(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년09월08일
(11) 등록번호 10-2153791
(24) 등록일자 2020년09월02일

(51) 국제특허분류(Int. Cl.)
G06N 3/08 (2006.01) *G06N 3/04* (2006.01)
(52) CPC특허분류
G06N 3/08 (2013.01)
G06N 3/04 (2013.01)
(21) 출원번호 10-2018-0099090
(22) 출원일자 2018년08월24일
심사청구일자 2018년08월24일
(65) 공개번호 10-2019-0074938
(43) 공개일자 2019년06월28일
(30) 우선권주장
1020170176305 2017년12월20일 대한민국(KR)
(56) 선행기술조사문헌
KR100168975 B1*
(뒷면에 계속)

(73) 특허권자
연세대학교 산학협력단
서울특별시 서대문구 연세로 50 (신촌동, 연세대학교)
(72) 발명자
김시호
서울특별시 종로구 창경궁로 265, 101동 809호(명륜2가, 아남아파트)
박현빈
서울특별시 영등포구 영중로27길 10-1, 201호(영등포동7가, 블루빌)
(74) 대리인
민영준

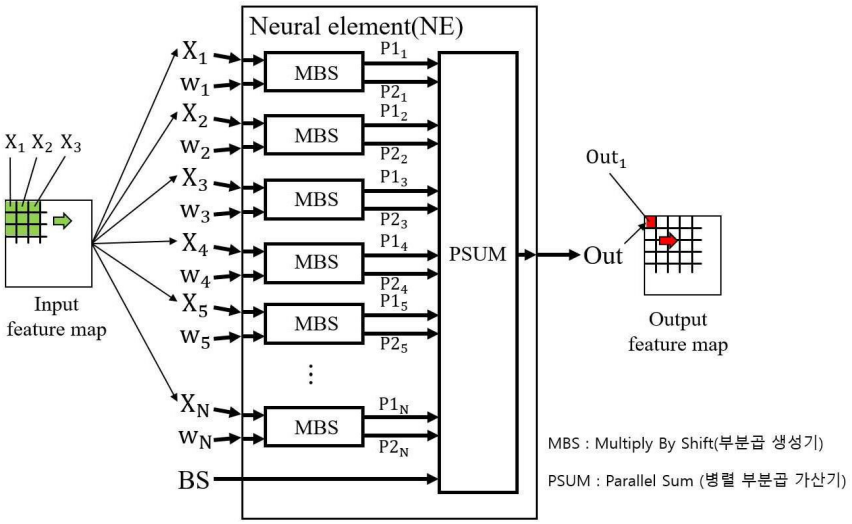
전체 청구항 수 : 총 11 항 심사관 : 유진태

(54) 발명의 명칭 **인공 신경망을 위한 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진**

(57) 요 약

본 발명의 실시예에 따른 인공 신경망을 위한 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진은 정수화된 가중치를 인가받아 입력값과의 곱셈 연산을 비트 천이 연산으로 획득되는 부분곱의 합산 방식으로 수행하도록 함으로써, 소형의 디지털 하드웨어로 구현될 수 있고, 전력 소모를 줄일 수 있으며, 연산 속도를 크게 향상시킬 수 있도록 한다.

대 표 도



(56) 선행기술조사문헌
WO2016175925 A1*
KR100321705 B1
KR1020050076459 A
KR1019930020288 A
*는 심사관에 의하여 인용된 문헌

이 발명을 지원한 국가연구개발사업

과제고유번호	20001311
부처명	산업통상자원부
과제관리(전문)기관명	한국산업기술평가관리원
연구사업명	산업기술혁신사업
연구과제명	[RCMS]엠텍비젼(주)/인공지능 카메라 임베디드 추론 엔진 및 자율 주행차 운전자 탑승자 상태 인지 카메라 모듈 응용 제품 개발(1/2,1단계)
기 여 율	1/1
과제수행기관명	연세대학교 산학협력단
연구기간	2018.04.01 ~ 2018.12.31

명 세 서

청구범위

청구항 1

기지정된 비트 수를 갖는 정수 포멧의 다수의 입력값을 포함하는 입력 벡터와 기지정된 비트 수를 갖는 정수 포멧의 다수의 가중치를 포함하는 가중치 벡터를 인가받아 내적 연산을 수행하는 디지털 하드웨어로 구현되는 디지털 뉴런에 있어서,

상기 디지털 뉴런은 신경소자를 포함하고, 상기 신경소자는

상기 가중치 벡터에서 대응하는 가중치를 인가받고, 인가된 정수 포멧의 상기 가중치를 -1, 0, 1 중 적어도 하나의 값을 갖는 계수(R)와 2의 승수(2^n)의 곱으로 표현되는 다수의 부분곱($R \cdot 2^n$)의 합으로 분해하고, 분해된 부분곱의 계수(R)와 지수(n)에 따라 제어 신호를 출력하는 다수의 가중치 분해부;

상기 입력 벡터에서 대응하는 입력값을 인가받고, 상기 제어 신호에 응답하여 상기 입력값을 지수(n)만큼 상위 비트 방향으로 비트 천이하여, 다수의 부분곱을 각각 출력하는 다수의 부분곱 생성기; 및

상기 다수의 부분곱을 병렬로 합산하여 채널 연산값을 출력하는 부분곱 가산기; 를 포함하되,

상기 다수의 부분곱 생성기 각각은

상기 계수(R)에 따라 상기 다수의 부분곱에 대해 1의 보수 연산 또는 0 전환 연산을 추가로 수행하는 디지털 뉴런.

청구항 2

삭제

청구항 3

제1 항에 있어서, 상기 디지털 뉴런은

상기 다수의 부분곱 생성기에서 1의 보수 연산된 횟수를 카운트하여 카운트 값을 획득하는 네거티브 부분곱 카운터; 를 더 포함하고,

상기 부분곱 가산기는

상기 다수의 부분곱과 함께 상기 카운트 값을 합산하여 상기 채널 연산값을 계산하는 디지털 뉴런.

청구항 4

제3 항에 있어서, 상기 부분곱 가산기는

상기 다수의 부분곱에서 부호 비트의 값이 제외된 값과 상기 카운트 값을 합산하고, 상기 합산 결과에 상기 카운트 값의 2의 보수를 부호 비트에 추가하여 상기 채널 연산값을 계산하는 디지털 뉴런.

청구항 5

제4 항에 있어서, 상기 다수의 부분곱 생성기 각각은

상기 제어 신호에 따라 구동되어 각각 부분곱을 계산하여 출력하는 다수의 부분곱 계산기를 포함하고,

상기 다수의 부분곱 계산기 각각은

상기 입력값을 상기 지수(n)만큼 비트 천이하는 비트 시프터;

상기 계수(R)가 -1인 경우, 상기 비트 시프터의 출력값에 대해 1의 보수 연산하여 상기 부분곱을 출력하는 1의 보수 연산기; 및

상기 계수(R)가 0인 경우, 상기 제어 신호에 응답하여 상기 부분곱을 0으로 전환하는 제로 곱셈기; 를 더 포함하는 디지털 뉴런.

청구항 6

제4 항에 있어서, 상기 부분곱 가산기는

각각 다수의 전가산기를 포함하는 다수의 스테이지로 구성되고,

상기 다수의 스테이지 중 제1 스테이지는 상기 다수의 부분곱을 인가받고, 서로 다른 기설정된 개수의 부분곱의 동일 비트의 값들을 그룹화하여 가산하고,

나머지 스테이지는 이전 스테이지의 가산값들을 그룹화하여 가산하며,

최종 스테이지는 이전 스테이지의 가산값의 하위 비트에 상기 카운트 값을 합산하여 상기 채널 연산값을 계산하는 디지털 뉴런.

청구항 7

제6 항에 있어서, 상기 부분곱 가산기는

상기 카운트 값을 인가받아 2의 보수 연산하여 카운트 보수값을 계산하는 2의 보수 연산기; 를 더 포함하고,

상기 최종 스테이지는 이전 스테이지의 가산값의 상위 비트에 상기 카운트 보수값을 가산하여 상기 채널 연산값의 부호 비트를 확장하는 디지털 뉴런.

청구항 8

제1 항에 있어서, 상기 디지털 뉴런은

3차원 가중치 필터의 다수의 가중치을 갖는 다수의 2차원 가중치 필터 중 대응하는 2차원 가중치 필터와 3차원 입력 특징 맵의 다수의 입력값을 갖는 다수의 2차원 입력 특징 맵 중 대응하는 2차원 입력 특징 맵을 각각 인가받는 다수의 신경소자를 포함하고,

상기 다수의 신경소자 각각에서 출력되는 상기 채널 연산값과 기지정된 바이어스 값을 가산하여 상기 디지털 뉴런의 출력값을 출력하는 채널 가산기; 를 더 포함하는 디지털 뉴런.

청구항 9

제1 항에 있어서, 상기 가중치는

상기 부분곱의 항목의 개수를 줄이기 위해, 미리 지정된 정수가 제외되도록 지정된 인접 정수로 대치되어 인가되는 디지털 뉴런.

청구항 10

기지정된 비트 수를 갖는 정수 포멧의 다수의 입력값을 포함하는 입력 벡터와 기지정된 비트 수를 갖는 정수 포멧의 다수의 가중치를 포함하는 가중치 벡터를 인가받아 내적 연산을 수행하는 디지털 하드웨어로 구현되는 신경소자를 포함하는 디지털 뉴런에 있어서,

상기 신경소자는

상기 가중치 벡터에서 대응하는 가중치를 인가받고, 인가된 정수 포멧의 상기 가중치를 -1, 0, 1 중 적어도 하나의 값을 갖는 계수(R)와 2의 승수(2^n)의 곱으로 표현되는 다수의 부분곱($R \cdot 2^n$)의 합으로 분해하는 다수의 가중치 분해부;

상기 입력 벡터에서 대응하는 입력값과 상기 입력값의 네거티브 입력값 및 0값에 대응하는 전압을 인가받아 상기 계수(R)에 대응하는 값을 선택하고, 선택된 값을 지수(n)만큼 상위 비트 방향으로 비트 천이하여, 다수의 부분곱을 각각 출력하는 다수의 부분곱 생성기; 및

상기 다수의 부분곱을 병렬로 합산하여 채널 연산값을 출력하는 부분곱 가산기; 를 포함하되,

상기 다수의 부분곱 생성기 각각은

상기 계수(R)에 따라 상기 다수의 부분곱에 대해 1의 보수 연산 또는 0 전환 연산을 추가로 수행하는 디지털 뉴런.

청구항 11

제10 항에 있어서, 상기 다수의 부분곱 생성기 각각은

각각 부분곱을 계산하여 출력하는 다수의 부분곱 계산기를 포함하며,

상기 다수의 부분곱 계산기 각각은

상기 입력값과 상기 입력값의 2의 보수인 네거티브 입력값 및 0값에 대응하는 전압을 인가받고, 상기 계수(R)에 대응하는 값을 선택하여 출력하는 먹스; 및

상기 먹스에서 선택되어 인가되는 값을 상기 지수(n)만큼 비트 천이하는 비트 시프터; 를 포함하는 디지털 뉴런.

청구항 12

제10 항에 있어서, 상기 가중치는

상기 부분곱의 항목의 개수를 줄이기 위해, 미리 지정된 정수가 제외되도록 지정된 인접 정수로 대치되어 인가되는 디지털 뉴런.

발명의 설명

기 술 분 야

[0001] 본 발명은 인공 신경망을 위한 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진에 관한 것으로, 특히 정수 기반으로 다수의 곱셈 및 덧셈 연산을 단순화하여 연산 속도를 향상시키고 전력 소비를 저감할 수 있으며, 작은 면적의 디지털 하드웨어로 구현 가능한 인공 신경망을 위한 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진에 관한 것이다.

배 경 기 술

[0002] 최근 인간의 두뇌가 패턴을 인식하는 방법을 모사하여 두뇌와 비슷한 방식으로 여러 정보를 처리하도록 구성된 인공 신경망(artificial neural network)을 이용한 딥 러닝에 대한 연구가 활발하게 진행되고 있다. 딥 러닝은 일예로 객체 분류, 객체 검출, 음성 인식, 자연어 처리와 분야에 적용되고 있으며, 적용 분야가 계속 확장되어 가고 있다.

[0003] 딥 러닝(Deep Learning) 기법을 이용하는 인공 신경망은 일반적으로 다수의 인공 뉴런(artificial neuron)을 포함하고, 다수의 인공 뉴런은 행렬(또는 벡터)의 내적(dot product) 연산을 수행하도록 구성된다. 즉 다수의 인공 뉴런은 인공 신경망의 추론 엔진을 구성하며, 연산기로서 기능한다.

[0004] 또한 인공 신경망은 추론을 위해 사용되기 이전에 방대한 학습 데이터를 기반으로 학습이 수행되어야, 요구되는 패턴 인식 성능을 나타낼 수 있다. 그러므로 일반적으로 학습은 서버 또는 GPU(Graphics Processing Unit)/TPU(Tensor Processing Units)와 같은 고성능의 범용 연산 가속기가 장착된 장치에서 수행되고, 실제 사용 시의 추론은 임베디드 시스템에서 수행되는 전략이 채택되어 왔다. 그리고 인공 신경망은 학습 및 추론 과정에서 대량의 덧셈 및 곱셈 연산을 수행하게 된다.

[0005] 그러나 인공 신경망이 학습이 아닌 추론을 위해 이용되는 경우, 적용된 시스템의 하드웨어의 성능에 따라 처리 속도에 큰 차이가 발생하게 된다. 특히 임베디스 시스템(embedded system)에서의 경우, 크기, 전력 소비 등과 같은 여러 제약 사항이 존재하므로, 인공 신경망을 위해 고성능의 범용 연산 가속기 이용하기 어렵다는 한계가 있다. 따라서 임베디스 시스템에서는 일반적으로 인공 신경망의 인공 뉴런이 추론을 위해 별도로 설계되는 디지털 하드웨어인 추론 엔진의 형태로 구현된다. 그럼에도 임베디드 시스템의 크기, 전력 소비 및 연산 속도의 문제는 추론 엔진의 설계에도 제약 사항으로 작용하여, 인공 신경망이 소형 기기 등에 적용되기 어렵게 하는 요

인이 되어, 인공 신경망의 적용 분야를 제한하는 요인이 되고 있다.

[0006] 이에 적은 면적으로 구현되면서 저전력을 소모하여 인공 신경망에서 요구되는 대량의 연산에 대한 연산 속도를 획기적으로 향상 시킬 수 있는 추론 엔진을 위한 최적화된 인공 뉴런이 요구되고 있다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 한국 공개 특허 제10-2016-0143505호 (2016.12.14 공개)

발명의 내용

해결하려는 과제

[0008] 본 발명의 목적은 소형으로 구현 가능하고 저전력을 소모하며, 연산 속도를 향상시킬 수 있는 인공 신경망을 위한 디지털 하드웨어로 구현되는 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진을 제공하는데 있다.

[0009] 본 발명의 다른 목적은 정수화된 가중치를 입력받아, 정수 입력값과의 곱셈 연산을 제한된 개수의 비트 천이 및 가산 연산으로 수행함으로써, 연산 속도를 향상시키고, 저전력을 소모하며 구현 면적을 줄일 수 있는 인공 신경망을 위한 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진을 제공하는데 있다.

[0010] 본 발명의 다른 목적은 다수의 입력값과 다수의 가중치의 곱셈을 병렬로 동시에 연산하여 연산 속도를 향상 시킬 수 있는 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진을 제공하는데 있다.

[0011] 본 발명의 또 다른 목적은 부분곱의 합산 시에 부호 비트에 대한 연산을 별도로 수행하여, 적은 면적으로 구현 가능한 인공 신경망을 위한 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진을 제공하는데 있다.

[0012] 본 발명의 또 다른 목적은 다수의 디지털 뉴런을 포함하여 기지정된 크기 이하의 가중치 필터에 대해서는 병렬로 연산 처리를 수행하고, 기지정된 크기를 초과하는 크기를 갖는 가중치 필터에 대해서는 가중치 필터의 가중치를 분할하여 동시에 연산 처리를 수행할 수 있도록 하여 가중치 필터의 크기 조절이 가능한 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진을 제공하는데 있다.

[0013] 본 발명의 또 다른 목적은 다수의 레이어로 구성되는 인공 신경망에서 다수의 레이어가 디지털 뉴런을 공유할 수 있도록 하여, 최소 개수의 디지털 뉴런으로 다수의 레이어를 구현할 수 있도록 하는 추론 엔진을 제공하는데 있다.

과제의 해결 수단

[0014] 상기 목적을 달성하기 위한 본 발명의 일 실시예에 따른 디지털 뉴런은 기지정된 비트 수를 갖는 정수 포멧의 다수의 입력값을 포함하는 입력 벡터와 기지정된 비트 수를 갖는 정수 포멧의 다수의 가중치를 포함하는 가중치 벡터를 인가받아 내적 연산을 수행하는 디지털 하드웨어로 구현되는 디지털 뉴런에 있어서, 상기 디지털 뉴런은 신경소자를 포함하고, 상기 신경소자는 상기 가중치 벡터에서 대응하는 가중치를 인가받고, 인가된 정수 포멧의 상기 가중치를 -1, 0, 1 중 적어도 하나의 값을 갖는 계수(R)와 2의 승수(2^n)의 곱으로 표현되는 다수의 부분곱 ($R \cdot 2^n$)의 합으로 분해하고, 분해된 부분곱의 계수(R)와 지수(n)에 따라 제어 신호를 출력하는 다수의 가중치 분해부; 상기 입력 벡터에서 대응하는 입력값을 인가받고, 상기 제어 신호에 응답하여 상기 입력값을 지수(n)만큼 상위 비트 방향으로 비트 천이하여, 다수의 부분곱을 각각 출력하는 다수의 부분곱 생성기; 및 상기 다수의 부분곱을 병렬로 합산하여 채널 연산값을 출력하는 부분곱 가산기; 를 포함한다.

[0015] 상기 다수의 부분곱 생성기 각각은 상기 계수(R)에 따라 상기 다수의 부분곱에 대해 1의 보수 연산 또는 0 전환 연산을 추가로 수행할 수 있다.

[0016] 상기 디지털 뉴런은 상기 다수의 부분곱 생성기에서 1의 보수 연산된 횟수를 카운트하여 카운트 값을 획득하는 네거티브 부분곱 카운터; 를 더 포함하고, 상기 부분곱 가산기는 상기 다수의 부분곱과 함께 상기 카운트 값을 합산하여 상기 채널 연산값을 계산할 수 있다.

[0017] 상기 부분곱 가산기는 상기 다수의 부분곱에서 부호 비트의 값이 제외된 값과 상기 카운트 값을 합산하고, 상기

합산 결과에 상기 카운트 값의 2의 보수를 부호 비트에 추가하여 상기 채널 연산값을 계산할 수 있다.

[0018] 상기 다수의 부분곱 생성기 각각은 상기 제어 신호에 따라 구동되어 각각 부분곱을 계산하여 출력하는 다수의 부분곱 계산기를 포함하고, 상기 다수의 부분곱 계산기 각각은 상기 입력값을 상기 지수(n)만큼 비트 천이하는 비트 시프터; 상기 계수(R)가 -1인 경우, 상기 비트 시프터의 출력값에 대해 1의 보수 연산하여 상기 부분곱을 출력하는 1의 보수 연산기; 및 상기 계수(R)가 0인 경우, 상기 제어 신호에 응답하여 상기 부분곱을 0으로 전환하는 제로 곱셈기; 를 더 포함할 수 있다.

[0019] 상기 부분곱 가산기는 각각 다수의 전가산기를 포함하는 다수의 스테이지로 구성되고, 상기 다수의 스테이지 중 제1 스테이지는 상기 다수의 부분곱을 인가받고, 서로 다른 기설정된 개수의 부분곱의 동일 비트의 값들을 그룹화하여 가산하고, 나머지 스테이지는 이전 스테이지의 가산값들을 그룹화하여 가산하며, 최종 스테이지는 이전 스테이지의 가산값의 하위 비트에 상기 카운트 값을 합산하여 상기 채널 연산값을 계산할 수 있다.

[0020] 상기 부분곱 가산기는 상기 카운트 값을 인가받아 2의 보수 연산하여 카운트 보수값을 계산하는 2의 보수 연산기; 를 더 포함하고, 상기 최종 스테이지는 이전 스테이지의 가산값의 상위 비트에 상기 카운트 보수값을 가산하여 상기 채널 연산값의 부호 비트를 확장할 수 있다.

[0021] 상기 디지털 뉴런은 다수의 가중치을 갖는 2차원 가중치 필터와 다수의 입력값을 갖는 2차원 입력 특징 맵을 인가받는 하나의 신경소자를 포함하고, 상기 부분곱 가산기는 기지정된 바이어스 값을 인가받아 상기 채널 연산값에 가산하여 상기 디지털 뉴런의 출력값을 출력할 수 있다.

[0022] 상기 디지털 뉴런은 3차원 가중치 필터의 다수의 가중치을 갖는 다수의 2차원 가중치 필터 중 대응하는 2차원 가중치 필터와 3차원 입력 특징 맵의 다수의 입력값을 갖는 다수의 2차원 입력 특징 맵 중 대응하는 2차원 입력 특징 맵을 각각 인가받는 다수의 신경소자를 포함하고, 상기 다수의 신경소자 각각에서 출력되는 상기 채널 연산값과 기지정된 바이어스 값을 가산하여 상기 디지털 뉴런의 출력값을 출력하는 채널 가산기; 를 더 포함할 수 있다.

[0023] 상기 가중치는 상기 부분곱의 개수를 줄이기 위해, 미리 지정된 정수가 제외되도록 지정된 인접 정수로 대치되어 인가될 수 있다.

[0024] 상기 목적을 달성하기 위한 본 발명의 일 실시예에 따른 디지털 뉴런은 기지정된 비트 수를 갖는 정수 포멧의 다수의 입력값을 포함하는 입력 벡터와 기지정된 비트 수를 갖는 정수 포멧의 다수의 가중치를 포함하는 가중치 벡터를 인가받아 내적 연산을 수행하는 디지털 하드웨어로 구현되는 신경소자를 포함하는 디지털 뉴런에 있어서, 상기 신경소자는 상기 가중치 벡터에서 대응하는 가중치를 인가받고, 인가된 정수 포멧의 상기 가중치를 -1, 0, 1 중 적어도 하나의 값을 갖는 계수(R)와 2의 승수(2^n)의 곱으로 표현되는 다수의 부분곱($R \cdot 2^n$)의 합으로 분해하는 다수의 가중치 분해부; 상기 입력 벡터에서 대응하는 입력값과 상기 입력값의 2의 보수인 네거티브 입력값 및 0값에 대응하는 전압을 인가받아 상기 계수(R)에 대응하는 값을 선택하고, 선택된 값을 지수(n)만큼 상위 비트 방향으로 비트 천이하여, 다수의 부분곱을 각각 출력하는 다수의 부분곱 생성기; 및 상기 다수의 부분곱을 병렬로 합산하여 채널 연산값을 출력하는 부분곱 가산기; 를 포함한다.

[0025] 상기 다수의 부분곱 생성기 각각은 각각 부분곱을 계산하여 출력하는 다수의 부분곱 계산기를 포함하며, 상기 다수의 부분곱 계산기 각각은 상기 입력값과 상기 입력값의 2의 보수인 네거티브 입력값 및 0값에 대응하는 전압을 인가받고, 상기 계수(R)에 대응하는 값을 선택하여 출력하는 먹스; 및 상기 먹스에서 선택되어 인가되는 값을 상기 지수(n)만큼 비트 천이하는 비트 시프터; 를 포함할 수 있다.

발명의 효과

[0026] 따라서, 본 발명의 인공 신경망을 위한 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진은 디지털 뉴런이 곱셈 연산을 비트 쉬프트 및 합산 방식으로 수행하도록 함으로써, 소형의 디지털 하드웨어로 구현될 수 있고, 전력 소모를 줄일 수 있으며, 연산 속도를 크게 향상시킬 수 있다. 이때, 디지털 뉴런은 부분곱의 개수를 최소화할 수 있도록 네거티브 부분곱을 2의 보수 연산을 이용하여 제공함으로써, 연산 속도를 더욱 향상시킬 수 있으며, 크기 및 전력 소비를 추가적으로 줄일 수 있다.

[0027] 그리고 다수의 가중치와 다수의 입력값에 대해 동시에 병렬로 곱셈 연산을 수행하도록 하여 연산 속도를 더욱 향상 시킬 수 있다.

[0028] 또한 디지털 뉴런은 다수의 네거티브 부분곱을 획득하기 위한 2의 보수 연산을 1의 보수 연산 이후, 네거티브

부분곱의 개수에 따라 일괄적으로 가산하여 회로 구성을 더욱 간략화할 수 있다. 뿐만 아니라 디지털 뉴런은 부분곱의 합산 시에 부호 비트에 대한 연산을 별도로 수행할 수 있도록 하여, 크기와 전력소모를 최소화 화면서도 연산 속도를 크게 높일 수 있다.

[0029] 그리고 추론 엔진은 다수의 디지털 뉴런을 포함하여 기지정된 크기 이하의 가중치 필터에 대해서는 병렬로 연산 처리를 수행하고, 기지정된 크기를 초과하는 크기를 갖는 가중치 필터에 대해서는 가중치 필터의 가중치를 분할하여 연산 처리를 수행할 수 있도록 하여, 하드웨어적 구조 변경없이 다양한 크기의 가중치 필터를 이용할 수 있다.

[0030] 추가적으로 다수의 레이어로 구성되는 인공 신경망의 추론 엔진은 다수의 레이어가 디지털 뉴런을 공유하도록 함으로써, 다수의 레이어를 최소 개수의 하드웨어 레이어로 구현함으로써, 인공 신경망에 포함되는 디지털 뉴런의 개수를 최소화 할 수 있으며, 최소 회로 구성으로 복잡한 인공 신경망을 용이하게 구현할 수 있도록 한다.

도면의 간단한 설명

[0031] 도1 은 인공 신경망 중 심층 신경망의 일예를 나타낸다.

도2 는 인공신경망의 인공 뉴런의 수학적 모델의 일예를 나타낸다.

도3 은 본 발명의 실시예에 따른 인공 신경망의 디지털 뉴런에 대한 블록 다이어그램을 나타낸다.

도4 는 가중치를 지정된 개수의 부분곱으로 표현하는 경우의 오차를 설명하기 위한 도면이다.

도5 는 도3 의 인공 뉴런에서 신경소자의 상세 구성의 일예를 나타낸다.

도6 은 도5 의 부분곱 가산기의 상세 구성의 일예를 나타낸다.

도7 은 도6 의 2의 보수 연산기의 동작 개념을 설명하기 위한 도면이다.

도8 은 디지털 뉴런의 신경소자의 다른 예를 나타낸다.

도9 는 본 발명의 다른 실시예에 따른 인공 신경망의 디지털 뉴런에 대한 블록 다이어그램을 나타낸다.

도10 은 도9 의 인공 뉴런에서 신경소자의 상세 구성의 일예를 나타낸다.

도11 은 도10 의 부분곱 가산기의 상세 구성의 일예를 나타낸다.

도12 는 본 실시예에 따른 디지털 뉴런의 전력 소비를 시뮬레이션 한 결과를 나타낸다.

도13 은 본 발명의 실시예에 따른 디지털 뉴런을 이용하여 가중치 필터 크기를 가변할 수 있는 인공 뉴런 구조의 일예를 나타낸다.

도14 는 본 발명의 다른 실시예에 따른 인공 신경망의 추론 엔진의 개략적 구성을 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0032] 본 발명과 본 발명의 동작상의 이점 및 본 발명의 실시에 의하여 달성되는 목적을 충분히 이해하기 위해서는 본 발명의 바람직한 실시예를 예시하는 첨부 도면 및 첨부 도면에 기재된 내용을 참조하여야만 한다.

[0033] 이하, 첨부한 도면을 참조하여 본 발명의 바람직한 실시예를 설명함으로써, 본 발명을 상세히 설명한다. 그러나, 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며, 설명하는 실시예에 한정되는 것이 아니다. 그리고, 본 발명을 명확하게 설명하기 위하여 설명과 관계없는 부분은 생략되며, 도면의 동일한 참조부호는 동일한 부재임을 나타낸다.

[0034] 명세서 전체에서, 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. 또한, 명세서에 기재된 "...부", "...기", "모듈", "블록" 등의 용어는 적어도 하나의 기능이나 동작을 처리하는 단위를 의미하며, 이는 하드웨어나 소프트웨어 또는 하드웨어 및 소프트웨어의 결합으로 구현될 수 있다.

[0035] 도1 은 인공 신경망 중 컨볼루션 신경망의 일예를 나타내고, 도2 는 인공신경망의 인공 뉴런의 수학적 모델의 일예를 나타낸다.

[0036] 도1 에 도시된 바와 같이, 일반적으로 인공 신경망(artificial neural network), 특히 심층 신경망(Deep

Neural Network: DNN)은 다수의 레이어를 포함하고, 다수의 레이어 각각에 포함된 적어도 하나의 인공 뉴런 각각은 입력 레이어 또는 이전 레이어의 출력을 입력 벡터(X)로서 인가받아 기지정된 연산을 수행한다.

[0037] 다수의 레이어에 포함된 다수의 인공 뉴런 각각은 인가된 입력 벡터(X)에 대해 미리 지정된 가중치 벡터(w)를 이용하여 기지정된 방식으로 연산하여 출력하도록 구성될 수 있다.

[0038] 도2 를 참조하면, 다수의 인공 뉴런 각각에는 입력 벡터(X)의 원소(X_1, X_2, …, X_N)가 입력값으로 인가된다. 그리고 가중치 벡터(w)의 원소인 가중치(w_1, w_2, …, w_N)는 입력 벡터(X)의 다수의 입력값(X_1, X_2, …, X_N) 중 대응하는 입력값의 중요도를 조절하기 위한 값이다.

[0039] 여기서 가중치 벡터(w)의 다수의 가중치(w_1, w_2, …, w_N)는 심층 신경망의 학습 과정 동안 오차 역전파에 의해 가변되어 다양한 값을 갖도록 지정될 수 있다. 학습에 의해 지정된 가중치 벡터(w)는 각각의 레이어 또는 별도의 메모리에 미리 저장될 수 있다.

[0040] 일예로, 인공 뉴런은 입력 벡터(X)의 다수의 입력값(X_1, X_2, …, X_N)에 가중치 벡터(w)의 다수의 가중치 각각(w_1, w_2, …, w_N) 중 대응하는 가중치를 곱하고, 각각의 곱한 결과를 바이어스 값(b)와 함께 합산한다.

[0041] 즉 인공 뉴런은 입력 벡터(X)에 대해 기지정된 가중치 벡터(w)를 내적(inner product)하여 출력하는 연산을 수행할 수 있으며, 이러한 인공 뉴런의 수학적 모델의 프로세스는 수학식 1과 같이 표현될 수 있다.

수학식 1

[0042] $$\text{Output} = b + \sum_{i=1}^{N}(X_i \cdot w_i)$$

[0043] 여기서 X_i, w_i, b 및 Output는 각각 입력값, 가중치, 바이어스 값 및 출력값을 나타낸다.

[0044] 수학식 1에서는 입력 벡터(X) 및 가중치 벡터(w)가 각각 1차원 벡터인 것으로 가정하였으나, 입력 벡터(X) 및 가중치 벡터(w)는 다차원 벡터(n차원 벡터(여기서 n은 자연수))일 수 있으며, 행렬로 표현될 수 있다. 이 때, 가중치 벡터(w)는 입력 벡터(X)의 차원에 대응하는 차원을 가질 수 있다. 예를 들면, 입력 벡터(X)가 2차원 행렬인 경우, 가중치 벡터(w) 또한 2차원 행렬로 구성될 수 있다.

[0045] 도1 은 인공 신경망의 일예로서, 컨볼루션 신경망(Convolution Neural Network: CNN)을 도시하였으며, 컨볼루션 신경망(CNN)은 영상 인식, 음성 인식, 자연어 처리, 필기체 인식 등에 주로 사용되는 신경망이다. 도1 에서는 대표적인 컨볼루션 신경망으로 잘알려진 LeNet-5의 개략적 구조를 나타내었다.

[0046] 도1 에 도시된 바와 같이, 컨볼루션 신경망은 입력된 입력 영상의 특징을 추출하는 특징 추출부(Feature Extraction) 및 분류부(Classification)를 포함할 수 있다.

[0047] 그리고 도1 의 컨볼션 신경망의 특징 추출부는 적어도 하나의 컨볼루션 레이어(Convolution Layer)(C1, C2, C3)과 적어도 하나의 풀링 레이어(Pooling Layer)(S1, S2, S3)를 포함하고, 분류부는 적어도 하나의 완전 연결 레이어(Fully Connected Layer)(FC1, FC2)를 포함할 수 있다.

[0048] 도1 에서는 일예로 컨볼루션 신경망이 3개의 컨볼루션 레이어(C1, C2, C3)와 3개의 풀링 레이어(S1, S2, S3) 및 2개의 완전 연결 레이어(FC1, FC2)를 포함하는 것으로 도시하였으나, 레이어의 개수는 다양하게 조절될 수 있다.

[0049] 컨볼루션 레이어(C1, C2, C3)는 이전 단의 레이어에서 출력되는 적어도 하나의 출력 특징 맵(또는 입력 영상)을 입력 벡터(X)로서 인가받아 기지정된 가중치 벡터(w)와 컨볼루션 연산을 수행하여 출력 특징 맵을 생성한다.

[0050] 컨볼루션 신경망(CNN)에서는 입력 벡터(X)를 입력 특징 맵(Input feature map)이라 하며, 가중치 벡터(w)를 가중치 필터(weight filter)라 한다. 즉 컨볼루션 신경망(CNN)의 컨볼루션 레이어는 입력 특징 맵을 인가받아 가중치 필터로 필터링하여 출력 특징 맵을 출력한다.

[0051] 그리고 풀링 레이어(S1, S2, S3)는 인공 신경망의 오버피팅을 줄이고 성능을 향상 시키기 위해 이전 레이어(C1, C2, C3)에서 출력되는 출력값에 대해 서브 샘플링(sub-sampling)을 수행한다.

[0052] 완전 연결 레이어 (FC1, FC2)는 특징 추출부에서 추출된 특징 맵 각각을 기지정된 클래스로 분류한다.

[0053] 컨볼루션 레이어(C1, C2, C3)는 적어도 하나의 입력 특징 맵 중 대응하는 적어도 하나의 입력 특징 맵을 인가받아, 서로 다른 기저장된 가중치 필터로 컨볼루션 연산을 수행하는 다수의 디지털 뉴런을 포함할 수 있다. 이때 다수의 컨볼루션 레이어 각각은 서로 다른 개수의 인공 뉴런을 포함하도록 구성될 수 있다.

[0054] 컨볼루션 레이어(C1, C2, C3)의 인공 뉴런은 인가된 적어도 하나의 입력 특징 맵을 기지정된 가중치 필터와 컨볼루션 연산한다. 컨볼루션 레이어(C1, C2, C3)의 인공 뉴런은 컨볼루션 연산을 수행하는 동안 가중치 필터를 입력 특징 맵 상을 이동시키며, 입력 특징 맵에서 가중치 필터가 투영된 영역의 입력값과 가중치를 곱한다. 그리고 곱해진 결과와 바이어스 값(Bias)을 합하고, 활성화 함수로 필터링하여 출력한다. 이때 출력값은 입력 특징 맵에 가중치 필터가 투영된 위치에 대응하여 출력 특징 맵의 지정된 위치의 값으로 지정된다. 즉 출력 특징 맵은 행렬 형태로 표현되는 벡터로 획득될 수 있다.

[0055] 따라서 컨볼루션 레이어(C1, C2, C3)의 인공 뉴런은 입력 특징 맵(X)에 대해 기지정된 가중치 필터(w)를 이동시키면서 내적하여 컨볼루션 연산을 수행할 수 있으며, 컨볼루션 레이어의 인공 뉴런의 프로세스는 수학식 2와 같이 표현될 수 있다.

수학식 2

$$\mathrm{Output}[x][y] = b + \sum_{i=0}^{R-1}\sum_{j=0}^{S-1} \mathbf{X}[x+i][y+j] \cdot \mathbf{w}[x+i][y+j]$$

[0056] $$= \mathbf{X} \bullet \mathbf{w} + b$$

[0057] 여기서 X, w, b, F 및 Output은 각각 입력값, 가중치, 바이어스 값 및 출력값을 나타낸다. 그리고 R 및 S는 각각 입력 벡터(X) 상에서 가중치 벡터(w)가 행 방향 및 열 방향 이동 가능 거리를 나타내며, x, y는 입력 벡터(X)에 대한 가중치 벡터(w)의 위치를 나타낸다.

[0058] 수학식 2 를 참조하면, 컨볼루션 신경망(CNN)의 인공 뉴런 또한 수학식 1과 마찬가지로, 입력 벡터(X)와 가중치 벡터(w)를 내적 연산함을 알 수 있다.

[0059] 다만 컨볼루션 연산의 특성상 컨볼루션 레이어(C1, C2, C3)의 인공 뉴런은 입력 특징 맵을 가중치 필터의 크기에 대응하여 다수 횟수로 곱셈을 반복하고, 반복 수행되는 곱셈의 결과를 누적합하여 수행한다.

[0060] 일반적으로 회로 구현에 있어서 곱셈 연산은 덧셈 연산에 비해 큰 전력과 연산 시간을 요구하며, 특히 칩 구현에 있어서 더 큰 면적을 필요로 한다. 따라서 임베디드 시스템에 적용되는 인공 신경망을 위한 연산 가속기로서 디지털 회로로 구현되는 디지털 뉴런은 곱셈 연산을 효율적으로 수행할 수 있도록 구성되어야 한다.

[0061] 도3 은 본 발명의 실시예에 따른 인공 신경망의 디지털 뉴런에 대한 블록 다이어그램을 나타낸다.

[0062] 도3 을 참조하면, 본 실시예에 따른 디지털 뉴런은 적어도 하나의 신경소자(Neural Element: NE)를 포함하도록 구성된다.

[0063] 신경소자(NE)는 입력 특징 맵과 가중치 필터를 인가받고, 인가된 입력 특징 맵 및 가중치 필터에 대해 기지정된 연산을 수행하여 출력한다. 여기서 기지정된 연산은 일예로 내적 연산이거나, 내적 연산을 누적하여 수행하는 컨볼루션 연산일 수 있다.

[0064] 상기한 바와 같이 컨볼루션 연산은 입력 특징 맵과 가중치 필터의 내적의 누적합이며, 내적은 입력 특징 맵의 원소인 다수의 입력값(X_i)과 가중치 필터의 원소인 다수의 가중치(w_i = w_1, w_2, …, w_N)의 곱셈값의 합으로 계산된다.

[0065] 이때 다수의 입력값(X_i)은 정수 포멧으로 인가되며, 가중치 필터의 가중치(w_i) 또한 기지정된 비트수를 갖는 정수 포멧의 가중치로 인가될 수 있다. 일예로 가중치(w_i)는 5비트의 정수 포멧의 가중치일 수 있다.

[0066] 일반적으로 인공 신경망은 학습 과정을 통해 32비트의 단정밀도 부동 소수점 포멧(single-precision Floating Point format(FP32)) 또는 16 비트의 반정밀도 부동 소수점 포멧(half-precision Floating Point format(FP16))의 가중치가 획득된다. 그러나 부동 소수를 곱셈 연산하기 위한 곱셈 회로의 구조는 매우 복잡하여, 전력 소모가 크며, 큰 면적을 요구한다.

[0067] 반면, 가중치(w_i)가 정수인 경우, 정수 포멧의 입력값과 가중치의 곱셈은 비트 천이 연산과 덧셈 연산의 조합으로 수행될 수 있다. 비트 천이 연산을 수행하는 비트 시프터 및 덧셈 연산을 수행하는 가산기는 곱셈 회로에 비해 매우 간단한 구조로 구현될 수 있어, 소형으로 구현될 수 있으며 전력 소모를 크게 줄일 수 있다. 특히 곱셈 회로에 비해 연산 속도가 비약적으로 증가된다.

[0068] 다만 디지털 뉴런이 부동 소수점 포멧의 가중치가 아닌 정수 포멧의 가중치를 이용하는 경우에도, 인공 신경망이 동일한 성능을 나타낼 수 있는지 고려되어야 한다. 이에 본 실시예에서는 부동 소수점 포멧의 가중치와 정수화된 가중치를 이용하는 경우 각각에서 인공 신경망의 성능을 시뮬레이션 하였다. 시뮬레이션은 필기체 숫자를 인식하기 위해 개발된 대표적인 컨볼루션 신경망인 LeNet-5에서 MNIST(Modified National Institute of Standards and Technology database)를 이용하여 수행되었으며, 시뮬레이션 결과는 32비트의 단정밀도 부동 소수점 포멧의 가중치 대신 8비트 정수 포멧의 가중치가 이용되어도 LeNet-5의 필기체 숫자 인식 정확도는 동일하게 나타나는 것으로 확인되었다. 즉 32비트의 부동 소수점 포멧의 가중치보다 적은 비트수를 갖는 8비트의 정수 포멧의 가중치를 이용하여도 동일한 정확도를 나타내며, 인공 신경망의 성능이 동일하게 유지됨이 확인되었다.

[0069] 따라서 본 실시예에서 디지털 뉴런은 곱셈 연산을 비트 천이 연산 및 덧셈 연산의 조합으로 수행할 수 있도록 정수화된 가중치를 인가받는다.

[0070] 특히 본 실시예에서 디지털 뉴런은 32비트의 단정밀도 부동 소수점 포멧과 동일한 정확도를 나타내는 8비트의 정수 포멧 보다 적은 비트수(예를 들면 5비트)의 정수 포멧의 가중치(w_i)를 인가받을 수 있다.

[0071] 인공 신경망은 패턴 인식 및 분류를 위해 주로 이용되며, 이러한 인공 신경망은 매우 정확한 정밀도의 산술적 계산을 요구하지 않는다. 오히려 인공 신경망에서는 학습 데이터에 과도하게 최적화되는 오버피팅(Overfitting)과 같은 문제를 방지하여, 다양한 입력 데이터에 대해 유연하게 처리할 수 있도록 풀링 레이어(Pooling layer)을 추가하는 경우가 빈번하다.

[0072] 이에 본 실시예에서는 신경소자(NE)의 회로를 더욱 간단하게 구현하고 전력 소모를 줄일 수 있도록 32비트의 단정밀도 부동 소수점 포멧의 가중치에 대응하는 정확도를 갖는 8비트의 정수 포멧 보다 적은 비트 수(여기서는 일예로 5비트)의 정수 가중치를 인가받는 경우에 대해서 추가적으로 시뮬레이션을 수행하였다. 시뮬레이션 결과, 디지털 뉴런이 5비트의 정수 포멧의 가중치를 인가받더라도 인공 신경망의 성능 저하는 미미한 수준인 것으로 확인하였다. 따라서 본 실시예에서는 디지털 뉴런이 32비트의 단정밀도 부동 소수점 포멧과 동일한 성능을 나타내는 8비트의 정수보다 적은 비트 수를 갖는 정수 포멧의 가중치(w_i)를 인가받는 것으로 가정하여 설명한다.

[0073] 여기서 적은 비트 수를 갖는 정수 포멧의 가중치(w_i)는 일예로 인공 신경망의 학습 장치가 32비트의 단정밀도 부동 소수점 포멧의 가중치 각각에 기지정된 정수화값(일예로 16(이진수 10000))를 곱하고, 이하 소수점 자리수를 버리거나 반올림하여 획득될 수 있다.

[0074] 또한 가중치(w_i)는 파지티브 정수(positive integer)가 아닌 부호 정수(signed integer)로 인가될 수 있다. 즉, 5비트의 부호 정수 포멧의 가중치(w_i)는 -16(11111) ~ 15(01111)의 범위의 값으로 인가될 수 있다.

[0075] 본 실시예에서 가중치(w_i)가 기지정된 비트 수(여기서는 일예로 5비트)의 정수값을 가지므로, 신경소자(NE)는 입력값(X)과 가중치(w_i)의 곱셈 연산을 비트 천이 연산과 비트 천이 연산으로 계산되는 부분곱을 가산하는 덧셈 연산의 조합으로 수행할 수 있다. 여기서 비트 천이 연산은 가중치(w_i)의 각 비트의 값에 따라 입력값(X)을 비트 천이하여 부분곱을 계산하도록 수행된다.

[0076] 따라서 곱셈 연산을 비트 천이와 덧셈 연산으로 수행하므로, 신경소자(NE)는 디지털 뉴런의 연산 속도를 향상시키고, 전력 소비를 줄일 뿐만 아니라 소형으로 구현될 수 있도록 한다.

[0077] 도3 을 참조하면, 신경소자(NE)는 N(여기서 N은 자연수)개의 부분곱 생성기(MBS)와 부분곱 가산기(PSUM)를 포함

한다.

[0078] 부분곱 생성기(MBS)의 개수(N)은 2차원 가중치 필터의 원소 개수, 즉 가중치($w_i = w_1, w_2, \cdots, w_N$)의 개수와 동일하다. 일예로 가중치 필터가 5 X 5인 크기의 행렬(N = 25)인 경우, 신경소자(NE)는 25개의 부분곱 생성기(MBS)를 포함한다.

[0079] 그리고 N개의 부분곱 생성기(MBS)는 인가된 2차원 입력 특징 맵의 대응하는 입력값(X)과 2차원 가중치 필터의 대응하는 가중치(w_i)의 부분곱(($P1_1, P2_1$), ($P1_2, P2_2$), …, ($P1_N, P2_N$))을 생성하여 출력한다.

[0080] 부분곱 생성기(MBS)는 가중치(w)가 기지정된 비트수의 정수값을 가지므로, 입력값(X)과 가중치(w_i)의 곱셈 연산을 입력값(X)을 가중치(w_i) 각각의 비트 값에 따라 비트 천이하여 2개씩의 부분곱(($P1_1, P2_1$), ($P1_2, P2_2$), …, ($P1_N, P2_N$))을 계산한다. 여기서 다수의 부분곱 생성기(MBS)는 병렬로 부분곱을 계산하므로, 동시에 다수의 부분곱(($P1_1, P2_1$), ($P1_2, P2_2$), …, ($P1_N, P2_N$))을 계산할 수 있다.

[0081] 본 실시예에서는 N개의 부분곱 생성기(MBS) 각각이 2개의 부분곱(($P1_1, P2_1$), ($P1_2, P2_2$), …, ($P1_N, P2_N$))을 생성하여 출력하는 것으로 가정하며, 부분곱 생성기(MBS)가 2개의 부분곱을 생성하는 이유는 후술한다.

[0082] 부분곱 가산기(PSUM)는 N개의 부분곱 생성기(MBS) 각각에서 2개씩 전달되는 부분곱(($P1_1, P2_1$), ($P1_2, P2_2$), …, ($P1_N, P2_N$))과 바이어스 값(BS)을 가산하여, 2차원 입력 특징 맵과 2차원 가중치 필터의 내적 연산 결과인 출력값(Out)을 출력한다. 여기서 부분곱 가산기(PSUM)에서 출력되는 출력값(Out)이 컨볼루션 연산의 결과가 아닌 내적 연산의 결과인 것은 2차원 가중치 필터가 2차원 입력 특징 맵의 일부와 연산된 결과이기 때문이다. 그리고 바이어스 값(BS)는 수학식 1 및 2의 바이어스 값(b)을 의미한다.

[0083] 기존의 GPU/TPU의 경우, 한 사이클 동안 가중치 필터에서 하나의 가중치(w_i)와 대응하는 하나의 입력값(X)를 곱셈 연산하여 누적하는 방식으로 연산값을 계산한다. 따라서 가중치 필터의 원소인 가중치(w_i)의 개수(N)에 대응하는 사이클 동안 반복 연산을 수행해야 한다. 즉 N 사이클 동안, 반복 연산을 수행해야 한다.

[0084] 그에 반해 본 실시예에 따른 디지털 뉴런은 신경소자(NE)의 N개의 부분곱 생성기(MBS)가 병렬로 동시에 연산을 수행하므로, 고속으로 연산을 수행할 수 있다. 또한 부분곱 생성기(MBS)가 비트 천이 방식으로 부분곱을 계산하므로, 디지털 회로 구현이 간단하여 연산 속도를 향상시키고, 전력 소비를 줄일 뿐만 아니라 소형으로 구현될 수 있다.

[0085] 한편, 본 실시예에 따른 신경소자(NE)는 부호 정수 포멧의 가중치를 인가받도록 구성됨에 따라, 부분곱 생성기(MBS)가 부분곱의 부호가 음수로 나타나는 네거티브 부분곱을 생성할 수 있도록 한다. 네거티브 부분곱을 이용하는 경우, 입력값(X)과 가중치(w_i)의 곱셈 연산을 위해 생성되어야 하는 부분곱의 개수를 더욱 줄일 수 있다.

[0086] 예를 들어 가중치(w_i)가 7인 경우, 가중치(w_i)와 입력값(X)의 곱셈 연산은 7(00111)·X이다. 그리고 가중치(w_i)와 입력값(X)의 곱셈 연산을 비트 천이 연산을 이용하는 부분곱과 부분곱의 덧셈으로 연산을 수행하는 경우, 가중치(w_i)의 각 비트별 값(즉 2의 승수(2^n))과 입력값(X)의 부분곱의 합으로 4(00100)·X + 2(00010) ·X +1(00001)·X 와 같이 계산된다. 즉 3개의 부분곱이 생성되어야 한다. 여기서 부분곱은 입력값(X)을 가중치(w_i)의 1의 값을 갖는 비트(2의 승수(2^n)의 지수(n))만큼 상위 비트 방향으로 시프트하여 획득될 수 있다. 즉 3개의 비트 천이 회로와 가산기를 요구한다.

[0087] 그러나 신경소자(NE)가 네거티브 부분곱을 생성할 수 있는 경우, 가중치(w_i) 7과 입력값(X)의 곱셈은 8(01000) ·X + (-X)로 계산될 수 있다. 여기서 네거티브 부분곱인 -X는 입력값(X)에 대한 2의 보수 연산을 수행하여 획득될 수 있다. 즉 2개의 부분곱과 2의 보수를 이용하여 계산될 수 있다. 따라서 2개의 비트 천이 회로와 2의 보수 회로 및 가산기로 동일한 연산을 수행할 수 있다.

[0088] 이와 같은 방식으로 부분곱과 2의 보수를 이용하여, 부분곱의 개수가 최소화되도록 가중치(w_i)를 재구성하는 경우, 5비트의 가중치(w_i) 중 양의 정수(1 ~ 15) 값과 입력값(X)의 곱셈에 대한 모든 경우의 수는 표1 과 같이 나타난다.

표 1

가중치(w_i)	수식($w_i \cdot X$)	동작
1 (00001) = 1	00001 • X	X
2 (00010) = 2	00010 • X	X를 상위 비트 방향으로 시프트 1
3 (00011) = 2+1	00010 • X + 00001 • X	X를 상위 비트 방향으로 시프트 1 + X
4 (00100) = 4	00100 • X	X를 상위 비트 방향으로 시프트 2
5 (00101) = 4+1	00100 • X + 00001 • X	X를 상위 비트 방향으로 시프트 2 + X
6 (00110) = 4+2	00100 • X + 00010 • X	X를 상위 비트 방향으로 시프트 2 + X를 상위 비트 방향으로 시프트 1
7 (00111) = 8-1	01000 • X - 00001 • X	X를 상위 비트 방향으로 시프트 3 + X의 2의 보수
8 (01000) = 8	01000 • X	X를 상위 비트 방향으로 시프트 3
9 (01001) = 8+1	01000 • X + 00001 • X	X를 상위 비트 방향으로 시프트 3 + X
10 (01010) = 8+2	01000 • X + 00010 • X	X를 상위 비트 방향으로 시프트 3 + X를 상위 비트 방향으로 시프트 1
11 (01011) = 8+2+1	01000 • X + 00010 • X + 00001 • X	X를 상위 비트 방향으로 시프트 3 + X를 상위 비트 방향으로 시프트 1 + X
12 (01100) = 8+4	01000 • X + 00100 • X	X를 상위 비트 방향으로 시프트 3 + X를 상위 비트 방향으로 시프트 2
13 (01101) = 8+4+1	01000 • X + 00100 • X + 00001 • X	X를 상위 비트 방향으로 시프트 3 + X를 상위 비트 방향으로 시프트 2 + X를 상위 비트 방향으로 시프트 1
14 (01110) = 16-2	10000 • X - 00010 • X	X를 상위 비트 방향으로 시프트 4 + X를 상위 비트 방향으로 시프트 1 이후 2의 보수
15 (01111) = 16-1	10000 • X – 00001 • X	X를 상위 비트 방향으로 시프트 4 + X의 2의 보수

[0089]

[0090] 표1 의 부호 정수 포멧의 가중치(w_i)에서 이진수 10000은 -16을 의미하지만, 표 1에서와 같이 가중치 14 및 15를 계산하기 위해 이용되는 경우에, 이진수 10000은 +16으로 고려한다.

[0091] 표1 을 살펴보면, 1 ~ 15 범위 이내의 가중치(w_i)와 입력값(X)의 곱셈은 최대 3개의 부분곱의 합으로 계산될 수 있다. 그리고 표1 에서 네거티브 부분곱이 요구되는 경우에는 계산된 부분곱에 2의 보수 연산을 수행하여 획득될 수 있다.

[0092] 5비트의 정수 값을 갖는 가중치(w_i)와 입력값(X)과의 일반적인 곱셈 연산은 최대 5개의 부분곱을 합산이 필요하다. 그에 반해, 표1 과 같이, 네거티브 부분곱을 이용하게 되면, 최대 3개의 부분곱만을 합산하면 되므로, 곱셈 연산을 수행하기 위한 회로의 구성을 간략하게 할 수 있다.

[0093] 한편 표1 에 나타난 바와 같이 5비트의 정수 값을 갖는 가중치(w_i)와 입력값(X)의 곱셈에서 최대 개수인 3개의 부분곱을 요구하는 가중치는 11 및 13뿐이며, 나머지 가중치(1 ~ 10, 12, 14, 15)와 입력값(X)의 곱셈은 2개 이하의 부분곱의 합산으로 계산될 수 있다.

[0094] 따라서 5비트의 정수 값을 갖는 가중치(w_i) 중에서 최대 개수의 부분곱을 요구하는 가중치 11 및 13을 제외한 나머지 가중치와 입력값(X)의 곱셈은 수학식 3과 같이 나타날 수 있다.

수학식 3

[0095]
$$w \cdot X = \left(w_{b1}2^{i} + w_{b2}2^{j}\right) \cdot X$$
$$= w_{b1}2^{i} \cdot X + w_{b2}2^{j} \cdot X \quad \left(w_{b1,b2} \in -1,0,1\right)$$

[0096] 수학식 3을 참조하면, 5비트의 정수 값을 갖는 가중치(w_i)는 -1, 0 또는 1의 값을 갖는 계수(coefficient)(w_{b1}, w_{b2})와 2의 승수(2^i, 2^j)의 부분곱($w_{b1}2^i$, $w_{b2}2^j$)의 합으로 표현되고, 계수(w_{b1}, w_{b2}) 중 -1 의 값은 1에 대한 2의 보수(-1)로 표현될 수 있다.

[0097] 이에 가중치(w_i)와 입력값(X)의 곱셈은 2개의 부분곱($w_{b1}2^i \cdot X$, $w_{b2}2^j \cdot X$)과 2의 보수(비트 값(w_{b1}, w_{b2})이 -1인 경우)의 조합의 합산으로 계산될 수 있음을 나타낸다.

[0098] 한편, 가중치(w_i)가 -1 ~ -15의 범위인 경우에는 1 ~ 15 범위 이내의 가중치(w_i)의 각 입력값(X)의 곱에 2의 보수를 취하여, 표2 와 같이 계산할 수 있다. 따라서 가중치 -11 및 -13을 제외하면, 네거티브 가중치(w_i) 또한 2의 승수(2^n)에 대한 2개의 부분곱과 2의 보수의 조합을 합산하여 계산할 수 있다.

표 2

[0099]

가중치(w_i)	수식($w_i \cdot X$)	동작
-1 (11111) = -1	-00001 • X	X의 2의 보수
-2 (11110) = -2	-00010 • X	X를 상위 비트 방향으로 시프트 1 이후 2의 보수
-3 (11101) = -2-1	-00010 • X - 00001 • X	X를 상위 비트 방향으로 시프트 1 이후 2의 보수 + X의 2의 보수
-4 (11100) = -4	-00100 • X	X를 상위 비트 방향으로 시프트 2 이후 2의 보수
-5 (11011) = -4-1	-00100 • X - 00001 • X	X를 상위 비트 방향으로 시프트 2 이후 2의 보수 + X의 2의 보수
-6 (11010) = -4-2	-00100 • X - 00010 • X	X를 상위 비트 방향으로 시프트 2 이후 2의 보수 + X를 상위 비트 방향으로 시프트 1 이후 2의 보수
-7 (11001) = -8+1	-01000 • X + 00001 • X	X를 상위 비트 방향으로 시프트 3 이후 2의 보수 + X
-8 (11000) = -8	-01000 • X	X를 상위 비트 방향으로 시프트 3 이후 2의 보수
-9 (10111) = -8-1	-01000 • X - 00001 • X	X를 상위 비트 방향으로 시프트 3 이후 2의 보수 + X의 2의 보수
-10 (10110) = -8-2	-01000 • X - 00010 • X	X를 상위 비트 방향으로 시프트 3 이후 2의 보수 + X를 상위 비트 방향으로 시프트 1 이후 2의 보수
-11 (10101) = -8-2-1	-01000 • X - 00010 • X - 00001 • X	X를 상위 비트 방향으로 시프트 3 이후 2의 보수 + X를 상위 비트 방향으로 시프트 1 이후 2의 보수 + X의 2의 보수
-12 (10100) = -8-4	-01000 • X - 00100 • X	X를 상위 비트 방향으로 시프트 3 이후 2의 보수 + X를 상위 비트 방향으로 시프트 2 이후 2의 보수
-13 (10011) = -8-4-1	-01000 • X - 00100 • X - 00001 • X	X를 상위 비트 방향으로 시프트 3 이후 2의 보수 + X를 상위 비트 방향으로 시프트 2 이후 2의 보수 + X의 2의 보수
-14 (10010) = -16+2	10000 • X + 00010 • X	X를 상위 비트 방향으로 시프트 4 + X를 상위 비트 방향으로 시프트 1
-15 (10001) = -16+1	10000 • X + 00001 • X	X를 상위 비트 방향으로 시프트 4 + X
-16 (10000) = -16	10000 • X	X를 상위 비트 방향으로 시프트 4

[0100] 표2 에서도 -11 및 -13만이 3개의 부분곱이 필요한 것을 알 수 있다.

[0101] 이에 본 실시예에 따른 디지털 뉴런은 신경소자(NE)의 구조를 간략화하기 위하여, 기지정된 적어도 하나의 정수(여기서는 일예로 ±11 및 ±13)가 제외된 지정된 비트 수의 정수 포멧의 가중치(w_i)를 인가 받도록 구성될 수 있다.

[0102] 결과적으로 2의 승수(2^n)에 대한 2개의 부분곱을 수행하기 위한 2개의 비트 천이 회로와 2의 보수 회로만 구비하여도 5비트 범위의 가중치(w_i)와 입력값(X)의 곱셈을 수행할 수 있게 되어, 신경소자(NE)의 구조를 최대한 간략하게 할 수 있다. 즉 N개의 부분곱 생성기(MBS) 각각이 2개의 부분곱(($P1_1$, $P2_1$), ($P1_2$, $P2_2$), …, ($P1_N$, $P2_N$))을 출력하도록 구성될 수 있다.

[0103] 만일 가중치(w_i)가 6비트 또는 그 이상의 범위의 값으로 지정되는 경우에도, 유사하게 2개 또는 3개의 비트 천이 회로와 2의 보수 회로를 이용하여 가중치(w)와 입력값(X)의 곱셈을 수행하도록 할 수 있다.

[0104] 그리고 이와 별도로 가중치(w_i)가 0인 경우에는 입력값(X)에 무관하게 0으로 변환하는 회로가 신경소자(NE)에 포함될 수 있으며, 일예로 입력값(X)을 인가받아 0과의 논리곱(And) 연산을 수행하는 논리 회로를 포함할 수 있다.

[0105] 비록 -16 ~ 15 범위의 5비트 가중치(w_i)에서 ±11 및 ±13이 제외되어, 가중치(w_i)로 선택될 수 있는 값이 제한되지만, 이러한 가중치(w_i)의 제한이 인공 신경망의 성능에 미치는 영향은 크지 않다.

[0106] 도4 는 가중치를 지정된 개수의 부분곱으로 표현하는 경우의 오차를 설명하기 위한 도면이다.

[0107] 도4 에서는 일예로 가중치(w)가 8비트 정수 포멧인 경우에 대해, 0 ~ 128까지의 양의 정수 범위만을 고려한 것으로, 범위 내의 가중치(w_i)를 모두 표현하는 경우와 2의 승수(2^n)에 대한 2개의 부분곱과 2의 보수의 조합으로 표현되도록 일부 정수를 제외한 경우 및 2의 승수(2^n)에 대한 3개의 부분곱과 2의 보수의 조합으로 표현되도록 일부 정수를 제외한 경우의 오차를 나타낸다.

[0108] 도4 에서 X축은 7비트 양의 정수 포멧의 실제 가중치(w_i)를 나타내고, Y축은 7비트 양의 정수 포멧의 가중치(w_i)가 2개 또는 3개의 부분곱의 합으로 표현되도록 일부 값이 제외되어 인접한 정수값으로 대체된 가중치를 나타낸다. 여기서 녹색은 3개의 부분곱의 합으로 표현되는 가중치를 나타내고, 청색은 2개의 부분곱의 합으로 표현되는 가중치를 나타낸다.

[0109] 그러나 도4 에 나타난 바와 같이, 7비트 양의 정수 포멧의 가중치(w_i)가 2개 또는 3개의 부분곱의 합으로 표현되도록 일부 값이 제외되어 인접한 정수값으로 대체될지라도, 2개의 부분곱의 합으로 표현되는 경우에 최대 오차는 대략 9.4% 수준이고, 3개의 부분곱으로 표현되는 경우에 대략 2.3% 수준이다. 즉 7비트 양의 정수 포멧의 가중치(w_i)를 0 ~ 128까지의 범위의 정수로 표현하지 않고, 2개의 부분곱의 합 또는 3개의 부분곱의 합으로 표현 가능한 정수로 대체하여 표현하더라도 오차는 크지 않다.

[0110] 도4 에서는 양의 정수 범위(0 ~ 128)만을 고려하였으나, 음의 정수 범위(0 ~ -128)에 대해서도 동일하게 표현될 수 있음은 자명하다. 즉 가중치가 8비트의 부호화 정수인 경우, 2개의 부분곱 또는 3개의 부분곱으로 표현 가능한 정수로 대체하더라도, 오차는 10% 미만으로 나타난다.

[0111] 따라서 사전에 시뮬레이션을 통해, 가중치(w_i)를 표현하는 부분곱의 개수의 변화에 따른 인공 신경망의 오차를 검토하고, 검토된 오차가 미리 지정된 허용 오차 범위 이내인 경우에, 디지털 뉴런은 적은 개수의 부분곱과 2의 보수의 조합으로 표현 가능한 정수로 표현되는 가중치(w_i)를 인가받도록 구성될 수 있다.

[0112] 표3 은 심층 컨볼루션 신경망 중 하나인 LeNet-5가 가중치(w_i)에 따라 MNIST에서 제공된 필기된 숫자를 판별한 추측 정확도를 시뮬레이션한 결과를 나타낸다.

표 3

[0113]

	Inference accuracy
32-bit floating-point representation	99.10 %
8-bit integer weights	99.10 %
5-bit integer weights	98.95 %
5-bit integer weights except for ±13 and ±11	98.92 %

[0114] 표3 을 참조하면, 가중치(w_i)가 32비트의 단정밀도 부동 소수점 포멧인 경우와 8비트 정수 포멧인 경우에 LeNet-5의 추측 정확도는 99.10%로 동일하게 나타남을 알 수 있다. 그리고 가중치(w_i)가 5비트 정수 포멧인 경우에 98.95%의 정확도를 나타내고, ±11 및 ±13가 제외된 5비트 정수 포멧인 경우에는 98.92%의 정확도를 나타낸다. 즉 가중치(w_i)가 ±11 및 ±13이 제외된 5비트 정수 포멧으로 인가되더라도, 인공 신경망의 성능에 미치는 영향은 크지 않음을 알 수 있다.

[0115] 인공 신경망의 학습 장치는 부동 소수점 포멧의 가중치를 5비트의 정수로 변환하여 인공 신경망의 디지털 뉴런으로 전달할 수 있으며, 변환 과정에서 제외되도록 지정된 값(±11, ±13)으로 가중치가 획득되면, 이를 인접한 다른 정수로 변환하여 전달할 수 있다. 일예로, 학습 장치는 학습된 가중치(w_i)가 10.7, 11.3인 경우, 각각 10 및 12로 변환할 수 있다.

[0116] 따라서 디지털 뉴런의 신경소자(NE)에 인가되는 가중치(w_i)는 상기한 바와 같이, 일부 지정된 수를 제외한 기지정된 비트 수를 갖는 정수일 수 있다.

[0117] 가중치(w_i)가 2개의 부분곱의 합으로 표현될 수 있으므로, 부분곱 생성기(MBS) 각각은 수학식 3과 같이 2개의 부분곱 회로와 2의 보수 회로 및 가중치 0 계산기를 포함하여 구현될 수 있다. 그리고 N개의 부분곱 생성기(MBS) 각각은 2개의 부분곱(($P1_1$, $P2_1$), ($P1_2$, $P2_2$), …, ($P1_N$, $P2_N$))을 부분곱 가산기(PSUM)로 출력한다.

[0118] 부분곱 가산기(PSUM)은 N개의 부분곱 생성기(MBS)에서 인가되는 2N개의 (($P1_1$, $P2_1$), ($P1_2$, $P2_2$), …, ($P1_N$, $P2_N$))과 바이어스 값(BS)를 합산하여 출력값(Out)을 출력한다.

[0119] 여기서 출력값(Out)은 2차원 가중치 필터가 2차원 입력 특징 맵의 일부 영역에 대해 내적 연산한 결과로서, 2차원의 출력 특징 맵의 원소이다.

[0120] 인공 뉴런은 컨볼루션 연산을 수행하기 위해, 2차원 가중치 필터를 2차원 입력 특징 맵 상에서 이동시켜 다시 가중치 필터와 대응하는 2차원 입력 특징 맵을 내적 연산할 수 있다.

[0121] 도5 는 도3 의 인공 뉴런에서 신경소자의 상세 구성의 일예를 나타낸다.

[0122] 도5 를 참조하여, 디지털 뉴런의 신경소자(NE)를 설명하면, 신경소자(NE)는 N개의 가중치 분해부(WDC)와 N개의 부분곱 생성기(MBS), 부분곱 가산기(PSUM) 및 네거티브 부분곱 카운터(NCCP)를 포함한다.

[0123] 상기한 바와 같이, 신경소자(NE)는 2차원 입력 특징 맵을 인가받아 2차원의 가중치 필터로 내적 연산을 수행한다. 그리고 2차원 입력 특징 맵의 N개의 입력값(X)은 m 비트(예를 들면, 8비트)의 정수 값을 가지며, 가중치 필터의 N개의 가중치(w_i)는 지정된 정수(예를 들면, ±11 및 ±13)가 제외된 기지정된 비트 수(n)(예를 들면, 5비트)의 정수 값을 갖는 것으로 가정한다.

[0124] N개의 가중치 분해부(WDC) 각각은 N개의 가중치 중 대응하는 가중치(w_i)를 인가받아, 계수(w_{b1}, w_{b2})와 2의 승수(2^i, 2^j)의 부분곱($w_{b1}2^i$, $w_{b2}2^j$) 형태로 분해하고, 분해된 가중치($w_{b1}2^i$, $w_{b2}2^j$)에 따라 N개의 부분곱 생성기(MBS) 중 대응하는 부분곱 생성기(MBS_i)의 동작을 제어한다. 이때, 가중치 분해부(WDC)는 가중치(w_i)를 부분곱의 개수가 최소화되도록 2의 승수(2^n)와 2의 보수의 조합으로 구성된 표1 및 표2 에 따라 가중치(w_i)를 분해하고, 부분곱 생성기(MBS_i)의 동작을 판별한다. 그리고 판별된 동작을 수행하도록 부분곱 생성기(MBS_i)를 제어하여, 부분

곱 생성기(MBS_i)가 가중치(w_i)와 입력값(X)의 곱셈을 수행할 수 있도록 한다.

[0125] 가중치 분해부(WDC)는 표1 및 표2 를 이용하여 가중치(w_i)에 대응하여, 입력 신호(X)가 시프트되어야 하는 비트 수를 판별한다. 그리고 판별된 비트수에 따라 시프트 제어 신호(SH1, SH2)를 출력한다. 여기서 시프트되어야 하는 비트 수는 분해된 가중치(w_i)의 2의 승수(2^i, 2^j)에서 지수(i, j)에 대응한다.

[0126] 또한 가중치 분해부(WDC)는 표1 및 표2 로부터 분해된 가중치(w_i)의 계수(w_{b1}, w_{b2})로부터 네거티브 부분곱이 요구되는, 즉 2의 보수를 취해야 하는 값을 판별한다. 가중치 분해부(WDC)는 계수(w_{b1}, w_{b2})가 음수, 즉 -1 인 경우, 2의 보수를 취하는 것으로 판단한다. 그리고 네거티브 부분곱이 필요한 값에 따라 반전 제어 신호(INV1, INV2)를 출력한다.

[0127] 한편 가중치 분해부(WDC)는 분해된 가중치(w_i)의 계수(w_{b1}, w_{b2})가 0의 값을 갖는 경우, 제로 연산 신호(ALZ1, ALZ2)를 출력한다. 그리고 가중치 분해부(WDC)는 N개의 부분곱 생성기(MBS)로 인가되는 모든 반전 제어 신호(INV1, INV2)를 네거티브 부분곱 카운터(NCCP)로 전송한다.

[0128] 가중치 분해부(WDC)는 표1 및 표2 를 기반으로 가중치(w_i)에 따른 시프트 제어 신호(SH1, SH2), 반전 제어 신호(INV1, INV2) 및 제로 연산 신호(ALZ1, ALZ2)를 생성하기 위한 적어도 하나의 룩업 테이블(Look-Up Table)을 포함할 수 있다. 도5 에서는 일예로 가중치 분해부(WDC)가 시프트 제어 신호(SH1, SH2), 반전 제어 신호(INV1, INV2) 및 제로 연산 신호(ALZ1, ALZ2) 각각을 생성하기 위한 3개의 룩업 테이블을 포함하는 것으로 가정하여 도시하였다.

[0129] N개의 부분곱 생성기(MBS) 각각은 수학식 3과 같이 2개의 비트별 부분곱 및 2의 보수 연산을 수행하여, 입력값(X)과 가중치(w_i)의 곱셈 연산을 수행할 수 있다. 이에 N개의 부분곱 생성기(MBS) 각각은 2개의 부분곱 계산기(PP1, PP2)를 포함한다. 본 실시예에서는 가중치 분해부(WDC)가 가중치(w_i)를 2개의 부분곱($w_{b1}2^i$, $w_{b2}2^j$) 형태로 분해하는 것으로 가정함에 따라, N개의 부분곱 생성기(MBS) 각각이 2개의 부분곱 계산기(PP1, PP2)를 포함하는 것으로 도시되었다. 그러나 가중치 분해부(WDC)가 가중치(w_i)를 3개 이상의 부분곱($w_{b1}2^i$, $w_{b2}2^j$) 형태로 분해하는 경우, N개의 부분곱 생성기(MBS) 각각은 분해된 가중치의 부분곱 개수에 대응하는 개수의 부분곱 계산기를 포함하도록 구성된다.

[0130] 2개의 부분곱 계산기(PP1, PP2)는 각각 비트 시프터(BSH), 1의 보수 연산기(1CMP) 및 제로 곱셈기(ZOP)를 포함하여, 가중치(w_i)의 특정 비트와 입력값(X)을 부분곱한 부분곱(P1, P2)을 출력한다.

[0131] 비트 시프터(BSH)는 가중치 분해부(WDC)에서 인가되는 시프트 제어 신호(SH1, SH2)에 응답하여, 입력값(X)을 상위 비트 방향으로 비트 천이 시킨다. 즉 입력값(X)에 분해된 가중치($w_{b1}2^i$, $w_{b2}2^j$)의 2의 승수(2^i, 2^j), 즉 특정 비트를 곱한다. 여기서 비트 시프터(BSH)는 일예로 배럴 시프터(Barrel shifter)로 구현될 수 있다. 비트 시프터(BSH)는 표1 및 표2 에 나타난 바와 같이, 분해된 가중치($w_{b1}2^i$, $w_{b2}2^j$)에서 2의 승수(2^i, 2^j)의 지수(i, j)에 대응하여, 입력값(X)을 상위 비트 방향으로 비트 천이 시킨다. 시프트 제어 신호(SH1, SH2)의 비트 수(q)는 표1 및 표2 에 나타난 가중치(w_i)에 대해 비트 천이(BSH)가 시프트시키는 최대 범위에 따라 결정된다.

[0132] 그리고 1의 보수 연산기(1CMP)는 가중치 분해부(WDC)에서 인가되는 반전 제어 신호(INV1, INV2)에 응답하여, 비트 시프터(BSH)에서 인가되는 값을 반전하여 1의 보수를 계산한다.

[0133] 제로 곱셈기(ZOP)는 가중치 분해부(WDC)에서 인가되는 제로 연산 신호(ALZ1, ALZ2)에 응답하여, 1의 보수 연산기(1CMP)에서 인가되는 값을 0으로 전환하여 출력한다. 제로 곱셈기(ZOP)는 가중치(w_i)가 0으로 인가된 경우에, 입력값(X)에 무관하게 0의 값을 출력하기 위해 추가된 구성이다.

[0134] 여기서 2개의 부분곱 계산기(PP1, PP2)가 2의 보수 연산기 대신 1의 보수 연산기를 포함하는 것은 부분곱 계산기(PP1, PP2)의 구조를 더욱 간략화하기 위해서이다.

[0135] 2의 보수 연산기는 인가된 모든 비트값을 반전한 후 1을 더하는 연산을 수행해야 한다. 즉 2의 보수연산기는 인가된 값의 비트별 반전을 수행하는 반전 회로와 더불어 가산기가 필요하다.

[0136] 그에 반해 1의 보수 회로는 인가된 값의 모든 비트값을 단순히 반전하여 출력하므로, 가산기가 필요하지 않다. 따라서 2의 보수 회로에 비해 1의 보수 회로는 회로 구성이 간단하다.

[0137] 가산기 1개의 크기나 전력 소모는 크지 않다. 그러나 신경소자(NE)가 N개의 부분곱 생성기(MBS)를 포함하고, N개의 부분곱 생성기(MBS) 각각이 2개의 부분곱 계산기(PP1, PP2)를 포함하므로, 인공 뉴런은 N * 2 개씩의 2의 보수 회로를 포함해야 한다. 따라서 2의 보수 회로를 1의 보수 회로로 전환하면, 디지털 뉴런의 하나의 신경소자(NE)에서 N * 2개의 가산기를 줄일 수 있다. 즉 인공 뉴런의 크기를 크게 줄일 수 있고, 연산 속도를 향상시킬 수 있으며, 전력 소모를 절감할 수 있다.

[0138] 다만, 부분곱 계산기(PP1, PP2)가 2의 보수 연산기 대신 1의 보수 연산기를 포함함에 따라, 부분곱 계산기(PP1, PP2)에서 출력되는 부분곱(P1, P2)의 합은 가중치(w_i)와 입력값(X)의 곱셈값과 차이가 발생할 수 있다. 이러한 차이는 부분곱(P1, P2) 중 네거티브 부분곱을 계산할 때, 2의 보수가 가 아닌 1의 보수가 이용되어 1이 추가되지 않았기 때문에 발생한다.

[0139] 이에 본 실시예에서는 네거티브 부분곱 카운터(NCCP)를 별도로 포함하여, 부분곱 계산기(PP1, PP2)가 2의 보수가 아닌 1의 보수 연산을 수행하여, 가산되지 않은 1을 계산하도록 한다. 네거티브 부분곱 카운터(NCCP)는 가중치 분해부(WDC)에서 인가되는 반전 제어 신호(INV1, INV2)를 카운트한다. 여기서 카운트된 값은 N개의 가중치(w_i)에서 표1 및 표2 에 따라 네거티브 부분곱이 요구되는 개수를 의미한다. 즉 네거티브 부분곱 카운터(NCCP)는 신경소자(NE) 전체에서 2의 보수 연산에 추가되어야 하는 1의 값을 일괄로 카운트하여 카운트 값(NUM_P)을 획득한다. 그리고 카운트 값(NUM_P)을 부분곱 가산기(PSUM)으로 전달한다.

[0140] 부분곱 가산기(PSUM)는 N개의 부분곱 생성기(MBS) 각각으로부터 2개씩의 부분곱($P1_i$, $P2_i$)을 인가받고, 네거티브 부분곱 카운터(NCCP)로부터 카운트 값(NUM_P)을 인가받으며, 바이어스 값(BS)를 인가받아 모두 합산하여, 출력값(Out)를 출력한다.

[0141] 도6 은 도5 의 부분곱 가산기의 상세 구성의 일예를 나타낸다.

[0142] 상기한 바와 같이, 부분곱 가산기(PSUM)는 N개의 부분곱 생성기(MBS) 각각으로부터 2개씩의 부분곱($P1_i$, $P2_i$)을 인가받으며, 여기서는 N은 25인 경우를 가정하여 설명한다. N이 25이면, 부분곱 가산기(PSUM)는 50(= 2N)개의 부분곱(($P1_1$ ~ $P1_N$), ($P2_1$ ~ $P2_N$))과 카운트 값(NUM_P) 및 바이어스 값(BS)를 인가받아 합산하여, 출력값(Out)를 출력한다.

[0143] 본 실시예에서 부분곱 가산기(PSUM)는 기본적으로 월러스 트리 가산기(Wallace Tree Adder)와 유사한 방식으로 덧셈 연산을 수행한다. 본 실시예에서 부분곱 가산기(PSUM)는 가산해야 하는 부분곱의 개수에 대응하는 다수의 스테이지(ST1 ~ STk)와 CLA 가산기(CAL)를 포함하고, 다수의 스테이지(ST1 ~ STk) 각각은 다수의 전가산기(Full Adder: FA)를 포함하는 다수의 가산 그룹(GP)을 포함한다.

[0144] 제1 스테이지(ST1)에서 다수의 가산 그룹(GP) 각각은 2N개의 부분곱(($P1_1$ ~ $P1_N$), ($P2_1$ ~ $P2_N$))에서 동일위치의 비트들을 기설정된 a개씩 그룹화하여 가산한다. 2N개의 부분곱(($P1_1$ ~ $P1_N$), ($P2_1$ ~ $P2_N$)) 각각이 j비트(j는 자연수, 일예로 j = m+n)의 값인 경우, 각각의 가산 그룹(GP)는 j개의 전가산기(FA)를 포함할 수 있다.

[0145] 일예로 제1 스테이지(ST1)의 제1 가산 그룹(GP)의 전가산기(FA)들은 3개의 부분곱($P1_1$, $P1_2$, $P1_3$)의 동일 비트를 입력으로 인가받아 가산할 수 있다. 그리고 제2 가산 그룹(GP)의 전가산기(FA)들 또한 3개의 부분곱($P1_4$, $P1_5$, $P1_6$)의 동일 비트를 입력으로 인가받아 가산할 수 있다.

[0146] 그러나 경우에 따라서 제1 스테이지(ST1)의 다수의 가산 그룹(GP) 각각의 전가산기(FA)들은 5개의 부분곱($P1_1$, $P1_2$, $P1_3$, $P1_4$, $P1_5$)의 동일 비트를 입력으로 인가받아 가산할 수도 있다. 즉 각 가산 그룹(GP)가 부분곱의 동일 비트를 인가받아 가산하는 개수는 다양하게 조절될 수 있다.

[0147] 제1 스테이지(ST1)가 2N개의 부분곱(($P1_1$ ~ $P1_N$), ($P2_1$ ~ $P2_N$))을 a개씩 그룹화하여 가산을 수행하는 경우, 제1 스테이지(ST1)에는 A(= 2N/a)개의 가산 그룹(GP)이 포함된다.

[0148] 그리고 제2 스테이지(ST2) 이후 단의 스테이지는 이전 스테이지에서 가산된 결과를 인가받아 다시 a개씩 그룹화하여 가산을 수행한다. 이때 도4 에 도시된 바와 같이 전가산기(FA)가 a비트의 입력을 인가받아 캐리(carry)

및 합값(sum)의 2비트를 출력하므로, 각 스테이지(ST2 ~ STk-1)는 가산 그룹(GP)의 수가 순차적으로 2/a씩 줄어들게 된다.

[0149] 일예로 각 가산 그룹(GP)의 전가산기(FA)가 3비트의 입력을 받아 2비트를 출력하도록 구성되고 부분곱의 개수(2N)가 50개인 경우, 제1 스테이지(ST1)는 17(50/3 = 16.6)개의 가산 그룹(GP)을 포함한다. 그리고 제2 스테이지(ST2)는 12개(17* 2/3 = 11.3)의 가산 그룹(GP)를 포함하고, 제3 스테이지(ST3)는 8개의 가산 그룹(GP)을 포함한다. 스테이지별로 가산 그룹(GP)의 수가 2/3씩로 줄어들므로, 제7 스테이지(ST7)에서 가산 그룹(GP)의 수는 2가 된다.

[0150] 그리고 각 스테이지(ST1 ~ STk) 로 가산 결과가 전이되는 동안, 가산 결과의 비트 폭은 1비트씩 확장된다. 예로서, 2N개의 부분곱(($P1_1$ ~ $P1_N$), ($P2_1$ ~ $P2_N$)) 각각이 j비트의 값인 경우, 제1 스테이지(ST1)의 A개의 가산 그룹(GP)은 각각 j개의 전가산기(FA)를 포함하여, A개의 j비트 가산 값을 출력한다. 그러나 제2 스테이지(ST2)의 B개의 가산 그룹(GP) 각각은 j+1개의 전가산기(FA)를 포함하여, B개의 j+1비트 가산값을 출력한다. 또한 제3 스테이지(ST3)의 C(C = 2B/a)개의 가산 그룹(GP) 각각은 j+2개의 전가산기(FA)를 포함하여, C개의 j+2비트 가산값을 출력한다.

[0151] 상기한 부분곱 가산기(PSUM)의 2N개의 부분곱(($P1_1$ ~ $P1_N$), ($P2_1$ ~ $P2_N$))에 대한 가산 방법은 월러스 트리 가산기와 유사하다. 그러나 이러한 부분곱 가산기(PSUM)의 구조는 입력되는 2N개의 부분곱(($P1_1$ ~ $P1_N$), ($P2_1$ ~ $P2_N$))이 자연수인 경우에 동일한 것으로 네거티브 부분곱에 대한 부호 비트가 고려되어 있지 않은 것이다.

[0152] 부분곱 가산기(PSUM)가 개별 부호가 있는 숫자의 합계를 계산하고, 부호 비트(s)를 포함하여 채널 연산값(MO)을 출력하는 경우, j비트의 2N개의 부분곱(($P1_1$ ~ $P1_N$), ($P2_1$ ~ $P2_N$)) 각각은 부호 비트(1비트)가 s비트(일예로 6비트)만큼 더 확장되어 j+s 비트의 값으로 입력되어야 한다. 즉 부호 비트로 s비트(s= T-1)가 추가로 확장되어야 한다.

[0153] 이 경우, 부분곱 가산기(PSUM)의 각 스테이지(ST1 ~ STk)에서 각각의 가산 그룹(GP)은 s개의 전가산기(FA)를 더 구비해야 하며, 이는 부분곱 가산기(PSUM)의 회로 면적을 약 21% 증가시키게 되는 문제가 있다.

[0154] 이에 본 실시예에 따른 부분곱 가산기(PSUM)는 확장되어야 하는 부호 비트의 값을 별도로 인가받아 최종 스테이지(STk)에서 가산되도록 함으로써, 부분곱 가산기(PSUM)의 크기와 전력 소모를 줄일 수 있도록 한다.

[0155] 이를 위해, 부분곱 가산기(PSUM)는 네거티브 부분곱 카운터(NCCP)에서 인가되는 카운트 값(NUM_P)을 2의 보수로 변환하는 2의 보수 연산기(2CMP)를 더 포함하고, 2의 보수 연산기(2CMP)는 카운트 값(NUM_P)에 대한 2의 보수를 연산하여, 카운트 보수값(N_NUM_P)을 출력한다. 이때 카운트 보수값(N_NUM_P)의 비트수(T)는 확장되는 부호 비트 수에 대응하는 비트 수를 갖는다.

[0156] 2의 보수 연산기(2CMP)가 카운트 값(NUM_P)을 2의 보수로 변환하는 것은, 카운트 값(NUM_P)을 2의 보수가 2N개의 부분곱(($P1_1$ ~ $P1_N$), ($P2_1$ ~ $P2_N$)) 전체에서 네거티브 부분곱의 부호 비트를 모두 더한 값과 동일하기 때문이다.

[0157] 도7 은 도6 의 2의 보수 연산기의 동작 개념을 설명하기 위한 도면이다.

[0158] 도7 에서는 일예로, 부분곱 가산기(PSUM)가 5비트 이진 정수 포멧의 6개의 부분곱(11101, 10110, 10010, 00101, 10101, 10000)을 가산하여 출력하는 경우를 가정하여 설명한다. 도7 에서 6개의 부분곱 중 사인비트가 1인 5개의 부분곱(11101, 10110, 10010, 10101, 10000)은 음수로서 네거티브 부분곱이고, 사인비트가 0인 1개의 부분곱(00101)은 양수로서 포지티브 부분곱이다.

[0159] 그리고 6개의 부분곱은 부호 비트가 s비트(일예로 6비트)만큼 더 확장되어 j+s 비트(도7 에서는 11비트)의 값으로 전환된다. 확장된 부호 비트는 결국 부호 비트와 동일한 비트값을 갖고 확장된다.

[0160] 그러나 확장된 전체 부호비트의 값은 5개의 -1의 값과 1개의 0의 값의 합이므로, (-1) X 5 + 0 X 1 = -5 와 동일하다. 즉 음수의 개수, 즉 네거티브 부분곱의 개수(5)에 2의 보수를 취한 값(-5)과 동일하다.

[0161] 이는 5비트 이진 정수 포멧의 6개의 부분곱(11101, 10110, 10010, 00101, 10101, 10000)을 가산할 때, 부호 비트 부분을 제외한 2비트의 양의 이진 정수 포멧을 가산하고, 이후, 네거티브 부분곱의 개수(5)에 2의 보수를 취한 값(-5)을 별도로 가산해도 동일하다는 것을 의미한다.

[0162] 이에 본 발명의 실시예에 따른 부분곱 가산기(PSUM)는 2의 보수 연산기(2CMP)를 포함하여, 네거티브 부분곱 카

운터(NCCP)에서 인가되는 카운트 값(NUM_P)을 2의 보수로 변환하여, 확장되는 부호 비트 수에 대응하는 비트 수를 갖는 카운트 보수값(N_NUM_P)을 획득함으로써, 부분곱 가산기(PSUM)의 크기와 전력 소모를 줄일 수 있도록 한다.

[0163] 다시 도6 을 참조하면, 2의 보수 연산기(2CMP)에서 출력되는 카운트 보수값(N_NUM_P)은 도4 에 도시된 바와 같이, 최종 스테이지(STk)에서 이전 스테이지까지 누적된 합산 결과의 상위 비트에 대응하는 전가산기(FA)의 다수의 입력 중 하나의 입력으로 인가되어 가산된다.

[0164] 한편 본 실시예에 따른 부분곱 가산기(PSUM)는 부분곱 생성기(MSB)의 부분곱 계산기(PP1, PP2)가 2의 보수가 아닌 1의 보수를 이용하여 부분곱을 수행하였으므로, 네거티브 부분곱 카운터(NCCP)에서 인가되는 카운트 값(NUM_P)을 추가로 가산해야 한다. 이에 도4 에 도시된 바와 같이, 카운트 값(NUM_P)의 각 비트는 최종 스테이지(STk)의 하위 비트에 대응하는 전가산기(FA)의 다수의 입력 중 하나의 입력으로 인가되어 가산된다.

[0165] 또한 부분곱 가산기(PSUM)는 최종 스테이지(STk) 이전의 스테이지에 바이어스 값(BS)을 인가받아 추가로 가산한다. 이때 바이어스 값(BS)은 도6 에 도시된 바와 같이, 부호 비트가 추가 확장된 비트수(BS'[j-1+s:0])를 갖는 2진 포멧의 값으로 전환되어 인가될 수 있다. 추가 확장된 비트수를 갖는 2진 포멧의 바이어스 값(BS')의 각 비트는 전가산기(FA)의 다수의 입력 중 하나의 입력으로 인가되어 가산된다.

[0166] 그리고 CLA 가산기(CLA)는 최종 스테이지(STk)에서 가산된 각 비트값을 인가받아 가산하여 최종적으로 출력값(Out[j+s:0])을 출력한다.

[0167] 결과적으로, 본 실시예에 따른 부분곱 가산기(PSUM)는 고속 덧셈 연산이 가능한 월러스 트리 가산 방식을 이용하여, 고속으로 연산할 수 있다. 그리고 부호 비트에 대한 연산을 네거티브 부분곱 카운터(NCCP)에서 인가되는 카운트 값(NUM_P)에 대한 2의 보수로 계산하여, 별도로 상위 비트에 가산함으로써, 회로 크기 및 전력 소비를 줄일 수 있다. 또한 카운트 값(NUM_P)을 하위 비트에 가산함으로써, 부분곱 생성기(MBS)의 부분곱 계산기(PP1, PP2)들이 2의 보수 연산기(2CMP)가 아닌 1의 보수 연산기(1CMP)를 구비할 수 있도록 한다.

[0168] 도8 은 디지털 뉴런의 신경소자의 다른 예를 나타낸다.

[0169] 도8 의 신경소자(NE)는 도5 의 신경소자(NE)와 마찬가지로 N개의 가중치 분해부(WDC)와 N개의 부분곱 생성기(MBS), 부분곱 가산기(PSUM) 및 네거티브 부분곱 카운터(NCCP)를 포함한다.

[0170] N개의 가중치 분해부(WDC) 각각은 N개의 가중치 중 대응하는 가중치(w_i)를 인가받아, 계수(R_a, R_b, R_c)와 2의 승수(2^a, 2^b, 2^c)의 부분곱($R_a \cdot 2^a$, $R_b \cdot 2^b$, $R_c \cdot 2^c$) 형태로 분해($w_i = R_a \cdot 2^a + R_b \cdot 2^b + R_c \cdot 2^c$ (여기서 계수(R_a, R_b, $R_c \in \{-1, 0, 1\}$))하고, 분해된 가중치($R_a \cdot 2^a$, $R_b \cdot 2^b$, $R_c \cdot 2^c$)에 따라 N개의 부분곱 생성기(MBS) 중 대응하는 부분곱 생성기(MBS_i)의 동작을 제어한다.

[0171] 가중치 분해부(WDC)는 분해된 가중치($R_a \cdot 2^a$, $R_b \cdot 2^b$, $R_c \cdot 2^c$)에서 계수(R_a, R_b, R_c)와 2의 승수(2^a, 2^b, 2^c)의 지수(a, b, c)를 대응하는 부분곱 생성기(MBS_i)로 전달하여, 부분곱 생성기(MBS_i)의 동작을 제어할 수 있다.

[0172] 도8 에서는 일예로 가중치 분해부(WDC)가 도5 의 가중치 분해부(WDC)와 달리 가중치(w_i)를 3개의 부분곱($R_a \cdot 2^a$, $R_b \cdot 2^b$, $R_c \cdot 2^c$)으로 분해하는 것으로 가정하였으나, 가중치 분해부(WDC)는 도5 에서와 마찬가지로 가중치(w_i)를 2개의 부분곱으로 분해하도록 설정될 수도 있다.

[0173] N개의 부분곱 생성기(MBS) 각각은 입력값(X_i)과 네거티브 입력값($\overline{X_i}+1$)을 인가받는다. 여기서 네거티브 입력값($\overline{X_i}+1$)은 입력값(X_i)의 2의 보수값이다. 그리고 N개의 부분곱 생성기(MBS) 각각은 3개의 부분곱 계산기(PP1 ~ PP3)를 포함할 수 있다. 이는 상기한 바와 같이, 가중치 분해부(WDC)가 가중치(w_i)를 3개의 부분곱($R_a \cdot 2^a$, $R_b \cdot 2^b$, $R_c \cdot 2^c$)으로 분해하는 것으로 가정하였기 때문이다.

[0174] 3개의 부분곱 계산기(PP1 ~ PP3) 각각은 먹스(PMX)와 비트 시프터(BSH)를 포함한다. 먹스(PMX)는 입력값(X_i)

과 네거티브 입력값($\overline{X_i}$+1)에 분해된 가중치($R_a \cdot 2^a$, $R_b \cdot 2^b$, $R_c \cdot 2^c$)에서 계수(R_a, R_b, R_c)를 곱하기 위한 구성으로, 가중치 분해부(WDC)에서 인가되는 계수(R_a, R_b, R_c) 중 대응하는 계수에 응답하여, 입력값(X_i)과 네거티브 입력값($\overline{X_i}$+1) 및 접지 전원 중 하나를 선택하여 비트 시프터(BSH)로 전달한다.

[0175] 상기한 바와 같이, 계수(R_a, R_b, R_c)는 -1, 0, 1의 값 중 하나로 지정될 수 있다. 도8 의 부분곱 계산기(PP1 ~ PP3)에서 먹스(PMX)는 계수(R_a, R_b, R_c)가 -1인 경우, 입력값(X_i)을 반전하는 도5 와 달리, 반전되어 인가되는 네거티브 입력값($\overline{X_i}$+1)을 선택함으로써, 네거티브 부분곱을 획득할 수 있도록 한다. 또한, 계수(R_a, R_b, R_c)가 0인 경우에 제로 곱셈을 수행하지 않고, 논리 0을 나타내는 접지 전원을 선택하여 출력하도록 한다. 먹스(PMX)가 네거티브 입력값($\overline{X_i}$+1) 및 논리 0의 접지 전원을 선택할 수 있도록 구성됨에 따라 도8 의 부분곱 계산기(PP1 ~ PP3)는 1의 보수 연산기(1CMP) 및 제로 곱셈기(ZOP)를 포함하지 않아도 무방하다.

[0176] 비트 시프터(BSH)는 먹스(PMX)에서 선택된 값을 인가받고, 가중치 분해부(WDC)에서 인가되는 지수(a, b, c)에 응답하여, 인가된 값을 상위 비트 방향으로 비트 천이 시켜서 부분곱(P1, P2, P3)을 출력한다.

[0177] 부분곱 가산기(PSUM)는 동일하게 N개의 부분곱 생성기(MBS) 각각으로부터 3개씩의 부분곱($P1_i$, $P2_i$, $P3_i$)을 인가받고, 바이어스 값(BS)를 인가받아 모두 합산하여, 출력값(Out)를 출력한다.

[0178] 도9 는 본 발명의 다른 실시예에 따른 인공 신경망의 디지털 뉴런에 대한 블록 다이어그램을 나타내고, 도10 은 도9 의 인공 뉴런에서 신경소자의 상세 구성의 일예를 나타내며, 도11 은 도10 의 부분곱 가산기의 상세 구성의 일예를 나타낸다.

[0179] 도3 에서는 일예로 디지털 뉴런이 하나의 2차원 입력 특징 맵과 2차원 입력 특징 맵에 대응하는 하나의 2차원의 가중치 필터를 인가받는 것으로 가정하여, 디지털 뉴런이 하나의 신경소자(NE)를 포함하는 것으로 도시하였다.

[0180] 그러나 도1 에 나타난 바와 같이, 컨볼루션 레이어(C1, C2, C3)의 인공 뉴런은 3차원 입력 특징 맵을 인가받고, 3차원 가중치 필터를 이용하여 3차원 컨볼루션 연산을 수행할 수 있다.

[0181] 여기서 3차원 입력 특징 맵은 M개(여기서 M은 자연수)의 2차원 입력 특징 맵으로 구성될 수 있으며, M개의 2차원 입력 특징 맵은 3차원 입력 특징 맵의 M개의 채널이다. 이때 M개의 2차원 입력 특징 맵으로 구성되는 3차원 입력 특징 맵은 이전 레이어에서 출력되는 출력 특징 맵의 일부 일 수 있다. 일예로 이전 레이어에서 출력되는 출력 특징 맵은 L(L > M인 자연수)(도1 에서는 일예로 32개) 개의 2차원 출력 특징 맵으로 구성될 수 있다. 그리고 인공 뉴런은 L개의 2차원 출력 특징 맵에서 M개의 2차원 출력 특징 맵을 선택하여 3차원 입력 특징 맵을 구성하는 M개의 입력 특징 맵으로서 인가받을 수 있다.

[0182] 또한 동일한 컨볼루션 레이어에서 다수의 인공 뉴런 각각은 서로 다른 개수의 2차원 입력 특징맵으로 구성된 3차원 입력 특징 맵을 인가받을 수 있다.

[0183] 한편 3차원 입력 특징 맵의 M개의 채널에 대응하여, 3차원의 가중치 필터 또한 M개의 2차원 가중치 필터를 갖는다. 여기서 M개의 2차원 입력 특징 맵 각각은 이전 컨볼루션 레이어의 인공 뉴런 각각에서 생성된 출력 특징 맵일 수 있다.

[0184] 이에 도9 내지 도11 에서는 디지털 뉴런이 M개의 2차원 입력 특징 맵과 M개의 2차원 가중치 필터를 인가받도록 구성된 경우를 도시하였으며, 이에 디지털 뉴런은 M개의 신경소자(NE)를 포함하도록 구성되었다.

[0185] 도9 에 도시된 M개의 신경소자(NE) 각각은 기본적으로 도3 에 도시된 신경소자(NE)와 유사한 구성을 갖는다. 다만, 도9 및 도10 에 도시된 디지털 뉴런에서 M개의 신경소자(NE) 각각은 바이어스값(BS)를 합산하지 않고, N개의 부분곱 생성기(MBS)에서 인가되는 2N개의 (($P1_1$, $P2_1$), ($P1_2$, $P2_2$), …, ($P1_N$, $P2_N$))만을 합산하여 채널 연산값(MO_1 ~ MO_M)을 출력한다.

[0186] 그리고 도9 의 디지털 뉴런은 M개의 신경소자(NE) 각각에서 출력되는 M개의 채널 연산값(MO_1 ~ MO_M)과 바이어스값(BS)를 합산하여, 출력값(Out)을 출력한다. 여기서 출력값(Out)는 3차원 가중치 필터가 3차원 입력 특징 맵의 일부 영역에 대해 내적 연산한 결과로서, 2차원의 출력 특징 맵의 원소이다.

[0187] 디지털 뉴런은 이후, 3차원 가중치 필터를 3차원 입력 특징 맵 상에서 이동시켜 다시 내적 연산하여, 2차원의

출력 특징 맵의 다른 원소를 계산할 수 있다.

[0188] 도10 에서는 일예로서 다수의 신경소자(NE)가 도3 과 동일한 구성을 갖는 것으로 도시하였으나, 다수의 신경소자(NE)는 도8 에 따를 구성을 가질 수도 있다.

[0189] 도12 는 본 실시예에 따른 디지털 뉴런의 전력 소비를 시뮬레이션 한 결과를 나타낸다.

[0190] 도12 에서는 일반적인 인공 신경망의 학습 및 실행을 위해 주로 이용되는 범용 연산 장치로서 다수의 상용 GPU와 본 실시예의 디지털 뉴런의 전력 소비를 비교하여 나타내었다. 여기서 본 실시예에 따른 디지털 뉴런을 상용 GPU와 비교한 것은 단순 연산을 반복하여 수행하는 인공 신경망에서는 CPU보다 GPU가 더욱 우수한 성능을 나타내고, 전력 소모를 적게 하기 때문이다.

[0191] 도12 를 참조하면, 본 실시예에 따라 하드웨어적으로 구현된 디지털 뉴런은 300mW 미만의 전력을 소비하는데 비해, 상용 GPU는 100W 이상의 전력을 소비함을 알 수 있다. 즉 상용 GPU에 비해 본 실시예의 디지털 뉴런은 전력 소모를 1/300 미만으로 줄일 수 있다.

[0192] 표4 는 본 실시예에 따른 디지털 뉴런의 성능을 다른 인공 뉴런과 비교한 결과를 나타낸다.

표 4

[0193]

	Adder tree with booth multiplier (N=50)	YodaNN (N=50) [13]	Proposed neural element (N=50)
Precision	5-bit integers (exact)	1-bit	5-bit integers (except for ±11 and ±13)
Top-1 accuracy degradation than single precision (ImageNet /AlexNet)	3.84 %	21.7 % [14]	3.86 %
Gate delay	57	40	36
Simulated power consumption	44.619 mW	13.199 mW	23.776 mW

[0194] 표4 를 참조하면, 본 실시예에 따른 디지털 뉴런은 5비트의 정수 가중치를 이용하여 부스 곱셈(booth multiplier)을 수행하는 방식과 유사한 정확도를 나타내는 반면, 게이트 지연 및 전력 소비가 크게 줄어듬을 알 수 있다. 또한 1비트의 정수 가중치를 이용하는 YodaNN 인공 뉴런에 비해 전력 소비는 높으나 정확도가 월등하게 높다는 것을 알 수 있다.

[0195] 결과적으로 본 실시예에 따른 인공 신경망을 위한 디지털 뉴런, 인공 뉴런 및 이를 포함하는 추론 엔진은 일부 값을 제외한 지정된 비트 수의 정수로 변환된 가중치를 이용하므로 부분곱의 개수를 최소화한다. 그리고 네거티브 부분곱에 대해 2의 보수가 아닌 1의 보수 연산을 수행하여 획득함으로써 부분곱 회로의 구조를 간략화 할 수 있다. 또한 채널 연산값(MO) 계산 시에 부호 비트값을 카운트 값(NUM_P)을 이용하여 별도로 계산한 후, 추가함으로써, 부분곱 가산기 구조를 간소화 할 수 있다. 따라서 고속 연산이 가능하며, 소비 전력을 줄일 수 있으며 저면적으로 구현될 수 있다.

[0196] 도13 은 본 발명의 실시예에 따른 디지털 뉴런을 이용하여 가중치 필터 크기를 가변할 수 있는 인공 뉴런의 일예를 나타낸다.

[0197] 도9 의 디지털 뉴런은 M개의 2차원 입력 특징 맵을 포함하는 3차원 입력 특징 맵을 인가받고, M개의 2차원 가중치 필터를 포함하는 3차원 가중치 필터를 이용하여 컨볼루션 연산 또는 내적 연산을 수행할 수 있도록 구성되었다.

[0198] 다만 도9 의 디지털 뉴런은 가중치 필터에 포함되는 가중치의 개수의 조절이 용이하지 않았다. 즉 가중치 필터

의 크기를 조절하기 용이하지 않았다.

[0199] 만일 가중치 필터가 5 X 5인 크기의 M개의 2차원 가중치 필터로 구성되는 경우, 도9 의 디지털 뉴런은 5 X 5인 크기 미만(예를 들면 3 X 3), M개 미만의 2차원 가중치 필터에 대해서는 내적 연산에 영향을 주지않는 더미 가중치(예를 들면 0)를 추가하여, 5 X 5인 크기의 M개의 2차원 가중치 필터로 변환하여 연산을 수행함으로써, 입력 특징 맵과 가중치 필터의 내적 연산을 수행할 수 있다.

[0200] 즉 디지털 뉴런은 디지털 하드웨어 설계 시에 지정된 크기 미만의 가중치 필터에 대해서는 용이하게 내적 연산 또는 컨볼루션 연산을 수행할 수 있다. 그러나 지정된 크기를 초과하는 가중치 필터에 대해서는 내적 연산 또는 컨볼루션 연산을 수행할 수 없다.

[0201] 이러한 문제를 방지하기 위해, 디지털 뉴런의 설계 시에 미리 연산 가능한 가중치 필터의 크기를 크게 확장하게 되면, 불필요한 다수의 연산을 수행할 뿐만 아니라, 디지털 뉴런의 크기와 전력 소모가 증가된다.

[0202] 도13 에서는 디지털 뉴런에 지정된 크기를 초과하는 가중치 필터를 이용하여 연산을 수행할 수 있으며, 지정된 크기 이하의 가중치 필터를 이용하여 연산을 수행하는 경우, 연산 속도를 향상시킬 수 있는 인공 뉴런의 구조를 개념적으로 도시한다.

[0203] 도13 을 참조하면, 본 실시예에 따른 인공 뉴런은 입력값 추출부(INEX), 다수의 디지털 뉴런(DN), 출력 가산기(AD), 적어도 하나의 입력 선택 먹스(Mux1) 및 출력 선택 먹스(Mux2)를 포함한다.

[0204] 입력값 추출부(INEX)는 입력 특징 맵에서 미리 지정된 위치의 지정된 크기의 입력값을 추출한다. 이때 입력값 추출부(INEX)는 모드 신호(MD)에 응답하여 서로 다른 위치의 입력값을 추출할 수 있으며, 모드 신호(MD)는 기본 모드 및 확장 모드를 지정할 수 있다.

[0205] 여기서 기본 모드는 다수의 디지털 뉴런(DN) 각각이 연산 가능한 지정된 크기 이하의 가중치 필터를 이용하여 연산을 수행하기 위한 모드이고, 확장 모드는 지정된 크기를 초과하는 가중치 필터를 이용하여 연산을 수행하기 위한 모드이다.

[0206] 도13 에서는 일예로 디지털 뉴런(DN)의 지정된 가중치 필터의 크기가 5 X 5 크기의 M개의 2차원 가중치 필터인 것으로 가정하였다. 따라서 기본 모드는 디지털 뉴런(DN)에 지정된 크기에 따라 가중치 필터가 5 X 5 크기 이하, M개 이하의 2차원 가중치 필터를 포함하는 경우에 설정될 수 있다.

[0207] 그러나 가중치 필터의 크기가 5 X 5 크기를 초과하거나 M개를 초과하는 2차원 가중치 필터를 포함하는 경우, 확장 모드가 지정된다.

[0208] 여기서는 가중치 필터가 5 X 5 크기의 M개의 2차원 가중치 필터로 구성되는 경우와, 7 X 7 크기의 M개의 2차원 가중치 필터로 구성되는 경우를 가정하여 설명한다.

[0209] 입력값 추출부(INEX)는 모드 신호(MD)가 기본 모드로 지정된 경우, 입력 특징 맵에서 가중치 필터의 크기에 대응하는 크기를 갖는 2개의 입력값 세트를 추출한다.

[0210] 기본 모드에서 입력값 추출부(INEX)는 각각 가중치 필터의 크기에 대응하는 2개의 입력값 세트를 추출한다.

[0211] 도13 을 참조하면, 기본 모드에서 입력값 추출부(INEX)는 입력 특징 맵의 제1 영역(IN_{T1})의 입력값과 제2 영역(IN_{T2})의 입력값을 제1 기본 입력값 세트로 추출하고, 제2 영역(IN_{T2})의 입력값과 제4 영역(IN_{T4})의 입력값을 제2 기본 입력값 세트로 추출한다.

[0212] 그러나 확장 모드에서 입력값 추출부는 입력 특징 맵의 제1 영역(IN_{T1})의 입력값과 제2 영역(IN_{T2})의 입력값을 제1 확장 입력값 세트로 추출하고, 제3 영역(IN_{T3})의 입력값과 제4 영역(IN_{T4})의 입력값을 제2 확장 입력값 세트로 추출한다.

[0213] 즉 제1 기본 입력값 세트와 제1 확장 입력값 세트는 동일한 입력값들이 추출되는 반면, 제2 기본 입력값 세트와 제2 확장 입력값 세트는 서로 다른 입력값들이 추출된다.

[0214] 여기서 추출된 제1 기본 입력값 세트와 제1 확장 입력값 세트는 다수의 디지털 뉴런 중 대응하는 제1 디지털 뉴런(DN1)으로 인가되고, 제2 기본 입력값 세트와 제2 확장 입력값 세트는 대응하는 제2 디지털 뉴런(DN2)로 인가된다.

[0215] 기본 모드에서 제1 및 제2 디지털 뉴런(DN1, DN2)은 각각 인가되는 제1 기본 입력값 세트와 제2 기본 입력값 세

트와 지정된 가중치 필터를 이용하여 내적 연산을 수행하여 출력값(Out_1, Out_2)를 출력한다. 여기서 제1 및 제2 디지털 뉴런(DN1, DN2)의 가중치 필터는 동일한 가중치 필터이다.

[0216] 즉 기본 모드에서 제1 및 제2 디지털 뉴런(DN1, DN2)은 입력 특징 맵상을 가중치 필터가 이동하면서 내적 연산을 수행하는 컨볼루션 연산에서 가중치 필터가 위치가 이동된 연산을 동시에 수행한다. 즉 다수의 내적 연산으로 구성되는 컨볼루션 연산에서 2번의 내적 연산을 2개의 디지털 뉴런(DN1, DN2)를 이용하여 병렬로 수행할 수 있도록 함으로써, 연산 속도를 2배로 향상시킬 수 있다.

[0217] 도13 에서는 일예로 인공 뉴런이 2개의 디지털 뉴런(DN1, DN2)을 포함하는 것으로 가정하였으므로, 연산 속도가 2배로 향상되지만, 인공 뉴런에 포함되는 디지털 개수가 증가하면 연산 속도 또한 비례하여 향상될 수 있다.

[0218] 한편, 확장 모드에서 제1 및 제2 디지털 뉴런(DN1, DN2) 각각은 제1 확장 입력값 세트와 제2 확장 입력값 세트와 지정된 가중치 필터의 대응하는 영역을 이용하여 내적 연산을 수행하여 출력값(Out_1)을 출력한다.

[0219] 확장 모드에서는 가중치 필터의 크기가 디지털 뉴런(DN1, DN2)이 연산할 수 있는 크기보다 크기 때문에 하나의 디지털 뉴런만으로 요구되는 내적 연산을 수행할 수 없다. 이에 본 실시에에서는 다수의 디지털 뉴런(DN1, DN2)이 크기가 큰 가중치 필터의 대응하는 영역의 입력값과 가중치를 인가받아 각각 내적 연산을 수행할 수 있도록 함으로써, 디지털 뉴런(DN1, DN2)이 연산할 수 있는 크기보다 큰 크기의 가중치 필터에 대해서도 연산을 수행할 수 있도록 한다.

[0220] 상기에서 각 디지털 뉴런(DN1, DN2)이 연산할 수 있는 가중치 필터의 크기가 5 X 5 X M인 것으로 가정하였므로, 각각의 디지털 뉴런(DN1, DN2)은 25 X M개의 입력값과 가중치에 대한 연산을 수행할 수 있다. 따라서 2개의 디지털 뉴런(DN1, DN2)은 50 X M개의 입력값과 가중치에 대한 연산을 수행할 수 있다.

[0221] 확장 모드에서 인가된 가중치 필터가 7 X 7 X M (= 49 X M)인 것으로 가정하였으므로, 5 X 5 X M 크기의 2개의 디지털 뉴런(DN1, DN2)을 이용하는 인공 뉴런은 용이하게 7 X 7 X M 크기의 가중치 필터에 대한 내적 연산을 수행할 수 있게 된다.

[0222] 다만, 기본 모드에서의 병렬 연산은 개별적 출력값(Out_1, Out_2)으로 출력되어야 하는 반면, 확장 모드에서는 다수의 디지털 뉴런(DN1, DN2)의 출력값이 통합되어야 단일 출력값(Out_1)으로 출력되어야 한다.

[0223] 이에 도13 의 인공 뉴런은 출력 가산기(ADD)와 출력 선택 먹스(Mux2)를 더 포함한다. 출력 가산기(ADD)는 확장 모드에서 다수의 디지털 뉴런(DN1, DN2)의 출력값을 합하여 출력 선택 먹스(Mux2)로 출력한다.

[0224] 출력 선택 먹스(Mux2)는 모드 신호(MD)에 응답하여 기본 모드에서는 제1 디지털 뉴런(DN1)에서 출력되는 출력값을 선택하는 반면, 확장 모드에서는 출력 가산기(ADD)에서 출력되는 합계 값을 출력한다.

[0225] 한편, 도13 에 도시된 바와 같이, 제1 기본 입력값 세트와 제2 기본 입력값 세트는 제2 영역(IN_{T2})의 입력값이 중복된다. 그리고 제2 기본 입력값 세트와 제2 확장 입력값 세트는 제4 영역(IN_{T4})의 입력값이 중복되는 반면, 제3 영역(IN_{T3})의 입력값은 제2 확장 입력값 세트에만 포함된다.

[0226] 이렇게 중복되는 입력값과 차별되는 입력값을 다수의 디저털 뉴런(DN1, DN2)로 각각 개별적으로 인가하는 경우, 인공 뉴런에 다수의 데이터 라인이 배치되어야 한다. 그리고 M개 채널 각각에서 입력값 세트가 출력되므로, 중복되는 입력값을 전송하기 위해 구비되어야 하는 데이터 라인의 수는 매우 많으며, 이로 인해 인공 뉴런의 크기와 전력 소비가 증가하는 문제가 있다.

[0227] 이에 본 실시예에서는 다수개의 입력 선택 먹스(Mux1)을 구비하여, 제2 기본 입력값 세트와 제2 확장 입력값 세트에서 중복되는 입력값과 차별화 된 입력값이 선택적으로 제2 디지털 뉴런(DN2)으로 인가되도록 한다.

[0228] 여기서 적어도 하나의 입력 선택 먹스(Mux1)는 모드 신호(MD)에 응답하여, 제2 기본 입력값 세트와 제2 확장 입력값 세트의 중복 입력값과 차별 입력값 중 하나를 선택할 수 있다.

[0229] 도14 는 본 발명의 다른 실시예에 따른 인공 신경망의 추론 엔진의 개략적 구성을 나타낸다.

[0230] 도14 는 컨볼루션 신경망에 대한 추론 엔진의 일예를 도시하였다. 특히 도14 의 추론 엔진은 하나의 컨볼루션 엔진부(CEN)와 하나의 완전 연결 엔진부(FEN) 및 메모리(MEM)를 포함한다. 도1 에 도시된 바와 같이, 컨볼루션 신경망은 다수의 컨볼루션 레이어(C1, C2, C3)와 다수의 완전 연결 레이어(FC1, FC2)를 포함할 수 있다. 그러나 다수의 컨볼루션 레이어(C1, C2, C3)와 다수의 완전 연결 레이어(FC1, FC2)를 각각 구분하여 개별적인 회로

로 구현하는 것은 크기, 전력 소모, 제조 비용 등에 있어서 매우 비효율적이다.

[0231] 이에 도14 에서는 유사한 동작을 수행하는 다수의 컨볼루션 레이어의 동작을 수행하기 위한 하나의 컨볼루션 엔진부(CEN)와 다수의 완전 연결 레이어의 동작을 수행하기 위한 하나의 완전 연결 엔진부(FEN)를 포함하여 컨볼루션 신경망이 디지털 하드웨어로 구현될 수 있도록 한다.

[0232] 도14 에서는 일예로 왼쪽에 도시한 바와 같이, 인공 신경망이 2개의 컨볼루션 레이어(C1, C2)와 3개의 완전 연결 레이어(FC1, FC2, INF)를 포함하는 것으로 가정한다.

[0233] 우선 메모리(MEM)는 컨볼루션 신경망으로 인가되는 입력 데이터(또는 입력 영상)와 다수의 컨볼루션 레이어(C1, C2)와 다수의 완전 연결 레이어(FC1, FC2, INF)를 위한 가중치 필터가 저장된다.

[0234] 컨볼루션 엔진부(CEN)는 입력값 획득부(INOB), 컨볼루션 필터뱅크 버퍼(FB1)와 입력 먹스(MuxIN), 입력 선택 먹스(MuxDI), 디먹스(Demux), 출력 특징 맵 버퍼(OFMB) 및 적어도 하나의 디지털 뉴런(DN1)을 포함한다.

[0235] 입력 먹스(MuxIN)는 인공 신경망의 입력부로서 메모리(MEN)에 저장된 입력 데이터 또는 컨볼루션 엔진부(CEN)의 출력값을 인가받고, 레이어 선택 신호(SelINL)에 응답하여, 입력 영상 또는 출력값 중 하나를 선택하여 입력 특징 맵의 원소인 입력값을 입력값 획득부(INOB)로 출력한다.

[0236] 여기서 레이어 선택 신호($SelIN_L$)는 다수의 컨볼루션 레이어(C1, C2)를 포함하는 인공 신경망에서 컨볼루션 엔진부(CEN)의 초기 입력값을 선택하기 위한 신호로서, 인공 신경망의 초기 레이어(여기서는 제1 컨볼루션 레이어(C1))인지 아니면, 컨볼루션 레이어(C1, C2) 이후의 레이어(여기서는 완전 연결 레이어(FC1, FC2, INF))인지를 구분하는 신호이다.

[0237] 상기한 바와 같이, 인공 신경망이 2개의 컨볼루션 레이어(C1, C2)를 포함하는 경우, 제1 컨볼루션 레이어(C1)는 입력 데이터를 인가받는데 반해, 제2 컨볼루션 레이어(C2)는 제1 컨볼루션 레이어(C1)의 출력값을 인가받도록 구성될 수 있다.

[0238] 상기한 구조에 기반하여 입력 먹스(MuxIN)는 레이어 선택 신호($SelIN_L$)에 응답하여, 초기 레이어(C1)이면 메모리(MEM)으로부터 입력 데이터를 선택하여 입력값 획득부(INOB)로 출력한다. 반면, 컨볼루션 레이어(C1, C2)가 아니면, 컨볼루션 엔진부(CEN)의 출력값을 선택하여 입력값 획득부(INOB)로 출력한다.

[0239] 여기서 컨볼루션 레이어(C1, C2)가 아닌 경우, 컨볼루션 엔진부(CEN)의 출력값은 기지정된 설정값 또는 초기화된 값으로 출력될 수 있다. 즉 입력 먹스(MuxIN)는 컨볼루션 레이어(C1, C2)가 아닌 경우에, 초기화된 값을 입력값 획득부(INOB)로 출력하여, 입력값 획득부(INOB)를 기지정된 초기화 값으로 초기화 시킬 수 있다.

[0240] 입력값 획득부(INOB)는 입력 먹스(MuxIN)로부터 입력 데이터를 인가받아 입력 특징 맵을 구성하여 저장하고, 저장된 입력 특징 맵에서 지정된 영역에 포함된 입력값을 추출하여 입력 선택 먹스(MuxDI)로 출력한다.

[0241] 상기한 바와 같이 컨볼루션 레이어는 컨볼루션 연산을 수행하기 위해, 입력 특징 맵 상을 지정된 가중치 필터가 이동하면서 내적 연산을 수행해야 한다.

[0242] 이는 입력값 획득부(INOB)에 저장된 입력 특징 맵의 다수의 입력값 중 가중치 필터에 대응하는 위치의 입력값을 위치 이동하면서 선택하여 디지털 뉴런(DN1)으로 전달해야 하며, 이러한 특징을 하드웨어적으로 구현하기 위해서는 매우 어렵다.

[0243] 그러나 본 실시예에서는 입력값 획득부(INOB)를 도14 에 도시된 바와 같이 다수의 시프트 레지스터(shift register)로 구현하고, 입력 특징 맵의 다수의 입력값이 순차적으로 지정된 위치로 이동하도록 구성함에 따라, 항상 동일한 위치의 입력값을 추출하여 입력 선택 먹스(MuxDI)로 출력할 수 있도록 한다.

[0244] 즉 시프트 레지스터를 이용하여 매우 용이하게 입력값 획득부(INOB)를 하드웨어로 구현할 수 있도록 하였다.

[0245] 그리고 입력값 획득부(INOB)는 현재 수행되어야 하는 레이어가 컨볼루션 레이어(C1, C2)가 아닌 경우에, 입력 먹스(MuxIN)로부터 인가되는 초기화값을 저장함으로써, 초기화된다.

[0246] 입력 선택 먹스(MuxDI)는 입력 데이터 선택 신호($SelDI_L$)에 응답하여, 입력값 획득부(INOB)에서 인가되는 입력값 또는 출력 특징 맵 버퍼(OFMB)에서 인가되는 출력값 중 하나를 선택하여 디지털 뉴런(DN1)으로 출력한다.

[0247] 여기서 입력 데이터 선택 신호($SelDI_L$)는 인공 신경망에서 컨볼루션 엔진부(CEN)가 현재 컨볼루션 연산을 수행

해야 하는 컨볼루션 레이어가 초기 컨볼루션 레이어(C1)인지 아닌지를 판별하기 위한 신호이다.

[0248] 입력 선택 먹스(MuxDI)는 인공 신경망의 다수의 컨볼루션 레이어(C1, C2) 중 초기 컨볼루션 레이어(C1)이면, 입력값 획득부(INOB)에서 인가되는 입력값을 디지털 뉴런(DN1)으로 출력한다.

[0249] 그러나 초기 컨볼루션 레이어(C1)이 아니면, 출력 특징 맵 버퍼(OFMB)에서 인가되는 출력 특징 맵의 값을 입력값으로 디지털 뉴런(DN1)으로 출력한다. 즉 이전 컨볼루션 레이어에서 획득된 출력 특징 맵을 입력 특징 맵으로서, 디지털 뉴런(DN1)으로 전달한다.

[0250] 컨볼루션 필터뱅크 버퍼(FB1)는 메모리(MEN)에 저장된 가중치 필터를 인가받아 임시 저장한다. 이때 컨볼루션 필터뱅크 버퍼(FB1)는 다수의 컨볼루션 레이(C1, C2)어 각각에 대응하는 다수의 가중치 필터를 한번에 인가받아 저장할 수도 있으나, 현재 컨볼루션 엔진(CEN)이 수행하는 동작에 대응하는 컨볼루션 레이어를 위한 가중치 필터를 인가받아 저장할 수도 있다.

[0251] 디지털 뉴런(DN1)은 도9 에 도시된 인공 뉴런으로서, 입력 선택 먹스(MuxDI)에서 인가되는 다수의 입력값과 컨볼루션 필터뱅크 버퍼(FB1)에서 인가되는 가중치 필터를 내적하여 출력한다.

[0252] 디먹스(Demux)는 출력 데이터 선택 신호($SelDO_L$)에 응답하여, 디지털 뉴런(DN1)에서 출력되는 출력값을 출력 특징 맵 버퍼(OFMB) 또는 완전 연결 엔진(FEN)으로 출력한다.

[0253] 여기서 출력 데이터 선택 신호($SelDO_L$)는 입력 데이터 선택 신호($SelDI_L$)과 상이하게 인공 신경망에서 컨볼루션 엔진부(CEN)가 현재 컨볼루션 연산을 수행해야 하는 컨볼루션 레이어가 최종 컨볼루션 레이어인지 아닌지를 판별하기 위한 신호이다.

[0254] 디먹스(Demux)는 다수의 컨볼루션 레이어(C1, C2) 중 최종 컨볼루션 레이어(C2)이면, 디지털 뉴런(DN1)의 출력값을 완전 연결 엔진(FEN)으로 출력한다. 그러나 최종 컨볼루션 레이어(C2)가 아니면, 디지털 뉴런(DN1)의 출력값을 출력 특징 맵 버퍼(OFMB)으로 출력한다.

[0255] 또한 디먹스(Demux)의 출력은 현재 연산이 수행되는 레이어가 컨볼루션 레이어(C1, C2)가 아닌 경우에, 입력 먹스(MuxIN)으로 인가된다.

[0256] 그리고 출력 특징 맵 버퍼(OFMB)는 디먹스(Demux)에서 출력되는 출력값을 순차적으로 누적하여 출력 특징 맵으로서 저장하고, 저장된 출력값을 입력 선택 먹스(MuxDI)로 출력한다.

[0257] 상기와 같이 구성된 컨볼루션 엔진(CEN)은 특징 추출부의 다수의 컨볼루션 레이어 중 초기 컨볼루션 레이어에 대해서는 메모리(MEM)에 저장된 입력 영상을 입력 특징 맵으로서 인가받아 입력값을 추출하여 디지털 뉴런(DN1)으로 전달하고, 디지털 뉴런(DN1)에서 출력되는 출력값을 출력 특징 맵 버퍼(OFMB)에 저장한다.

[0258] 그리고 중간 단계의 컨볼루션 레이어에 대해서는 출력 특징 맵 버퍼(OFMB)로부터 출력 특징 맵을 입력 특징 맵으로서 인가받아 다시 출력 특징 맵 버퍼(OFMB)에 저장한다.

[0259] 그러나 최종 컨볼루션 레이어에 대해서는 디지털 뉴런(DN1)에서 출력되는 출력값을 완전 연결 엔진(FEN)으로 출력한다.

[0260] 즉 하나의 컨볼루션 엔진(CEN)으로 다수의 컨볼루션 레이어(C1, C2)를 포함하는 특징 추출부를 구현할 수 있다.

[0261] 이때 컨볼루션 엔진(CEN)은 출력 특징 맵 버퍼(OFMB)에 저장된 출력 특징 맵을 기지정된 방식으로 서브 샘플링하는 풀링부(미도시)를 더 포함할 수 있다. 풀링부는 일예로 평균값 풀링 또는 맥스 풀링 작업을 수행하도록 구성될 수 있다.

[0262] 그리고 상기에서는 출력 특징 맵 버퍼(OFMB)가 컨볼루션 엔진(CEN) 내에 포함되는 것으로 도시하였으나, 경우에 따라서 출력 특징 맵은 메모리(MEM)에 저장되도록 구성될 수도 있다.

[0263] 한편, 완전 연결 엔진(FEN)은 입력 추론 선택 먹스(MuxFI)와 입력 레지스터(FIRG), 출력 레지스터(FORG), 완전 연결 필터뱅크 버퍼(FB2) 및 디지털 뉴런(DN2)를 포함할 수 있다.

[0264] 입력 추론 선택 먹스(MuxFI)는 컨볼루션 엔진(CEN)의 입력 선택 먹스(MuxDI)와 유사하게 연결 레이어 선택 신호($SelFI_L$)에 응답하여 컨볼루션 엔진(CEN)에서 인가되는 출력값 또는 출력 레지스터(FORG)에 저장된 출력값을 선택하여 입력 레지스터(FIRG)로 전달한다.

[0265] 여기서 연결 레이어 선택 신호($SelFI_L$)는 분류부의 다수의 완전 연결 레이어(FC1, FC2, INF) 중 초기 완전 연결 레이어(FC1)인지 나머지 완전 연결 레이어(FC2, INF)인지를 구분하기 위해 인가되는 신호이다.

[0266] 입력 추론 선택 먹스(MuxFI)는 연결 레이어 선택 신호($SelFI_L$)에 응답하여 초기 완전 연결 레이어(FC1)이면, 컨볼루션 엔진(CEN)에서 인가되는 출력값을 선택하여, 입력 레지스터(FIRG)로 전달한다. 그러나 나머지 완전 연결 레이어(FC2, INF)이면, 출력 레지스터(FORG)에 저장된 출력값을 선택하여 입력 레지스터(FIRG)로 전달한다.

[0267] 입력 레지스터(FIRG)는 입력 추론 선택 먹스(MuxFI)에서 전달되는 출력값을 인가받아 임시 저장하여 디지털 뉴런(DN2)로 출력하고, 출력 레지스터(FORG)는 디지털 뉴런(DN2)에서 출력되는 출력값을 임시 저장하여 입력 추론 선택 먹스(MuxFI)로 출력한다. 그리고 완전 연결 필터뱅크 버퍼(FB2)는 메모리(MEN)에 저장된 가중치 필터를 인가받아 임시 저장한다.

[0268] 디지털 뉴런(DN2)는 입력 레지스터(FIRG)에서 인가되는 출력값과 완전 연결 필터뱅크 버퍼(FB2)에서 저장된 가중치 필터를 인가받아 기지정된 연산을 수행하여 출력값을 출력한다. 이때 디지털 뉴런(DN2)에서 출력되는 출력값은 추론값(Inferencr output)일 수 있다.

[0269] 여기서 완전 연결 엔진(FEN)은 디지털 뉴런(DN2)에서 출력되는 출력값을 출력 레지스터(FORG)로 출력하거나 외부로 출력하도록 선택하는 디먹스를 추가로 더 구비할 수 있다.

[0270] 이 경우 컨볼루션 엔진(CEN)과 완전 연결 엔진(FEN)은 거의 동일한 구조를 가질 수 있다. 그러나 추론 엔진이 컨볼루션 엔진(CEN)과 완전 연결 엔진(FEN)을 구분하여 각각 구비하는 것은 일반적으로 컨볼루션 레이어에서 이용되는 가중치 필터의 크기와 완전 연결 레이어에서 이용되는 가중치 필터의 크기 차이가 매우 크므로, 컨볼루션 엔진(CEN)을 이용하여 완전 연결 레이어를 구현하는 것은 매우 비효율적이기 때문이다.

[0271] 즉 도14 에서 컨볼루션 엔진(CEN)의 디지털 뉴런(DN1)과 완전 연결 엔진(FEN)의 디지털 뉴런(DN2)은 모두 도9과 같은 인공 뉴런으로 구현될 수 있으나, 가중치 필터와 입력 특징 맵의 크기 차이가 크기 때문에 별도로 구현되는 것이 바람직하다. 특히 컨볼루션 엔진(CEN)의 디지털 뉴런(DN1)은 3차원 가중치 필터를 이용하는 반면, 완전 연결 엔진(FEN)의 디지털 뉴런(DN2)은 대부분 1차원 가중치 필터를 이용한다.

[0272] 이에 본 실시예에서는 컨볼루션 엔진(CEN)의 디지털 뉴런(DN1)은 도9 의 인공 뉴런으로 구성하고, 완전 연결 엔진(FEN)의 디지털 뉴런(DN2)은 별도로 구현되는 완전 연결 인공 뉴런이라 할 수 있다.

[0273] 상기한 도14 에서는 컨볼루션 엔진(CEN)이 하나의 디지털 뉴런(DN1)을 포함하는 것으로 가정하여 도시하였으나, 도14 의 인공 신경망의 다수의 컨볼루션 레이어 각각은 서로 다른 크기의 가중치 필터를 이용할 수도 있다. 이때 가중치 필터의 크기는 디지털 뉴런(DN1)에서 지정된 가중치 필터의 크기보다 클 수 있다.

[0274] 이에 도14 의 인공 신경망을 위한 추론 엔진은 디지털 뉴런(DN1) 대신 도13 의 인공 뉴런이 추가될 수 있다. 일예로 도14 의 추론 엔진에서 컨볼루션 엔진(CEN)은 도13 에서와 같이 다수개의 디지털 뉴런을 포함할 수 있다. 그리고 도14 의 입력값 획득부(INOB)는 도13 의 입력값 추출부(INEX)와 같이 모드 신호(MD)에 응답하여 서로 다른 위치의 입력값을 추출하도록 구성될 수 있다. 또한 입력값 획득부(INOB)와 다수의 디지털 뉴런 사이에 다수의 입력 선택 먹스(MuxDI)를 포함하고, 다수의 디지털 뉴런과 디먹스(Demux) 사이에 가산기(AD)와 적어도 하나의 출력 선택 먹스(Mux2)를 포함하도록 구성될 수 있다.

[0275] 즉 컨볼루션 엔진(CEN)이 수행해야하는 동작에 대응하는 컨볼루션 레이어의 가중치 필터의 크기에 따라 지정된 크기 이내의 가중치 필터에 대해서는 둘 이상의 내적 연산을 병렬로 수행하고, 지정된 크기를 초과하는 가중치 필터에 대해서는 분할하여 내적 연산을 수행할 수 있도록 한다.

[0276] 경우에 따라서는 완전 연결 엔진(FEN) 또한 컨볼루션 엔진(CEN)과 마찬가지로 다수의 인공 뉴런을 포함하고, 가중치의 크기에 따라 병렬 또는 분할 방식으로 내적 연산을 수행하도록 구성될 수도 있다.

[0277] 본 발명에 따른 방법은 컴퓨터에서 실행 시키기 위한 매체에 저장된 컴퓨터 프로그램으로 구현될 수 있다. 여기서 컴퓨터 판독가능 매체는 컴퓨터에 의해 액세스 될 수 있는 임의의 가용 매체일 수 있고, 또한 컴퓨터 저장 매체를 모두 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 포함하며, ROM(판독 전용 메모리), RAM(랜덤 액세스 메모리), CD(컴팩트 디스크)-ROM, DVD(디지털 비디오 디스크)-ROM, 자기 테이프, 플로피 디스크, 광데이터 저장장치 등을 포함할 수 있다.

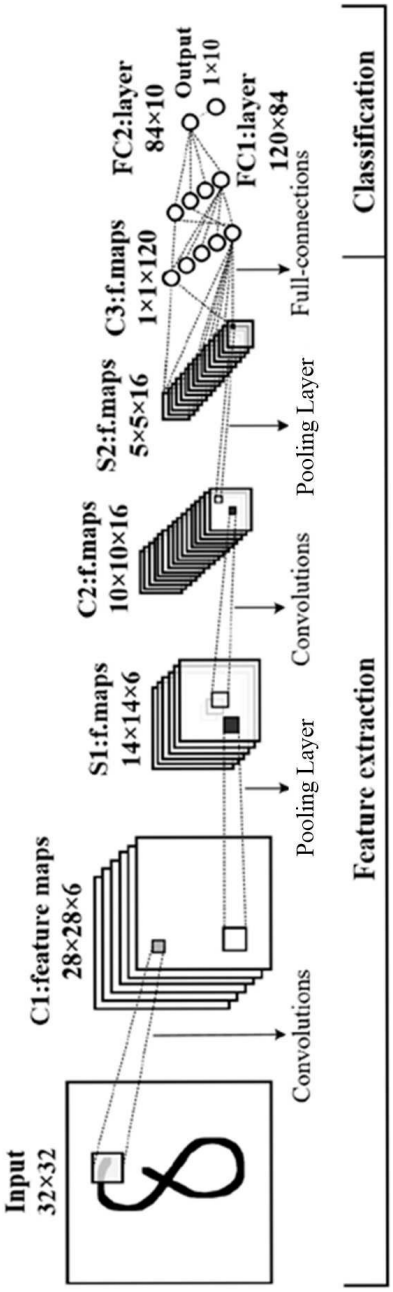
[0278] 본 발명은 도면에 도시된 실시예를 참고로 설명되었으나 이는 예시적인 것에 불과하며, 본 기술 분야의 통상의

지식을 가진 자라면 이로부터 다양한 변형 및 균등한 타 실시예가 가능하다는 점을 이해할 것이다.

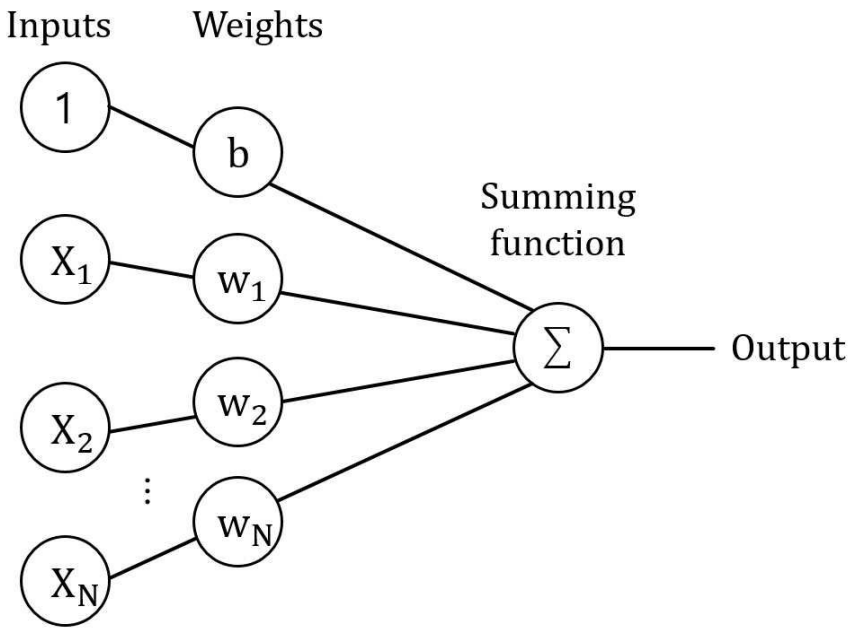
[0279] 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

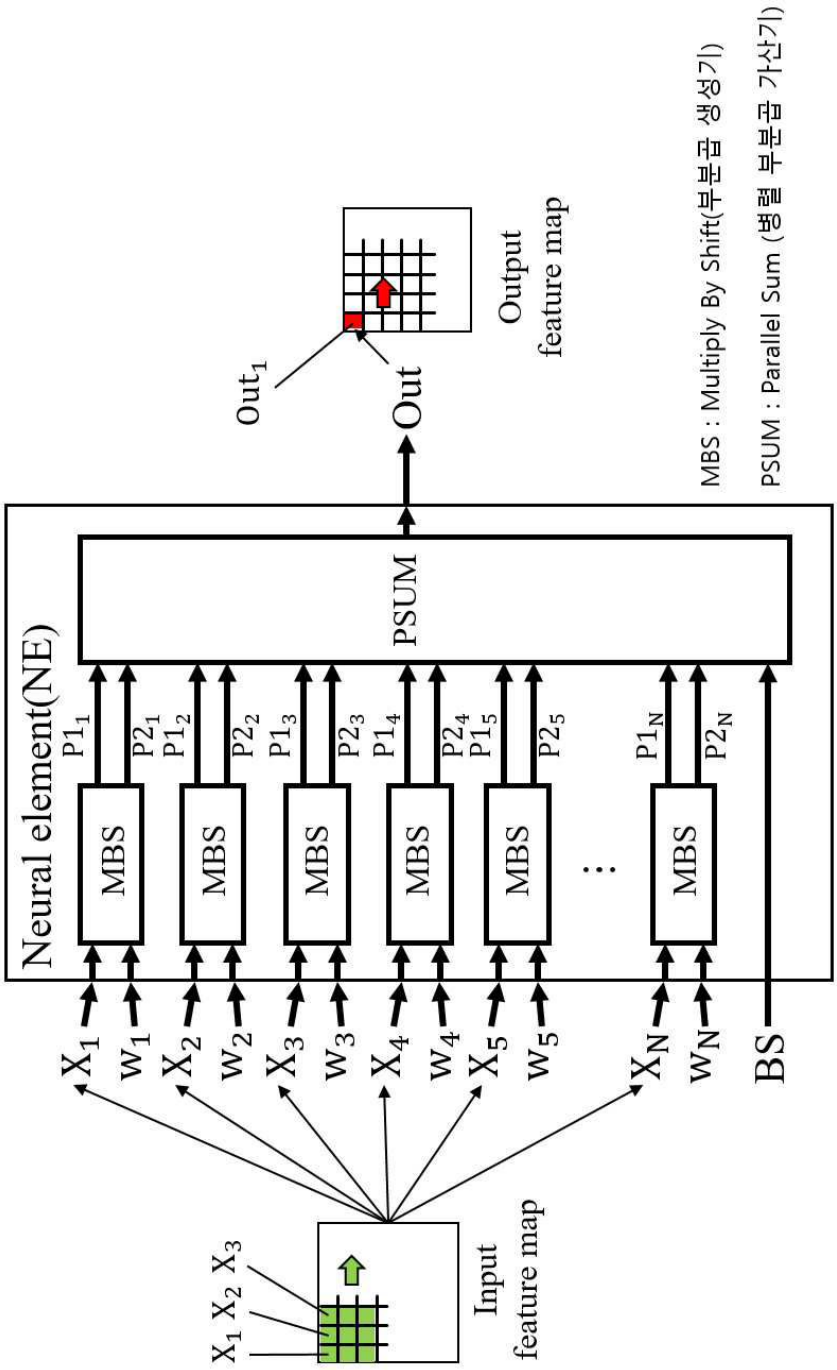
도면1



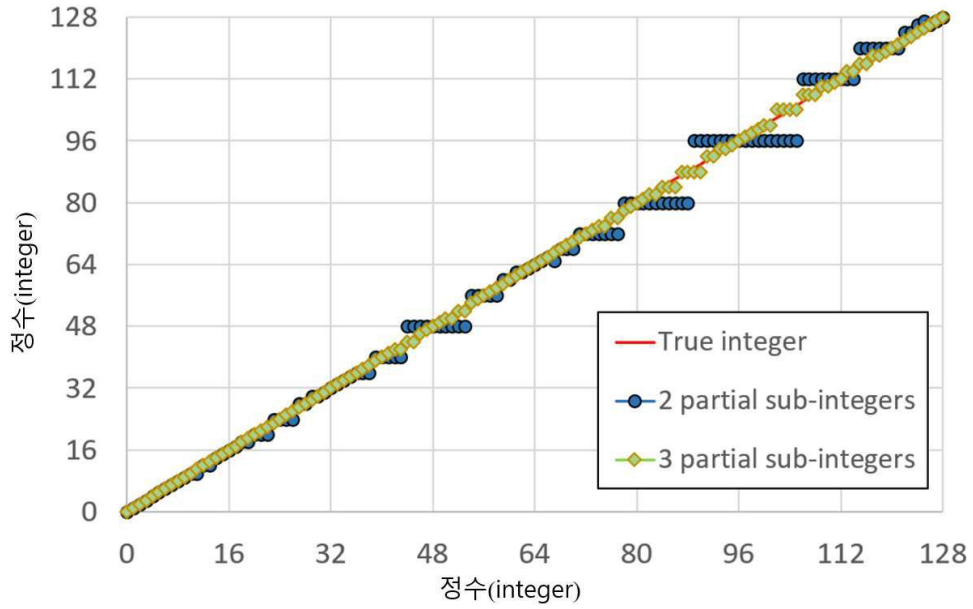
도면2



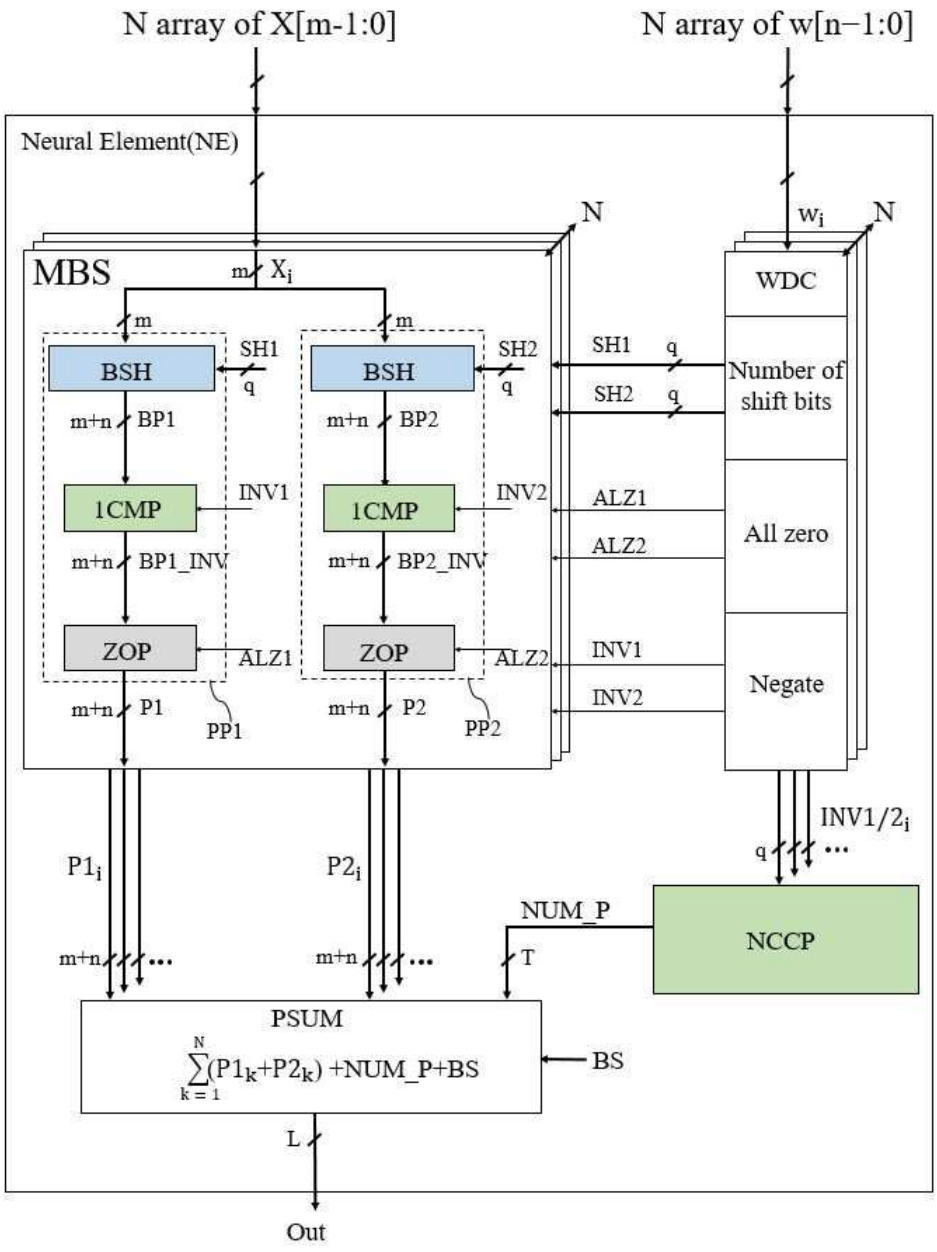
도면3


MBS : Multiply By Shift(부분곱 생성기)
PSUM : Parallel Sum (병렬 부분곱 가산기)
Output feature map
Out1
Out
PSUM
Neural element(NE)
MBS
P1N
P2N
BS
Input feature map
X1 X2 X3

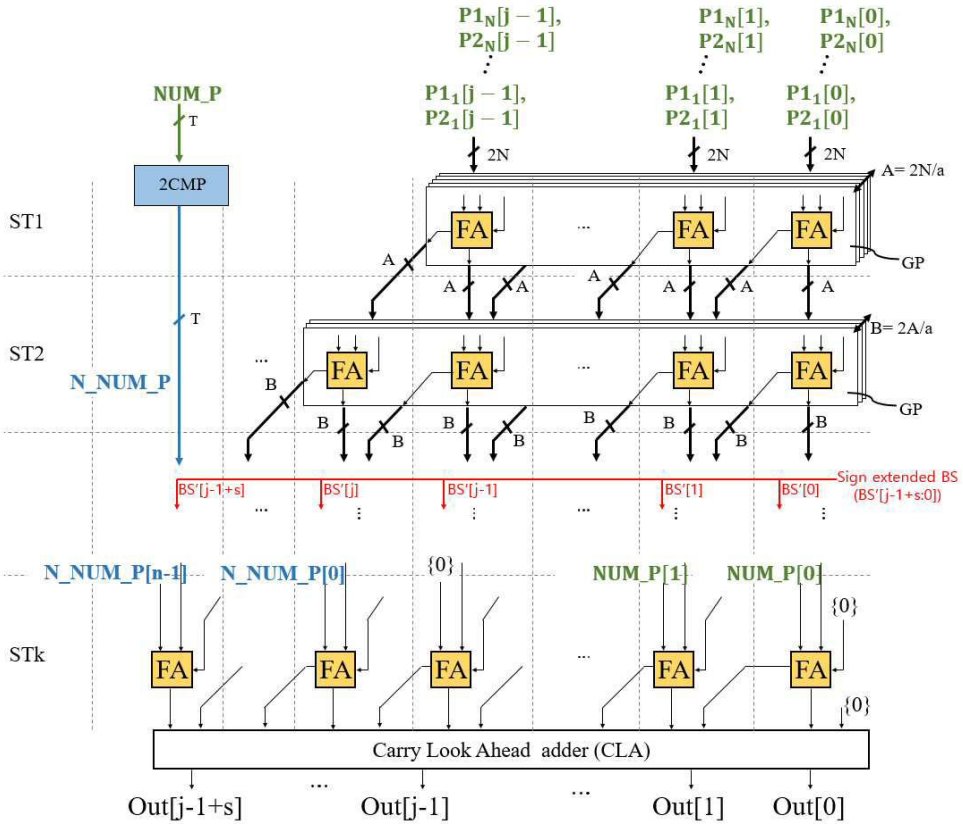
도면4


128
112
96
80
64
48
32
16
0
정수(integer)
0
16
32
48
64
80
96
112
128
정수(integer)
True integer
2 partial sub-integers
3 partial sub-integers

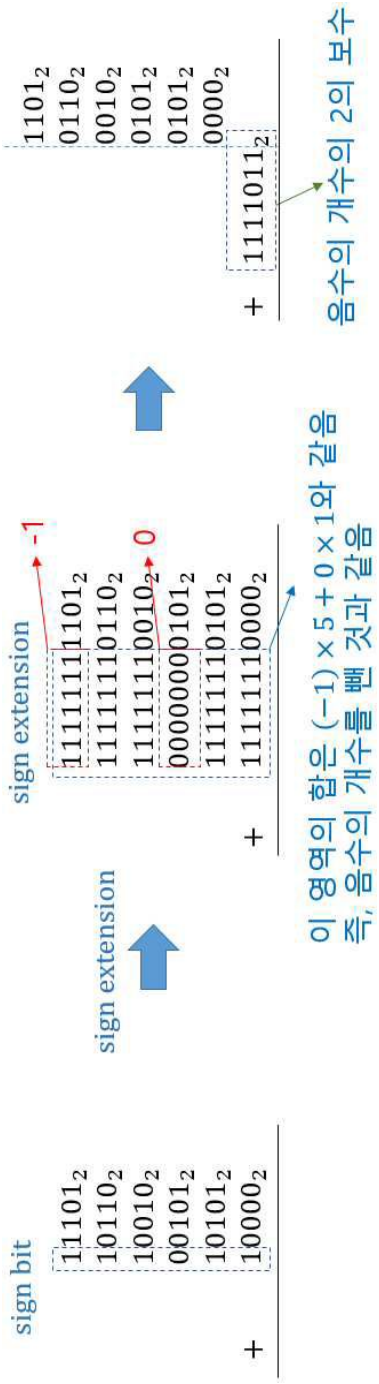
도면5


N array of X[m-1:0]
N array of w[n−1:0]
Neural Element(NE)
MBS
m
Xi
N
wi
WDC
BSH
SH1
q
SH2
Number of shift bits
m+n
BP1
BP2
1CMP
INV1
INV2
ALZ1
ALZ2
All zero
BP1_INV
BP2_INV
ZOP
Negate
P1
P2
PP1
PP2
P1i
P2i
INV1/2i
NUM_P
T
NCCP
PSUM
BS
L
Out

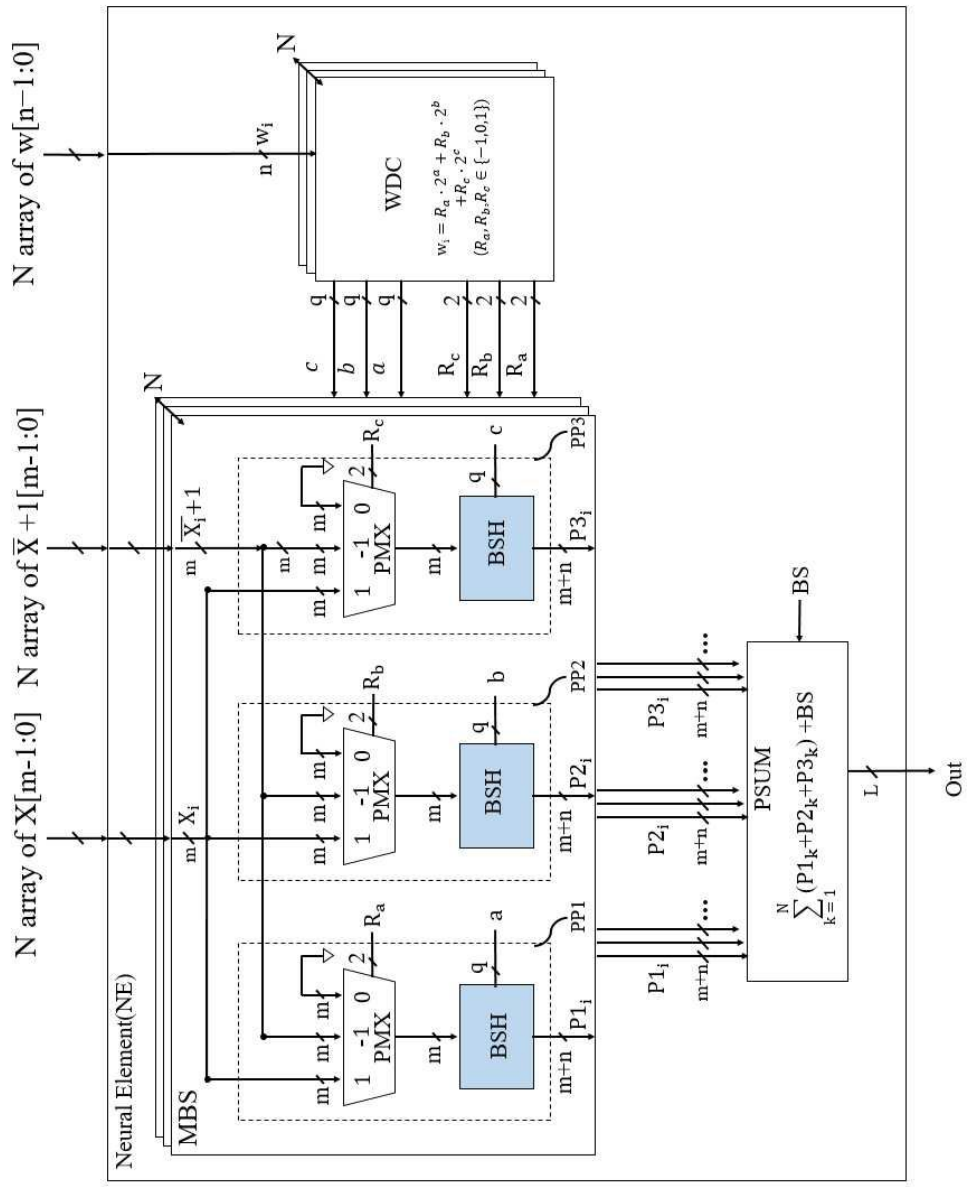
도면6


NUM_P
T
2CMP
ST1
ST2
N_NUM_P
STk
FA
A
B
2N
A= 2N/a
B= 2A/a
GP
Sign extended BS
(BS'[j-1+s:0])
BS'[j-1+s]
BS'[j]
BS'[j-1]
BS'[1]
BS'[0]
N_NUM_P[n-1]
N_NUM_P[0]
NUM_P[1]
NUM_P[0]
{0}
Carry Look Ahead adder (CLA)
Out[j-1+s]
Out[j-1]
Out[1]
Out[0]

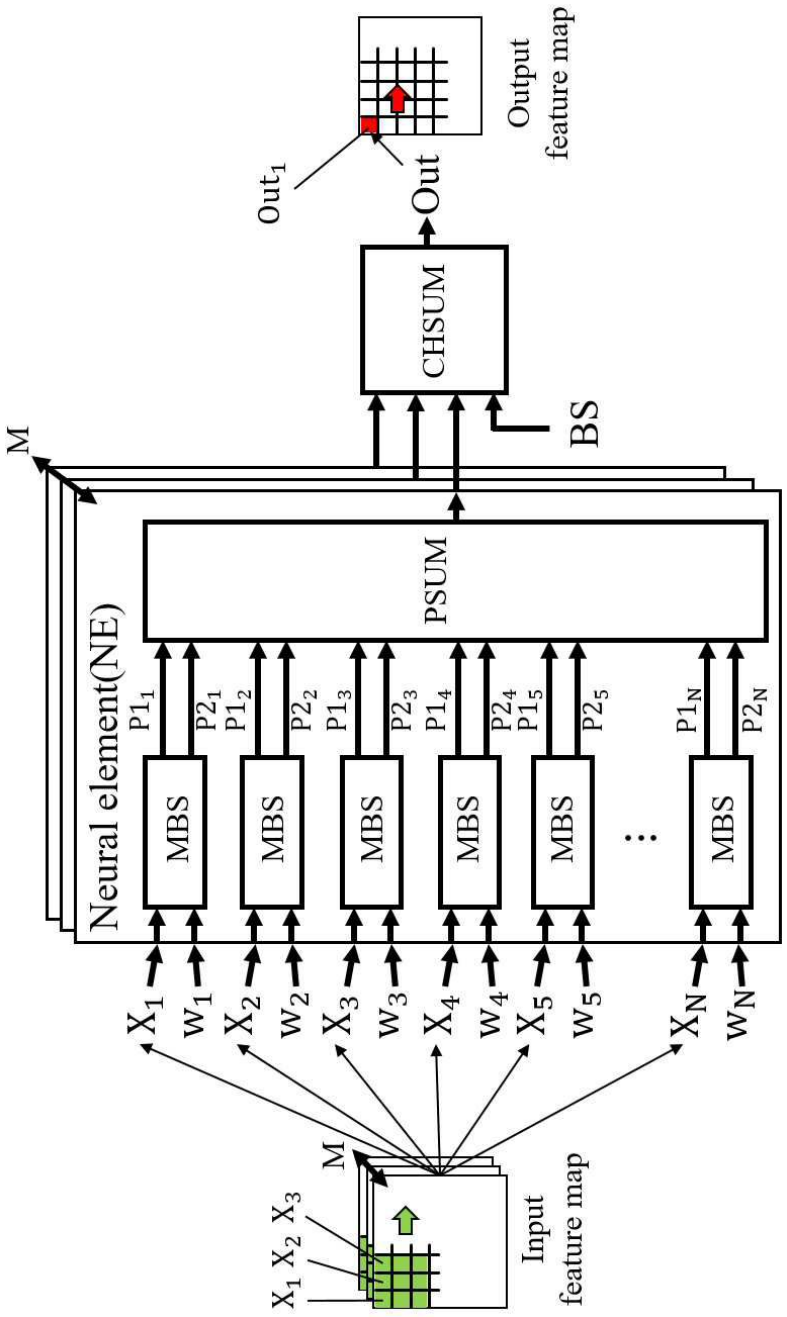
도면7


sign bit
sign extension
sign extension
-1
0
이 영역의 합은 (−1) × 5 + 0 × 1와 같음
즉, 음수의 개수를 뺀 것과 같음
음수의 개수의 2의 보수

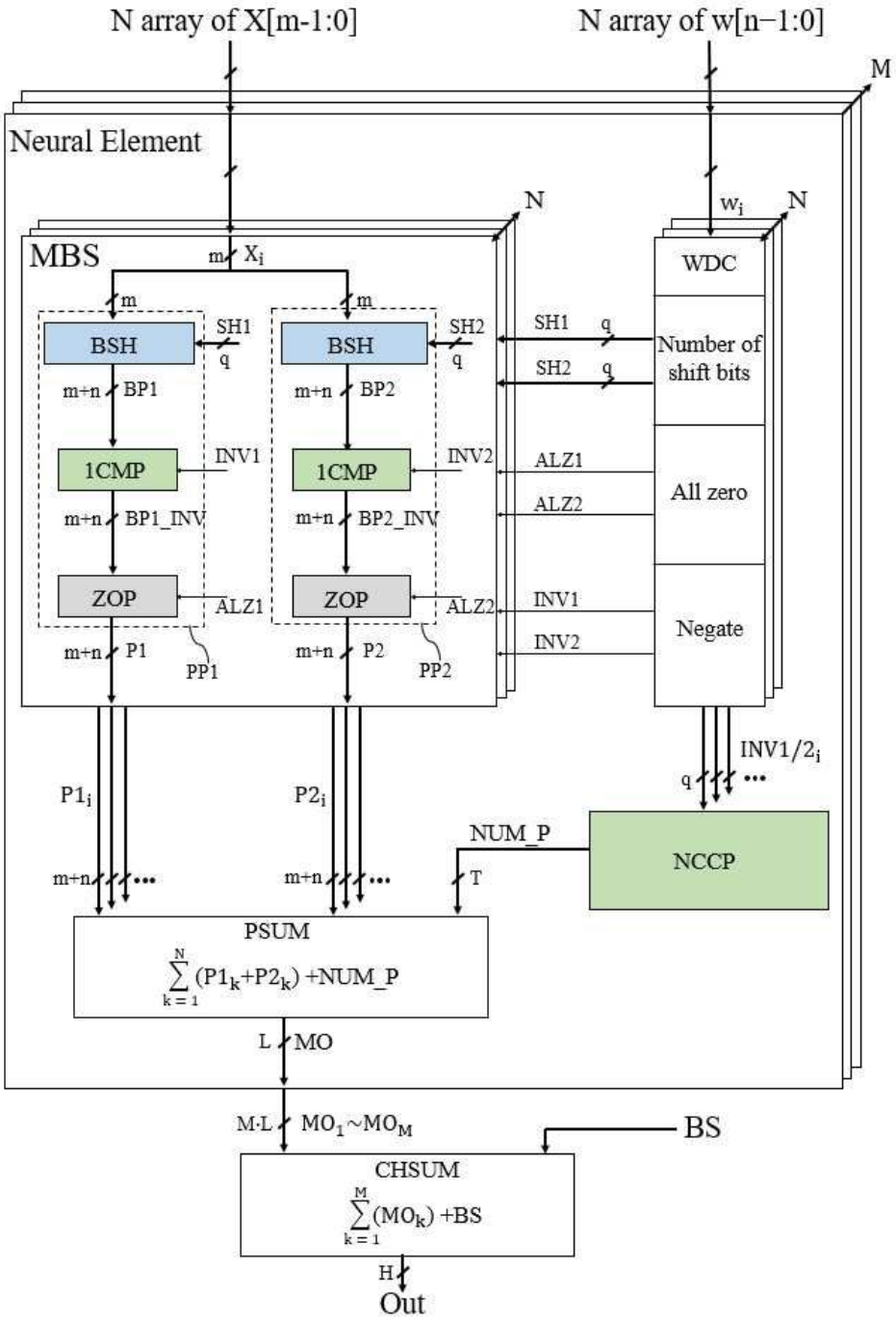
도면8


N array of w[n-1:0]
N array of X[m-1:0]
N array of X̄+1[m-1:0]
WDC
$w_i = R_a \cdot 2^a + R_b \cdot 2^b + R_c \cdot 2^c$
$(R_a, R_b, R_c \in \{-1,0,1\})$
Neural Element(NE)
MBS
PMX
BSH
PSUM
$\sum_{k=1}^{N}(P1_k+P2_k+P3_k)+BS$
BS
Out

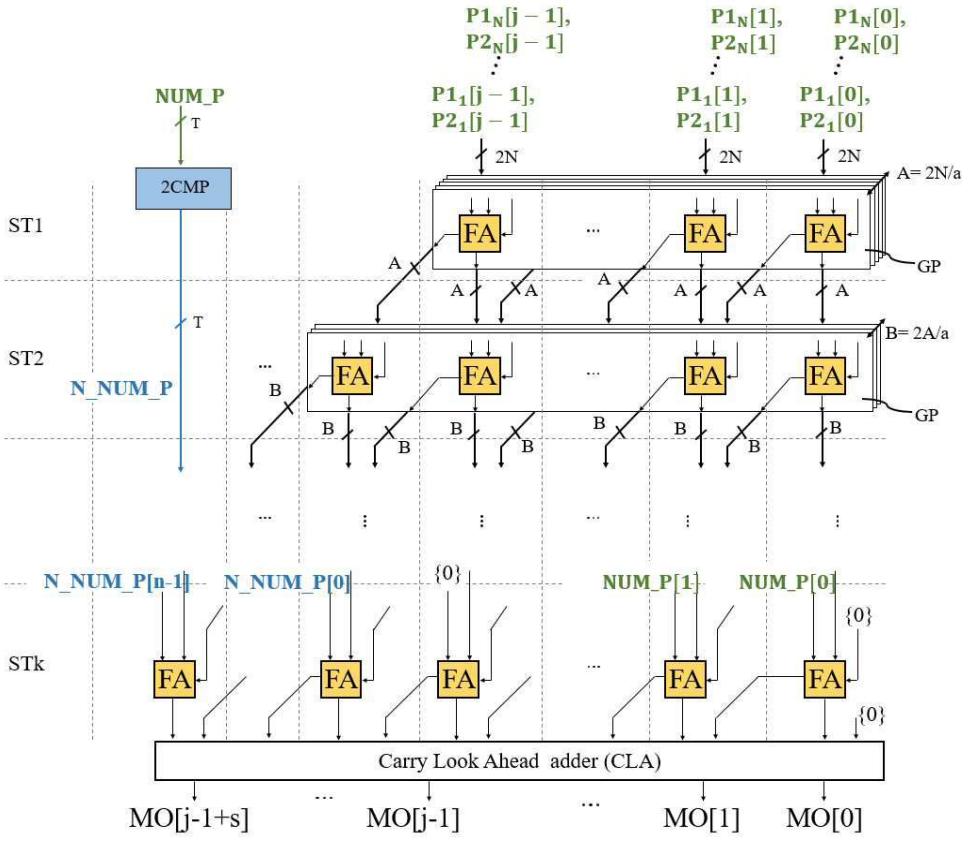
도면9


Output feature map
Out1
Out
CHSUM
BS
M
PSUM
Neural element(NE)
MBS
P11
P21
P12
P22
P13
P23
P14
P24
P15
P25
P1N
P2N
X1
W1
X2
W2
X3
W3
X4
W4
X5
W5
XN
WN
Input feature map
X1 X2 X3

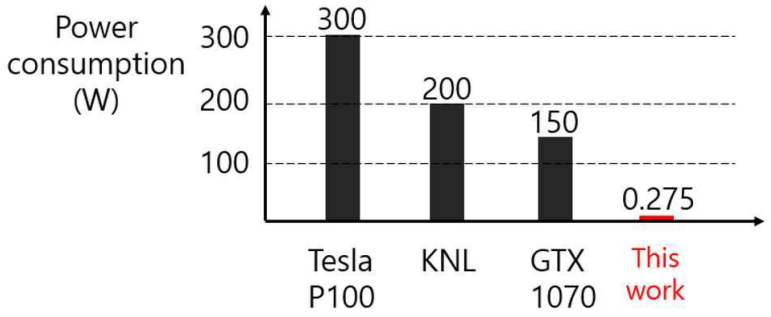
도면10


N array of X[m-1:0]
N array of w[n−1:0]
M
Neural Element
N
w_i
N
MBS
m
X_i
WDC
m
m
BSH
SH1
q
BSH
SH2
q
SH1
q
SH2
q
Number of shift bits
m+n
BP1
m+n
BP2
1CMP
INV1
1CMP
INV2
ALZ1
ALZ2
All zero
m+n
BP1_INV
m+n
BP2_INV
ZOP
ALZ1
ZOP
ALZ2
INV1
INV2
Negate
m+n
P1
PP1
m+n
P2
PP2
$INV1/2_i$
q
$P1_i$
$P2_i$
NUM_P
NCCP
m+n
m+n
T
PSUM
$\sum_{k=1}^{N}(P1_k+P2_k)+NUM\_P$
L
MO
M·L
$MO_1 \sim MO_M$
BS
CHSUM
$\sum_{k=1}^{M}(MO_k)+BS$
H
Out

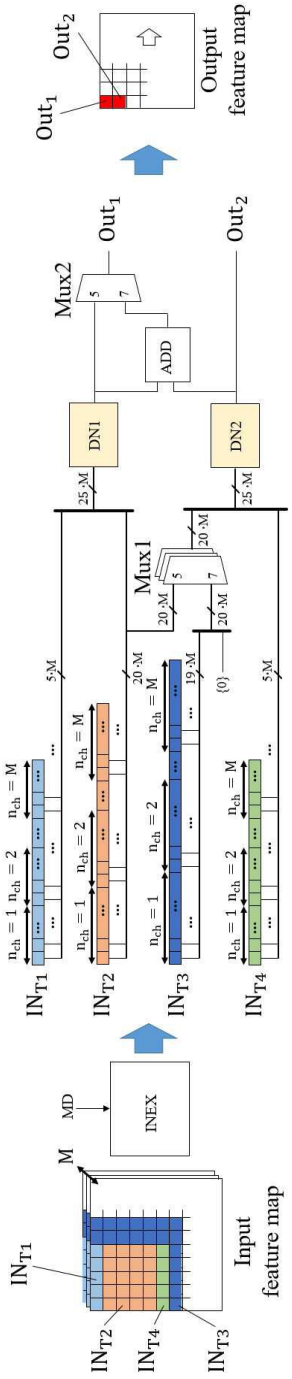
도면11


NUM_P
T
2CMP
ST1
ST2
N_NUM_P
STk
2N
A= 2N/a
B= 2A/a
GP
FA
{0}
N_NUM_P[n-1]
N_NUM_P[0]
NUM_P[1]
NUM_P[0]
Carry Look Ahead adder (CLA)
MO[j-1+s]
MO[j-1]
MO[1]
MO[0]

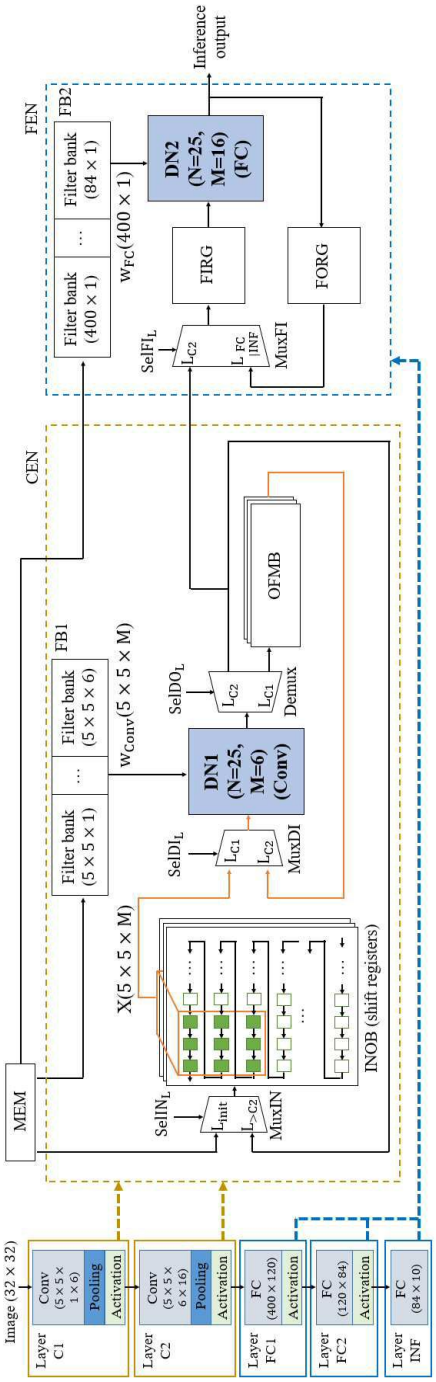
도면12


Power consumption (W)
300
200
100
300
200
150
0.275
Tesla P100
KNL
GTX 1070
This work

도면13


Out1
Out2
Output feature map
Mux2
ADD
DN1
DN2
25·M
Mux1
20·M
5·M
19·M
{0}
INEX
MD
M
Input feature map
IN_T1
IN_T2
IN_T3
IN_T4
n_ch = 1
n_ch = 2
n_ch = M

도면14


Inference output
FEN
FB2
Filter bank (400 × 1)
...
Filter bank (84 × 1)
$w_{FC}(400 \times 1)$
DN2 (N=25, M=16) (FC)
FIRG
FORG
$SelFI_L$
L_{C2}
$L_{FC\ INF}$
MuxFI
CEN
OFMB
FB1
Filter bank (5 × 5 × 6)
...
Filter bank (5 × 5 × 1)
$w_{Conv}(5 \times 5 \times M)$
$SelDO_L$
L_{C2}
L_{C1}
Demux
DN1 (N=25, M=6) (Conv)
$SelDI_L$
L_{C1}
L_{C2}
MuxDI
$X(5 \times 5 \times M)$
INOB (shift registers)
MEM
$SelIN_L$
L_{init}
$L_{>C2}$
MuxIN
Image (32 × 32)
Layer C1
Conv (5 × 5 × 1 × 6)
Pooling
Activation
Layer C2
Conv (5 × 5 × 6 × 16)
Pooling
Activation
Layer FC1
FC (400 × 120)
Activation
Layer FC2
FC (120 × 84)
Activation
Layer INF
FC (84 × 10)