(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2022-0033314
(43) 공개일자 2022년03월16일

(51) 국제특허분류(Int. Cl.)
G06N 3/063 (2006.01) *G06N 3/04* (2006.01)
(52) CPC특허분류
G06N 3/063 (2013.01)
G06F 9/28 (2013.01)
(21) 출원번호 10-2020-0115565
(22) 출원일자 2020년09월09일
심사청구일자 없음

(71) 출원인
삼성전자주식회사
경기도 수원시 영통구 삼성로 129 (매탄동)
(72) 발명자
신상규
경기도 수원시 권선구 효원로256번길 33, 220호 (권선동, 한라비발디파크)
이영식
서울특별시 송파구 법원로4길 6, 1146호 (문정동, 문정아이파크)
(74) 대리인
특허법인 무한

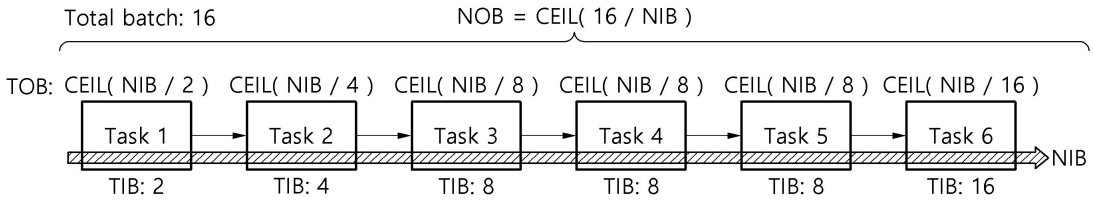
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **호스트 프로세서 및 가속기의 동작 방법 및 이들을 포함한 전자 장치**

(57) 요 약

호스트 프로세서 및 가속기의 동작 방법 및 이들을 포함한 전자 장치가 개시된다. 개시된 호스트 프로세서의 동작 방법은 가속기에서 실행하고자 하는 모델을 복수의 스테이지들로 분할하는 단계, 복수의 스테이지들 각각에 대해 가속기의 온-칩 메모리 내에서 처리 가능한 최대 배치 크기를 결정하는 단계 및 결정된 최대 배치 크기들을 모델에 적용할 후보 배치 크기로 결정하고, 후보 배치 크기들 중 모델이 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나를 모델에 적용되는 최종 배치 크기로 결정하는 단계를 포함한다.

대 표 도



(52) CPC특허분류
G06F 9/5077 (2013.01)
G06N 3/04 (2013.01)

명 세 서

청구범위

청구항 1

호스트 프로세서의 동작 방법에 있어서,

가속기에서 실행하고자 하는 모델을 복수의 스테이지들(plurality of stages)로 분할하는 단계;

상기 복수의 스테이지들 각각에 대해 상기 가속기의 온-칩 메모리 내에서 처리 가능한 최대 배치 크기(maximum batch size)를 결정하는 단계; 및

상기 결정된 최대 배치 크기들을 상기 모델에 적용할 후보 배치 크기로 결정하고, 상기 후보 배치 크기들 중 상기 모델이 상기 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나를 상기 모델에 적용되는 최종 배치 크기로 결정하는 단계

를 포함하는

호스트 프로세서의 동작 방법.

청구항 2

제1항에 있어서,

상기 메모리 액세스 비용은

상기 모델의 가중치에 대한 메모리 액세스 비용, 상기 모델의 중간 특징 맵(intermediate feature map; IMFM)에 대한 메모리 액세스 비용, 상기 모델에 대한 전체 배치 크기, 상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기 및 상기 가속기의 오프-칩 메모리의 대역폭에 기초하여 결정되는,

호스트 프로세서의 동작 방법.

청구항 3

제2항에 있어서,

상기 메모리 액세스 비용은 상기 가속기의 오프-칩 메모리에 대한 액세스 비용인,

호스트 프로세서의 동작 방법.

청구항 4

제2항에 있어서,

상기 중간 특징 맵에 대한 메모리 액세스 비용은

상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기보다 작은 최대 배치 크기를 가지는 스테이지에서 발생하는,

호스트 프로세서의 동작 방법.

청구항 5

제4항에 있어서,

상기 중간 특징 맵에 대한 메모리 액세스 비용은

상기 스테이지에서 발생하는 중간 특징 맵에 대한 1회 메모리 액세스 비용, 상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기 및 해당 스테이지에 대해 결정된 최대 배치 크기에 기초하여 결정되는,

호스트 프로세서의 동작 방법.

청구항 6

제2항에 있어서,

상기 가중치에 대한 메모리 액세스 비용은 상기 모델에 적용되는 가중치들에 대한 1회 메모리 액세스 비용인,

호스트 프로세서의 동작 방법.

청구항 7

제1항에 있어서,

상기 연산 비용은

상기 가속기에서 상기 복수의 스테이지들 각각을 처리하는 연산 시간, 상기 연산 비용을 계산하고자 하는 후보 배치 크기, 상기 모델에 대한 전체 배치 크기 및 상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기에 기초하여 결정되는,

호스트 프로세서의 동작 방법.

청구항 8

제1항에 있어서,

상기 복수의 스테이지들 각각의 상기 최대 배치 크기를 결정하는 단계는

대응하는 스테이지의 가중치, 입력 특징 맵, 출력 특징 맵의 크기, 상기 온-칩 메모리의 크기에 기초하여 결정되는,

호스트 프로세서의 동작 방법.

청구항 9

제1항에 있어서,

상기 분할하는 단계는

상기 모델을 상기 가속기에 포함된 연산 유닛에서 한 번에 처리 가능한 연산 단위 또는 상기 모델에 포함된 레이어 단위에 기반하여 상기 복수의 스테이지들로 분할하는,

호스트 프로세서의 동작 방법.

청구항 10

제1항에 있어서,

상기 최종 배치 크기에 기반하여 상기 모델을 상기 가속기에서 실행시키기 위한 명령어 집합(instruction set)을 생성하는 단계

를 더 포함하는,

호스트 프로세서의 동작 방법.

청구항 11

가속기의 동작 방법에 있어서,

상기 가속기에서 실행하고자 하는 모델에 대한 명령어 집합을 수신하는 단계; 및

상기 명령어 집합에 기초하여, 상기 모델에 포함된 상기 복수의 스테이지들에 동일하게 적용되는 공통 배치 크기에 따른 복수의 입력들을 상기 복수의 스테이지들 각각에서 처리하는 단계

를 포함하고,

상기 복수의 스테이지들 중에서 상기 공통 배치 크기보다 작은 상기 가속기의 온-칩 메모리 내에서 처리 가능한 최대 배치 크기를 가지는 스테이지가 반복 수행되어 상기 공통 배치 크기에 따른 중간 특징 맵이 다음 스테이지로 전달되는,

가속기의 동작 방법.

청구항 12

제11항에 있어서,

상기 모델에서 처리되어야 할 전체 배치 크기가 상기 공통 배치 크기보다 크다면, 상기 복수의 스테이지들이 반복 수행되어 상기 전체 배치 크기에 따른 입력들이 처리되는,

가속기의 동작 방법.

청구항 13

제11항에 있어서,

상기 공통 배치 크기는

상기 복수의 스테이지들 각각에 대해 결정된 상기 온-칩 메모리 내에서 처리 가능한 최대 배치 크기들 중 상기 모델이 상기 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나로 결정되는,

가속기의 동작 방법.

청구항 14

제1항 내지 제13항 중에서 어느 하나의 항의 방법을 실행시키기 위한 프로그램이 기록된 컴퓨터 판독 가능한 저장 매체.

청구항 15

모델을 가속기에서 실행하기 위한 요청에 응답하여 상기 가속기에 의해 실행 가능한 명령어 집합을 생성하는 호스트 프로세서; 및

상기 명령어 집합이 실행되면, 상기 모델에 포함된 상기 복수의 스테이지들에 동일하게 적용되는 공통 배치 크기에 따른 복수의 입력들을 상기 복수의 스테이지들 각각에서 처리하는 가속기

를 포함하고,

상기 호스트 프로세서는

상기 복수의 스테이지들 각각에 대해 결정된 상기 온-칩 메모리 내에서 처리 가능한 최대 배치 크기들 중 상기 모델이 상기 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나를 상기 공통 배치 크기로 결정하는,

전자 장치.

청구항 16

제15항에 있어서,

상기 메모리 액세스 비용은

상기 모델의 가중치에 대한 메모리 액세스 비용, 상기 모델의 중간 특징 맵에 대한 메모리 액세스 비용, 상기 모델에 대한 전체 배치 크기, 상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기에 기초하여 결정되는,

전자 장치.

청구항 17

제16항에 있어서,

상기 중간 특징 맵에 대한 메모리 액세스 비용은

상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기보다 작은 최대 배치 크기를 가지는 스테이지에서 발생하는

전자 장치.

청구항 18

제17항에 있어서,

상기 중간 특징 맵에 대한 메모리 액세스 비용은

상기 스테이지에서 발생하는 중간 특징 맵에 대한 1회 메모리 액세스 비용, 상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기 및 해당 스테이지에 대해 결정된 최대 배치 크기에 기초하여 결정되는,

전자 장치.

청구항 19

제15항에 있어서,

상기 복수의 스테이지들 각각의 상기 최대 배치 크기는

대응하는 스테이지의 가중치, 입력 특징 맵, 출력 특징 맵의 크기, 상기 온-칩 메모리의 크기에 기초하여 결정되는,

전자 장치.

청구항 20

제15항에 있어서,

상기 복수의 스테이지들은

상기 가속기에 포함된 연산 유닛에서 한 번에 처리 가능한 연산 단위 또는 상기 모델에 포함된 레이어 단위에 기반하여 상기 모델로부터 분할되는,

전자 장치.

발명의 설명

기 술 분 야

[0001] 아래 실시예들은 호스트 프로세서 및 가속기의 동작 방법 및 이들을 포함한 전자 장치에 관한 것이다.

배 경 기 술

[0002] 뉴럴 네트워크에 기반한 추론 서비스에서 낮은 레이턴시 달성을 위해 1개의 배치 처리가 아닌 여러 배치들을 함께 처리하는 기술들에 대한 연구가 활발히 진행되고 있다. 또한, 뉴럴 네트워크에서 요구되는 메모리의 캐패시티(capacity)와 대역폭(bandwidth)가 점차 증가하게 되면서 제한된 리소스 내에서 효율적으로 여러 배치들을 빠르게 처리하려는 연구가 진행되고 있다.

발명의 내용

해결하려는 과제

과제의 해결 수단

[0004] 일실시예에 따른 호스트 프로세서의 동작 방법은 가속기에서 실행하고자 하는 모델을 복수의 스테이지들(plurality of stages)로 분할하는 단계; 상기 복수의 스테이지들 각각에 대해 상기 가속기의 온-칩 메모리 내에서 처리 가능한 최대 배치 크기(maximum batch size)를 결정하는 단계; 및 상기 결정된 최대 배치 크기들을 상기 모델에 적용할 후보 배치 크기로 결정하고, 상기 후보 배치 크기들 중 상기 모델이 상기 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나를 상기 모델에 적용되는 최종 배치 크기로 결정하는 단계를 포함한다.

[0005] 일실시예에 따른 호스트 프로세서의 동작 방법에서 상기 메모리 액세스 비용은 상기 모델의 가중치에 대한 메모리 액세스 비용, 상기 모델의 중간 특징 맵(intermediate feature map; IMFM)에 대한 메모리 액세스 비용, 상기 모델에 대한 전체 배치 크기, 상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기 및 상기 가속기의 오프-칩 메모리의 대역폭에 기초하여 결정될 수 있다.

[0006] 일실시예에 따른 호스트 프로세서의 동작 방법에서 상기 메모리 액세스 비용은 상기 가속기의 오프-칩 메모리에 대한 액세스 비용일 수 있다.

[0007] 일실시예에 따른 호스트 프로세서의 동작 방법에서 상기 중간 특징 맵에 대한 메모리 액세스 비용은 상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기보다 작은 최대 배치 크기를 가지는 스테이지에서 발생할 수 있다.

[0008] 일실시예에 따른 호스트 프로세서의 동작 방법에서 상기 중간 특징 맵에 대한 메모리 액세스 비용은 상기 스테이지에서 발생하는 중간 특징 맵에 대한 1회 메모리 액세스 비용, 상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기 및 해당 스테이지에 대해 결정된 최대 배치 크기에 기초하여 결정될 수 있다.

[0009] 일실시예에 따른 호스트 프로세서의 동작 방법에서 상기 가중치에 대한 메모리 액세스 비용은 상기 모델에 적용되는 가중치들에 대한 1회 메모리 액세스 비용일 수 있다.

[0010] 일실시예에 따른 호스트 프로세서의 동작 방법에서 상기 연산 비용은 상기 가속기에서 상기 복수의 스테이지들 각각을 처리하는 연산 시간, 상기 연산 비용을 계산하고자 하는 후보 배치 크기, 상기 모델에 대한 전체 배치

크기 및 상기 메모리 액세스 비용을 계산하고자 하는 후보 배치 크기 에 기초하여 결정될 수 있다.

[0011] 일실시예에 따른 호스트 프로세서의 동작 방법에서 상기 복수의 스테이지들 각각의 상기 최대 배치 크기를 결정하는 단계는 대응하는 스테이지의 가중치, 입력 특징 맵, 출력 특징 맵의 크기, 상기 온-칩 메모리의 크기에 기초하여 결정될 수 있다.

[0012] 일실시예에 따른 호스트 프로세서의 동작 방법에서 상기 분할하는 단계는 상기 모델을 상기 가속기에 포함된 연산 유닛에서 한 번에 처리 가능한 연산 단위 또는 상기 모델에 포함된 레이어 단위에 기반하여 상기 복수의 스테이지들로 분할할 수 있다.

[0013] 일실시예에 따른 호스트 프로세서의 동작 방법은 상기 최종 배치 크기에 기반하여 상기 모델을 상기 가속기에서 실행시키기 위한 명령어 집합(instruction set)을 생성하는 단계를 더 포함할 수 있다.

[0014] 일실시예에 따른 가속기의 동작 방법은 상기 가속기에서 실행하고자 하는 모델에 대한 명령어 집합을 수신하는 단계; 및 상기 명령어 집합에 기초하여, 상기 모델에 포함된 상기 복수의 스테이지들에 동일하게 적용되는 공통 배치 크기에 따른 복수의 입력들을 상기 복수의 스테이지들 각각에서 처리하는 단계를 포함하고, 상기 복수의 스테이지들 중에서 상기 공통 배치 크기보다 작은 상기 가속기의 온-칩 메모리 내에서 처리 가능한 최대 배치 크기를 가지는 스테이지가 반복 수행되어 상기 공통 배치 크기에 따른 중간 특징 맵이 다음 스테이지로 전달된다.

[0015] 일실시예에 따른 가속기의 동작 방법에서 상기 모델에서 처리되어야 할 전체 배치 크기가 상기 공통 배치 크기보다 크다면, 상기 복수의 스테이지들이 반복 수행되어 상기 전체 배치 크기에 따른 입력들이 처리될 수 있다.

[0016] 일실시예에 따른 가속기의 동작 방법에서 상기 공통 배치 크기는 상기 복수의 스테이지들 각각에 대해 결정된 상기 온-칩 메모리 내에서 처리 가능한 최대 배치 크기들 중 상기 모델이 상기 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나로 결정될 수 있다.

[0017] 일실시예에 따른 전자 장치는 모델을 가속기에서 실행하기 위한 요청에 응답하여 상기 가속기에 의해 실행 가능한 명령어 집합을 생성하는 호스트 프로세서; 및 상기 명령어 집합이 실행되면, 상기 모델에 포함된 상기 복수의 스테이지들에 동일하게 적용되는 공통 배치 크기에 따른 복수의 입력들을 상기 복수의 스테이지들 각각에서 처리하는 가속기를 포함하고, 상기 호스트 프로세서는 상기 복수의 스테이지들 각각에 대해 결정된 상기 온-칩 메모리 내에서 처리 가능한 최대 배치 크기들 중 상기 모델이 상기 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나를 상기 공통 배치 크기로 결정한다.

도면의 간단한 설명

[0019] 도 1은 일실시예에 따른 전자 장치를 나타낸 도면이다.

도 2는 일실시예에 따른 가속기 보드를 나타낸 도면이다.

도 3 및 도 4는 일실시예에 따라 모델에 적용할 배치 개수를 결정하는 과정을 설명하기 위한 도면이다.

도 5는 일실시예에 따른 배치 처리에 기반한 추론 과정을 설명하기 위한 도면이다.

도 6은 일실시예에 따른 컴파일 타임과 런타임에서의 동작을 설명하기 위한 도면이다.

도 7은 일실시예에 따른 호스트 프로세서의 동작 방법을 나타낸 도면이다.

도 8은 일실시예에 따른 가속기의 동작 방법을 나타낸 도면이다.

도 9 및 도 10은 일실시예에 따른 전자 장치의 예시들을 나타낸 도면이다.

발명을 실시하기 위한 구체적인 내용

[0020] 실시예들에 대한 특정한 구조적 또는 기능적 설명들은 단지 예시를 위한 목적으로 개시된 것으로서, 다양한 형태로 변경되어 실시될 수 있다. 따라서, 실시예들은 특정한 개시형태로 한정되는 것이 아니며, 본 명세서의 범위는 기술적 사상에 포함되는 변경, 균등물, 또는 대체물을 포함한다.

[0021] 제1 또는 제2 등의 용어를 다양한 구성요소들을 설명하는데 사용될 수 있지만, 이런 용어들은 하나의 구성요소를 다른 구성요소로부터 구별하는 목적으로만 해석되어야 한다. 예를 들어, 제1 구성요소는 제2 구성요소로 명

명될 수 있고, 유사하게 제2 구성요소는 제1 구성요소로도 명명될 수 있다.

[0022] 어떤 구성요소가 다른 구성요소에 "연결되어" 있다고 언급된 때에는, 그 다른 구성요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 할 것이다.

[0023] 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 설명된 특징, 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것이 존재함으로 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0024] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 해당 기술분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 것과 동일한 의미를 가진다. 일반적으로 사용되는 사전에 정의되어 있는 것과 같은 용어들은 관련 기술의 문맥상 가지는 의미와 일치하는 의미를 갖는 것으로 해석되어야 하며, 본 명세서에서 명백하게 정의하지 않는 한, 이상적이거나 과도하게 형식적인 의미로 해석되지 않는다.

[0025] 이하, 실시예들을 첨부된 도면을 참조하여 상세하게 설명한다. 아래의 특정한 구조적 내지 기능적 설명들은 단지 실시예들을 설명하기 위한 목적으로 예시된 것으로, 실시예의 범위가 본문에 설명된 내용에 한정되는 것으로 해석되어서는 안된다. 관련 기술 분야에서 통상의 지식을 가진 자라면 이러한 기재로부터 다양한 수정 및 변형이 가능하다. 또한, 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타내며, 공지된 기능 및 구조는 생략하도록 한다.

[0027] 도 1은 일실시예에 따른 전자 장치를 나타낸 도면이다.

[0028] 도 1을 참조하면, 일실시예에 따른 전자 장치(100)은 호스트 프로세서(110) 및 가속기 보드(accelerator board)(120)를 포함할 수 있다. 호스트 프로세서(110) 및 가속기 보드(120)는 버스(bus), NoC(Network on a Chip), PCIe(Peripheral Component Interconnect Express) 등을 통하여 서로 통신할 수 있다.

[0029] 호스트 프로세서(110)는 전자 장치(100)에 포함된 컴포넌트들의 동작을 제어하는 장치로, 예를 들어, 중앙 처리 장치(CPU; Central Processing Unit) 등을 포함할 수 있다. 호스트 프로세서(110)는 뉴럴 네트워크를 가속기 보드(120)에서 처리하기 위한 하나 이상의 요청을 수신하고, 해당 요청에 응답하여 가속기 보드(120)에서 실행 가능한 명령어 집합을 생성한다. 요청은 뉴럴 네트워크에 기반한 데이터 추론을 위한 것으로, 예를 들어, 객체 인식, 패턴 인식, 컴퓨터 비전, 음성 인식, 기계 번역, 기계 통역, 추천 서비스, 개인 맞춤 서비스, 영상 처리, 자율 주행 등을 위해 가속기 보드(120)로 하여금 뉴럴 네트워크를 실행하게 하여 데이터 추론 결과를 얻기 위한 것일 수 있다. 호스트 프로세서(110)에서 명령어 집합이 생성되는 과정은 가속기 보드(120)에서 추론이 수행되기 이전에 미리 한 번만 실행되고, 실제 사용자로부터 추론이 요청되면 기 생성된 명령어 집합이 가속기 보드(120)에서 실행될 수 있다.

[0030] 호스트 프로세서(110)는 가속기 보드(120)에서 실행하고자 하는 뉴럴 네트워크를 복수의 스테이지들로 분할하고, 복수의 스테이지들 각각에 대해 가속기 보드(120)의 온-칩 메모리 내에서 처리 가능한 최대 배치 크기를 결정하고, 결정된 최대 배치 크기들을 뉴럴 네트워크에 적용할 후보 배치 크기로 결정하고, 후보 배치 크기들 중 뉴럴 네트워크가 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나를 뉴럴 네트워크에 적용되는 최종 배치 크기로 결정한다. 그리고, 호스트 프로세서(110)는 최종 배치 크기에 기반하여 뉴럴 네트워크를 가속기에서 실행시키기 위한 명령어 집합을 생성할 수 있다.

[0031] 가속기 보드(120)는 호스트 프로세서(110)의 명령어 집합에 따른 뉴럴 네트워크를 실행하여 입력되는 데이터를 추론하는 AI 가속기(Artificial Intelligence accelerator)로서, 호스트 프로세서(110)와 구별되는 별도의 프로세서일 수 있다. 예를 들어, 가속기 보드(120)은 NPU(Neural Processing Unit), GPU(Graphics Processing Unit), TPU(Tensor Processing Unit), DSP(Digital Signal Processor) 등일 수 있다. 가속기 보드(120)는 뉴럴 네트워크에 따른 연산들의 특성 상 범용의 호스트 프로세서(110)에서 처리되기 보다는 별도의 전용 프로세서(다시 말해, 가속기 보드(120))에서 처리되는 것이 보다 효율적인 작업들을 처리할 수 있다.

[0032] 뉴럴 네트워크는 복수의 레이어들을 포함한다. 일실시예에서, 뉴럴 네트워크는 입력 레이어, 복수의 히든 레이어들 및 출력 레이어를 포함한다. 각각의 레이어들은 인공 뉴런이라고도 불리는 복수의 노드들을 포함한다. 각 노드는 하나 이상의 입력 및 출력을 가지는 계산 단위를 나타내고, 노드들은 상호 연결될 수 있다. 노드들

간의 연결에는 가중치가 설정될 수 있으며, 이러한 가중치는 조정 또는 변경될 수 있다. 가중치는 연관된 데이터 값을 증폭, 감소 또는 유지시킴으로써 해당 데이터 값이 최종 결과에 미치는 영향도를 결정할 수 있다. 출력 레이어에 포함된 각각의 노드에는 이전 레이어에 포함된 노드들의 가중된 입력들이 입력될 수 있다. 가중된 데이터가 임의의 레이어로부터 다음 레이어로 입력되는 과정을 전파(propagation)라고 지칭할 수 있다.

[0033] 일실시예에 따른 가속기 보드(120)에서 추론 대상이 되는 데이터가 다수일 수 있으며, 이 경우 여러 데이터를 함께 추론하는 배치 처리를 통해 추론 동작 효율을 향상시킬 수 있다. 이때, 가속기 보드(120)의 연산 리소스와 메모리 액세스 리소스를 고려한 배치 개수를 결정함으로써, 가속기 보드(120)에 최적화된 배치 처리를 기대할 수 있다.

[0034] 본 명세서에서는 설명의 편의를 위해 가속기 보드(120)를 가속기라고 표현할 수 있고, 뉴럴 네트워크를 모델이라고 표현할 수 있다.

[0036] 도 2는 일실시예에 따른 가속기 보드를 나타낸 도면이다.

[0037] 도 2를 참조하면, 일실시예에 따른 가속기 보드(200)는 오프-칩 메모리(off-chip memory)(210) 및 가속기 칩(220)을 포함할 수 있다. 가속기 칩(220)은 프로세서(221), DMA 엔진(Direct Memory Access Engine)(223), 버퍼(buffer)(225) 및 복수의 연산 유닛들(plurality of computation units)(227)을 포함할 수 있다.

[0038] 오프-칩 메모리(210)는 가속기 칩(220)의 외부에 배치된 메모리로서, 예를 들어, DRAM(Dynamic Random Access Memory) 등을 포함할 수 있다. 오프-칩 메모리(210)는 추론 대상 데이터 및/또는 가속기 칩(220)에서 실행할 뉴럴 네트워크의 파라미터들을 저장할 수 있으며, 저장된 데이터는 이후 추론 수행을 위해 가속기 칩(220)으로 전달될 수 있다. 또한, 오프-칩 메모리(210)는 가속기 칩(220)에서 뉴럴 네트워크를 실행하는 데 가속기 칩(220) 내부의 온-칩 메모리(on-chip memory)가 충분하지 않은 경우에 활용될 수도 있다.

[0039] 오프-칩 메모리(210)는 가속기 칩(220) 내부의 온-칩 메모리보다 큰 메모리 용량을 가지나, 뉴럴 네트워크 실행 시 가속기 칩(220)이 오프-칩 메모리(210)로 액세스하는 비용이 내부의 온-칩 메모리로 액세스하는 비용보다 크다. 따라서, 가속기 칩(220)에서 뉴럴 네트워크 실행 시 빠른 추론을 위해서는 오프-칩 메모리(210)에 대한 액세스 비용을 줄이는 것이 중요하게 작용할 수 있다. 메모리 액세스 비용은 해당 메모리에 액세스하여 데이터를 읽거나 쓸 때 요구되는 전력 및/또는 시간을 나타낼 수 있다.

[0040] 가속기 칩(220)는 뉴럴 네트워크에 따른 연산들의 특성 상 범용의 호스트 프로세서에서 처리되기 보다는 별도의 전용 프로세서(다시 말해, 가속기 칩(220))에서 처리되는 것이 보다 효율적인 작업들을 처리할 수 있다. 이때 가속기 칩(220)에 포함된 하나 이상의 프로세싱 엘리먼트들(PEs; Processing Elements) 및 온-칩 메모리가 활용될 수 있다.

[0041] 온-칩 메모리는 가속기 칩(220) 내부의 글로벌 쉐어드 버퍼(global shared buffer) 및/또는 로컬 버퍼(local buffer) 등을 포함하는 장치로서, 가속기 칩(220) 외부에 위치하는 오프-칩 메모리(210)와 구분될 수 있다. 예를 들어, 온-칩 메모리는 주소 공간(address space)을 통해 액세스 가능한 스크래치패드 메모리(scratchpad memory), SRAM(Static Random Access Memory) 등을 포함할 수 있다.

[0042] 도 2에서 온-칩 메모리는 버퍼(225)로 표현될 수 있다. 버퍼(225)는 추후 설명한 태스크의 동작이 정의된 태스크 커널(task kernel), 해당 태스크에 적용되는 가중치(weight), 입력 특징 맵(input feature map; IFM), 출력 특징 맵(output feature map; OFM)을 포함할 수 있다. 이때, 입력 특징 맵과 출력 특징 맵은 해당 태스크에서 처리되는 배치 크기만큼 버퍼(225)에 저장될 수 있다. 버퍼(225)로 데이터가 로딩되거나, 버퍼(225)로부터 데이터가 출력되는 동작은 DMA 엔진(223)에 의해 제어될 수 있다.

[0043] 프로세서(221)는 태스크 커널에 정의된 동작에 따라 DMA 엔진(223)의 데이터 이동 및/또는 복수의 연산 유닛들(227)의 연산 처리를 제어할 수 있다.

[0044] 앞서 설명한 프로세싱 엘리먼트들은 도 2에서 복수의 연산 유닛들(227)로 표현될 수 있다. 복수의 연산 유닛들(227)은 뉴럴 네트워크에 따른 연산(예컨대, MAC(multiply-accumulation) 연산 등)을 수행할 수 있다.

[0045] 일실시예에서 가속기 보드(200)는 도 1의 호스트 프로세서(110)로부터의 명령어 집합에 따른 배치 처리를 수행할 수 있으며, 최적의 배치 개수를 정하는 과정에 대해서는 이하 도면들을 참조하여 상세히 설명한다.

[0047] 도 3 및 도 4는 일실시예에 따라 모델에 적용할 배치 개수를 결정하는 과정을 설명하기 위한 도면이다.

[0048] 도 3을 참조하면, 일실시예에 따라 가속기에서 실행하고자 하는 모델은 복수의 스테이지들로 분할된다. 복수의 스테이지들은 모델이 가속기에 포함된 연산 유닛에서 한 번에 처리 가능한 연산 단위 또는 모델에 포함된 레이어 단위로 분할된 것일 수 있다. 연산 유닛에서 한 번에 처리 가능한 연산 단위는 태스크로 표현될 수 있다. 하나의 태스크는 하나의 레이어에 대응하거나, 또는 한 레이어보다 작거나 클 수 있다. 모델을 태스크 단위로 분할하여 배치 처리를 수행함으로써, 연산 유닛의 활용률을 높게 유지시킬 수 있다. 각 태스크는 실행되는 순간에 가속기 리소스를 점유하여 처리될 수 있다. 이처럼, 모델이 태스크 단위 또는 레이어 단위에 기반하여 복수의 스테이지들로 분할될 수 있지만, 아래에서는 설명의 편의를 위해 태스크 단위로 분할된 경우를 예로 설명한다. 도 3에서는 모델이 6개의 태스크들로 분할된 예시를 기준으로 설명한다. 다만, 이러한 설명이 모델을 한정하는 것은 아니고, 다양한 개수의 태스크들 또는 레이어들에도 본 명세서의 설명이 적용될 수 있다.

[0049] 그리고, 복수의 태스크들 각각에 대해 가속기의 온-칩 메모리 내에서 처리 가능한 최대 배치 크기가 결정될 수 있다. 태스크마다 필요로 하는 입력 특징 맵, 가중치, 출력 특징 맵의 크기가 다르고, 가중치는 배치 크기가 복수로 결정되더라도 재사용되므로 배치 크기와 상관없이 일정하게 고정되나 배치 크기에 따라 입력 특징 맵과 출력 특징 맵의 총 크기가 달라지는 특성을 고려하여, 각 태스크들에 대한 최대 배치 크기가 결정될 수 있다. 이러한 최대 배치 크기는 TIB(Task-Inner batch)로 지칭될 수 있다.

[0050] 각 태스크의 TIB는 weight + (IFM + OFM) x batch < on-chip memory를 만족하는 최대 배치 개수로 결정될 수 있다. 다시 말해, 입력 특징 맵의 크기 및 출력 특징 맵의 크기 간 합과 해당 태스크의 최대 배치 크기의 곱이 온-칩 메모리의 크기에서 가중치의 크기를 뺀 값을 초과하지 않도록, 최대 배치 크기가 결정될 수 있다. 만약 입력되는 총 배치 크기보다 큰 TIB를 가지는 태스크에 대해서는 TIB를 총 배치 크기와 동일하게 설정할 수 있다.

[0051] 앞서 설명한 것처럼, 오프-칩 메모리에 대한 액세스 비용이 온-칩 메모리에 대한 액세스 비용보다 크므로, 복수의 태스크들 각각에 대해 온-칩 메모리 내에서 처리 가능한 최대 배치 크기를 결정함으로써, 오프-칩 메모리 액세스가 발생하는 것을 최대한 억제할 수 있다.

[0052] 실시예에 따라 모델이 레이어 단위로 분할되는 경우에는 최대 배치 크기가 LIB(Layer-Inner batch)로 지칭되고, 각 레이어의 LIB는 weight + (IFM + OFM) x batch $\leq$ on-chip memory를 만족하는 최대 배치 개수로 결정될 수 있다.

[0053] 그리고, 각 태스크로부터 생성되는 중간 특징 맵을 최대한 온-칩 메모리에 유지하는 상태로 다음 태스크를 처리하기 위해 전체 태스크들이 공통으로 처리해야 하는 공통 배치 크기가 결정될 수 있다. 이러한 공통 배치 크기는 복수의 태스크들을 포함한 네트워크 전체에 적용되는 것으로 NIB(Network-Inner batch)로 지칭될 수 있다.

[0054] NIB는 각 태스크에서 생성되어야 할 IMFM의 배치 크기로, NIB보다 크거나 같은 TIB를 갖는 태스크는 NIB만큼만 IMFM를 생성하여 온-칩 메모리 내에서 다음 태스크로 생성된 IMFM를 전달할 수 있으나, NIB보다 작은 TIB를 갖는 태스크는 NIB만큼 생성된 IMFM를 모두 온-칩 메모리에 저장할 수 없기에 오프-칩 메모리를 활용할 수 밖에 없다. 다음 태스크는 오프-칩 메모리에서 IMFM를 로딩해야 해서, 오프-칩 메모리 액세스가 발생하게 된다. 따라서, 해당 모델에 최적의 NIB를 구하는 것이 중요하다.

[0055] 복수의 태스크들 각각에 대해 결정된 TIB를 모델에 적용할 후보 NIB로 결정하고, 후보 NIB들 중 레이턴시가 가장 적은 하나를 최종 NIB로 선정할 수 있다. 다시 말해, 후보 NIB들 중 모델이 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나를 최종 NIB로 결정한다. 각 후보 NIB의 비용은 다음과 같이 표현될 수 있다.

수학식 1

[0056]
$$Cost(NIB) = Memory\_cost(NIB) + Computation\_cost(NIB)$$

[0057] 위의 수학식 1에서, $Cost(NIB)$는 해당 후보 NIB의 비용을 나타내고, $Memory_cost(NIB)$은 해당 부호의 NIB의 메

모리 액세스 비용을 나타내며, *Computation_cost(NIB)*은 해당 부호의 NIB의 연산 비용을 나타낸다.

[0058] 연산 비용은 각 태스크가 후보 NIB를 처리하는 데 걸리는 순수 연산 시간의 총 합으로 결정될 수 있으며, 다음과 같이 표현될 수 있다.

수학식 2

[0059] $$Computation\_cost() = SUM(comp\_time\_per\_task \times NIB) \times NOB$$

[0060] 위의 수학식 2에서, NOB(Network-Outer batch)는 전체 배치를 처리하기 위해 해당 모델이 몇 번 반복 수행되는지를 나타내는 것으로, *CEIL(Total batch / NIB)*으로 결정될 수 있다.

[0061] 메모리 액세스 비용은 오프-칩 메모리에 대한 액세스 비용을 나타내며, 각 태스크의 가중치에 대한 메모리 액세스 비용과 IMFM에 대한 메모리 액세스 비용으로 구분될 수 있다. 메모리 액세스 비용은 다음과 같이 표현될 수 있다.

수학식 3

[0062] $$Memory\_cost() = (cost(weight\_access) + cost(IMFM\_access)) \times NOB \times 1/Bandwidth$$

[0063] 위의 수학식 3에서, *CEIL*은 소수점 이하를 올림한 정수를 반환하는 함수이고, *Bandwidth*는 가속기의 오프-칩 메모리의 대역폭을 나타낸다. 해당 모델이 한 번 수행되면 NIB만큼의 배치가 처리되므로, 전체 배치를 처리하기 위해서는 해당 모델이 NOB만큼 반복 수행될 필요가 있다. 메모리 액세스 비용은 모델이 실행될 때마다 발생하므로, 가중치에 대한 메모리 액세스 비용과 IMFM에 대한 메모리 액세스 비용의 합과 NOB 간 곱에 의해 결정될 수 있다.

[0064] 후보 NIB보다 작은 TIB를 갖는 태스크가 NIB만큼의 배치를 처리하기 위해 반복 수행되더라도 동일한 가중치가 사용되기 때문에, 가중치에 대한 메모리 액세스 비용은 태스크의 반복 처리 횟수와는 무관하며, 앞선 설명처럼 NOB만큼 모델이 반복 수행되는 것에 영향 받을 수 있다. 가중치에 대한 메모리 액세스 비용은 다음과 같이 표현될 수 있다.

수학식 4

[0065] $$cost(weight\_access) = Total\_weight$$

[0066] IMFM에 대한 메모리 액세스 비용의 경우, 후보 NIB보다 작은 TIB를 갖는 태스크에서 NIB만큼 배치를 처리하기 위해서는 IMFM가 외부-칩 메모리에 저장되기 때문에, 외부-칩 메모리에 대한 액세스 비용이 발생한다. 반대로, 후보 NIB보다 크거나 같은 TIB를 갖는 태스크에서는 온-칩 메모리만으로도 NIB만큼 배치를 처리할 수 있기 때문에, 오프-칩 메모리 액세스가 발생하지 않는다. 이러한 IMFM에 대한 메모리 액세스 비용은 다음과 같이 표현될 수 있다.

수학식 5

[0067] $$Cost(IMFM\_access) = Task\_IMFM \times TOB, \text{ where } TIB < NIB$$

[0068] 위의 수학식 5에서, TOB(Task-Outer batch)는 각 태스크가 NIB만큼의 배치를 처리하기 위해 반복 수행해야 하는

횟수를 나타내는 것으로, *CEIL(NIB / TIB)*으로 결정될 수 있다. 해당 태스크가 한 번 수행되면 TIB만큼의 배치가 처리되므로, 총 NIB만큼의 배치를 처리하기 위해서는 해당 태스크가 TOB만큼 반복 수행될 필요가 있다. TOB가 2 이상이면, 해당 태스크의 TIB가 NIB보다 작아 외부-칩 메모리에 대한 액세스 비용이 발생함을 의미할 수 있다. 이때, 메모리 액세스 비용은 해당 태스크에서 생성되는 IMFM 크기와 TOB 간 곱에 의해 결정될 수 있다.

[0069] 실시예에 따라 모델이 레이어 단위로 분할되는 경우에는 해당 레이어는 LOB(Layer-Outer batch)만큼 반복 수행될 수 있으며, LOB는 *CEIL(NIB / LIB)*으로 결정될 수 있다.

[0070] 앞서 설명한 비용을 후보 TIB마다 결정하고, 가장 적은 비용을 갖는 TIB를 최종 TIB로 결정하여 모델에 적용시킴으로써, 추론 시 오프-칩 메모리 액세스가 발생하는 것을 최소화시키고 연산 비용도 고려하여 빠른 추론을 기대할 수 있다.

[0072] 도 4를 참조하면, 일실시예에 따라 전체 배치 16개에 대한 입력을 처리하는 경우에 최종 NIB를 결정하는 예시가 도시된다. 도 4에 도시된 구체적인 숫자는 설명의 편의를 위한 예시로서, 실시예가 이로 한정되지 않는다.

[0073] 최종 NIB는 복수의 TIB들에 해당하는 후보 NIB들 중 어느 하나로 결정되는데, 도 4의 예시에서 후보 NIB들은 중복되는 TIB를 제외한 2, 4, 8, 16일 수 있다. 후보 NIB들 각각에 대한 비용을 계산하고, 가장 적은 비용을 갖는 후보 NIB가 모델에 적용되는 최종 NIB로 결정될 수 있다.

[0074] 도 4의 예시에서 최종 NIB는 4로 결정될 수 있는데, 태스크 1은 TIB가 2이므로 2개의 입력을 온-칩 메모리에 로딩하여 한 번 처리하고, TOB가 2이기 때문에 태스크 1를 2번 반복하여 총 4개의 OFM(달리 표현하면, IMFM)이 생성될 수 있다. 이때, 오프-칩 메모리에 대한 액세스 비용이 발생할 수 있다. 태스크 2 내지 6의 경우 각 TIB가 최종 NIB보다 크거나 같으므로, 오프-칩 메모리 액세스 없이 온-칩 메모리만으로 4개의 배치를 처리할 수 있다. 이러한 과정을 NOB에 해당하는 4번 반복하여 전체 배치 16개가 모두 처리될 수 있다.

[0075] 태스크 2 내지 6은 온-칩 메모리 내에서 처리 가능한 최대 배치 크기보다 적은 배치를 처리하게 되어, 가속기 리소스를 충분히 사용하지 않아 모델을 더 많이 반복해야 하는 것으로 보일 수 있다. 그러나, 앞선 도 3의 설명처럼 NIB의 비용 계산 시 메모리 액세스 비용뿐만 아니라 연산 비용도 종합적으로 고려함으로써, 모델의 반복 수행을 줄이기 위해 최종 NIB를 크게 결정하여 오프-칩 메모리 액세스 비용의 급격한 증가가 전체 비용 증가로 이어지는 것을 방지하고, 최적의 NIB를 얻을 수 있다.

[0076] 호스트 프로세서는 결정된 TIB, NIB, TOB, NOB에 기초하여 추론 시 필요한 데이터의 메모리 주소를 설정해서 명령어 집합의 메타데이터에 기록하고 가속기로 전달할 수 있다. 가속기 내 펌웨어가 해당 정보를 토대로 모델을 반복 실행함으로써 전체 배치가 처리될 수 있다.

[0078] 도 5는 일실시예에 따른 배치 처리에 기반한 추론 과정을 설명하기 위한 도면이다.

[0079] 도 5를 참조하면, 일실시예에 따라 가속기에서 추론 수행 시 배치 처리에 따른 데이터 이동을 설명하기 위한 예시가 도시된다. 도 5의 각 태스크 박스는 해당 태스크 단계에서 온-칩 메모리에 저장되는 데이터들을 나타낼 수 있다.

[0080] 도 5의 예시에서, 최종 NIB가 4로 결정되었기 때문에, 각 태스크에서는 배치 4개에 대한 입력을 처리해야 한다. 태스크 1의 TIB가 2이기 때문에, 먼저 2개의 입력을 처리한 후 그 결과를 오프-칩 메모리(510)에 저장한 후, 다음 2개의 입력을 처리해야 한다. 따라서, 오프-칩 메모리(510)에 대한 액세스 비용이 발생하게 된다. 나머지 태스크 2 내지 6은 배치 4개에 대한 입력을 한 번에 처리할 수 있기 때문에 오프-칩 메모리(510)에 대한 액세스가 발생하지 않아 실행 속도가 빨라질 수 있다. 태스크 1 내지 6의 한 번 실행을 통해 4개의 배치들을 처리할 수 있기에, 전체 배치 16개를 처리하기 위해서는 태스크 1 내지 6의 실행이 4번 반복될 수 있다.

[0082] 도 6은 일실시예에 따른 컴파일 타임과 런타임에서의 동작을 설명하기 위한 도면이다.

[0083] 도 6을 참조하면, 일실시예에 따른 컴파일 타임에서 호스트 프로세서의 동작과 런타임에서 가속기의 동작이 도시된다.

[0084] 단계(610)에서, 호스트 프로세서는 가속기에서 실행하고자 하는 뉴럴 네트워크에 대한 정보와 처리하고자 하는

전체 배치 크기를 입력 받을 수 있다. 뉴럴 네트워크에 대한 정보는 뉴럴 네트워크에 포함된 레이어 수, 노드 수, 가중치 등을 포함할 수 있다. 또한, 호스트 프로세서는 가속기의 연산 리소스 및 메모리 액세스 리소스 등에 관한 하드웨어 정보를 입력 받을 수 있다.

[0085] 단계(620)에서, 호스트 프로세서는 뉴럴 네트워크를 태스크 단위로 분할함으로써, 태스크 단위 최적화를 수행할 수 있다. 실시예에 따라서는 타일링(Tiling), 벡터화(Vectorization), 텐서화(Tensorize) 등의 최적화 기법이 적용될 수 있다.

[0086] 단계(630)에서, 호스트 프로세서는 복수의 태스크들 각각에 대해 온-칩 메모리 내에서 처리 가능한 최대 배치 크기인 TIB를 계산할 수 있다.

[0087] 단계(640)에서, 호스트 프로세서는 계산된 TIB들 중 레이턴시가 최소가 되는 하나를 최적 NIB로 선정할 수 있다.

[0088] 단계(650)에서, 호스트 프로세서는 최적 NIB에 기반하여 뉴럴 네트워크를 가속기에서 실행시키기 위한 명령어 집합인 커널을 생성할 수 있다.

[0089] 단계(660)에서, 가속기는 호스트 프로세서에서 생성된 커널과 추론 대상이 되는 배치 데이터를 입력 받을 수 있다.

[0090] 단계(670)에서, 가속기는 커널을 실행하여 최적 NIB에 기반한 배치 처리를 통해 전체 배치에 대한 입력을 처리할 수 있다.

[0091] 단계(680)에서, 가속기는 배치 결과를 추론 수행 결과로 리턴할 수 있다.

[0093] 도 7은 일실시예에 따른 호스트 프로세서의 동작 방법을 나타낸 도면이다.

[0094] 도 7을 참조하면, 단계(710)에서 호스트 프로세서는 가속기에서 실행하고자 하는 모델을 복수의 스테이지들로 분할한다. 단계(720)에서 호스트 프로세서는 복수의 스테이지들 각각에 대해 가속기의 온-칩 메모리 내에서 처리 가능한 최대 배치 크기를 결정한다. 단계(730)에서 호스트 프로세서는 결정된 최대 배치 크기들을 모델에 적용할 후보 배치 크기로 결정하고, 후보 배치 크기들 중 모델이 가속기에서 실행될 때 발생하는 연산 비용 및 메모리 액세스 비용의 합이 최소가 되는 하나를 모델에 적용되는 최종 배치 크기로 결정한다.

[0095] 도 7에 도시된 각 단계들에는 도 1 내지 도 6을 통하여 전술한 사항들이 그대로 적용되므로, 보다 상세한 설명은 생략한다.

[0097] 도 8은 일실시예에 따른 가속기의 동작 방법을 나타낸 도면이다.

[0098] 도 8을 참조하면, 단계(810)에서 가속기는 실행하고자 하는 모델에 대한 명령어 집합을 수신한다. 단계(820)에서 가속기는 명령어 집합에 기초하여, 모델에 포함된 복수의 스테이지들에 동일하게 적용되는 공통 배치 크기에 따른 복수의 입력들을 복수의 스테이지들 각각에서 처리한다. 이때, 복수의 스테이지들 중에서 공통 배치 크기보다 작은 가속기의 온-칩 메모리 내에서 처리 가능한 최대 배치 크기를 가지는 스테이지가 반복 수행되어 공통 배치 크기에 따른 중간 특징 맵이 다음 스테이지로 전달된다.

[0099] 도 8에 도시된 각 단계들에는 도 1 내지 도 6을 통하여 전술한 사항들이 그대로 적용되므로, 보다 상세한 설명은 생략한다.

[0100] 실시예들은 뉴럴 프로세서를 이용하여 배치 입력을 효율적으로 최적화 및 처리할 수 있으며, 온-칩 메모리를 효율적으로 사용하여 오프-칩 메모리의 액세스를 최소화하고, 전체적인 레이턴시를 효과적으로 감소시킬 수 있다. 이처럼, 불필요한 오프-칩 메모리의 사용을 줄여 추론 시간을 효과적으로 단축시킬 수 있다.

[0102] 도 9 및 도 10은 일실시예에 따른 전자 장치의 예시들을 나타낸 도면이다.

[0103] 도 9를 참조하면, 일실시예에 따른 전자 장치는 서버(900)로 구현될 수 있다.

[0104] 서버(900)는 사용자에 의해 제어되는 사용자 단말과 구분되는 별도의 장치로서, 유선 및/또는 무선 네트워크를

통해 하나 이상의 사용자 단말과 통신을 수행할 수 있다. 서버(900)는 다수의 사용자들이 각자 자신의 단말을 통해 동시다발적으로 전송하는 추론 실행 요청들을 수신할 수 있다. 호스트 프로세서(910)는 가속기(920)에서 모델 실행 시 최적의 NIB에 기반한 배치 처리를 위한 명령어 집합을 생성할 수 있다. 가속기(920)는 호스트 프로세서(910)에서 생성된 명령어 집합에 기반한 배치 처리를 통해 다수의 입력들에 빠른 추론을 수행할 수 있다. 그리고, 서버(900)는 추론 결과들을 각각 대응하는 사용자 단말로 리턴할 수 있다. 예를 들어, 사용자 단말은 스마트폰, 태블릿, 랩탑, 퍼스널 컴퓨터 등 다양한 컴퓨팅 장치, 스마트 시계, 스마트 안경 등 다양한 웨어러블 기기, 스마트 스피커, 스마트 TV, 스마트 냉장고 등 다양한 가전장치, 스마트 자동차, 스마트 키오스크, IoT(Internet of Things) 기기, 드론, 로봇 등을 포함할 수 있다.

[0105] 도 10을 참조하면, 일실시예에 따른 전자 장치는 사용자 단말(1000)로 구현될 수 있다. 도 10에서는 설명의 편의를 위해 사용자 단말(1000)이 스마트 폰으로 도시되었지만, 이외에도 사용자에 의해 제어되는 기기라면 제한 없이 적용될 수 있다. 사용자 단말(1000)은 직접 사용자로부터 추론 실행 요청들을 획득하고, 앞선 설명처럼 호스트 프로세서(1010)에서 결정된 명령어 집합에 따른 배치 처리에 기반하여 가속기(1020)에서 다수의 입력들이 처리될 수 있다. 사용자 단말(1000)은 추론 결과들을 단순히 사용자로 제공하거나, 그에 기반한 후속 동작을 수행할 수 있다.

[0107] 이상에서 설명된 실시예들은 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치, 방법 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0108] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

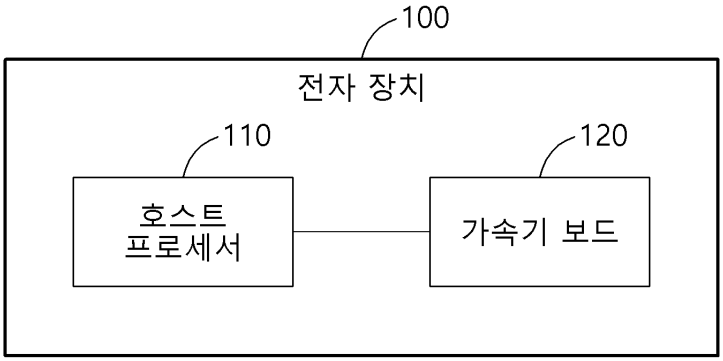
[0109] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0110] 이상과 같이 실시예들이 비록 한정된 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기를 기초로 다양한 기술적 수정 및 변형을 적용할 수 있다. 예를 들어, 설명된 기술들이 설명된 방법과 다

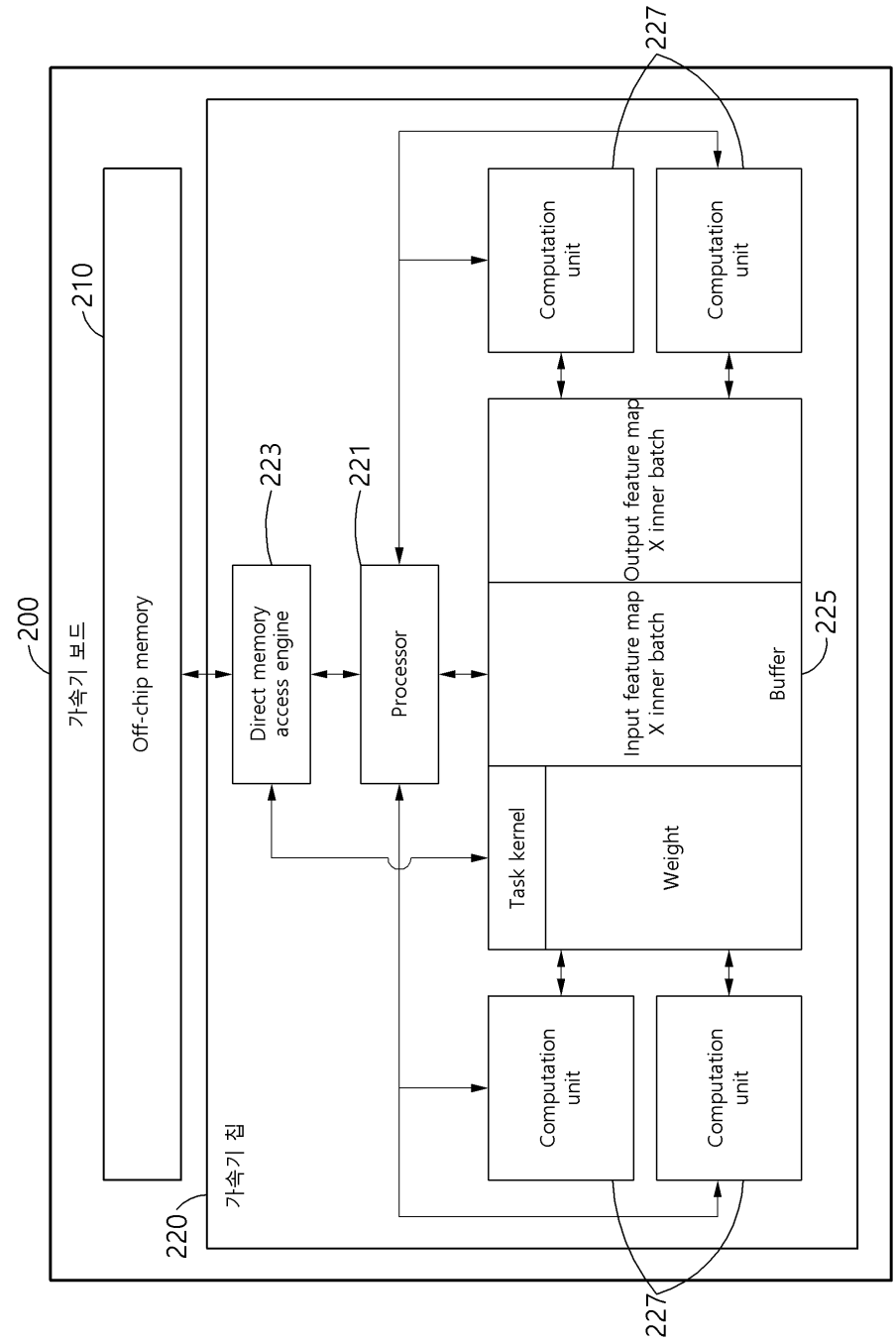
른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

도면

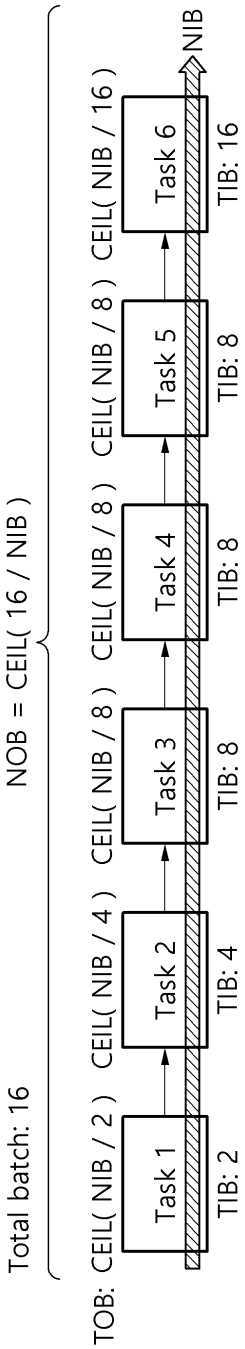
도면1



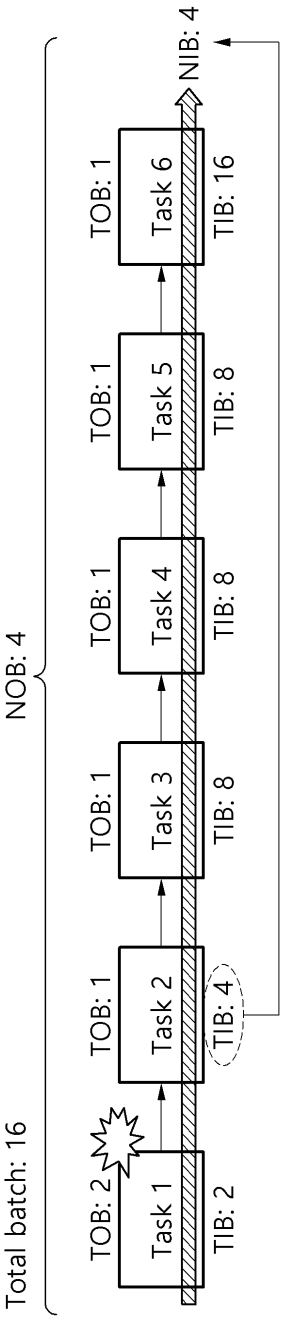
도면2


200
가속기 보드
210
Off-chip memory
220
가속기 칩
223
Direct memory access engine
221
Processor
227
Computation unit
Computation unit
Output feature map X inner batch
Input feature map X inner batch
Buffer
225
Task kernel
Weight
Computation unit
Computation unit
227

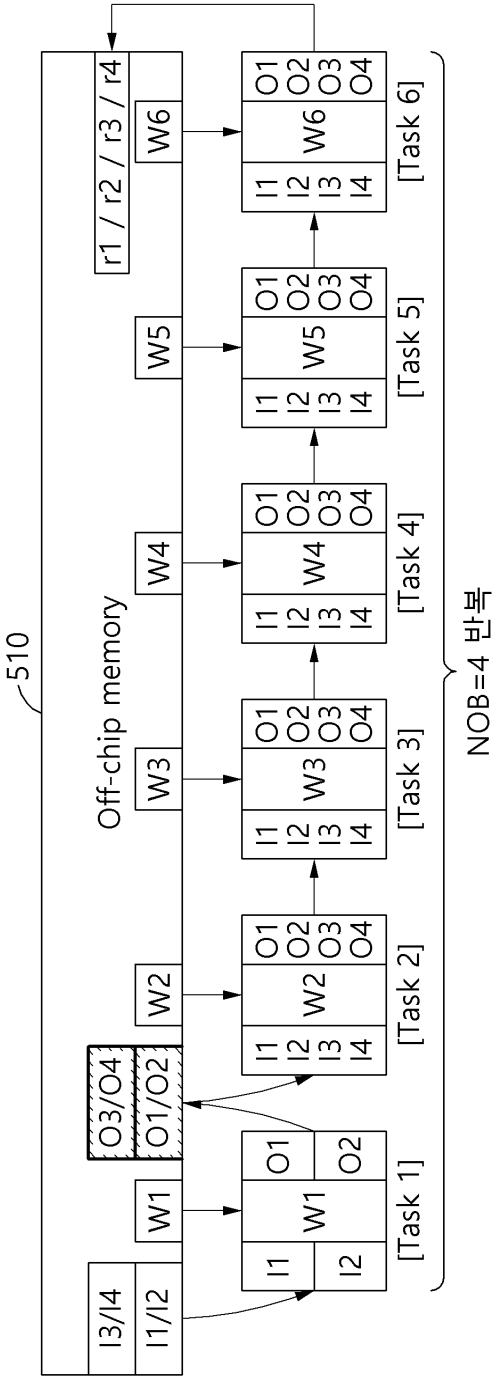
도면3


Total batch: 16
NOB = CEIL(16 / NIB)
TOB: CEIL(NIB / 2)
CEIL(NIB / 4)
CEIL(NIB / 8)
CEIL(NIB / 8)
CEIL(NIB / 8)
CEIL(NIB / 16)
Task 1
Task 2
Task 3
Task 4
Task 5
Task 6
NIB
TIB: 2
TIB: 4
TIB: 8
TIB: 8
TIB: 8
TIB: 16

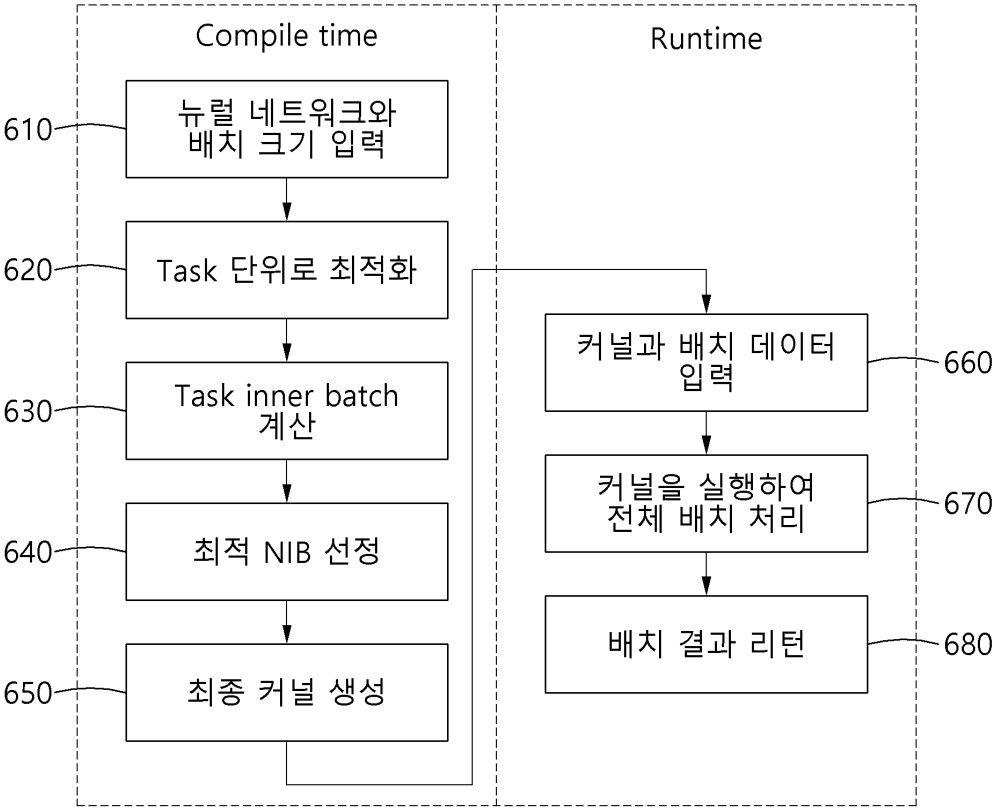
도면4


Total batch: 16
NOB: 4
TOB: 2
Task 1
TIB: 2
TOB: 1
Task 2
TIB: 4
TOB: 1
Task 3
TIB: 8
TOB: 1
Task 4
TIB: 8
TOB: 1
Task 5
TIB: 8
TOB: 1
Task 6
TIB: 16
NIB: 4

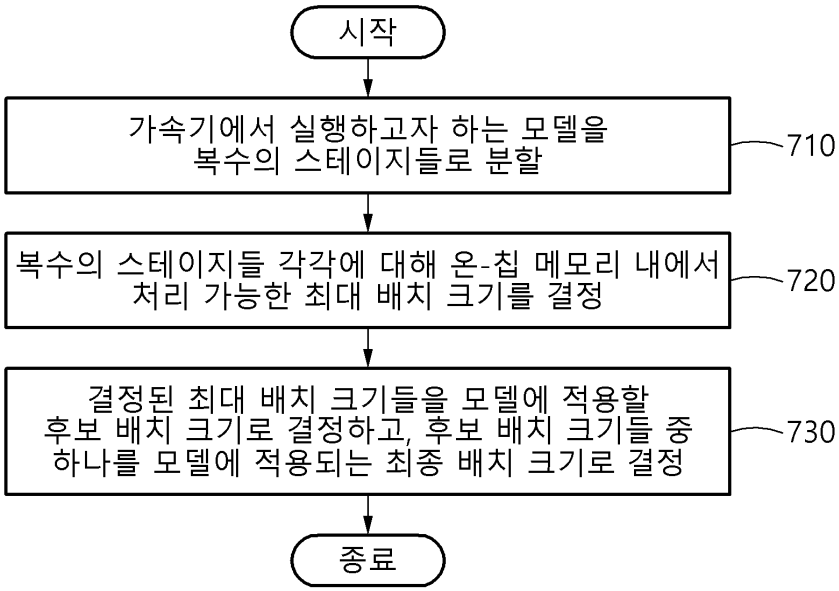
도면5


510
Off-chip memory
I3/I4
I1/I2
W1
O3/O4
O1/O2
W2
W3
W4
W5
r1 / r2 / r3 / r4
W6
I1
I2
W1
O1
O2
[Task 1]
I1
I2
I3
I4
W2
O1
O2
O3
O4
[Task 2]
I1
I2
I3
I4
W3
O1
O2
O3
O4
[Task 3]
I1
I2
I3
I4
W4
O1
O2
O3
O4
[Task 4]
I1
I2
I3
I4
W5
O1
O2
O3
O4
[Task 5]
I1
I2
I3
I4
W6
O1
O2
O3
O4
[Task 6]
NOB=4 반복

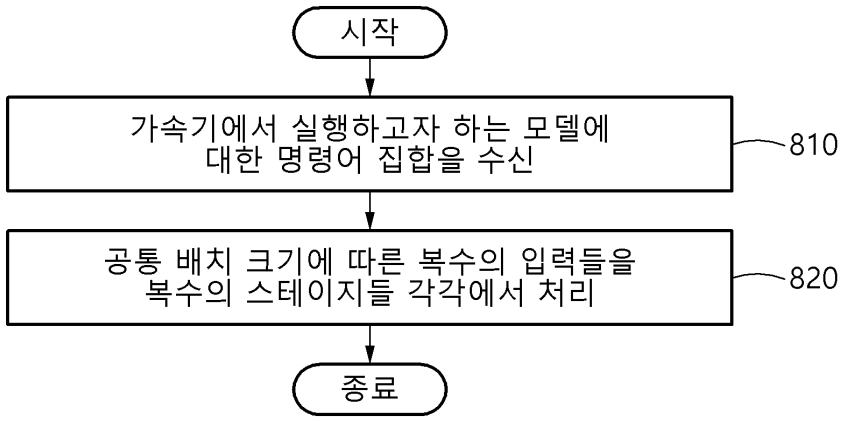
도면6


Compile time
Runtime
610
뉴럴 네트워크와 배치 크기 입력
620
Task 단위로 최적화
630
Task inner batch 계산
640
최적 NIB 선정
650
최종 커널 생성
660
커널과 배치 데이터 입력
670
커널을 실행하여 전체 배치 처리
680
배치 결과 리턴

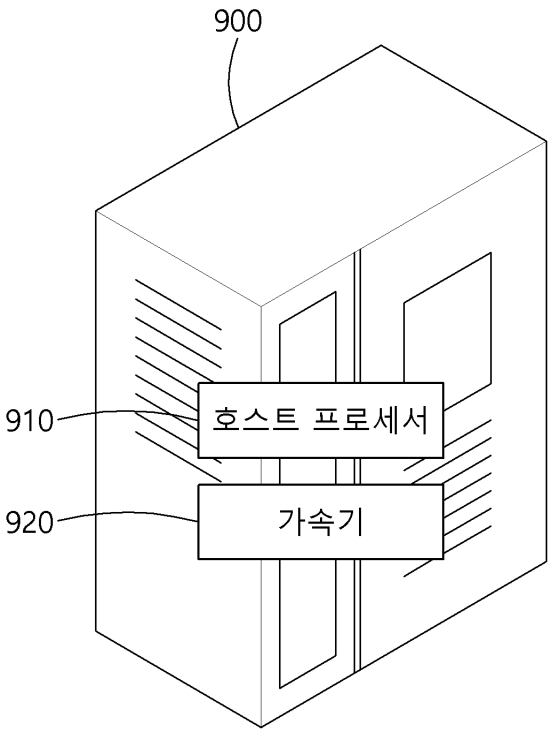
도면7


시작
가속기에서 실행하고자 하는 모델을 복수의 스테이지들로 분할
710
복수의 스테이지들 각각에 대해 온-칩 메모리 내에서 처리 가능한 최대 배치 크기를 결정
720
결정된 최대 배치 크기들을 모델에 적용할 후보 배치 크기로 결정하고, 후보 배치 크기들 중 하나를 모델에 적용되는 최종 배치 크기로 결정
730
종료

도면8


시작
가속기에서 실행하고자 하는 모델에
대한 명령어 집합을 수신
810
공통 배치 크기에 따른 복수의 입력들을
복수의 스테이지들 각각에서 처리
820
종료

도면9


900
910
호스트 프로세서
920
가속기

도면10