 나누었을 때 출력되는 나머지 값과, 상기 열 요소를 상기 필터 행렬의 열의 개수에 상기 채널의 개수를 곱한 값으로 나누었을 때 출력되는 나머지 값을 더하여 상기 엘레멘트 아이디를 생성하고, 상기 입력 데이터 행렬의 오프셋은, 상기 패치 아이디에 상기 입력 데이터 행렬의 열의 개수 및 채널의 개수를 곱한 값일 수 있다.

[0022] 또한, 상기 프로세서는, 원본 입력 데이터 행렬의 크기, 복수의 성분들의 개수 및 순서를 워크스페이스에 해당하는 메모리 영역으로 변경하여 상기 입력 데이터 행렬을 생성하고, 상기 입력 데이터 행렬은, 상기 원본 입력 데이터 행렬이 필터 행렬과 GEMM 연산으로 상기 특징 데이터 행렬을 출력할 수 있도록, 상기 원본 입력 데이터 행렬의 복수의 성분들이 규칙을 가지고 재조합되어 배열된 행렬일 수 있다.

[0023] 또한, 상기 프로세서는, 상기 원본 입력 데이터 행렬을 특징 데이터 행렬의 행의 개수와 동일한 행의 개수를 가지고, 상기 필터 행렬의 크기와 동일한 열의 개수를 가지는 워크스페이스로 변환하여 상기 입력 데이터 행렬을 생성할 수 있다.

[0024] 또한, 상기 프로세서는, 텐서 코어 로드 데이터를 입력받고, 상기 텐서 코어 로드 데이터에 포함된 제1 목적 레지스터 주소를 가지는 성분의 식별자 및 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있는지 식별하고, 상기 식별자 및 상기 제2 목적 레지스터 주소가 상기 로드 기록 버퍼에 기록되어 있지 않은 경우, 상기 성분의 데이터를 패치하고, 상기 식별자 및 상기 패치된 데이터가 저장된 레지스터의 제2 목적 레지스터 주소를 상기 로드 기록 버퍼에 기록하고, 상기 로드 기록 버퍼에 기록된 제2 목적 레지스터 주소가 상기 제1 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신하고, 상기 식별자 및 상기 제2 목적 레지스터

주소가 상기 로드 기록 버퍼에 기록되어 있는 경우, 상기 로드 기록 버퍼에 기록된 제2 목적 레지스터 주소가 상기 제1 목적 레지스터 주소에 대응하도록 상기 레지스터 매핑 테이블을 갱신할 수 있다.

발명의 효과

[0025] 개시된 발명의 일 측면에 따르면, 원본 입력 데이터 행렬을 워크스페이스로 변환하고, 변환된 입력 데이터 행렬의 각 성분에 대하여 서로 중복되는 중복 성분들에게 공통 목적 레지스터를 할당하고, 공통 목적 레지스터에 저장된 입력 데이터 행렬의 데이터를 재사용하여 일반적인 합성곱 연산보다 연산 속도가 빠르고 에너지 효율이 좋은 연산을 수행할 수 있다.

도면의 간단한 설명

[0026] 도 1은 일 실시예에 따른 합성곱 연산 장치의 구성도이다.

도 2는 일 실시예에 따른 합성곱 연산 장치의 또다른 구성도이다.

도 3은 일 실시예에서 로드 기록 버퍼를 사용하는 것을 설명하기 위한 표이다.

도 4는 일 실시예에 따른 아이디 생성기 및 로드 기록 버퍼의 구성을 나타낸 도면이다.

도 5는 일 실시예에 따라 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있는지 식별하는 것을 설명하기 위한 도면이다.

도 6은 일 실시예에 따라 원본 입력 데이터 행렬을 필터 행렬과 합성곱 연산을 수행하는 것을 설명하기 위한 도면이다.

도 7은 일 실시예에 따라 워크스페이스로 변환된 입력 데이터 행렬을 나타낸 도면이다.

도 8은 일 실시예에 따른 입력 데이터 행렬의 성분들이 규칙을 가지고 재조합되어 배열된 행렬임을 나타내는 도면이다.

도 9는 일 실시예에 따른 입력 데이터 행렬의 성분들이 규칙을 가지고 배열된 행렬임을 나타내는 또다른 도면이다.

도 10은 일 실시예에 따라 생성된 입력 데이터 행렬의 복수의 성분들의 패치 아이디를 설명하기 위한 도면이다.

도 11은 일 실시 예에 따라 생성된 입력 데이터 행렬(202)의 복수의 성분들의 엘레멘트 아이디를 설명하기 위한 도면이다.

도 12는 또 다른 일 실시 예에 따른 아이디 생성기 및 로드 기록 버퍼의 구성을 나타낸 도면이다.

도 13은 일 실시예에 따른 합성곱 연산 방법의 순서도이다.

도 14는 본 발명의 합성곱 연산 방법의 효과를 나타내는 그래프이다.

발명을 실시하기 위한 구체적인 내용

[0027] 명세서 전체에 걸쳐 동일 참조 부호는 동일 구성요소를 지칭한다. 본 명세서가 실시예들의 모든 요소들을 설명하는 것은 아니며, 개시된 발명이 속하는 기술분야에서 일반적인 내용 또는 실시예들 간에 중복되는 내용은 생략한다. 명세서에서 사용되는 '~모듈'이라는 용어는 소프트웨어 또는 하드웨어로 구현될 수 있으며, 실시예들에 따라 복수의 '~모듈'이 하나의 구성요소로 구현되거나, 하나의 '모듈'이 복수의 구성요소들을 포함하는 것도 가능하다.

[0028] 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.

[0029] 제1, 제2 등의 용어는 하나의 구성요소를 다른 구성요소로부터 구별하기 위해 사용되는 것으로, 구성요소가 전술된 용어들에 의해 제한되는 것은 아니다.

[0030] 단수의 표현은 문맥상 명백하게 예외가 있지 않는 한, 복수의 표현을 포함한다.

[0031] 각 단계들에 있어 식별부호는 설명의 편의를 위하여 사용되는 것으로 식별부호는 각 단계들의 순서를 설명하는 것이 아니며, 각 단계들은 문맥상 명백하게 특정 순서를 기재하지 않는 이상 명기된 순서와 다르게 실시될 수

있다.

[0032] 이하 첨부된 도면들을 참고하여 개시된 발명의 작용 원리 및 실시예들에 대해 설명한다.

[0033] 도 1은 일 실시예에 따른 합성곱 연산 장치의 구성도이며, 도 2는 일 실시예에 따른 합성곱 연산 장치의 또다른 구성도이다.

[0034] 도 1을 참조하면, 본 발명의 실시예에 따른 합성곱 연산 장치(100)는, 프로세서(110), 메모리(120), 아이디 생성기(130), 로드 기록 버퍼(140), 레지스터(150)를 포함할 수 있다.

[0035] 프로세서(110)는 합성곱 연산을 수행할 수 있다. 구체적으로, 프로세서(110)는 합성곱 연산을 수행하기 위하여 입력 데이터 행렬을 설정된 필터 행렬과 일반 행렬곱(GEneral Matrix Multiplication)연산을 수행하여 출력 데이터 행렬에 대응되는 특징 데이터 행렬을 생성할 수 있다.

[0036] 여기에서, 입력 데이터 행렬은 원본 입력 데이터 행렬이 워크스페이스로 변환된 행렬일 수 있다. 구체적으로, 입력 데이터 행렬은 원본 입력 데이터 행렬이 필터 행렬과 일반 행렬 곱 연산으로 특징 데이터 행렬을 출력할 수 있도록, 원본 입력 데이터 행렬의 복수의 성분들이 규칙을 가지고 재조합되어 배열된 행렬일 수 있다. 특징 데이터 행렬은 합성곱 연산으로 생성되는 출력 데이터 행렬에 대응되는 행렬일 수 있다. 구체적으로, 특징 데이터 행렬은 설정된 필터 행렬이 이용되는 GEMM 연산에서 입력 데이터 행렬을 입력값으로 했을 때의 결과값일 수 있다. 또한, 특징 데이터 행렬은 1개의 열로 된 행렬일 수 있다.

[0037] 메모리(120)는 입력 데이터 행렬의 복수의 성분들의 데이터 규칙에 기초하여 설정되는 레지스터 매핑 테이블을 저장할 수 있다.

[0038] 프로세서(110)는 메모리(120)에 저장된 레지스터 매핑 테이블을 갱신할 수 있다.

[0039] 구체적으로, 프로세서(110)는 입력 데이터 행렬의 복수의 성분들 중 서로 중복되는 데이터를 나타내는 중복 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블을 갱신할 수 있다.

[0040] 여기에서, 중복 성분은 입력 데이터 행렬의 성분들 중 서로 중복되는 성분일 수 있다.

[0041] 구체적으로, 원본 입력 데이터 행렬이 워크스페이스로 변환되는 과정에서 행렬의 크기는 확장되고, 원본 입력 데이터 행렬에서의 특정 성분이 위치를 달리하여 중복하여 배열될 수 있다. 이와 같이, 입력 데이터 행렬의 성분들 중 동일한 원본 입력 데이터 행렬의 성분에 기초하여 변환된 성분들은 서로 중복 성분이 될 수 있다.

[0042] 그리고, 프로세서(110)는 레지스터 매핑 테이블에 기초하여 중복 성분들 대해 동일한 제2 목적 레지스터 주소를 가지는 레지스터(150)를 재사용하여 입력 데이터 행렬과 필터 행렬 간의 합성곱 연산을 수행하도록 구성될 수 있다.

[0043] 즉, 입력 데이터 행렬의 성분들은 중복 성분들을 포함한다는 점에서, GEMM연산을 수행하는데 현재 필요한 데이터가 이미 메모리(120)에서 불러온 적이 있는 경우, 해당 데이터가 저장된 레지스터(150)가 있을 수 있다.

[0044] 이 경우, 프로세서(110)는 굳이 다시 메모리(120)에 접근하여 해당 데이터를 불러오지 않고, 해당 데이터가 저장된 레지스터(150)에서 불러와 GEMM 연산을 수행할 수 있다는 점에서, 메모리(120)에 대한 불필요한 액세스를 방지할 수 있다.

[0045] 도2를 참조하면, 합성곱 연산 장치(100)는 레지스터 매핑 테이블(121)을 포함할 수 있다.

[0046] 레지스터 매핑 테이블(121)은 복수의 제1 목적 레지스터 주소들과, 복수의 제2 목적 레지스터 주소들 간의 대응 관계를 나타내는 테이블일 수 있다.

[0047] 제1 목적 레지스터 주소는 입력 데이터 행렬의 복수의 성분들 각각에 대응하는 논리적 주소(logical address) 또는 가상 주소(virtual address)를 의미할 수 있다.

[0048] 제2 목적 레지스터 주소는 복수 개의 레지스터(150)들 각각에 대응하는 물리적 주소(physical address)를 의미할 수 있다.

[0049] 한편, 해당 성분에 대한 제2 목적 레지스터 주소는 해당 성분 및 해당 성분과 중복된 성분에 대하여 동일하게 설정될 수 있다.

[0050] 구체적으로, 제1 목적 레지스터 주소는 입력 데이터 행렬의 복수의 성분들 각각에 대응되므로, 입력 데이터 행

렬의 성분마다 다를 수 있다. 예를 들어, 입력 데이터 행렬의 1행 2열의 성분의 제1 목적 레지스터 주소와 입력 데이터 행렬의 2행 1열의 성분의 제1 목적 레지스터 주소는 다를 수 있다.

[0051] 반면, 제2 목적 레지스터 주소는 중복된 성분들에 대하여 동일하게 설정될 수 있다. 즉, 만약 입력 데이터 행렬의 1행 2열의 성분과 입력 데이터 행렬의 2행 1열의 성분이 서로 중복되는 성분이라면, 입력 데이터 행렬의 1행 2열의 성분의 제1 목적 레지스터 주소와 입력 데이터 행렬의 2행 1열의 성분의 제2 목적 레지스터 주소는 동일할 수 있다.

[0052] 그리고, 제1 목적 레지스터 주소들은 각각 대응되는 제2 목적 레지스터 주소들이 있을 수 있다. 즉, 제1 목적 레지스터 주소에 대응되는 성분의 데이터는 해당 제1 목적 레지스터 주소에 대응되는 제2 목적 레지스터 주소를 가지는 레지스터(150)에 저장되거나 저장될 수 있으며, 이러한 제1 목적 레지스터 주소와 제2 목적 레지스터 주소간의 대응 관계는 레지스터 매핑 테이블(121)에 나타날 수 있다.

[0053] 프로세서(110)는 레지스터 매핑 테이블(121)에 기초하여, 중복 성분들에 대하여 동일한 제2 목적 레지스터 주소를 가지는 레지스터(150)를 재사용하여 합성곱 연산을 수행할 수 있다.

[0054] 구체적으로, 프로세서(110)는 레지스터 매핑 테이블(121)에 기초하여 제1 목적 레지스터 주소를 이에 대응되는 제2 목적 레지스터 주소로 변환할 수 있다.

[0055] 그리고, 프로세서(110)는 입력 데이터 행렬의 복수의 성분들 중 특정 제1 목적 레지스터의 성분의 데이터가 이용되는 연산을 수행하는 경우, 해당 제1 목적 레지스터 주소가 변환된 제2 목적 레지스터 주소를 가지는 레지스터(150)로부터 해당 데이터를 획득하여 연산을 수행할 수 있다.

[0056] 아이디 생성기(130)는 입력 데이터 행렬의 복수의 성분들의 식별자를 생성할 수 있다. 여기에서, 식별자는 입력 데이터 행렬의 복수의 성분들 각각에 대하여 하나씩 생성될 수 있다. 이 경우, 아이디 생성기(130)는, 입력 데이터 행렬의 복수의 성분들 중 중복 성분들에 대하여 동일한 식별자를 생성할 수 있다.

[0057] 구체적으로, 아이디 생성기(130)는 로드 인스트럭션의 메모리 주소와 합성곱 연산의 인자(입력 데이터 행렬의 크기/폭/채널수, 필터 크기/폭/이동거리 등)를 기반으로 중복 성분들에 동일한 식별자를 생성함으로써 입력 데이터 행렬의 복수의 성분들 간의 중복성 여부가 판별될 수 있다. 이를 위해, 아이디 생성기(130)는 합성곱 연산이 시작할 때 해당 인자들에 맞게 프로그래밍될 수 있다.

[0058] 프로세서(110)는 식별자에 기초하여 레지스터 매핑 테이블(121)을 갱신할 수 있다. 즉, 프로세서(110)는 입력 데이터 행렬의 복수의 성분들 중 동일한 식별자를 가지는 성분들의 제1 목적 레지스터 주소가 동일한 제2 목적 레지스터에 대응하도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0059] 로드 기록 버퍼(140)에는 앞서 로드된 성분들의 식별자와 해당 성분이 저장된 레지스터의 제2 목적 레지스터 주소가 저장될 수 있다.

[0060] 그리고, 프로세서(110)는 입력 데이터 행렬의 특정 성분에 대한 로드 명령이 수신되면, 아이디 생성기(130)가 생성한 식별자를 기반으로 앞선 기록 중 해당 성분과 동일한 식별자를 가졌던 성분에 대한 기록이 있는지 로드 기록 버퍼(140)를 통해 확인할 수 있다.

[0061] 그리고, 프로세서(110)는 로드 기록 버퍼(140)에 동일한 식별자를 가졌던 성분에 대한 기록이 존재하는 것으로 확인되면, 해당 성분의 데이터를 메모리(120)로부터 읽어오는 대신 로드 기록 버퍼(140)가 알려준 제2 목적 레지스터 주소를 가지는 레지스터(150)에 저장된 값을 재사용함으로써 중복 데이터를 반복적으로 읽는 불필요한 메모리 접근을 제거할 수 있다.

[0062] 이처럼 본 발명의 합성곱 연산 방법은 합성곱 연산을 이용하는 심층 인공신경망에서 서로 다른 메모리 주소에 위치한 중복 데이터를 효율적으로 검출하고, 이용함으로써 데이터의 재사용 횟수를 획기적으로 늘리는 효과가 있다. 결과적으로 본 발명이 제안하는 기술은 많은 양의 데이터를 재사용하고 불필요한 메모리 접근을 제거함으로써 범용 그래픽스 처리장치의 성능 개선을 이끌어낼 수 있다.

[0063] 아이디 생성기(130)는 합성곱 연산 장치(100)에 포함된 복수개의 프로세서(110) 중 어느 하나의 프로세서(110)를 포함할 수 있다. 또한, 지금까지 설명된 본 발명의 실시예 및 앞으로 설명할 실시예에 따른 합성곱 연산 방법은, 프로세서(110) 및 아이디 생성기(130)에 의해 구동될 수 있는 프로그램의 형태로 구현될 수 있다.

[0064] 여기서 프로그램은, 프로그램 명령, 데이터 파일 및 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 프로그램은 기계어 코드나 고급 언어 코드를 이용하여 설계 및 제작된 것일 수 있다. 프로그램은 상술한 부호

수정을 위한 방법을 구현하기 위하여 특별히 설계된 것일 수도 있고, 컴퓨터 소프트웨어 분야에서 통상의 기술자에게 기 공지되어 사용 가능한 각종 함수나 정의를 이용하여 구현된 것일 수도 있다. 전술한 정보 표시 방법을 구현하기 위한 프로그램은, 프로세서(110) 및 아이디 생성기(130)에 의해 판독 가능한 기록매체에 기록될 수 있다. 이때, 기록매체는 메모리(120)일 수 있다.

[0065] 메모리(120)는 전술한 동작 및 후술하는 동작을 수행하는 프로그램을 저장할 수 있으며, 메모리(120)는 저장된 프로그램을 실행시킬 수 있다. 프로세서(110)와 메모리(120)가 복수인 경우에, 이들이 하나의 칩에 집적되는 것도 가능하고, 물리적으로 분리된 위치에 마련되는 것도 가능하다. 메모리(120)는 데이터를 일시적으로 기억하기 위한 S램(Static Random Access Memory, S-RAM), D램(Dynamic Random Access Memory) 등의 휘발성 메모리를 포함할 수 있다. 또한, 메모리(120)는 제어 프로그램 및 제어 데이터를 장기간 저장하기 위한 롬(Read Only Memory), 이피롬(Erasable Programmable Read Only Memory: EPROM), 이이피롬(Electrically Erasable Programmable Read Only Memory: EEPROM) 등의 비휘발성 메모리를 포함할 수 있다.

[0066] 프로세서(110) 및 아이디 생성기(130)는 각종 논리 회로와 연산 회로를 포함할 수 있으며, 메모리(120)로부터 제공된 프로그램에 따라 데이터를 처리하고, 처리 결과에 따라 제어 신호를 생성할 수 있다.

[0067] 도 3은 일 실시예에서 로드 기록 버퍼를 사용하는 것을 설명하기 위한 표이다.

[0068] 종래의 기술은 캐시를 이용하여 데이터의 지역성(locality)을 활용하는데, 심층 인공신경망의 데이터는 동일한 중복 성분들의 데이터가 서로 다른 메모리 주소에 위치하기 때문에 일반적인 캐시로는 데이터의 지역성을 전혀 활용할 수가 없다. 반면, 본 발명의 합성곱 연산 방법은 다양한 인자를 고려하여 동일한 중복 성분들의 데이터를 같은 코어에 할당함으로써 중복 데이터 검출 및 제거 효과를 최대화할 수 있다.

[0069] 도3을 참조하면, 텐서 코어 로드 데이터(500)는 제1 목적 레지스터 주소(501)를 포함할 수 있다.

[0070] 프로세서(110)는 텐서 코어 로드 데이터(500)를 입력 받을 수 있다. 구체적으로, 프로세서(110)는 현재 진행중인 합성곱 연산 과정에 대한 텐서 코어 로드 데이터(500)를 입력 받을 수 있다.

[0071] 텐서 코어 로드 데이터(500)는 텐서 코어 로드 명령에 포함된 데이터일 수 있다. 이때, 프로세서(110)는 텐서 코어 로드 명령에 기초하여 이미 이전 연산에서 이용되어 레지스터(150)에 저장된 중복 데이터를 로드할 수 있다.

[0072] 텐서 코어 로드 데이터(500)는 각각 하나의 제1 목적 레지스터 주소(501)를 포함할 수 있다. 구체적으로, 텐서 코어 로드 명령은 합성곱 연산을 수행하기 위하여 제1 목적 레지스터 주소(501)의 성분의 데이터를 로드하는 명령을 포함할 수 있다.

[0073] 즉, 어느 한 텐서 코어 로드 명령이 입력되면 해당 텐서 코어 로드 명령에 포함된 제1 목적 레지스터 주소에 해당하는 입력 데이터 행렬의 특정한 성분의 데이터가 로드되어야 할 수 있다.

[0074] 한편, 텐서 코어 로드 데이터(500)는 각각 하나의 어레이 인덱스(700)에 대응될 수 있다. 이때, 어레이 인덱스(700)는 로드되어야 하는 성분의 행렬에서의 위치에 대한 인자일 수 있다.

[0075] 구체적으로, 어레이 인덱스(700)는 입력 데이터 행렬의 복수의 성분들이 단일 차원 어레이로 배열되었을 때 각 성분들의 위치를 나타내는 값일 수 있다.

[0076] 결과적으로, 어레이 인덱스(700)는 텐서 코어 로드 명령에 의해 필요한 성분의 입력 데이터 행렬에서의 위치에 관한 정보일 수 있다. 또한, 각 어레이 인덱스(700)는 하나의 제1 목적 레지스터 주소에 대응될 수 있다.

[0077] 프로세서(110)는 입력 데이터 행렬의 복수의 성분들의 식별자를 생성할 수 있다.

[0078] 그리고, 프로세서(110)는 입력 데이터 행렬의 복수의 성분들 중 동일한 식별자가 생성된 성분들의 제1 목적 레지스터 주소(501)들이 동일한 제2 목적 레지스터 주소(142)에 대응하도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0079] 여기에서, 본 개시의 일 실시 예에 따른 식별자는 도 3과 같이, 엘레멘트 아이디(141)를 포함할 수 있다.

[0080] 여기에서, 텐서 코어 로드 명령의 연산에 필요한 입력 데이터 행렬 성분의 엘레멘트 아이디(141)는 해당 연산에 필요한 성분의 행렬에서의 위치, 즉 어레이 인덱스(700)에 기초하여 생성될 수 있다.

[0081] 예를 들어, 도 3을 참조하면, 프로세서(110)는 입력 받은 텐서 코어 로드 데이터(500)에 포함된 제1 목적 레지

스터 주소(501) 'r4'에 대응하는 성분의 엘레멘트 아이디(141)가 '2'인 것으로 생성할 수 있다.

[0082] 또한, 프로세서(110)는 입력 받은 텐서 코어 로드 데이터(500)에 포함된 제1 목적 레지스터 주소(501) 'r3'에 대응하는 성분의 엘레멘트 아이디(141)가 '10'인 것으로 생성할 수 있다.

[0083] 또한, 프로세서(110)는 입력 받은 텐서 코어 로드 데이터(500)에 포함된 제1 목적 레지스터 주소(501) 'r8'에 대응하는 성분의 엘레멘트 아이디(141)가 '6'인 것으로 생성할 수 있다.

[0084] 그리고, 프로세서(110)는 입력 데이터 행렬 성분의 엘레멘트 아이디(141)에 기초하여 제2 목적 레지스터 주소(142)를 결정할 수 있다. 구체적으로, 프로세서(110)는 동일한 엘레멘트 아이디(141)를 가지는 성분들의 제2 목적 레지스터 주소(142)가 같은 값을 가지도록 제2 목적 레지스터 주소(142)를 결정할 수 있다.

[0085] 그리고, 프로세서(110)는 결정된 제2 목적 레지스터 주소(142)가 해당 성분들의 제1 목적 레지스터 주소(501)와 대응되도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0086] 예를 들어, 도3을 참조하면, 입력 받은 텐서 코어 로드 데이터(500)의 제1 목적 레지스터 주소(501)에 대응하는 엘레멘트 아이디(141)가 '2'이고, 이에 대응하는 제2 목적 레지스터 주소(142)가 'p2'인 것으로 결정되면, 프로세서(110)는 제2 목적 레지스터 주소(142) 'p2'가 제1 목적 레지스터 주소(501) 'r4'에 대응되는 것으로 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0087] 또한, 입력 받은 텐서 코어 로드 데이터(500) 의 제1 목적 레지스터 주소(501)에 대응하는 엘레멘트 아이디(141)가 '6'이고, 이에 대응하는 제2 목적 레지스터 주소(142)가 'p6'인 것으로 결정되면, 프로세서(110)는 제2 목적 레지스터 주소(142) 'p6'가 제1 목적 레지스터 주소(501) 'r8'에 대응되는 것으로 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0088] 현재 입력 받은 텐서 코어 로드 명령에서 이용되는 입력 데이터 행렬의 성분에 대응하는 엘레멘트 아이디(141)와 동일한 엘레멘트 아이디(141)를 나타내는 텐서 코어 로드 명령이 이전에 있었을 수 있다.

[0089] 이 경우, 프로세서(110)는 현재 입력 받은 텐서 코어 로드 명령의 성분의 제1 목적 레지스터 주소(501)에 대응되는 제2 목적 레지스터 주소(142)가 이전의 텐서 코어 로드 명령의 성분의 제1 목적 레지스터 주소(501)에 대응되는 제2 목적 레지스터 주소(142)와 동일하도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0090] 이에 따라, 프로세서(110)는 이전에 있었던 텐서 코어 로드 명령에서 이용했던 레지스터와 동일한 레지스터를 이용하여 현재 입력 받은 텐서 코어 로드 명령의 연산을 수행할 수 있다.

[0091] 예를 들어, 도3을 참조하면, 세 번째로 입력 받은 텐서 코어 로드 명령의 경우, 엘레멘트 아이디(141)가 '2'이며, 이는 첫 번째로 입력 받은 텐서 코어 로드 명령의 엘레멘트 아이디(141)인 '2'와 동일하다. 따라서, 프로세서(110)는 세 번째로 입력 받은 텐서 코어 로드 명령에 포함된 제1 목적 레지스터 주소 'r3'에 대응되는 제2 목적 레지스터 주소(142)가 'p2'인 것으로 갱신할 수 있다.

[0092] 이에 따라, 프로세서(110)는 세 번째로 입력 받은 텐서 코어 로드 명령에 대한 연산에 이용되는 데이터를 로드하기 위하여 굳이 메모리(120)에 접근할 필요없이, 첫 번째 텐서 코어 로드 명령에 대한 연산에 이용되었던 제2 목적 레지스터 주소(142)가 'p2'인 레지스터(150)에 저장된 데이터를 이용하여 행렬 연산을 수행할 수 있게 된다.

[0093] 도 4는 일 실시예에 따른 아이디 생성기 및 로드 기록 버퍼의 구성을 나타낸 도면이며, 도 5는 일 실시예에 따라 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있는지 식별하는 것을 설명하기 위한 도면이다.

[0094] 도 4를 참조하면, 합성곱 연산 장치(100)는 범용 그래픽스 처리장치에 데이터의 중복성을 확인할 수 있는 감지 유닛(detection unit)을 포함할 수 있다. 또한, 프로세서(110)는 이러한 감지 유닛을 포함할 수 있다.

[0095] 감지 유닛은 앞서 프로세서(110)가 동일한 메모리 데이터에 이미 접근하였는지 확인하고, 해당 데이터가 레지스터 파일의 어느 위치에 저장되어 있는지 기록할 수 있다. 감지 유닛은 데이터 중복성을 확인할 수 있는 아이디 생성기(ID generator)(130)와 메모리 로드 인스트럭션의 행적을 추적하기 위한 로드 기록 버퍼(load history buffer)(140)로 구성될 수 있다.

[0096] 아이디 생성기(130)는 현재 행렬 연산에 필요한 입력 데이터 행렬의 성분의 엘레멘트 아이디(141)를 생성할 수 있다. 그리고, 로드 기록 버퍼(140)에는 생성된 엘레멘트 아이디(141) 및 이에 대응되는 제2 목적 레지스터 주소(142)가 기록될 수 있다.

[0097] 도 5를 참조하면, 감지 유닛, 즉 프로세서(110)는 현재 행렬 연산에 필요한 입력 데이터 행렬의 성분의 엘레멘트 아이디(141) 및 이에 대응되는 제2 목적 레지스터 주소(142)가 로드 기록 버퍼(140)에 기록되어 있는지 식별할 수 있다.

[0098] 프로세서(110)는 현재 행렬 연산에 필요한 입력 데이터 행렬의 성분의 엘레멘트 아이디(141) 및 이에 대응되는 제2 목적 레지스터 주소(142)가 로드 기록 버퍼(140)에 기록되어 있는 경우, 해당 제2 목적 레지스터 주소(142)가 해당 성분의 제1 목적 레지스터 주소(501)에 대응되도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0099] 이에 따라, 프로세서(110)는 레지스터 매핑 테이블(121)에 기초하여 현재 행렬 연산에 필요한 입력 데이터 행렬의 성분의 데이터를 로드하기 위하여, 메모리(120)에 접근하지 않고 로드 기록 버퍼(140)에 기록된 제2 목적 레지스터 주소(142)를 가지는 레지스터(150)에 저장된 데이터를 재사용할 수 있다.

[0100] 예를 들어, 도5의 (a), (b), (c)를 참조하면, 현재 연산에 필요한 입력 데이터 성분의 엘레멘트 아이디(141)가 '2'이고, 이에 대응되는 제2 목적 레지스터 주소(142)는 'p2'일 수 있다. 이때, 엘레멘트 아이디 '2' 및 제2 목적 레지스터 주소(142) 'p2'가 로드 기록 버퍼(140)에 기록되어 있는 경우, 프로세서(110)는 메모리(120)에 접근하지 않고, 로드 기록 버퍼(140)에 기록된 제2 목적 레지스터 주소 'p2'를 가지는 레지스터(150)에 저장된 데이터를 재사용할 수 있다.

[0101] 반면, 프로세서(110)는 현재 연산에 필요한 입력 데이터 행렬의 성분의 엘레멘트 아이디(141) 및 이에 대응되는 제2 목적 레지스터 주소(142)가 로드 기록 버퍼(140)에 기록되어 있지 않은 경우, 메모리(120)에 접근하여 메모리 계층으로부터 현재 연산에 필요한 입력 데이터 행렬의 성분의 데이터를 패치(fetch)할 수 있다.

[0102] 그리고, 프로세서(110)는 해당 엘레멘트 아이디(141) 및 해당 데이터가 패치된 레지스터(150)의 제2 목적 레지스터 주소를 로드 기록 버퍼(140)에 기록하고, 기록된 제2 목적 레지스터 주소(142)가 현재 연산에 필요한 입력 데이터 행렬의 성분의 제1 목적 레지스터 주소(501)와 대응되도록 레지스터 매핑 테이블(121)를 갱신할 수 있다.

[0103] 예를 들어, 도5의 (d)를 참조하면, 현재 연산에 필요한 입력 데이터 성분의 엘레멘트 아이디(141)가 '2'이고, 해당 성분의 제1 목적 레지스터 주소(501)는 'r4'일 수 있다. 이때, 엘레멘트 아이디(141) '2'와 제2 목적 레지스터 주소(142)인 'p2'가 로드 기록 버퍼(140)에 기록되어 있지 않은 경우, 메모리(120)에 접근하여 메모리 계층으로부터 제1 목적 레지스터 주소 'r4'의 데이터를 패치할 수 있다(L1$).

[0104] 그리고, 프로세서(110)는 엘레멘트 아이디 '2'및 패치된 데이터가 저장된 레지스터(150)의 제2 목적 레지스터 주소(142) 'p2'를 로드 기록 버퍼(140)에 기록할 수 있다.

[0105] 도 6은 일 실시예에 따라 원본 입력 데이터 행렬을 필터 행렬과 합성곱 연산을 수행하는 것을 설명하기 위한 도면이다.

[0106] 도6을 참조하면, 원본 입력 데이터 행렬(201)이 4×4 크기의 행렬이고, 필터 행렬(300)이 3×3 크기의 행렬인 합성곱 연산 과정의 예시를 확인할 수 있다.

[0107] 이때, 한번의 합성곱 연산 과정에서 4번의 행렬 연산이 진행되고, 2×2 크기의 출력 데이터 행렬(400)이 출력됨을 확인할 수 있다. 즉, 한번의 합성곱 연산을 위해서 행렬곱 연산이 여러 번 발생함을 알 수 있다. 따라서 연산 회수를 줄이기 위하여 한번의 행렬곱 연산을 통해 출력 데이터 행렬(400)을 출력할 수 있도록 원본 입력 데이터 행렬(201)을 변환하는 것이 필요할 수 있다.

[0108] 출력 데이터 행렬(400)은 원본 입력 데이터 행렬(201)을 입력값으로 했을 때의 합성곱 연산으로 인한 결과값일 수 있다. 즉, 특징 데이터 행렬(401)은 출력 데이터 행렬(400)의 성분이 순서대로 나열되어 열이 1개가 되도록 구성된 행렬일 수 있다.

[0109] 도 7은 일 실시예에 따라 워크스페이스로 변환된 입력 데이터 행렬을 나타낸 도면이다.

[0110] 도 7을 참조하면, 프로세서(110)는 합성곱을 GEMM 연산으로 수행하기 위하여 원본 입력 데이터 행렬(201)의 크기, 성분의 개수 및 성분의 순서를 워크스페이스에 해당하는 메모리 영역으로 변경하여 워크스페이스로 변환된 입력 데이터 행렬(202)을 생성할 수 있다.

[0111] 이 경우, 프로세서(110)는 원본 입력 데이터 행렬(201)을 특징 데이터 행렬(401)의 행의 개수와 동일한 행의 개수를 가지고, 필터 행렬(300)의 크기와 동일한 열의 개수를 가지는 워크스페이스로 변환하여 입력 데이터 행렬

(202)을 생성할 수 있다.

[0112] 예를 들어, 특징 데이터 행렬(401)의 행의 개수가 4이고, 필터 행렬(300)의 크기가 9이면, 워크스페이스로 변환된 입력 데이터 행렬(202)은 행의 개수가 4이고, 열의 개수가 9일 수 있다.

[0113] 이에 따라, 프로세서(110)는 필터 행렬(300)과의 한번의 행렬 연산으로 특징 데이터 행렬(401)을 출력할 수 있도록 원본 입력 데이터 행렬(201)을 워크스페이스로 변환된 입력 데이터 행렬(202)로 변환할 수 있다.

[0114] 도 8은 일 실시예에 따른 입력 데이터 행렬의 성분들이 규칙을 가지고 재조합되어 배열된 행렬임을 나타내는 도면이다.

[0115] 도8을 참조하면, 워크스페이스로 변환된 입력 데이터 행렬(202)은, 메모리 영역에서 필터 행렬(300)과의 한번의 행렬 연산으로 특징 데이터 행렬(401)을 출력할 수 있도록, 원본 입력 데이터 행렬(201)의 성분들이 규칙을 가지고 재조합되어 배열된 행렬일 수 있다.

[0116] 구체적으로, 도8의 워크스페이스로 변환된 입력 데이터 행렬(202)을 참조하면, 행렬에서의 1행 2열의 성분과 2행 1열의 성분이 서로 중복된 중복 성분이고, 1행 3열의 성분과 2행 2열의 성분이 서로 중복된 중복 성분일 수 있다.

[0117] 또한, 1행 8열의 성분, 2행 7열의 성분, 3행 5열의 성분 및 4행 4열의 성분이 서로 중복된 중복 성분일 수 있다.

[0118] 이처럼 워크스페이스로 변환된 입력 데이터 행렬(202)은 중복 성분들이 특정 규칙을 가지고 배열된 행렬이므로, 해당 규칙을 이용한다면 중복된 성분들에 대하여 동일한 엘레멘트 아이디(141)를 생성하는 것이 가능할 수 있다.

[0119] 다시 말해, 비록 워크스페이스로 변환된 입력 데이터 행렬(202) 상에서는 다른 위치에 위치한 서로 다른 성분이더라도, 원본 입력 데이터 행렬(201)에서 본래 같은 성분이었다면 해당 중복 성분들은 워크스페이스로 변환된 입력 데이터 행렬(202) 상에서 특정 규칙을 가지고 배열될 것이다. 이때, 해당 중복 성분들에 동일한 엘레멘트 아이디(141)를 생성하고, 해당 중복 성분들에 저장된 데이터를 동일한 레지스터(150)에서 불러올 수 있다면 연산량을 줄이고, 에너지를 절약하는 것이 가능할 수 있다.

[0120] 도 9는 일 실시예에 따른 입력 데이터 행렬의 성분들이 규칙을 가지고 배열된 행렬임을 나타내는 또 다른 도면이다.

[0121] 패치(Patch)는 워크스페이스로 변환된 입력 데이터 행렬(202)의 부분 행렬일 수 있다.

[0122] 도9를 참조하면, 워크스페이스로 변환된 입력 데이터 행렬(202)은 6개의 2×3크기의 행렬인 패치들로 구성될 수 있다. 이때,

[0123] 하나의 패치는 원본 입력데이터 행렬(201)의 어느 한 행의 구성 성분들이 재배열된 것일 수 있다.

[0124] 예를 들어, 원본 입력 데이터 행렬(201)의 1행의 성분 데이터가 순서대로 '3', '1,' '4', '-2'이면 첫 번째 패치의 성분 데이터는 '3', '1', '4', '1', '4', '-2'일 수 있다.

[0125] 또한, 원본 입력 데이터 행렬(201)의 3행의 성분 데이터가 순서대로 '4', '-2', '4', '0'이면 세 번째 패치의 성분 데이터는 '4', '-2', '4', '-2', '4', '0'일 수 있다.

[0126] 한편, 원본 입력 데이터 행렬(201)의 3행의 성분은 세번째 패치의 성분에만 나타나는 것은 아닐 수 있다. 도면을 참조하면, 다섯 번째 패치의 성분에도 원본 입력 데이터 행렬(201)의 3행의 성분이 나타나는 것을 확인할 수 있다.

[0127] 마찬가지로, 도면을 참조하면, 두 번째 패치의 성분과 네 번째 패치의 성분이 서로 대응되고, 이는 원본 입력 데이터 행렬(201)의 2행의 성분들이 재배열된 것임을 알 수 있다.

[0128] 이처럼, 워크스페이스로 변환된 입력 데이터 행렬(202)은 필터 행렬(300)과의 한번의 행렬 연산으로 특징 데이터 행렬(401)을 출력할 수 있도록, 복수개의 패치들이 규칙을 가지고 배열된 행렬일 수 있다.

[0129] 따라서, 프로세서(110)는 이와 같은 규칙에 기초하여 입력 데이터 행렬(202)의 복수의 성분들 중 서로 중복되는 데이터를 나타내는 중복 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0130] 구체적으로, 프로세서(110)는 워크스페이스로 변환된 입력 데이터 행렬(202)의 복수의 성분들의 식별자를 생성할 수 있다. 그리고, 복수의 성분들 중 동일한 식별자가 생성된 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0131] 이를 위해, 아이디 생성기(130)는 입력 데이터 행렬(202)의 복수의 성분들의 어레이 인덱스, 필터 행렬의 행의 개수 및 열의 개수 및 출력 데이터 행렬의 열의 개수에 기초하여 입력 데이터 행렬(202)의 복수의 성분들의 패치 아이디를 생성할 수 있다.

[0132] 그리고, 아이디 생성기(130)는 패치 아이디 및 입력 데이터 행렬(202)의 성분의 오프셋에 기초하여 복수의 성분들의 엘레멘트 아이디를 생성할 수 있다.

[0133] 여기에서, 패치 아이디는, 상기 입력 데이터 행렬의 열의 개수 및 채널의 개수에 기초하여 결정되는 값일 수 있다.

[0134] 그리고, 어레이 인덱스는, 상기 복수의 성분들이 단일 차원 어레이로 배열되었을 때 각 성분들의 위치를 나타내는 값일 수 있다.

[0135] 이하에서는, 아이디 생성기(130)가 입력 데이터 행렬(202)의 복수의 성분들에 대하여 엘레멘트 아이디(141)를 생성하는 방법을 자세히 설명한다.

[0136] 도 10은 일 실시예에 따라 생성된 입력 데이터 행렬(202)의 복수의 성분들의 패치 아이디를 설명하기 위한 도면이다. 여기에서, 필터 행렬(300)과 출력 데이터 행렬(400)은 도 6과 같은 것으로 가정하겠다.

[0137] 아이디 생성기(130)는 입력 데이터 행렬(202)의 복수의 성분들의 어레이 인덱스, 필터 행렬(300)의 행의 개수 및 열의 개수 및 출력 데이터 행렬(400)의 열의 개수에 기초하여 입력 데이터 행렬(202)의 복수의 성분들의 패치 아이디(600)를 생성할 수 있다.

[0138] 이를 위해, 아이디 생성기(130)는 입력 데이터 행렬(202)의 복수의 성분들의 어레이 인덱스를 할당할 수 있다.

[0139] 예를 들어, 도 10a와 같이, 입력 데이터 행렬(202)의 1행 1열에서 1행 9열까지 성분들에 대하여 0 에서 8의 어레이 인덱스 값이 순차적으로 할당되고, 2행 1열부터 2행 9열까지 성분들에 대하여 9 에서 17의 어레이 인덱스 값이 순차적으로 할당되고, 3행 1열부터 9열까지 성분들에 대하여 18 에서 26의 어레이 인덱스 값이 순차적으로 할당되고, 4행 1열부터 9열까지 성분들에 대하여 27 에서 35의 어레이 인덱스 값이 순차적으로 할당될 수 있다.

[0140] 그리고, 아이디 생성기(130)는 입력 데이터 행렬(202)의 복수의 성분들의 행 요소(1010) 및 열 요소(1020)를 산출할 수 있다.

[0141] 입력 데이터 행렬(202)의 성분의 행 요소(1010)는 해당 성분이 위치한 어레이 인덱스 값을 필터 행렬(300)의 크기(필터 행렬(300)의 열의 개수 x 필터 행렬(300)의 행의 개수)로 나누었을 때 출력되는 몫일 수 있다.

[0142] 예를 들어, 도 10a와 같이, 입력 데이터 행렬(202)의 2행 3열에 위치한 성분의 어레이 인덱스 값은 11일 수 있다. 또한, 4행 5열에 위치한 성분의 어레이 인덱스 값은31일 수 있다.

[0143] 이 때, 필터 행렬(300)의 크기는 9이므로, 2행 3열에 위치한 성분의 행 요소는 11을 9로 나누었을 때 출력되는 몫인 1일 수 있으며, 4행 5열에 위치한 성분의 행 요소(1010)는 31을 9로 나누었을 때 출력되는 몫인 3일 수 있다.

[0144] 입력 데이터 행렬(202)의 성분의 열 요소(1020)는 해당 성분이 위치한 어레이 인덱스 값을 필터 행렬(300)의 크기(필터 행렬(300)의 열의 개수 x 필터 행렬(300)의 행의 개수)로 나누었을 때 출력되는 나머지일 수 있다.

[0145] 예를 들어, 입력 데이터 행렬(202)의 2행 3열에 위치한 성분의 어레이 인덱스 값은 11일 수 있다. 또한, 4행 5열에 위치한 성분의 어레이 인덱스 값은31일 수 있다.

[0146] 이 때, 필터 행렬(300)의 크기는 9이므로, 2행 3열에 위치한 성분의 열 요소는 11을 9로 나누었을 때 출력되는 나머지인 2일 수 있으며, 4행 5열에 위치한 성분의 열 요소(1020)는 31을 9로 나누었을 때 출력되는 나머지인 4일 수 있다.

[0147] 그리고, 아이디 생성기(130)는 입력 데이터 행렬(202)의 성분마다 제1 패치 요소 및 제2 패치 요소를 산출할 수 있다.

[0148] 제1 패치 요소는 입력 데이터 행렬(202)의 성분의 행 요소를 출력 데이터 행렬(400)의 열의 개수로 나누었을 때

출력되는 몫일 수 있다.

[0149] 예를 들어, 도 10a와 같이, 입력 데이터 행렬(202)에서 2행 3열에 위치한 성분의 행 요소는 1일 수 있다. 또한, 4행 5열에 위치한 성분의 행 요소는 3일 수 있다.

[0150] 이때, 출력 데이터 행렬(400)의 열의 개수는 2이므로, 2행 3열에 위치한 성분의 제1 패치 요소는 1을 2로 나누었을 때 출력되는 몫인 0일 수 있으며, 4행 5열에 위치한 성분의 제1 패치 요소는 3을 2로 나누었을 때 출력되는 몫인 1일 수 있다.

[0151] 제2 패치 요소는 입력 데이터 행렬(202)의 성분의 열 요소를 필터 행렬(300)의 열의 개수로 나누었을 때 출력되는 몫일 수 있다.

[0152] 예를 들어, 도 10a의 입력 데이터 행렬(202)에서 2행 3열에 위치한 성분의 열 요소는 2일 수 있다. 또한, 4행 5열에 위치한 성분의 열 요소는 4일 수 있다.

[0153] 이때, 필터 행렬(300)의 열의 개수는 3이므로, 2행 3열에 위치한 성분의 제2 패치 요소는 2를 3으로 나누었 때 출력되는 몫인 0일 수 있다. 4행 5열에 위치한 성분의 제2 패치 요소는 4를 3으로 나누었을 때 출력되는 몫인 1일 수 있다.

[0154] 그리고, 아이디 생성기(130)는 제1 패치 요소에 필터 행렬(300)의 스트라이드를 곱한 값에 제2 패치 요소를 더하여 패치 아이디(600)를 생성할 수 있다.

[0155] 예를 들어, 필터 행렬(300)의 스트라이드가 1인 것으로 가정하면, 도 10b와 같이, 입력 데이터 행렬(202)에서 2행 3열에 위치한 성분의 패치 아이디(600)는 0에 1을 곱한 값에 0을 더한 값인 0일 수 있다. 또한, 4행 5열에 위치한 성분의 패치 아이디(600)는 1에 1을 곱한 값에 1을 더한 값인 2일 수 있다. 한편, 전술한 방식은 입력 데이터 행렬(202)의 어느 성분에 대하여 패치 아이디(600)를 구할 수 있는 하나의 예시일 뿐이며, 전혀 다른 방식으로 패치 아이디(600)가 생성되더라도 이러한 패치 아이디(600)를 이용하여 중복 성분들에 대하여 동일한 엘레멘트 아이디(141)를 생성할 수 있다면 문제없다.

[0156] 도 11은 일 실시 예에 따라 생성된 입력 데이터 행렬(202)의 복수의 성분들의 엘레멘트 아이디를 설명하기 위한 도면이다. 여기에서, 원본 데이터 행렬(201), 필터 행렬(300) 및 출력 데이터 행렬(400)은 도 6과 같고, 패치 아이디(600)는 도 10과 같이 생성된 것으로 가정한다.

[0157] 아이디 생성기(130)는 패치 아이디(600)에 기초하여, 중복된 데이터를 나타내는 입력 데이터 행렬(202)의 성분들에 동일한 엘레멘트 아이디(141)를 생성할 수 있다.

[0158] 구체적으로, 아이디 생성기(130)는 입력 데이터 행렬(202)의 성분의 오프셋에 기초하여, 중복된 데이터를 나타내는 입력 데이터 행렬(202)의 성분들에 동일한 엘레멘트 아이디(141)를 생성할 수 있다.

[0159] 여기에서, 입력 데이터 행렬(202)의 성분의 오프셋은 입력 데이터 행렬(202)의 성분의 패치 아이디(600)에 원본 입력 데이터 행렬(201)의 열의 개수 및 채널의 개수를 곱한 값일 수 있다.

[0160] 예를 들어, 원본 입력 데이터 행렬(201)의 채널의 개수가 1인 것으로 가정하면, 도 10b와 같이, 입력 데이터 행렬(202)에서, 2행 3열에 위치한 성분의 패치 아이디(600)가 0이므로, 2행 3열에 위치한 성분의 오프셋은 0에 입력 데이터 행렬(200)의 열의 개수인 4 및 채널의 개수 1을 곱한 값인 0일 수 있다.

[0161] 또한, 입력 데이터 행렬(202)에서, 4행 5열에 위치한 성분의 패치 아이디(600)가 2이므로, 4행 5열에 위치한 성분의 오프셋은 2에 입력 데이터 행렬(200)의 열의 개수인 4 및 채널의 개수 1을 곱한 값인 8일 수 있다.

[0162] 그리고, 아이디 생성기(130)는 입력 데이터 행렬(202)의 성분의 오프셋에 입력 데이터 행렬(202)의 성분의 행 요소(1010)를 출력 데이터 행렬(400)의 열의 개수에 채널의 개수 및 스트라이드를 곱한 값으로 나누었을 때 출력되는 나머지 값과, 입력 데이터 행렬(202)의 성분의 열 요소(1020)를 필터 행렬(300)의 열의 개수에 채널의 개수를 곱한 값으로 나누었을 때 출력되는 나머지 값을 더하여 입력 데이터 행렬(202)의 성분의 엘레멘트 아이디(141)를 생성할 수 있다.

[0163] 예를 들어, 도 11을 참조하면, 행렬(300)의 스트라이드 및 원본 입력 데이터 행렬(201)의 채널의 개수가 1인 것으로 가정하면, 2행 3열의 성분의 행 요소 1을 출력 데이터 행렬(400)의 열의 개수 2에 채널의 개수 1 및 스트라이드 1을 곱한 값인 2로 나누었을 때 출력되는 나머지 값은 1일 수 있다.

[0164] 그리고, 2행 3열의 성분의 열 요소 2를, 필터 행렬(300)의 열의 개수 3에 채널의 개수 1을 곱한 값인 3으로 나

누었을 때 출력되는 나머지 값은 2일 수 있다.

[0165] 이때, 2행 3열의 성분의 오프셋은 0이므로, 아이디 생성기(130)는 2행 3열의 성분에 대하여, 0에 1과 2를 더한 '3'으로 엘레멘트 아이디(141)를 생성할 수 있다.

[0166] 또한, 도 11을 참조하면, 4행 5열의 성분의 행 요소 3을, 출력 데이터 행렬(400)의 열의 개수 2에 채널의 개수 1 및 스트라이드 1을 곱한 값인 2로 나누었을 때 출력되는 나머지 값은 1일 수 있다.

[0167] 그리고, 4행 5열의 성분의 열 요소 4를, 필터 행렬(300)의 열의 개수 3에 채널의 개수 1을 곱한 값인 3으로 나누었을 때 출력되는 나머지 값은 1일 수 있다.

[0168] 이때, 4행 5열의 성분의 오프셋은 8이므로, 아이디 생성기(130)는 4행 5열의 성분에 대하여, 8에 1과 1을 더한 '10'으로 엘레멘트 아이디(141)를 생성할 수 있다.

[0169] 그리고, 프로세서(110)는 입력 데이터 행렬(202)의 복수의 성분들 중 동일한 엘레멘트 아이디가 생성된 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0170] 이 경우, 아이디 생성기(130)에 의해 생성된 복수의 성분들의 엘레멘트 아이디(141)는 중복 성분들이 동일한 값을 가지게 된다는 점에서, 결과적으로, 중복 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블(121)이 갱신될 수 있다.

[0171] 그리고, 프로세서(110)는 이와 같이 갱신되는 레지스터 매핑 테이블(121)에 기초하여 중복 성분들에 대해 동일한 제2 목적 레지스터 주소를 가지는 레지스터를 재사용하여 합성곱 연산을 수행할 수 있다.

[0172] 한편, 전술한 실시 예는 단일 배치의 합성곱이 수행되는 경우에 프로세서(110)가 레지스터 매핑 테이블(121)을 갱신하는 방법이 설명되었으나, 본 개시의 일 실시 예에 따르면, 다중 배치의 인자를 가지는 합성곱이 수행되더라도 프로세서(110)는 중복 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0173] 이를 위해, 아이디 생성기(130)는 입력 데이터 행렬(202)의 복수의 성분들의 엘레멘트 아이디 및 배치 아이디를 생성할 수 있다.

[0174] 그리고, 아이디 생성기(130)는 복수의 성분들 중 동일한 엘레멘트 아이디 및 배치 아이디가 생성된 성분들의 제1 목적 레지스터 주소들이 동일한 제2 목적 레지스터 주소에 대응하도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0175] 도 12는 또 다른 일 실시 예에 따른 아이디 생성기 및 로드 기록 버퍼의 구성을 나타낸 도면이다.

[0176] 도 12를 참조하면, 아이디 생성기(130)는 현재 행렬 연산에 필요한 입력 데이터 행렬의 성분의 엘레멘트 아이디(1210) 및 배치 아이디(1220)를 생성할 수 있다. 그리고, 로드 기록 버퍼(140)에는 생성된 엘레멘트 아이디(1210) 및 배치 아이디(1220)에 대응되는 제2 목적 레지스터 주소(142)가 기록될 수 있다.

[0177] 그리고, 프로세서(110)는 현재 연산에 필요한 입력 데이터 행렬의 성분의 엘레멘트 아이디(1210) 및 배치 아이디(1220)와 이에 대응되는 제2 목적 레지스터 주소(501)가 로드 기록 버퍼(140)에 기록되어 있는지 식별할 수 있다.

[0178] 프로세서(110)는 현재 연산에 필요한 입력 데이터 행렬의 성분의 엘레멘트 아이디(1210) 및 배치 아이디(1220)와 이에 대응되는 제2 목적 레지스터 주소(142)가 로드 기록 버퍼(140)에 기록되어 있는 경우, 해당 제2 목적 레지스터 주소(142)가 해당 성분의 제1 목적 레지스터 주소(501)에 대응되도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0179] 이에 따라, 프로세서(110)는 현재 행렬 연산에 필요한 입력 데이터 행렬의 성분의 데이터를 로드하기 위하여, 메모리(120)에 접근하지 않고 로드 기록 버퍼(140)에 기록된 제2 목적 레지스터 주소(142)를 가지는 레지스터(150)에 저장된 데이터를 재사용할 수 있다.

[0180] 반면, 프로세서(110)는 현재 연산에 필요한 입력 데이터 성분의 엘레멘트 아이디(1210) 및 배치 아이디(1220)와 이에 대응되는 제2 목적 레지스터 주소(142)가 로드 기록 버퍼(140)에 기록되어 있지 않은 경우, 메모리(120)에 접근하여 메모리 계층으로부터 제1 목적 레지스터 주소(501)의 성분의 데이터를 패치할 수 있다.

[0181] 그리고, 프로세서(110)는 해당 엘레멘트 아이디(1210) 및 배치 아이디(1220)와 해당 데이터가 패치된 레지스터

(150)의 제2 목적 레지스터 주소(142)를 로드 기록 버퍼(140)에 기록하고, 기록된 제2 목적 레지스터 주소(142)가 해당 성분의 제1 목적 레지스터 주소(501)와 대응되도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0182] 이를 위해, 아이디 생성기(130)는 입력 데이터 행렬(202)의 복수의 성분들의 엘레멘트 아이디(1210) 및 배치 아이디(1220)를 생성할 수 있다.

[0183] 여기에서, 아이디 생성기(130)가 입력 데이터 행렬(202)의 복수의 성분들의 어레이 인덱스를 할당하는 방법, 행 요소 및 열 요소를 산출하는 방법 및 엘레멘트 아이디(1210)를 생성하는 방법에 대한 설명은 도 6 내지 도 10에서 설명한 내용과 중복된다는 점에서, 생략하도록 한다.

[0184] 아이디 생성기(130)는 입력 데이터 행렬(202)의 복수의 성분의 어레이 인덱스, 필터 행렬(300)의 행의 개수 및 열의 개수 및 출력 데이터 행렬(400)의 행의 개수 및 열의 개수에 기초하여 입력 데이터 행렬(202)의 복수의 성분들의 배치 아이디(1220)를 생성할 수 있다.

[0185] 구체적으로, 아이디 생성기(130)는 입력 데이터 행렬(202)의 행 요소를 출력 데이터 행렬(400)의 크기(출력 데이터 행렬(400)의 열의 개수에 출력 데이터 행렬(400)의 행의 개수를 곱한 값)로 나누었을 때 출력되는 몫을 배치 아이디(1220)로 생성할 수 있다.

[0186] 그리고, 프로세서(110)는 입력 데이터 행렬(202)의 복수의 성분들 중 동일한 엘레멘트 아이디가 생성된 성분들의 제1 목적 레지스터 주소(501)들이 동일한 제2 목적 레지스터 주소(142)에 대응하도록 레지스터 매핑 테이블(121)을 갱신할 수 있다.

[0187] 이 경우, 아이디 생성기(130)에 의해 생성된 복수의 성분들의 엘레멘트 아이디(1210) 및 배치 아이디(1220)는 중복 성분들이 동일한 값을 가지게 된다는 점에서, 결과적으로, 중복 성분들의 제1 목적 레지스터 주소(501)들이 동일한 제2 목적 레지스터 주소(142)에 대응하도록 레지스터 매핑 테이블(121)이 갱신될 수 있다.

[0188] 그리고, 프로세서(110)는 이와 같이 갱신되는 레지스터 매핑 테이블(121)에 기초하여 중복 성분들에 대해 동일한 제2 목적 레지스터 주소(142)를 가지는 레지스터(150)를 재사용하여 합성곱 연산을 수행할 수 있다.

[0189] 이상에서 설명된 구성요소들의 성능에 대응하여 적어도 하나의 구성요소가 추가되거나 삭제될 수 있다. 또한, 구성요소들의 상호 위치는 시스템의 성능 또는 구조에 대응하여 변경될 수 있다는 것은 당해 기술 분야에서 통상의 지식을 가진 자에게 용이하게 이해될 것이다.

[0190] 도 13은 일 실시예에 따른 합성곱 연산 방법의 순서도이다. 이는 본 발명의 목적을 달성하기 위한 바람직한 실시 예일 뿐이며, 필요에 따라 일부 구성이 추가되거나 삭제될 수 있음은 물론이다.

[0191] 도 13을 참조하면, 프로세서(110)는 원본 입력 데이터를 워크스페이스에 해당하는 메모리 영역으로 변경할 수 있다(1310).

[0192] 이때, 적어도 하나의 프로세서(110)는, 원본 입력 데이터 행렬의 크기, 성분의 개수 및 성분의 순서를 워크스페이스에 해당하는 메모리 영역으로 변경하여 입력 데이터 행렬을 생성할 수 있다.

[0193] 구체적으로, 프로세서(110)는 원본 입력 데이터 행렬을 특징 데이터 행렬의 행의 개수와 동일한 행의 개수를 가지고, 필터 행렬의 크기와 동일한 열의 개수를 가지는 워크스페이스로 변환하여 입력 데이터 행렬을 생성할 수 있다.

[0194] 그리고, 프로세서(110)는 현재 연산에 필요한 입력 데이터 행렬의 성분의 제1 목적 레지스터 주소를 포함하는 텐서 코어 로드 데이터를 입력 받을 수 있다(1320).

[0195] 그리고, 프로세서(110)는 현재 연산에 필요한 입력 데이터 행렬의 성분의 식별자를 생성할 수 있다(1330). 이때, 식별자는 엘레멘트 아이디를 포함할 수 있다. 또한, 다중 배치 인자를 가지는 합성곱이 수행되는 경우, 식별자는 배치 아이디를 더 포함할 수 있다.

[0196] 이를 위해, 프로세서(110)는 현재 연산에 필요한 입력 데이터 행렬의 성분의 어레이 인덱스, 필터 행렬의 행의 개수 및 열의 개수 및 출력 데이터 행렬의 열의 개수에 기초하여 입력 데이터 행렬의 복수의 성분들의 패치 아이디를 생성할 수 있다.

[0197] 그리고, 프로세서(110)는 패치 아이디 및 현재 연산에 필요한 입력 데이터 행렬의 성분의 오프셋에 기초하여 엘레멘트 아이디를 생성할 수 있다.

[0198] 또한, 프로세서(110)는 현재 연산에 필요한 입력 데이터 행렬의 성분의 어레이 인덱스, 상기 필터 행렬의 행의

개수 및 열의 개수 및 출력 데이터 행렬의 행의 개수 및 열의 개수에 기초하여 배치 아이디를 생성할 수 있다.

[0199] 그리고, 프로세서(110)는 생성된 식별자 및 이에 대응되는 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있는지 식별할 수 있다(1340).

[0200] 그리고, 프로세서(110)는 생성된 식별자 및 이에 대응되는 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있는 것으로 식별되면(S1340-Y), 기록된 제2 목적 레지스터 주소가 현재 연산에 필요한 입력 데이터 행렬의 성분의 제1 목적 레지스터 주소에 대응되도록 레지스터 매핑 테이블(121)을 갱신할 수 있다(1350).

[0201] 반면, 프로세서(110)는 생성된 식별자 및 이에 대응되는 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있지 않은 것으로 식별되면(S1340-N), 메모리(120)에 접근하여 메모리 계층으로부터 현재 연산에 필요한 입력 데이터 행렬의 성분의 데이터를 패치할 수 있다(1360).

[0202] 그리고, 프로세서(110)는 생성된 식별자 및 해당 데이터가 패치된 레지스터의 제2 목적 레지스터 주소를 로드 기록 버퍼에 기록하고, 기록된 제2 목적 레지스터 주소가 현재 연산에 필요한 입력 데이터 행렬의 성분의 제1 목적 레지스터 주소에 대응되도록 레지스터 매핑 테이블을 갱신할 수 있다(1370). 그리고, 프로세서(110)는 레지스터 매핑 테이블(121)에 기초하여 제1 목적 레지스터 주소를 이에 대응되는 제2 목적 레지스터 주소로 변환할 수 있다(1380).

[0203] 이에 따라, 프로세서(110)는 입력 데이터 행렬의 복수의 성분들 중 특정 제1 목적 레지스터의 성분의 데이터가 이용되는 연산을 수행하는 경우, 해당 제1 목적 레지스터 주소가 변환된 제2 목적 레지스터 주소를 가지는 레지스터(150)로부터 해당 데이터를 획득하여 연산을 수행할 수 있다.

[0204] 본 발명의 실시예에 따른 합성곱 연산 방법의 성능을 검증하기 위하여 텐서 코어 모델과 함께 GPGPU-sim을 이용한 연구를 진행했다. 이때, GPGPU-sim은 NVIDIA Titan V로 구성된 것을 이용하였다.

[0205] 연구는 세가지의 대표적인 DNN(Deep Neural Network)에 의한 연산을 통해 이루어졌다. 구체적으로, 연구는 NVIDIA CUDA SDK 9.1의 cudaTensorCoreGemm 커널을 기반으로 구현되는 ResNet, GAN 및 YOLO에 의한 연산을 통하여 이루어졌다.

[0206] 도 14은 본 발명의 합성곱 연산 방법의 효과를 나타내는 그래프이다.

[0207] 도14의 (a)를 참조하면, Oracle상태에서 본 발명의 실시예에 따른 합성곱 연산 방법은 기존에 비하여 26%의 속도 증가 효과를 확인할 수 있다.

[0208] Oracle상태는 로드 기록 버퍼(140)의 크기가 무한이라고 가정된 상태일 수 있다. 다만, 실제로는 로드 기록 버퍼(140)의 크기가 1024-entry인 옵션으로 연구가 진행되었다. 평균적으로, 1024-entry의 로드 기록 버퍼(140)는 Oracle상태 경우의 약 4/5인 22.1%의 성능 향상을 보여주었다.

[0209] 또한, 도14의 (b)를 참조하면, Oracle상태에서 본 발명의 실시예에 따른 합성곱 연산 방법은 기존에 비하여 중복되는 텐서 코어의 로드를 최대 76%까지 줄일 수 있는 것을 확인할 수 있다.

[0210] 이때 전체 메모리 로드 중 약 3/4의 메모리 로드의 경우, 레지스터 주소가 바뀌게 되었다.

[0211] 이처럼 본 발명의 합성곱 연산 방법에 대한 연구에서는 텐서 코어 로드를 적극적으로 제거함으로써 26%의 연산 속도 증가 효과 및 34%의 에너지 절약 효과를 달성하는 것을 확인하였다.

[0212] 이상에서와 같이 첨부된 도면을 참조하여 개시된 실시예들을 설명하였다. 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 본 발명의 기술적 사상이나 필수적인 특징을 변경하지 않고도, 개시된 실시예들과 다른 형태로 본 발명이 실시될 수 있음을 이해할 것이다. 개시된 실시예들은 예시적인 것이며, 한정적으로 해석되어서는 안 된다.

부호의 설명

[0213] 100: 합성곱 연산 장치

110: 프로세서

120: 메모리

121: 레지스터 매핑 테이블

130: 아이디 생성기

140: 로드 기록 버퍼

141: 엘레멘트 아이디

142: 제2 목적 레지스터 주소

150: 레지스터

201: 원본 입력 데이터 행렬

202: 워크스페이스로 변환된 입력 데이터 행렬

300: 필터 행렬

400: 출력 데이터 행렬

401: 특징 데이터 행렬

500: 텐서 코어 로드 데이터

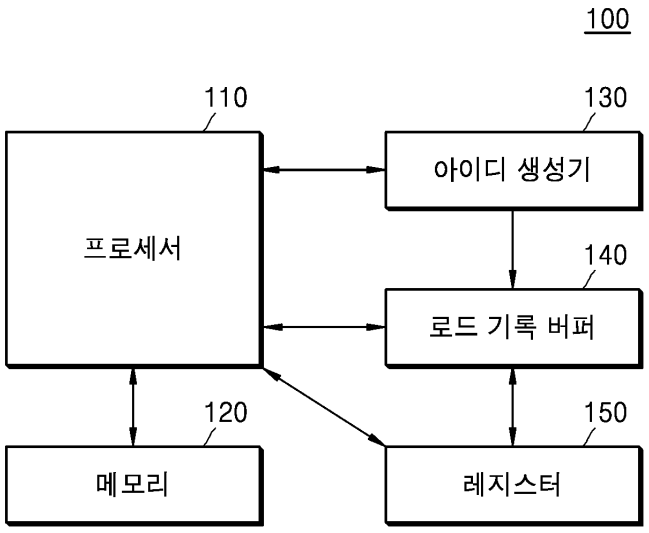
501: 제1 목적 레지스터 주소

600: 패치 아이디

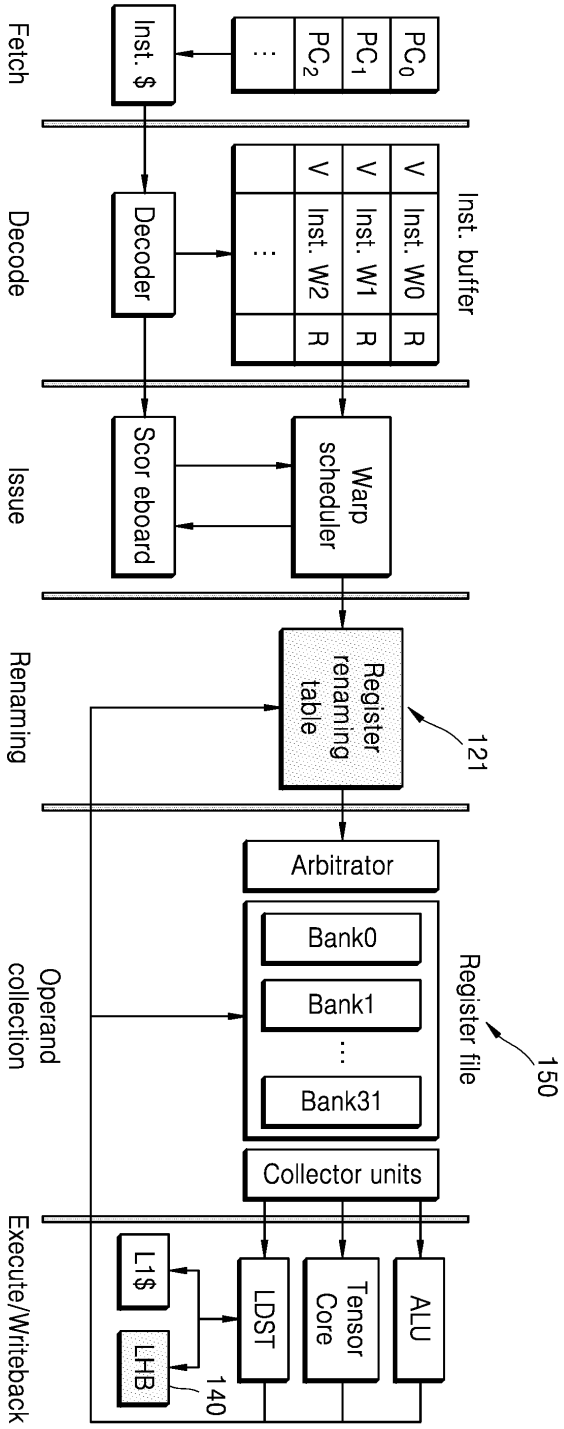
700: 어레이 인덱스

도면

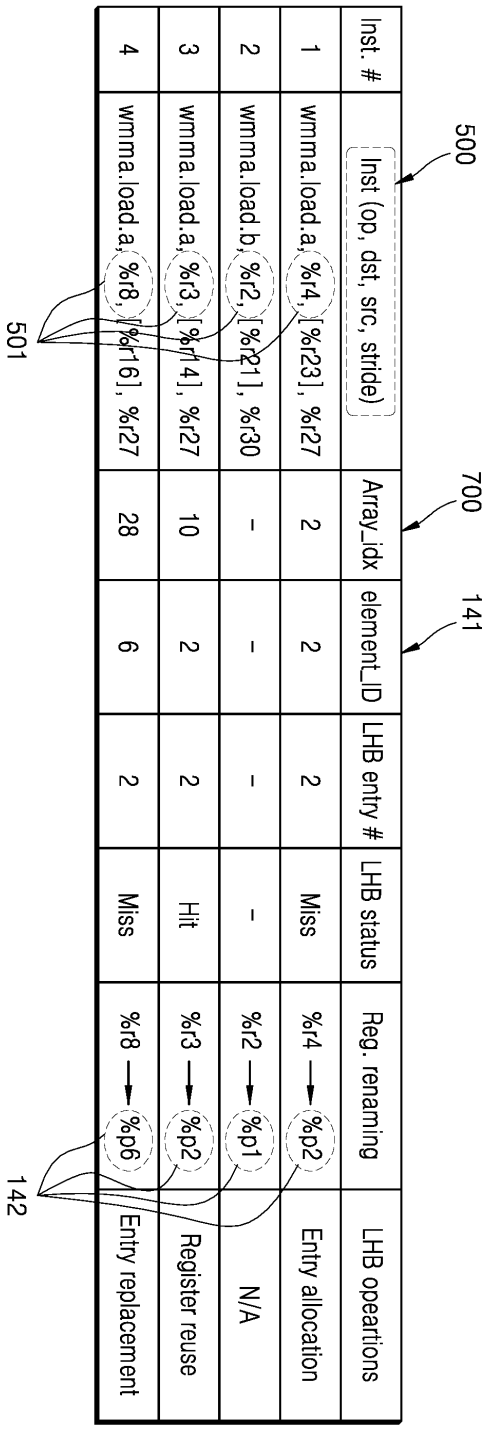
도면1



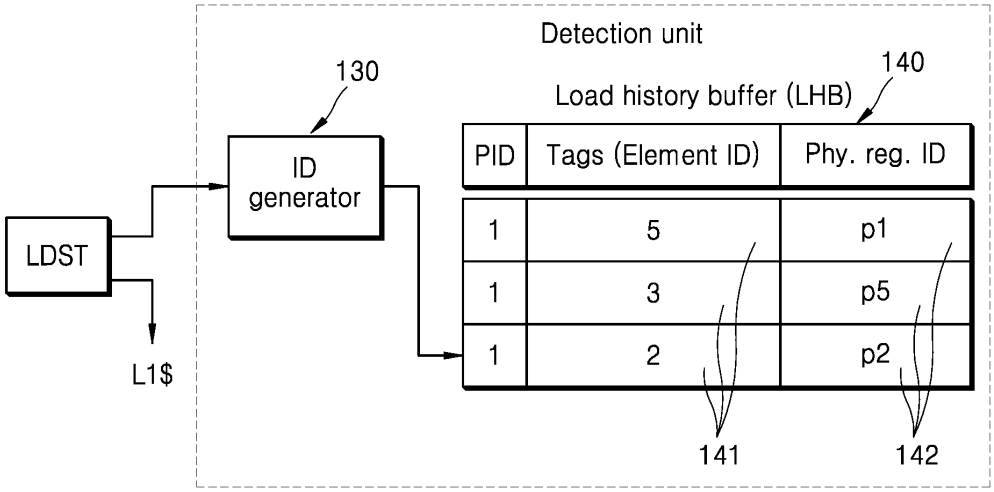
도면2


Fetch
Decode
Issue
Renaming
Operand collection
Execute/Writeback
Inst. $
PC0
PC1
PC2
Decoder
Inst. buffer
V
Inst. W0
Inst. W1
Inst. W2
R
Scor eboard
Warp scheduler
Register renaming table
121
Arbitrator
Bank0
Bank1
Bank31
Register file
150
Collector units
L1$
LHB
140
LDST
Tensor Core
ALU
100

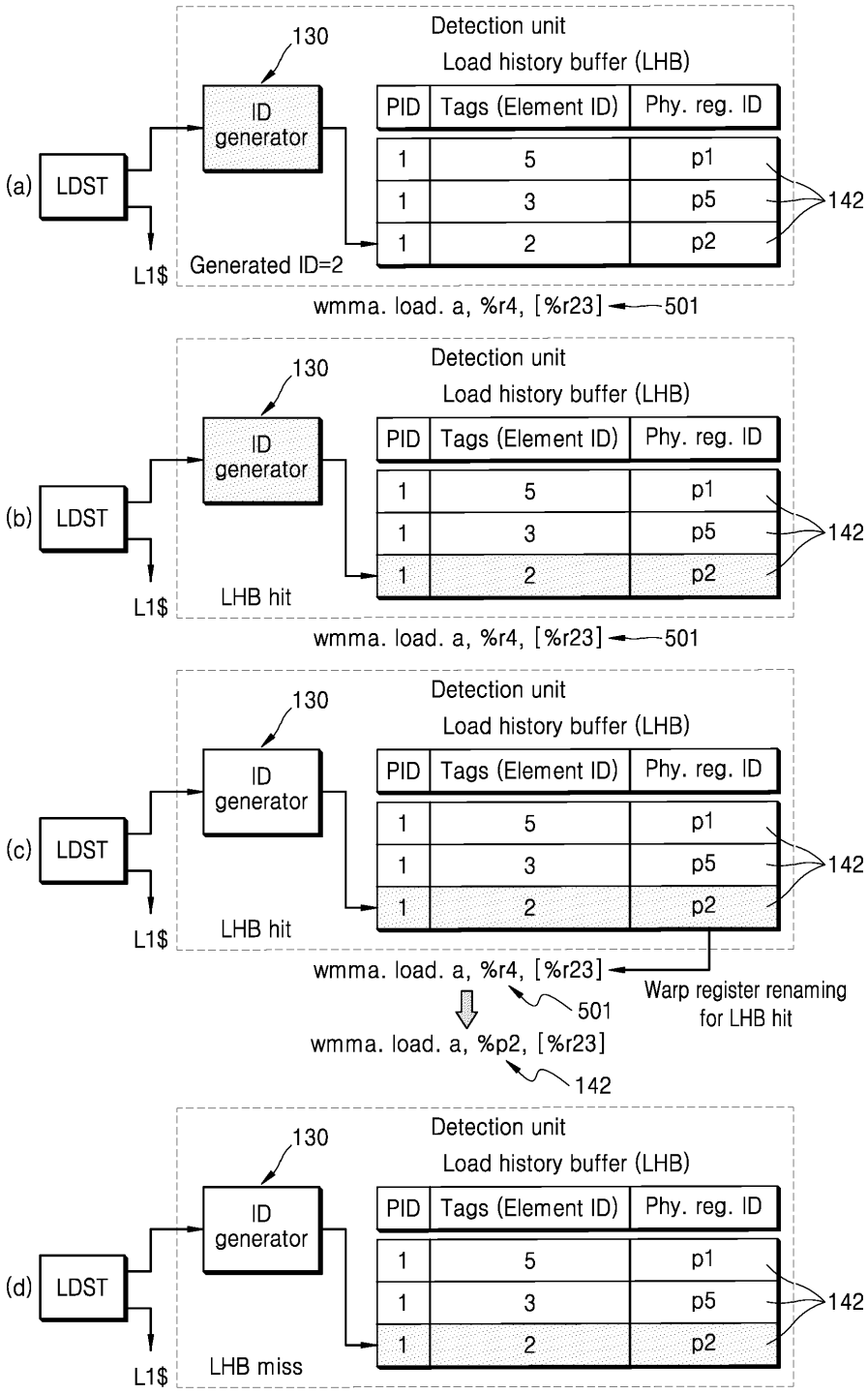
도면3


500
700
141
501
142
Inst. #
Inst (op, dst, src, stride)
Array_idx
element_ID
LHB entry #
LHB status
Reg. renaming
LHB opeartions
1
wmma.load.a, %r4, [%r23], %r27
2
2
2
Miss
%r4 → %p2
Entry allocation
2
wmma.load.b, %r2, [%r21], %r30
-
-
-
-
%r2 → %p1
N/A
3
wmma.load.a, %r3, [%r14], %r27
10
2
2
Hit
%r3 → %p2
Register reuse
4
wmma.load.a, %r8, [%r16], %r27
28
6
2
Miss
%r8 → %p6
Entry replacement

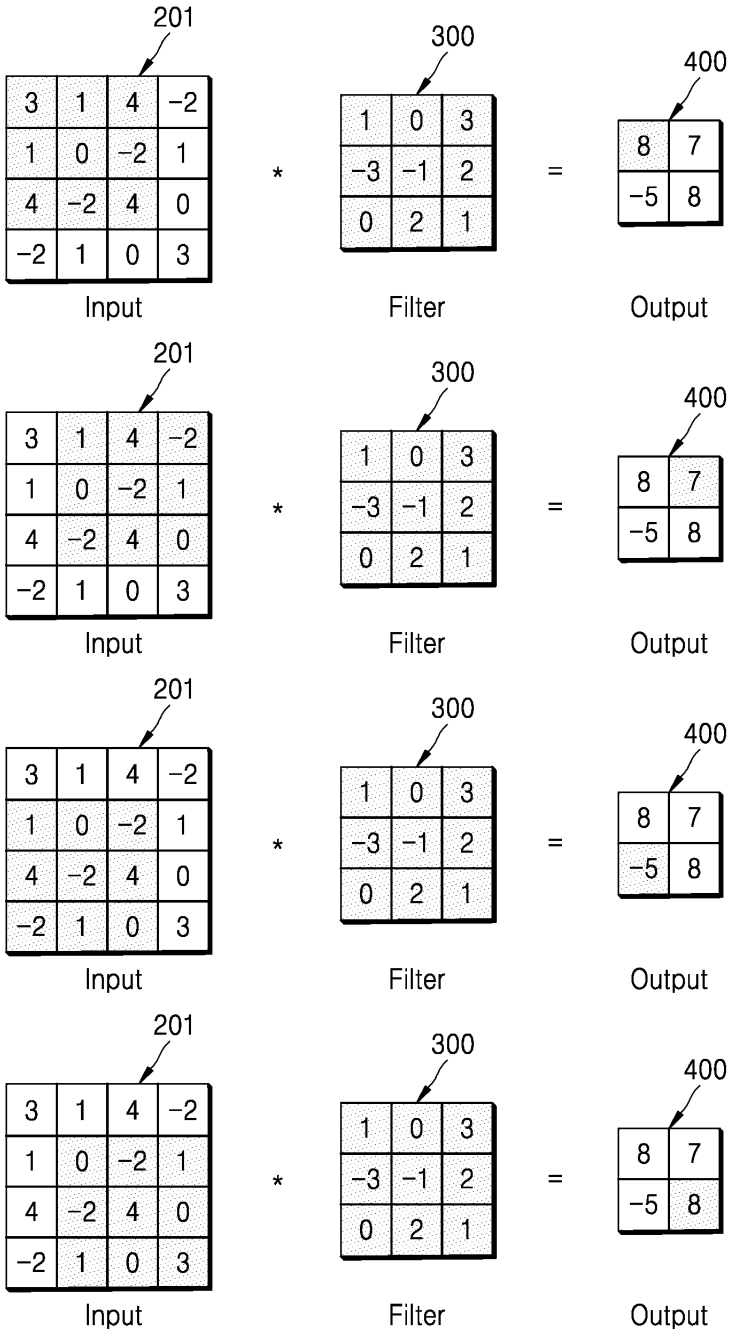
도면4


Detection unit
130
140
Load history buffer (LHB)
ID generator
LDST
L1$
PID
Tags (Element ID)
Phy. reg. ID
1
5
p1
1
3
p5
1
2
p2
141
142

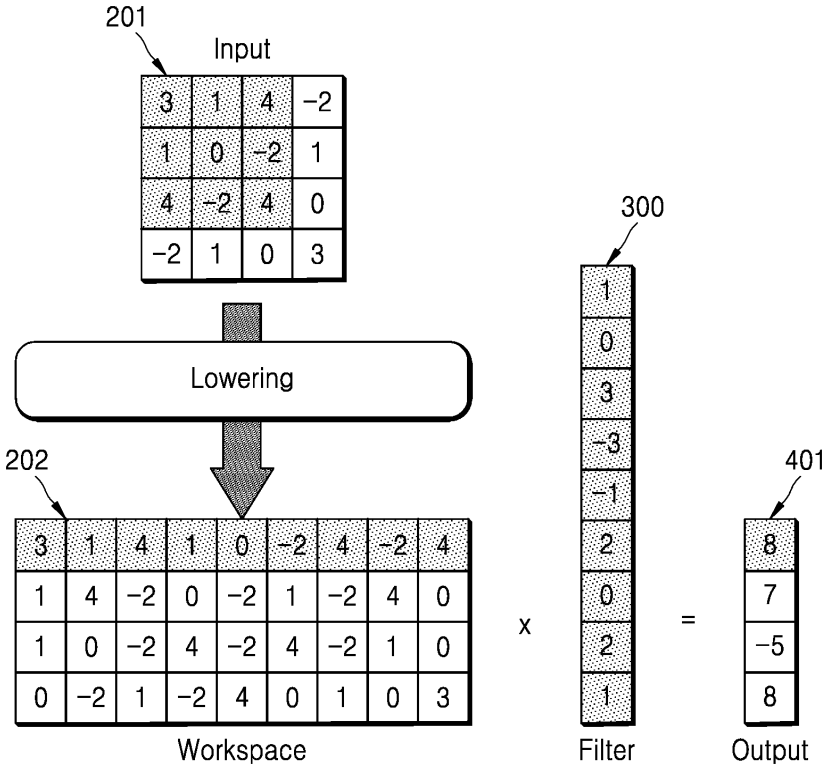
도면5


(a)
LDST
L1$
130
ID generator
Detection unit
Load history buffer (LHB)
PID
Tags (Element ID)
Phy. reg. ID
1
5
p1
1
3
p5
1
2
p2
142
Generated ID=2
wmma. load. a, %r4, [%r23]
501
(b)
LDST
L1$
130
ID generator
Detection unit
Load history buffer (LHB)
PID
Tags (Element ID)
Phy. reg. ID
1
5
p1
1
3
p5
1
2
p2
142
LHB hit
wmma. load. a, %r4, [%r23]
501
(c)
LDST
L1$
130
ID generator
Detection unit
Load history buffer (LHB)
PID
Tags (Element ID)
Phy. reg. ID
1
5
p1
1
3
p5
1
2
p2
142
LHB hit
wmma. load. a, %r4, [%r23]
501
Warp register renaming for LHB hit
wmma. load. a, %p2, [%r23]
142
(d)
LDST
L1$
130
ID generator
Detection unit
Load history buffer (LHB)
PID
Tags (Element ID)
Phy. reg. ID
1
5
p1
1
3
p5
1
2
p2
142
LHB miss

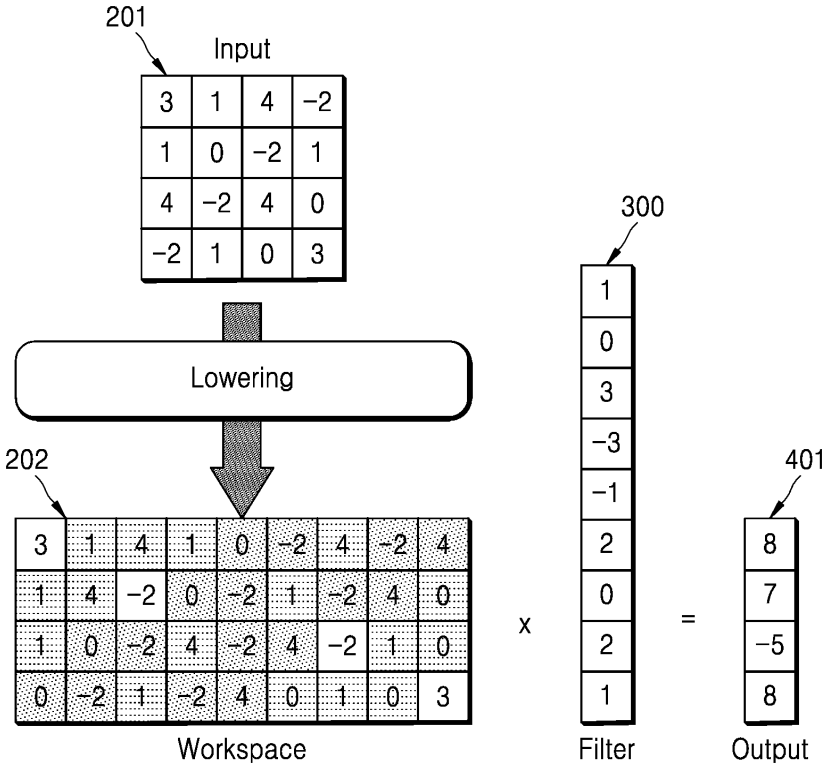
도면6


201
3 1 4 -2
1 0 -2 1
4 -2 4 0
-2 1 0 3
Input
*
300
1 0 3
-3 -1 2
0 2 1
Filter
=
400
8 7
-5 8
Output
201
3 1 4 -2
1 0 -2 1
4 -2 4 0
-2 1 0 3
Input
*
300
1 0 3
-3 -1 2
0 2 1
Filter
=
400
8 7
-5 8
Output
201
3 1 4 -2
1 0 -2 1
4 -2 4 0
-2 1 0 3
Input
*
300
1 0 3
-3 -1 2
0 2 1
Filter
=
400
8 7
-5 8
Output
201
3 1 4 -2
1 0 -2 1
4 -2 4 0
-2 1 0 3
Input
*
300
1 0 3
-3 -1 2
0 2 1
Filter
=
400
8 7
-5 8
Output

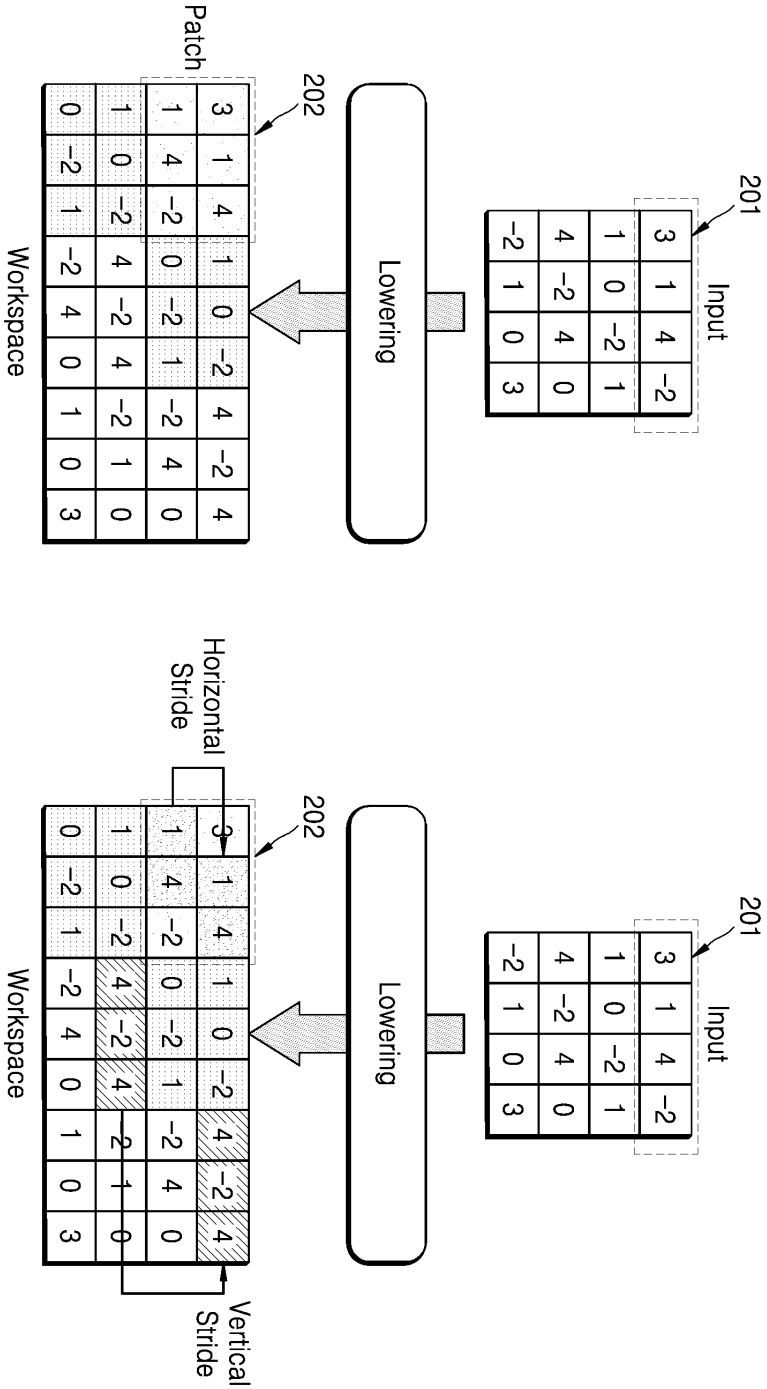
도면7


201
Input
3 1 4 -2
1 0 -2 1
4 -2 4 0
-2 1 0 3
Lowering
202
3 1 4 1 0 -2 4 -2 4
1 4 -2 0 -2 1 -2 4 0
1 0 -2 4 -2 4 -2 1 0
0 -2 1 -2 4 0 1 0 3
Workspace
x
300
1
0
3
-3
-1
2
0
2
1
Filter
=
401
8
7
-5
8
Output

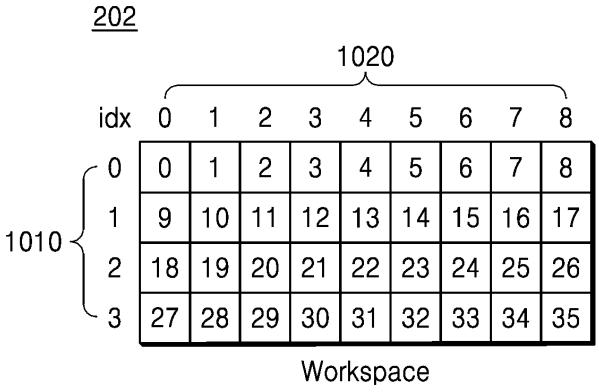
도면8


201
Input
3 1 4 -2
1 0 -2 1
4 -2 4 0
-2 1 0 3
Lowering
202
3 1 4 1 0 -2 4 -2 4
1 4 -2 0 -2 1 -2 4 0
1 0 -2 4 -2 4 -2 1 0
0 -2 1 -2 4 0 1 0 3
Workspace
x
300
1
0
3
-3
-1
2
0
2
1
Filter
=
401
8
7
-5
8
Output

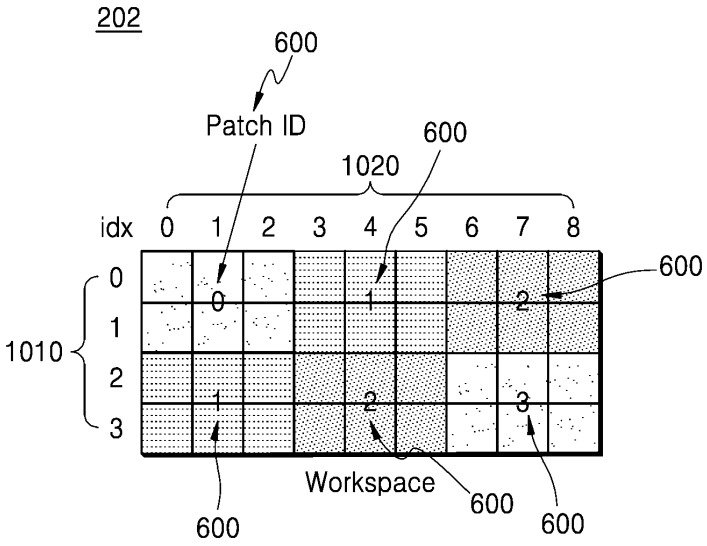
도면9


201
Input
Lowering
202
Patch
Workspace
Horizontal Stride
Vertical Stride

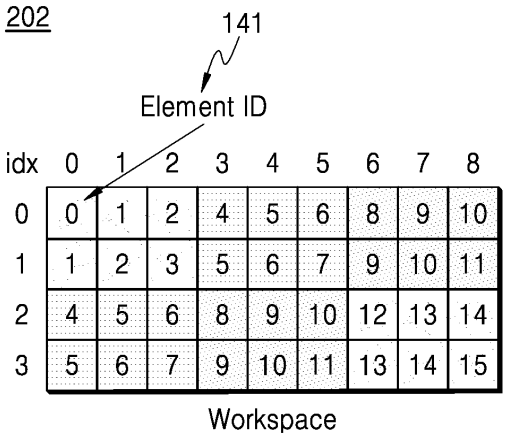
도면10a



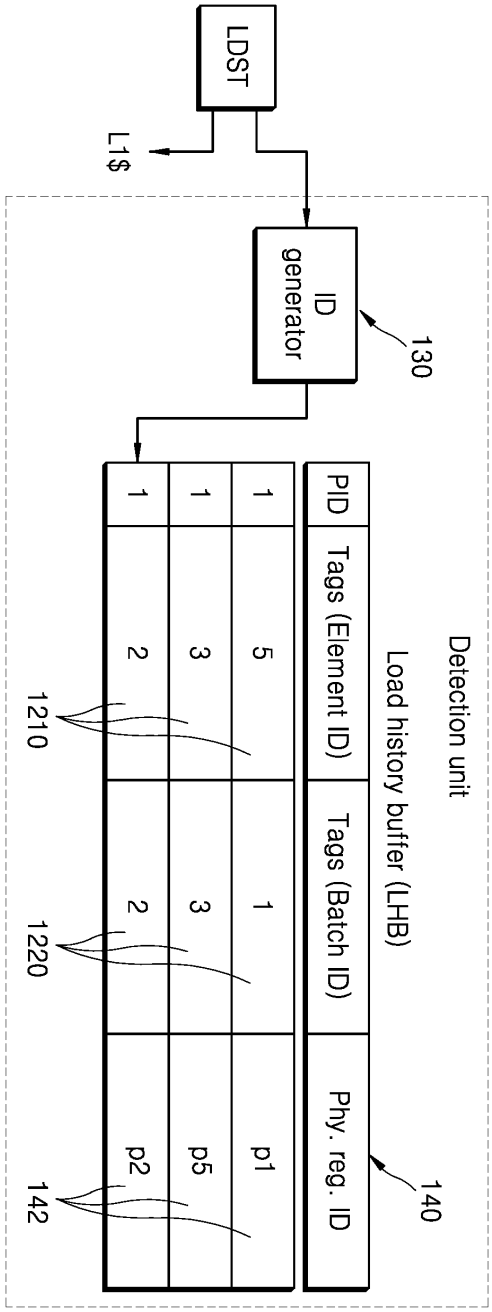
도면10b



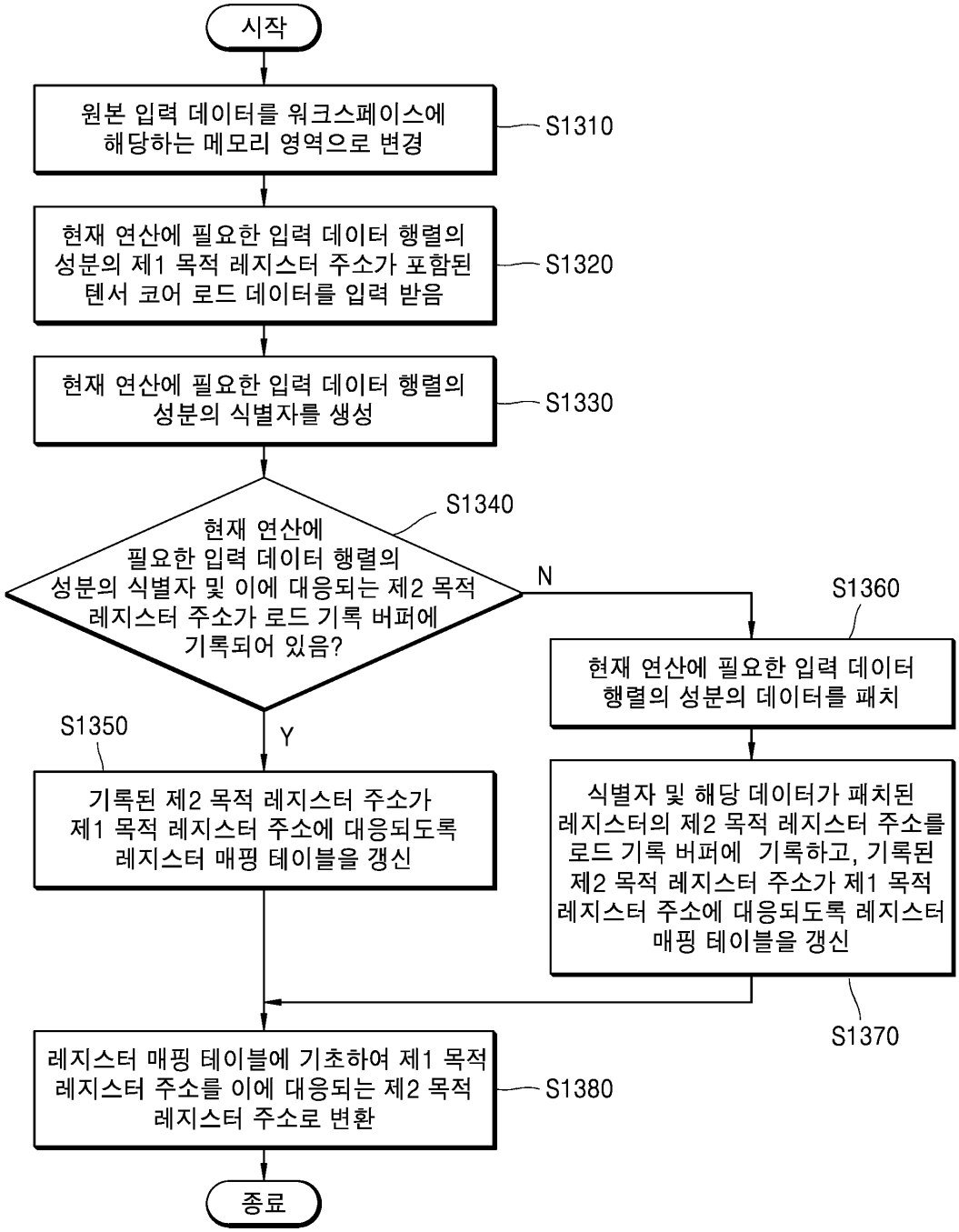
도면11



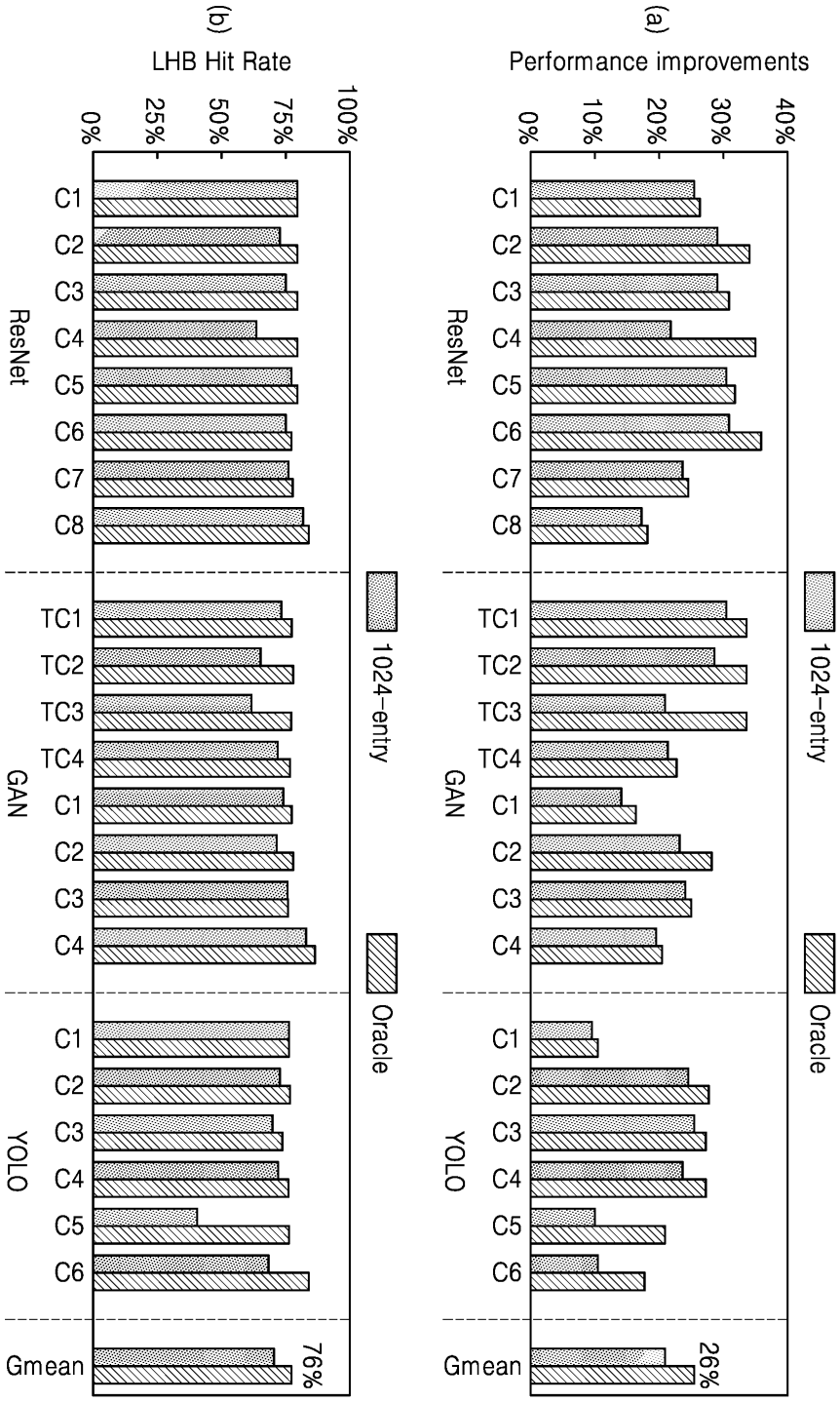
도면12


LDST
L1$
ID generator
130
Detection unit
Load history buffer (LHB)
PID
Tags (Element ID)
Tags (Batch ID)
Phy. reg. ID
140
1
1
5
p1
1
1
3
p5
1
2
2
p2
1210
1220
142

도면13


시작
원본 입력 데이터를 워크스페이스에 해당하는 메모리 영역으로 변경
S1310
현재 연산에 필요한 입력 데이터 행렬의 성분의 제1 목적 레지스터 주소가 포함된 텐서 코어 로드 데이터를 입력 받음
S1320
현재 연산에 필요한 입력 데이터 행렬의 성분의 식별자를 생성
S1330
현재 연산에 필요한 입력 데이터 행렬의 성분의 식별자 및 이에 대응되는 제2 목적 레지스터 주소가 로드 기록 버퍼에 기록되어 있음?
S1340
N
Y
S1350
기록된 제2 목적 레지스터 주소가 제1 목적 레지스터 주소에 대응되도록 레지스터 매핑 테이블을 갱신
S1360
현재 연산에 필요한 입력 데이터 행렬의 성분의 데이터를 패치
식별자 및 해당 데이터가 패치된 레지스터의 제2 목적 레지스터 주소를 로드 기록 버퍼에 기록하고, 기록된 제2 목적 레지스터 주소가 제1 목적 레지스터 주소에 대응되도록 레지스터 매핑 테이블을 갱신
S1370
레지스터 매핑 테이블에 기초하여 제1 목적 레지스터 주소를 이에 대응되는 제2 목적 레지스터 주소로 변환
S1380
종료

도면14


(a)
Performance improvements
0%
10%
20%
30%
40%
ResNet
C1
C2
C3
C4
C5
C6
C7
C8
GAN
TC1
TC2
TC3
TC4
C1
C2
C3
C4
YOLO
C1
C2
C3
C4
C5
C6
Gmean
26%
1024-entry
Oracle
(b)
LHB Hit Rate
0%
25%
50%
75%
100%
ResNet
C1
C2
C3
C4
C5
C6
C7
C8
GAN
TC1
TC2
TC3
TC4
C1
C2
C3
C4
YOLO
C1
C2
C3
C4
C5
C6
Gmean
76%
1024-entry
Oracle