

(12) 특허협력조약에 의하여 공개된 국제출원

(19) 세계지식재산권기구
국제사무국

(43) 국제공개일

2020년 9월 17일 (17.09.2020)



WIPO PCT



(10) 국제공개번호

WO 2020/184985 A1

(51) 국제특허분류:

G06T 1/20 (2006.01)

G06N 20/00 (2019.01)

G06T 1/60 (2006.01)

(21) 국제출원번호:

PCT/KR2020/003399

(22) 국제출원일:

2020년 3월 11일 (11.03.2020)

(25) 출원언어:

한국어

(26) 공개언어:

한국어

(30) 우선권정보:

10-2019-0027685 2019년 3월 11일 (11.03.2019) KR

10-2020-0029251 2020년 3월 9일 (09.03.2020) KR

(71) 출원인: 서울대학교산학협력단 (SEOUL NATIONAL UNIVERSITY R&DB FOUNDATION) [KR/KR]; 08826

서울시 관악구 관악로 1, Seoul (KR).

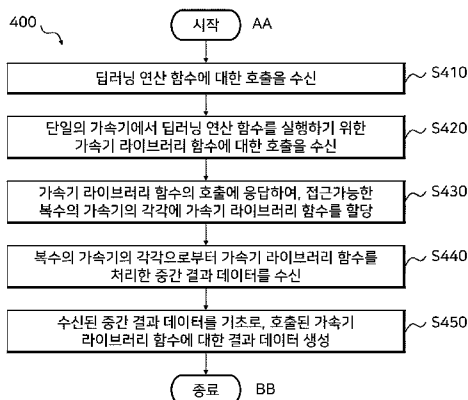
(72) 발명자: 이재진 (LEE, Jae Jin); 06610 서울시 서초구 서운로 200, 112동 604호, Seoul (KR). 박정호 (PARK, Jung Ho); 08725 서울시 관악구 성현로 80, 112동 1801호, Seoul (KR). 김형모 (KIM, Hyung Mo); 08758 서울시 관악구 봉천로 472, 1113호, Seoul (KR).

(74) 대리인: 안제성 등 (AHN, Je Sung et al.); 06253 서울특별시 강남구 도곡로 117, 옥신타워 14층, AnK 특허법률사무소, Seoul (KR).

(81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA,

(54) Title: METHOD AND COMPUTER PROGRAM FOR PROCESSING PROGRAM FOR SINGLE ACCELERATOR USING DNN FRAMEWORK IN PLURALITY OF ACCELERATORS

(54) 발명의 명칭: DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법 및 컴퓨터 프로그램



S410 ... Receive call for deep learning operation function

S420 ... Receive call for accelerator library function in order to execute deep learning operation function in single accelerator

S430 ... Assign accelerator library function to each of plurality of accessible accelerators in response to call for accelerator library function

S440 ... Receive, from each of plurality of accelerators, intermediate result data of processing accelerator library function

S450 ... Generate result data for called accelerator library function, on basis of received intermediate result data

AA ... Start

BB ... End

(57) Abstract: The present disclosure relates to a method for processing a program for a single accelerator using a DNN framework in a plurality of accelerators. The method for processing a program for a single accelerator in a plurality of accelerators comprises the steps of: receiving a call for a deep learning operation function; receiving a call for an accelerator library function in order to execute a deep learning operation function in a single accelerator; assigning an accelerator library function to each of the plurality of accessible accelerators in response to the call for the accelerator library function; receiving, from each of the plurality of accelerators, intermediate result data of processing the accelerator library function; and generating result data for a called accelerator library function, on the basis

[다음 쪽 계속]

PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

- (84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 유럽 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

공개:

— 국제조사보고서와 함께 (조약 제21조(3))

of the received intermediate result data.

(57) 요약서: 본 개시는 DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법에 관한 것이다. 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법은, 딥러닝 연산 함수에 대한 호출을 수신하는 단계, 단일의 가속기에서 딥러닝 연산 함수를 실행하기 위한 가속기 라이브러리 함수에 대한 호출을 수신하는 단계, 가속기 라이브러리 함수의 호출에 응답하여, 접근 가능한 복수의 가속기의 각각에 가속기 라이브러리 함수를 할당하는 단계, 복수의 가속기의 각각으로부터 가속기 라이브러리 함수를 처리한 중간 결과 데이터를 수신하는 단계 및 수신된 중간 결과 데이터를 기초로, 호출된 가속기 라이브러리 함수에 대한 결과 데이터를 생성하는 단계를 포함한다.

명세서

발명의 명칭: DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법 및 컴퓨터 프로그램 기술분야

- [1] 본 개시는 DNN(Deep Neural Network) 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법 및 컴퓨터 프로그램에 관한 것으로, 구체적으로, 단일의 가속기에서 딥러닝 연산 함수를 실행하기 위한 가속기 라이브러리 함수에 대한 호출을 수신하면, 접근 가능한 복수의 가속기의 각각에 가속기 라이브러리 함수 및 이러한 함수와 연관된 데이터를 할당하는 방법 및 컴퓨터 프로그램에 관한 것이다.

[2]

배경기술

- [3] 최근에 널리 사용되는 DNN 프레임워크인 Tensorflow, Pytorch 등은 단일 가속기에서 실행되는 고성능 가속기 연산 라이브러리인 cuDNN, cuBLAS 등을 바탕으로 하는 연산을, 함수의 형태로 사용자에게 제공할 수 있다. 이러한 종래의 DNN 프레임워크 하에서, 복수의 가속기를 활용하기 위하여, 사용자는 하나의 컴퓨터에 장치된 가속기 디바이스를 각각 명시하거나, 네트워크로 연결된 클러스터에서 각각의 컴퓨터를 명시하여 각 컴퓨터에서의 작업을 별도로 명시해 주어야 한다. 이러한 요구 사항은 DNN 프레임워크가 기반을 두고 있는 라이브러리가 하나의 가속기 대상이므로, DNN 프레임워크 수준 또는 사용자의 응용 프로그램 수준에서 작업 분배를 반드시 수행해 주어야 하기 때문에 발생할 수 있다. 또한, 클러스터 시스템의 구성에 따라, 사용자가 프로그램을 다시 작성해야 하고, 실행 환경에 따라 코드 수정이 요구되거나 성능이 저하되거나, 오류가 발생할 수도 있었다.
- [4] 일반적으로 가속기 간 및 서버 간 메모리 통신은 DNN 프레임워크에서 내부적으로 처리하므로 사용자가 신경 쓸 필요가 없다. 하지만 복수의 가속기 또는 각각의 컴퓨터간 딥 뉴럴 네트워크의 공유 여부, 가속기 별 작업 지정, 클러스터 구성 설정은 사용자(프로그래머)가 직접 응용 프로그램 수준에서 처리할 수 있도록 그와 관련된 함수를 제공하는데 그친다. 즉, DNN 프레임워크 내부적으로 단일 컴퓨터 혹은 복수의 컴퓨터로 구성된 클러스터에 장착된 복수의 가속기를 사용하기 위해 메모리 통신 등 기본적인 기능은 제공하나, 그 기능을 활용하여 복수의 가속기로 작업을 분배하는 등의 추가적인 일은 모두 사용자가 수행해야 한다.
- [5] 딥 러닝 분야가 빠르게 발전하고 변화함에 따라 구현의 대상이 되는 네트워크 모델(network model)이나 DNN 프레임워크가 빠르게 변경되고, 기술의 발전에 따라 딥 러닝을 수행하는 클러스터의 구성이나 규모 등 실행 환경이 바뀌기

때문에 코드 수정 및 추가가 불가피할 수 있다. 이에 따라, 사용자 입장에서는 실행 환경에 따라 코드를 직접 수정하거나, 옵션에 따라 상이하게 동작하도록 코드를 작성해 두어야 한다. 이 과정에 굉장히 많은 시간이 소요되며, 코드의 구조가 굉장히 복잡해질 수 있고, 이에 따라 유지 보수 및 변경이 어려울 수 있다.

[6]

발명의 상세한 설명

기술적 과제

[7] 본 개시는 상기와 같은 문제점을 해결하기 위한 DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법, 기록매체에 저장된 컴퓨터 프로그램 및 시스템을 제공한다.

[8] 본 발명의 목적은 프로그래머가 DNN 프레임워크를 이용하여, 하나의 가속기 대상의 딥러닝 학습/추론 프로그램을 작성하면, 이를 별도의 소스 코드(예를 들어, 사용자 프로그램 및 DNN 프레임워크 모두)의 수정 없이, 복수의 가속기가 설치된 컴퓨터 또는 클러스터 시스템에서 프로그램이 동작하도록 하는 방법 및 컴퓨터 프로그램이 제공된다.

[9] 본 발명은 소스 코드가 공개되지 않은 고성능 가속기 연산 라이브러리(예를 들어, cuDNN, cuBLAS 등)가 단일 가속기 대신에 하나의 컴퓨터 또는 클러스터 시스템에 포함된 다수의 가속기를 이용해 동작하도록 하는 방법 및 컴퓨터 프로그램이 제공된다.

[10]

과제 해결 수단

[11] 본 개시는 방법, 장치, 시스템, 컴퓨터 프로그램 또는 명령어들을 저장하는 컴퓨터 판독가능 저장 매체를 포함한 다양한 방식으로 구현될 수 있다.

[12] 본 개시의 일 실시예에 따른 DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법은, 딥러닝 연산 함수에 대한 호출을 수신하는 단계, 단일의 가속기에서 딥러닝 연산 함수를 실행하기 위한 가속기 라이브러리 함수에 대한 호출을 수신하는 단계, 가속기 라이브러리 함수의 호출에 응답하여, 접근 가능한 복수의 가속기의 각각에 가속기 라이브러리 함수를 할당하는 단계, 복수의 가속기의 각각으로부터 가속기 라이브러리 함수를 처리한 중간 결과 데이터를 수신하는 단계 및 수신된 중간 결과 데이터를 기초로, 호출된 가속기 라이브러리 함수에 대한 결과 데이터를 생성하는 단계를 포함한다.

[13] 일 실시예에 따르면, 접근 가능한 복수의 가속기의 각각에 가속기 라이브러리 함수를 할당하는 단계는, 가속기 라이브러리 함수의 입력 데이터를 복수의 부분 입력 데이터 세트로 분할하는 단계 및 가속기 라이브러리 함수 및 분할된 복수의 부분 입력 데이터 세트의 각각을 복수의 가속기의 각각에 할당하는 단계를 포함하고, 복수의 가속기의 각각으로부터 가속기 라이브러리 함수를 처리한

중간 결과 데이터를 수신하는 단계는, 복수의 가속기의 각각에서 가속기 라이브러리 함수를 이용하여 복수의 부분 입력 데이터 세트의 각각을 처리한 중간 결과 데이터를 수신하는 단계를 포함한다.

- [14] 일 실시예에 따르면, 가속기 라이브러리 함수 및 분할된 복수의 부분 입력 데이터 세트의 각각을 복수의 가속기의 각각에 할당하는 단계는, 가속기 라이브러리 함수의 메모리 접근 패턴을 분석하는 단계 및 분석된 접근 패턴에 기초하여 가속기 라이브러리 함수의 실행 전에 가속기 라이브러리 함수의 입력 데이터를 접근 가능한 복수의 가속기의 각각에 할당하는 단계를 포함한다.
- [15] 일 실시예에 따르면, 가속기 라이브러리 함수 및 분할된 복수의 부분 입력 데이터 세트의 각각을 복수의 가속기의 각각에 할당하는 단계는, 복수의 가속기의 각각에 할당되는 부분 입력 데이터 세트가 n 개의 부분 입력 데이터 세트를 포함하는 경우(여기서, n 은 2이상의 자연수임), n 개의 부분 입력 데이터 세트 중에서, m 번째 부분 입력 데이터 세트를 복수의 가속기의 각각에서 처리하는 동시에 $m+1$ 번째 부분 입력 데이터 세트가 복수의 가속기의 각각에 할당하는 단계(여기서, m 은 n 보다 작은 자연수임)를 포함한다.
- [16] 일 실시예에 따르면, 접근 가능한 복수의 가속기의 각각에 가속기 라이브러리 함수를 할당하는 단계는, 가속기 라이브러리 함수의 파라미터 데이터를 복수의 부분 파라미터 데이터 세트로 분할하는 단계 및 가속기 라이브러리 함수 및 분할된 복수의 부분 파라미터 데이터 세트의 각각을 복수의 가속기의 각각에 할당하는 단계를 포함한다.
- [17] 일 실시예에 따르면, 복수의 가속기의 각각으로부터 가속기 라이브러리 함수를 처리한 중간 결과 데이터를 수신하는 단계는 복수의 가속기의 각각으로부터 가속기 라이브러리 함수의 파라미터 데이터를 처리한 중간 결과 데이터를 수신하는 단계를 포함하고, 호출된 가속기 라이브러리 함수에 대한 결과 데이터를 생성하는 단계는, 가속기 라이브러리 함수의 파라미터 데이터를 처리한 중간 결과 데이터에 대한 결과 데이터를 생성하는 단계를 포함한다.
- [18] 일 실시예에 따르면, 수신된 중간 결과 데이터를 기초로, 호출된 가속기 라이브러리 함수에 대한 결과 데이터를 생성하는 단계는 수신된 중간 결과 데이터를 연결하여(concatenate) 결과 데이터를 생성하는 단계를 포함한다.
- [19] 일 실시예에 따르면, 수신된 중간 결과 데이터를 기초로, 호출된 가속기 라이브러리 함수에 대한 결과 데이터를 생성하는 단계는 수신된 중간 결과 데이터를 연산하여 결과 데이터를 생성하는 단계를 포함한다.
- [20] 일 실시예에 따르면, 복수의 가속기는 하나의 컴퓨팅 장치에 포함된다.
- [21] 일 실시예에 따르면, 복수의 가속기는 복수의 컴퓨팅 장치를 포함하는 클러스터 시스템에 포함된다.
- [22] 본 개시의 일 실시예에 따른 상술한 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법을 컴퓨터에서 실행하기 위해 컴퓨터 판독 가능한 기록 매체에 저장된 컴퓨터 프로그램이 제공된다.

[23]

발명의 효과

[24]

본 개시의 일부 실시예에 따르면, 주로 사용되는 DNN 프레임워크가 공통적으로 이용되고, 소스 코드가 공개되어 있지 않은 고성능 가속기 라이브러리는 하나의 가속기 대신에 하나의 컴퓨터 또는 클러스터에 포함된 복수의 가속기에서 동작될 수 있다.

[25]

본 개시의 일부 실시예에 따르면, 라이브러리 수준에서 자동으로 복수의 가속기를 활용하도록 하므로, DNN 프레임워크를 이용하는 사용자 응용 프로그램 수준에서는 단일 가속기를 대상으로 작성한 하나의 프로그램이, 실행 환경에서 적절하게 분산 처리될 수 있다. 또한, 복수의 가속기가 이용되어 연산이 수행되므로 실행 시간이 단축될 수 있다. 그리고, 사용자 응용 프로그램의 소스 코드가 변경될 필요가 없으므로, 이를 통해 프로그램 개발 과정이 크게 단축되어 프로그램의 생산성이 높아지고 유지 보수가 더욱 간편해질 수 있다.

[26]

본 개시의 일부 실시예에 따르면, DNN 프레임워크가 공통적으로 이용하는 라이브러리의 동작이 변경되므로 DNN 프레임워크의 소스 코드는 수정할 필요가 없다. 또한, DNN 프레임워크의 컴파일 과정에서 기존에 링크(link)하던 라이브러리를 바꿔치기하거나 가로채기하는 방식이 적용될 수 있다. 이러한 방식은 여러 종류의 DNN 프레임워크에 쉽게 적용될 수 있다. 새로운 DNN 프레임워크를 개발하는 경우에도, 하나의 가속기와 복수의 가속기의 처리가 일원화될 수 있다.

[27]

도면의 간단한 설명

[28]

본 개시의 실시예들은, 이하 설명하는 첨부 도면들을 참조하여 설명될 것이며, 여기서 유사한 참조 번호는 유사한 요소들을 나타내지만, 이에 한정되지는 않는다.

[29]

도 1은 본 개시의 일 실시예에 따른 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법을 제공하기 위하여, 호스트 컴퓨팅 장치가 복수의 컴퓨팅 장치와 통신 가능하도록 연결된 구성을 나타내는 개요도이다.

[30]

도 2는 본 개시의 일 실시예에 따른 컴퓨팅 장치의 내부 구성을 나타내는 블록도이다.

[31]

도 3은 본 개시의 일 실시예에 따른 DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법을 제공하기 위한, 프로세서의 내부 구성을 나타내는 블록도이다.

[32]

도 4는 본 개시의 일 실시예에 따른 DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법을 나타내는 흐름도이다.

- [33] 도 5는 본 개시의 일 실시예에 따른 입력 데이터를 복수의 부분 입력 데이터 세트로 분할하고, 분할된 복수의 부분 입력 데이터 세트의 각각을 복수의 가속기에 할당하는 예시를 나타내는 도면이다.
- [34] 도 6은 본 개시의 일 실시예에 따른 복수의 가속기에 입력 데이터를 복수의 부분 입력 데이터 세트로 분할하지 못할 경우, 데이터를 할당하는 예시를 나타내는 도면이다.
- [35] 도 7은 본 개시의 일 실시예에 따른 복수의 가속기로부터 수신한 중간 결과 데이터를 연결하여, 결과 데이터를 생성하는 예시를 나타내는 도면이다.
- [36] 도 8은 본 개시의 일 실시예에 따른 복수의 가속기로부터 수신한 중간 결과 데이터를 연산하여, 결과 데이터를 생성하는 예시를 나타내는 도면이다.
- [37] 도 9는 본 개시의 일 실시예에 따른 복수의 가속기에서 연산을 수행하는 동시에 복수의 부분 입력 데이터 세트의 복사를 수행하는 예시를 나타내는 도면이다.

[38]

발명의 실시를 위한 형태

- [39] 이하, 본 개시의 실시를 위한 구체적인 내용을 첨부된 도면을 참조하여 상세히 설명한다. 다만, 이하의 설명에서는 본 개시의 요지를 불필요하게 흐릴 우려가 있는 경우, 널리 알려진 기능이나 구성에 관한 구체적 설명은 생략하기로 한다.
- [40] 첨부된 도면에서, 동일하거나 대응하는 구성요소에는 동일한 참조부호가 부여되어 있다. 또한, 이하의 실시예들의 설명에 있어서, 동일하거나 대응되는 구성요소를 중복하여 기술하는 것이 생략될 수 있다. 그러나 구성요소에 관한 기술이 생략되어도, 그러한 구성요소가 어떤 실시예에 포함되지 않는 것으로 의도되지는 않는다.
- [41] 개시된 실시예의 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 후술되어 있는 실시예들을 참조하면 명확해질 것이다. 그러나 본 개시는 이하에서 개시되는 실시예들에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있으며, 단지 본 실시예들은 본 개시가 완전하도록 하고, 본 개시가 통상의 기술자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것일 뿐이다.
- [42] 본 명세서에서 사용되는 용어에 대해 간략히 설명하고, 개시된 실시예에 대해 구체적으로 설명하기로 한다. 본 명세서에서 사용되는 용어는 본 개시에서의 기능을 고려하면서 가능한 현재 널리 사용되는 일반적인 용어들을 선택하였으나, 이는 관련 분야에 종사하는 기술자의 의도 또는 관례, 새로운 기술의 출현 등에 따라 달라질 수 있다. 또한, 특정한 경우는 출원인이 임의로 선정한 용어도 있으며, 이 경우 해당되는 발명의 설명 부분에서 상세히 그 의미를 기재할 것이다. 따라서 본 개시에서 사용되는 용어는 단순한 용어의 명칭이 아닌, 그 용어가 가지는 의미와 본 개시의 전반에 걸친 내용을 토대로 정의되어야 한다.

- [43] 본 명세서에서의 단수의 표현은 문맥상 명백하게 단수인 것으로 특정하지 않는 한, 복수의 표현을 포함한다. 또한, 복수의 표현은 문맥상 명백하게 복수인 것으로 특정하지 않는 한, 단수의 표현을 포함한다. 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 있음을 의미한다.
- [44] 본 개시에서, '클러스터 시스템'은 네트워크를 통해 연결된 복수의 컴퓨터를 포함할 수 있다. 이러한 클라이언트 시스템 하에서, 한 클라이언트 장치는 클러스터 시스템을 하나의 컴퓨터처럼 이용할 수 있다. 이러한 클러스터 시스템은 연결된 사용자에게 복수의 컴퓨터를 하나의 컴퓨터처럼 이용할 수 있도록 제공하기 때문에, 하나의 컴퓨터에서의 처리 속도보다 훨씬 향상된 처리 속도가 구현될 수 있다.
- [45] 본 개시에서, '배치(Batch)'는 딥러닝에서 모델의 네트워크 파라미터를 업데이트 하기 전 샘플의 개수를 의미한다. 딥러닝 모델의 학습 데이터 세트는 하나 이상의 배치로 나뉘질 수 있다. 예를 들어, 이미지를 처리하는 딥러닝 모델에서 각각의 배치 크기는 한 번에 처리하는 이미지의 수를 의미한다. 배치 크기가 64이면, 이미지 파일 64개를 연속된 메모리 공간에 차례로 뭉칠 수 있다. 이에 따라, 배치 단위로 분할되는 데이터는 구조적으로 동일하며, 배치 단위로 분할되는 데이터는 복수의 가속기에 별다른 처리 없이 분배할 수 있다.
- [46] 본 개시에서, '라이브러리 래퍼(library wrapper)'는 소프트웨어를 개발하는데 사용되는 서브루틴 또는 클래스들의 모음으로, 라이브러리 래퍼는 라이브러리의 현재 인터페이스를 호환되는 인터페이스로 변환하는 코드의 얇은 층으로 구성될 수 있다. 예를 들어, 라이브러리 래퍼는 DNN 프레임워크가 공통적으로 이용하는 라이브러리의 동작을 변경하도록 구성된 함수를 포함할 수 있으므로, DNN 프레임워크의 소스 코드 자체를 수정할 필요가 없다. 이에 따라, 본 개시의 라이브러리 래퍼는 다양한 DNN 프레임워크에 유사한 방식으로 적용시킬 수 있음은 통상의 기술자에게 당연하다.
- [47] 도 1은 본 개시의 일 실시예에 따른 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법을 제공하기 위하여, 호스트 컴퓨팅 장치(120)가 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)와 통신 가능하도록 연결된 구성을 나타내는 개요도이다. 여기서, 복수의 가속기는 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)를 포함하는 클러스터 시스템(100)에 포함될 수 있다. 예를 들어, 복수의 가속기는 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)의 적어도 일부 컴퓨팅 장치에 포함될 수 있다. 또한, 클러스터 시스템(100)은 네트워크(110), 호스트 컴퓨팅 장치(120) 및 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)를 포함할 수 있다.
- [48] 네트워크(110)는, 클러스터 시스템(100)에 포함됨 호스트 컴퓨팅 장치(120) 및 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n) 사이의 통신이 가능하도록 구성될

수 있다. 네트워크(110)는 설치 환경에 따라, 예를 들어, 이더넷(Ethernet), 유선 홈 네트워크(Power Line Communication), 전화선 통신 장치 및 RS-serial 통신 등의 유선 네트워크, 이동통신망, WLAN(Wireless LAN), Wi-Fi, Bluetooth 및 ZigBee 등과 같은 무선 네트워크 또는 그 조합으로 구성될 수 있다. 다시 말해, 통신 방식은 제한되지 않으며, 네트워크(110)가 포함할 수 있는 통신망(일례로, 이동통신망, 유선 인터넷, 무선 인터넷, 방송망, 위성망 등)을 활용하는 통신 방식뿐만 아니라 호스트 컴퓨팅 장치(120) 및 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n) 사이의 근거리 무선 통신 역시 포함될 수 있다. 예를 들어, 네트워크(110)는 PAN(personal area network), LAN(local area network), CAN(campus area network), MAN(metropolitan area network), WAN(wide area network), BBN(broadband network), 인터넷 등의 네트워크 중 하나 이상의 임의의 네트워크를 포함할 수 있다. 또한, 네트워크(110)는 버스 네트워크, 스타 네트워크, 링 네트워크, 메쉬 네트워크, 스타-버스 네트워크, 트리 또는 계층적(hierarchical) 네트워크 등을 포함하는 네트워크 토폴로지 중 임의의 하나 이상을 포함할 수 있으나, 이에 제한되지 않는다.

- [49] 호스트 컴퓨팅 장치(120)는 DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 실행시키도록 구성될 수 있다. 또한, 호스트 컴퓨팅 장치(120)는 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)와 통신 가능하도록 구성되고, 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)에 포함된 복수의 가속기의 동작을 제어하도록 구성될 수 있다. 예를 들어, 호스트 컴퓨팅 장치(120)는 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)에게 가속기 라이브러리 함수, 입력 데이터를 할당하고, 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n) 각각으로부터 가속기 라이브러리 함수를 처리한 중간 결과 데이터를 수신하고, 수신된 중간 결과 데이터를 기초로, 호출된 가속기 라이브러리 함수에 대한 결과 데이터를 생성할 수 있다.
- [50] 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)는 클러스터 시스템(100) 상에서 정보처리 및 통신을 수행하는 컴퓨팅 장치이다. 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)는 컴퓨터 또는 원격처리 장치와 같은 단말의 형태로 구성될 수 있다. 또한, 각 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)는 독립적으로 정보처리 등을 수행할 수 있으나, 병렬 프로그래밍을 통하여 다른 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)들과 협력하면서 정보처리 등을 수행할 수도 있다. 각 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)는 네트워크(110)를 통해 딥러닝 어플리케이션의 동작을 위한 통신을 실행할 수 있다. 이러한 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n)는 데이터의 송신원, 수신처 또는 중계점 중 어느 하나에 해당할 수 있다.
- [51] 도 2는 본 개시의 일 실시예에 따른 컴퓨팅 장치(200)의 내부 구성을 나타내는 블록도이다. 복수의 가속기(230_1, 230_2, ..., 230_n)는 컴퓨팅 장치(200)에 포함될 수 있다. 일 실시예에 따르면, 컴퓨팅 장치(200)는 호스트 컴퓨팅

장치(120)를 지칭할 수 있다. 다른 실시예에서, 컴퓨팅 장치(200)는 복수의 컴퓨팅 장치(130_1, 130_2, ..., 130_n) 각각을 지칭할 수 있다. 도 2에 도시된 바와 같이, 하나의 컴퓨팅 장치(200)는 프로세서(210), 메인 메모리(220) 및 복수의 가속기(230_1, 230_2, ..., 230_n)를 포함할 수 있다.

[52] 프로세서(210)는, 예를 들어, CPU(Central Processing Unit, 중앙 처리 장치)와 같은 연산 처리를 위한 범용 프로세서로 구성될 수 있으며, 프로세서(210)는 복수의 가속기(230_1, 230_2, ..., 230_n)와 연결되어 복수의 가속기(230_1, 230_2, ..., 230_n)의 동작을 제어할 수 있다. 또한, 프로세서(210)는 메인 메모리(220)가 연결될 수 있다. 예를 들어, 프로세서(210)는 PCI-E(Peripheral component interconnect-Express) 버스를 통해 복수의 가속기(230_1, 230_2, ..., 230_n) 및/또는 메인 메모리(220)와 서로 연결될 수 있으며 복수의 가속기(230_1, 230_2, ..., 230_n) 및/또는 메인 메모리(220)의 제어를 위한 데이터를 송수신할 수 있다.

[53] 일 실시예에 따르면, 메인 메모리(220)는 전자 정보를 저장 가능한 임의의 전자 컴포넌트를 포함하도록 넓게 해석되어야 한다. 예를 들어, 메인 메모리(220)는 임의의 액세스 메모리(RAM), 판독-전용 메모리(ROM), 비-휘발성 임의의 액세스 메모리(NVRAM), 프로그램가능 판독-전용 메모리(PROM), 소거-프로그램가능 판독 전용 메모리(EPROM), 전기적으로 소거가능 PROM(EEPROM), 플래쉬 메모리, 자기 또는 광학 데이터 저장장치, 레지스터들 등과 같은 프로세서-판독가능 매체의 다양한 유형들을 지칭할 수도 있다. 프로세서(210) 및/또는 복수의 가속기(230_1, 230_2, ..., 230_n)로부터 정보를 판독하거나 메모리에 정보를 기록할 수 있다면 메인 메모리(220)는 프로세서(210) 및/또는 복수의 가속기(230_1, 230_2, ..., 230_n)와 전자 통신 상태에 있다고 불린다. 본 개시에서, 메인 메모리(220)는 프로세서(210) 및/또는 복수의 가속기(230_1, 230_2, ..., 230_n)에 의해 실행되는 프로그램(예를 들어, 단일 가속기용 프로그램, DNN 프레임워크, 가속기 라이브러리 래퍼 및 가속기 라이브러리 함수 등)과 연관된 임의의 데이터 및/또는 정보(예: 프로그램 실행 데이터, 입력 데이터, 파라미터 데이터, 결과 데이터 등)를 저장할 수 있다.

[54] 복수의 가속기(230_1, 230_2, ..., 230_n)는 범용의 CPU와는 달리 특정 패턴의 연산에 특화된 프로세서로 구성될 수 있다. 예를 들어, 복수의 가속기(230_1, 230_2, ..., 230_n)의 각각은 GPU, FPGA, DSP, Intel Xeon Phi, TPU, NPU, 멀티코어 CPU 등을 포함할 수 있다. 또한, 복수의 가속기(230_1, 230_2, ..., 230_n) 각각에는 메인 메모리와는 별도로 가속기 메모리(미도시)가 연결되거나 포함될 수 있다.

[55] 도 3은 본 개시의 일 실시예에 따른 DNN 프레임워크(320)를 이용하는 단일 가속기용 프로그램(310)을 복수의 가속기에서 처리하는 방법을 제공하기 위한, 프로세서(300)의 내부 구성을 나타내는 블록도이다. 일 실시예에 따르면, 프로세서(300)는 도 2의 프로세서(210)를 지칭할 수 있다. 도시된 바와 같이, 단일 가속기용 프로그램(310), DNN 프레임워크(320), 가속기 라이브러리 래퍼(330) 및 가속기 라이브러리(340)는 프로세서(300)에 의해 또는 프로세서(300) 상에서

동작되거나 처리될 수 있다.

- [56] 단일 가속기용 프로그램(310)은, 가속기의 연산 자원을 활용하기 위해 다양한 플랫폼의 가속기(230)를 위한 병렬 프로그래밍 모델을 의미할 수 있다. 일 실시예에 따르면, 단일 가속기용 프로그램(310)은 단일 GPU 대상 사용자 응용 프로그램을 지칭할 수 있다. 예를 들어, 단일 가속기용 프로그램(310)은 임의의 가속기에 대해 딥러닝 연산 함수를 지원하는 임의의 프로그램을 포함할 수 있으며, 예를 들어, OpenCL(Open Computing Language) 및 CUDA(Compute Unified Device Architecture)와 같은 병렬 프로그래밍 모델 또는 프로그램을 포함할 수 있으나, 이에 제한되지 않는다. 또한, 딥러닝 연산 함수는 단일 가속기용 프로그램(310)을 통해 호출될 수 있다.
- [57] DNN 프레임워크(320)는, 딥러닝 애플리케이션 작성 및 실행을 용이하게 하는 만들어진 소프트웨어 집합체를 포함할 수 있다. 이러한 DNN 프레임워크(320)는 개발자로 하여금 높은 숙련도가 요구되는 병렬 프로그래밍 모델 또는 프로그램을 보다 손쉽게 이용할 수 있도록, 딥러닝 처리 또는 딥러닝 연산 함수를 가속기에 적용하여 학습(training) 과정과 추론(inference) 과정을 가속할 수 있다. 예를 들어, DNN 프레임워크(320)는 최근에 널리 사용되고 있는 Caffe, Tensorflow, Pytorch, CNTK 및 Theano 등의 DNN 프레임워크를 포함할 수 있으나, 이에 제한되지 않는다.
- [58] 가속기 라이브러리(340)는, DNN 프레임워크(320)를 통해 호출될 수 있는 고성능 라이브러리(library)를 지칭할 수 있다. 가속기 라이브러리(340)는 사용자 또는 개발자에게 복수의 가속기 라이브러리 함수를 제공할 수 있다. 이러한 가속기 라이브러리 함수는 단일 가속기에서 실행되도록 구성될 수 있다.
- [59] 일 실시예에 따르면, 가속기 라이브러리(340)는 cuDNN 및 cuBLAS 등을 지칭할 수 있으나, 이에 한정되지 않는다. 일 실시예에 따르면, cuDNN은 convolutional layer, pooling layer 등의 forward propagation과 backward propagation 등을 API 형태로 제공할 수 있다. 예를 들어, cuDNN의 cudnnConvolutionForward() 함수는 네트워크의 파라미터 중 해당 레이어에 대응하는 가중치(weight), 그리고 이전 레이어의 피쳐 맵(feature map)(또는 이미지 데이터)을 배치(batch) 단위로 입력 받아 다음 레이어로의 출력 피쳐 맵(output feature map)을 배치 단위로 출력할 수 있다.
- [60] 가속기 라이브러리 래퍼(330)는, 호출된 가속기 라이브러리 함수를 가로채고, 가속기 라이브러리 함수를 래핑(wrapping)하여 복수의 가속기(230_1, 230_2, ..., 230_n)에 의해 병렬 처리 가능하도록 구성될 수 있다. 이러한 구조 하에서, 단일 가속기용 프로그램(310) 및 DNN 프레임워크(320) 수준에서 단일 가속기(230)를 대상으로 작성된 코드가 라이브러리 수준에서는 복수의 가속기(230_1, 230_2, ..., 230_n)에서 실행될 수 있다. 예를 들어, 가속기 라이브러리 래퍼(330)는 GPU 라이브러리 래퍼를 지칭할 수 있다. 일 실시예에서, 가속기 라이브러리 래퍼(330)를 사용하여 단일 가속기용 프로그램(310)이 가속기 라이브러리

래퍼(330)를 이용하여 처리될 때, DNN 프레임워크(320)의 빌드 과정의 수정을 통해 가속기 라이브러리 함수 및 소스 코드(예를 들어, 사용자 프로그램 및 DNN 프레임워크(320))의 수정 없이, 단일 가속기용 프로그램(310)을 복수의 가속기(230_1, 230_2, ..., 230_n)에서 병렬 처리할 수 있다. 예를 들어, 프로세서(210)는 복수의 가속기(230_1, 230_2, ..., 230_n) 각각에 가속기 라이브러리 함수를 할당할 수 있다.

- [61] 도 4는 본 개시의 일 실시예에 따른 DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법(400)을 나타내는 흐름도이다. 이러한 방법(400)은 프로세서에 의해 수행될 수 있다. 도시된 바와 같이, 방법(400)은 딥러닝 연산 함수에 대한 호출을 수신하는 단계(S410)로 개시될 수 있다. 예를 들어, 프로세서 상에서 동작되는 DNN 프레임워크는 단일 가속기용 프로그램으로부터 딥러닝 연산 함수에 대한 호출을 수신할 수 있다.
- [62] 그리고 나서, 단계(S420)에서, 단일의 가속기에서 딥러닝 연산 함수를 실행하기 위한 가속기 라이브러리 함수에 대한 호출을 수신할 수 있다. 예를 들어, 프로세서 상에서 동작하는 가속기 라이브러리 래퍼는 DNN 프레임워크로부터 단일의 가속기에서 딥러닝 연산 함수를 실행하기 위한 가속기 라이브러리 함수에 대한 호출을 수신할 수 있다.
- [63] 다음으로, 단계(S430)에서, 가속기 라이브러리 함수의 호출에 응답하여, 접근 가능한 복수의 가속기의 각각에 가속기 라이브러리 함수를 할당할 수 있다. 일 실시예에서, 프로세서 상에서 가속기 라이브러리 래퍼는, DNN 프레임워크로부터 단일 가속기에서 딥러닝 연산 함수를 실행하기 위한 가속기 라이브러리 함수에 대한 호출을 수신하여, 복수의 가속기의 각각에 가속기 라이브러리 함수를 할당할 수 있다. 이 때, 가속기 라이브러리 래퍼는, 가속기 라이브러리 함수의 입력 데이터를 복수의 부분 입력 데이터 세트로 분할하여, 복수의 가속기의 각각에 분할된 복수의 부분 입력 데이터 세트의 각각을 할당할 수 있다. 예를 들어, 가속기 라이브러리 함수의 입력 데이터를 배치(batch) 단위로 입력 받을 경우, 가속기 라이브러리 래퍼는 복수의 가속기의 각각에 입력 데이터를 배치 단위로 분배하여, 하나 이상의 데이터 세트로 분할해 할당할 수 있다.
- [64] 일 실시예에 따르면, 가속기 라이브러리 래퍼는, 가속기 라이브러리 함수 및 분할된 복수의 부분 입력 데이터 세트의 각각을 복수의 가속기의 각각에 할당할 수 있다. 여기서, 가속기 라이브러리 함수 및 분할된 복수의 부분 입력 데이터 세트의 각각을 복수의 가속기의 각각에 할당함에 있어서, 가속기 라이브러리 함수의 메모리 접근 패턴을 분석하고 분석된 접근 패턴에 기초하여 가속기 라이브러리 함수의 실행 전에 가속기 라이브러리 함수의 입력 데이터가 접근 가능한 복수의 가속기의 각각에 할당될 수 있다. 예를 들어, DNN 관련 가속기 라이브러리 함수의 입력과 출력의 메모리로의 접근 패턴은 컴파일러 분석 기법 및/또는 프로파일링 기법(예를 들어, 미리 함수를 실행하여 메모리의 입출력

패턴을 파악하는 기법)을 사용하여 분석될 수 있다. 또 다른 예로서, 가속기 라이브러리 함수의 실행 전에, 컴파일러 분석 및/또는 프로파일링 기법에 의해 분석된 접근 패턴을 이용하여 복수의 가속기에 데이터가 미리 분배될 수 있다. DNN 프레임워크의 동작 특성상 동일한 패턴의 작업이 전체 학습이나 추론 과정에서 여러 iteration 동안 반복될 수 있다. 이러한 특성을 이용하여 처음 혹은 몇 번(예를 들어, 3회)의 iteration에 대한 메모리 접근 패턴이 파악되면, 전체 학습이나 추론 과정에서 각 가속기 라이브러리 함수의 메모리 접근 패턴이 분석될 수 있다.

- [65] 다음으로, 단계(S440)에서, 복수의 가속기의 각각으로부터 가속기 라이브러리 함수를 처리한 중간 결과 데이터를 수신할 수 있다. 일 실시예에 따르면, 가속기 라이브러리 래퍼는, 가속기 라이브러리 함수를 처리한 복수의 가속기로부터 중간 결과 데이터를 수신할 수 있다. 예를 들어, 가속기 라이브러리 래퍼는, 복수의 가속기의 각각에서 가속기 라이브러리 함수를 이용하여 복수의 부분 입력 데이터 세트의 각각을 처리한 중간 결과 데이터를 수신할 수 있다. 다른 예로서, 가속기 라이브러리 함수의 출력 데이터가 배치 단위로 출력될 경우, 가속기 라이브러리 래퍼는 복수의 가속기로부터 입력 데이터 세트의 각각을 처리한 중간 결과 데이터를 배치 단위로 수신할 수 있다.
- [66] 마지막으로, 단계(S450)에서, 수신된 중간 결과 데이터를 기초로, 호출된 가속기 라이브러리 함수에 대한 결과 데이터가 생성될 수 있다. 예를 들어, 가속기 라이브러리 래퍼는, 복수의 가속기로부터 수신된 중간 결과 데이터를 기초로 결과 데이터를 생성할 수 있으며, 도 7 및 도 8을 통해 보다 상세히 후술한다.
- [67] 후술하는 도 5 내지 도 8에 도시된 복수의 가속기(530_1, 530_2, ..., 530_n; 여기서, n은 2이상의 자연수임)는 하나의 컴퓨팅 장치에 포함되거나, 복수의 컴퓨팅 장치를 포함하는 클러스터 시스템에 포함될 수 있다. 여기서, 복수의 가속기(530_1, 530_2, ..., 530_n)는 DNN 프레임워크가 동작되는 컴퓨팅 장치(노드)에 의해 접근 가능한 가속기를 포함할 수 있다.
- [68] 도 5는 본 개시의 일 실시예에 따른 입력 데이터(510)를 복수의 부분 입력 데이터 세트(520_1, 520_2, ..., 520_n)로 분할하고, 분할된 복수의 부분 입력 데이터 세트(520_1, 520_2, ..., 520_n)의 각각을 복수의 가속기(530_1, 530_2, ..., 530_n)에 할당하는 예시를 나타내는 도면이다.
- [69] 일 실시예에 따르면, 도시된 바와 같이, 가속기 라이브러리 함수의 입력 데이터(510)는 복수의 부분 입력 데이터 세트(520_1, 520_2, ..., 520_n)로 분할될 수 있다. 분할된 복수의 부분 입력 데이터 세트(520_1, 520_2, ..., 520_n)의 각각 및 해당 가속기 라이브러리 함수는 복수의 가속기(530_1, 530_2, ..., 530_n)의 각각에 할당될 수 있다.
- [70] 일 실시예에 따르면, 가속기 라이브러리 래퍼는, 입력 데이터(510)를 배치 단위로 입력 받은 경우, 복수의 가속기(530_1, 530_2, ..., 530_n)에 균등하게

분배할 수 있다. 예를 들어, Tensorflow나 Pytorch에서 뉴런에 해당하는 데이터의 메모리 레이아웃은 일반적으로 "NCHW" 또는 "NHWC"인데, 여기서 "N"이 가장 앞에 존재하므로, 가장 높은 차원이 배치인 레이아웃에 해당할 수 있다. 이에 따라, 클러스터 시스템에 포함된 가속기가 두 개라고 가정할 경우, 전체 데이터를 연속된 두 부분으로 나누어 두 개의 가속기에 분배할 수 있다.

[71] 다른 실시예에서, 분배되는 부분 입력 데이터의 처리 속도가 다르거나, 클러스터 시스템의 네트워크 구성 등으로 처리 성능의 차이가 있을 경우, 성능 차이를 고려하여 복수의 부분 입력 데이터 세트(520_1, 520_2, ..., 520_n)가 복수의 가속기에 차등하여 분배될 수 있다.

[72] 도 6은 본 개시의 일 실시예에 따른 복수의 가속기(530_1, 530_2, ..., 530_n)에 입력 데이터(610)를 복수의 부분 입력 데이터 세트로 분할하지 못할 경우, 데이터를 할당하는 예시를 나타내는 도면이다.

[73] 일 실시예에서, 가속기 라이브러리 래퍼는, 입력 데이터(610)를 배치 단위로 나눌 수 없는 경우, 복수의 가속기(530_1, 530_2, ..., 530_n)에게 동일한 입력 데이터(610)를 할당할 수 있다. 여기서, 입력 데이터(610)가 배치 단위로 나눌 수 없는 경우는 복수의 가속기(530_1, 530_2, ..., 530_n)에서 해당 입력 데이터(610)가 필요한 경우일 수 있으며, DNN에서 주로 네트워크 파라미터가 이에 해당할 수 있다. 예를 들어, Tensorflow나 Pytorch에서 파라미터 데이터의 메모리 레이아웃은 일반적으로 "CKHW" 또는 "HWKC"인데, 이는 배치와는 무관하여, 배치 단위로 나눌 수 없다. 따라서, 복수의 가속기(530_1, 530_2, ..., 530_n)가 동일한 데이터를 사용하도록 각각의 복수의 가속기(530_1, 530_2, ..., 530_n)에 입력 데이터(610)의 복사본을 할당할 수 있다.

[74] 다른 실시예에 따르면, 가속기 라이브러리 래퍼는 가속기 라이브러리 함수의 파라미터 데이터(즉, 가중치; 미도시)를 복수의 부분 파라미터 데이터 세트로 분할하고, 분할된 부분 파라미터 데이터 세트의 각각을 복수의 가속기(530_1, 530_2, ..., 530_n)에 제공할 수 있다. 이 때, 가속기 라이브러리 래퍼는 분할된 파라미터 데이터 세트와 연관된 가속기 라이브러리 함수를 함께 복수의 가속기(530_1, 530_2, ..., 530_n)에 제공할 수 있다. 또한, 가속기 라이브러리 래퍼는 분할된 파라미터 데이터 세트와 연관된 가속기 라이브러리 함수의 입력 데이터(610)를 복수의 가속기(530_1, 530_2, ..., 530_n)에 분할된 파라미터 데이터 세트 및 가속기 라이브러리 함수와 함께 제공할 수 있다. 여기서, 도시된 바와 같이, 입력 데이터(610)는 분할되지 않을 수 있다. 이와 달리, 가속기 라이브러리 래퍼는 분할된 파라미터 데이터 세트와 연관된 가속기 라이브러리 함수의 입력 데이터(610)를 복수의 부분 입력 데이터 세트로 분할하고, 분할된 복수의 부분 입력 데이터 세트를, 분할된 파라미터 데이터 세트 및 가속기 라이브러리 함수와 함께 복수의 가속기(530_1, 530_2, ..., 530_n)에 제공할 수 있다.

[75] 도 7은 본 개시의 일 실시예에 따른 복수의 가속기(530_1, 530_2, ..., 530_n)로부터 수신한 중간 결과 데이터(710_1, 710_2, ..., 710_n)를 연결하여,

결과 데이터(720)를 생성하는 예시를 나타내는 도면이다. 일 실시예에 따르면, 가속기 라이브러리 래퍼는 복수의 가속기(530_1, 530_2, ..., 530_n)로부터 수신된 중간 결과 데이터(710_1, 710_2, ..., 710_n)를 수신하고, 수신된 중간 결과 데이터(710_1, 710_2, ..., 710_n)를 연결하여(concatenate), 결과 데이터(720)를 생성할 수 있다. 예를 들어, CuDNN에서 제공하는 가속기 라이브러리 함수 중 convolution 함수의 경우, 복수의 가속기(530_1, 530_2, ..., 530_n)로부터 수신한 중간 결과 데이터(710_1, 710_2, ..., 710_n)를 연결함으로써, 결과 데이터(720)가 생성될 수 있다.

[76] 일 실시예에서, 가속기 라이브러리 래퍼(330)는, 중간 결과 데이터(710_1, 710_2, ..., 710_n)가 배치 단위로 병렬화되는 경우, 중간 결과 데이터(710_1, 710_2, ..., 710_n)를 연결하여(concatenate) 결과 데이터(720)를 생성할 수 있다. 예를 들어, 중간 결과 데이터(710_1, 710_2, ..., 710_n)가 배치 단위로 병렬화되는 경우, 중간 결과 데이터(710_1, 710_2, ..., 710_n)는 단순히 연결되어 결과 데이터(710)로서 저장할 수 있다.

[77] 도 8은 본 개시의 일 실시예에 따른 복수의 가속기(530_1, 530_2, ..., 530_n)로부터 수신된 중간 결과 데이터(810_1, 810_2, ..., 810_n)를 연산하여, 결과 데이터(820)를 생성하는 예시를 나타내는 도면이다. 도 8에 도시된 바와 같이, 가속기 라이브러리 래퍼는 복수의 가속기(530_1, 530_2, ..., 530_n)의 각각으로부터 가속기 라이브러리 함수의 파라미터 데이터를 처리한 중간 결과 데이터(810_1, 810_2, ..., 810_n)를 수신하고, 가속기 라이브러리 함수의 파라미터 데이터를 처리한 중간 결과 데이터(810_1, 810_2, ..., 810_n)를 연산하여, 결과 데이터(820)를 생성할 수 있다. 이러한 연산 과정은 리덕션(reduction)이라고 지칭될 수 있다. 예를 들어, CuDNN에서 제공하는 가속기 라이브러리 함수 중 max pooling의 경우, 복수의 가속기(530_1, 530_2, ..., 530_n)로부터 수신한 중간 결과 데이터(810_1, 810_2, ..., 810_n)를 연산 처리함으로써, 결과 데이터(820)가 생성될 수 있다.

[78] 일 실시예에 따르면, 중간 결과 데이터(810_1, 810_2, ..., 810_n)가 배치 방향으로 병렬화되지 않는 경우, 학습 과정에 필요한 네트워크 파라미터의 그라디언트(gradient) 데이터를 만들어 내는 연산이 사용될 수 있다. 복수의 가속기(530_1, 530_2, ..., 530_n)가 자신에게 분배된 입력 데이터로부터 네트워크 파라미터 그라디언트를 계산하고, 중간 결과 데이터(810_1, 810_2, ..., 810_n)를 서로 교환하여 합산할 수 있다.

[79] 도 9는 본 개시의 일 실시예에 따른 복수의 가속기(930_1, ..., 930_n; 여기서, n은 2 이상의 자연수임)에서 연산을 수행하는 동시에 복수의 부분 입력 데이터 세트의 복사를 수행하는 예시를 나타내는 도면이다. 일 실시예에 따르면, 복수의 가속기(930_1, ..., 930_n)의 각각에 할당되는 부분 입력 데이터 세트(910)가 n개의 부분 입력 데이터 세트를 포함하는 경우(여기서, n은 2이상의 자연수임), n개의 부분 입력 데이터 세트(910) 중에서, m번째 부분 입력 데이터 세트(920_1)를

복수의 가속기(930_1, ..., 930_n)의 각각에서 처리하는 동시에 m+1번째 부분 입력 데이터 세트(920_2)를 복수의 가속기(930_1, ..., 930_n)의 각각에 할당하는 단계(여기서, m은 n보다 작은 자연수임)를 포함할 수 있다. 이를 위해, 복수의 부분 입력 데이터 세트(910)의 각각을 복수의 가속기(930_1, ..., 930_n) 각각에 할당하기 이전에, 처리될 가속기 라이브러리 함수의 메모리 접근 패턴이 분석될 수 있다.

- [80] 일 실시예에서, 가속기 라이브러리 래퍼(330)는, 분석된 메모리 접근 패턴을 기초로, 복수의 부분 입력 데이터 세트(910) 중 일부인 m번째 부분 입력 데이터 세트(920_1)를 가속기(930_1)에 할당하여 처리할 수 있다. m번째 부분 입력 데이터 세트(920_1)가 가속기(930_1)에서 처리되는 동시에, 가속기 라이브러리 래퍼(330)는 m+1번째 부분 입력 데이터 세트(920_2)를 가속기(930_1)에 할당할 수 있다. 예를 들어, 복수의 가속기(930_1, ..., 930_n)의 기계적 특성으로 인해, 가속기 내의 메모리의 복사 동작과 연산 동작은 별도로 처리되므로, memory transfer overlapping 기법을 이용해 복수의 가속기(930_1, ..., 930_n)에서 연산이 수행되는 동안 메모리의 복사를 동시에 처리하여, 딥러닝 어플리케이션의 총 수행 시간을 줄일 수 있다. 이러한 동작 방식 하에서, 도시된 바와 같이, 복수의 부분 입력 데이터 세트(920_1, 920_3)가 복수의 가속기(930_1, 930_n)의 각각에 할당하고, 할당된 복수의 부분 입력 데이터 세트(920_1, 920_3)가 처리되는 동안, 또 다른 복수의 부분 입력 데이터 세트(920_2, 920_4)가 복수의 가속기(930_1, 930_n)의 각각에 할당될 수 있다.

- [81] 상술된 DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법은, 컴퓨터로 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현될 수도 있다. 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의해 판독될 수 있는 데이터가 저장되는 모든 종류의 기록장치를 포함한다. 컴퓨터가 읽을 수 있는 기록매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피 디스크, 광데이터 저장장치 등이 있다. 또한, 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터 시스템에 분산되어, 분산방식으로 컴퓨터가 읽을 수 있는 코드가 저장되고 실행될 수 있다. 그리고, 전술된 실시예들을 구현하기 위한 기능적인(functional) 프로그램, 코드 및 코드 세그먼트들은 본 발명이 속하는 기술분야의 프로그래머들에 의해 용이하게 추론될 수 있다.

- [82] 본 개시의 방법, 동작 또는 기법들은 다양한 수단에 의해 구현될 수도 있다. 예를 들어, 이러한 기법들은 하드웨어, 펌웨어, 소프트웨어, 또는 이들의 조합으로 구현될 수도 있다. 본원의 개시와 연계하여 설명된 다양한 예시적인 논리적 블록들, 모듈들, 회로들, 및 알고리즘 단계들은 전자 하드웨어, 컴퓨터 소프트웨어, 또는 양자의 조합들로 구현될 수도 있음을 통상의 기술자들은 이해할 것이다. 하드웨어 및 소프트웨어의 이러한 상호 대체를 명확하게 설명하기 위해, 다양한 예시적인 구성요소들, 블록들, 모듈들, 회로들, 및

단계들이 그들의 기능적 관점에서 일반적으로 위에서 설명되었다. 그러한 기능이 하드웨어로서 구현되는지 또는 소프트웨어로서 구현되는지의 여부는, 특정 애플리케이션 및 전체 시스템에 부과되는 설계 요구사항들에 따라 달라진다. 통상의 기술자들은 각각의 특정 애플리케이션을 위해 다양한 방식들로 설명된 기능을 구현할 수도 있으나, 그러한 구현들은 본 개시의 범위로부터 벗어나게 하는 것으로 해석되어서는 안된다.

- [83] 하드웨어 구현에서, 기법들을 수행하는 데 이용되는 프로세싱 유닛들은, 하나 이상의 ASIC들, DSP들, 디지털 신호 프로세싱 디바이스들(digital signal processing devices; DSPD들), 프로그램가능 논리 디바이스들(programmable logic devices; PLD들), 필드 프로그램가능 게이트 어레이들(field programmable gate arrays; FPGA들), 프로세서들, 제어기들, 마이크로제어기들, 마이크로프로세서들, 전자 디바이스들, 본 개시에 설명된 기능들을 수행하도록 설계된 다른 전자 유닛들, 컴퓨터, 또는 이들의 조합 내에서 구현될 수도 있다.
- [84] 따라서, 본 개시와 연계하여 설명된 다양한 예시적인 논리 블록들, 모듈들, 및 회로들은 범용 프로세서, DSP, ASIC, FPGA나 다른 프로그램 가능 논리 디바이스, 이산 게이트나 트랜지스터 로직, 이산 하드웨어 컴포넌트들, 또는 본원에 설명된 기능들을 수행하도록 설계된 것들의 임의의 조합으로 구현되거나 수행될 수도 있다. 범용 프로세서는 마이크로프로세서일 수도 있지만, 대안으로, 프로세서는 임의의 종래의 프로세서, 제어기, 마이크로제어기, 또는 상태 머신일 수도 있다. 프로세서는 또한, 컴퓨팅 디바이스들의 조합, 예를 들면, DSP와 마이크로프로세서, 복수의 마이크로프로세서들, DSP 코어와 연계한 하나 이상의 마이크로프로세서들, 또는 임의의 다른 구성의 조합으로서 구현될 수도 있다.
- [85] 펌웨어 및/또는 소프트웨어 구현에 있어서, 기법들은 랜덤 액세스 메모리(random access memory; RAM), 판독 전용 메모리(read-only memory; ROM), 비휘발성 RAM(non-volatile random access memory; NVRAM), PROM(programmable read-only memory), EPROM(erasable programmable read-only memory), EEPROM(electrically erasable PROM), 플래시 메모리, 컴팩트 디스크(compact disc; CD), 자기 또는 광학 데이터 스토리지 디바이스 등과 같은 컴퓨터 판독가능 매체 상에 저장된 명령들로서 구현될 수도 있다. 명령들은 하나 이상의 프로세서들에 의해 실행 가능할 수도 있고, 프로세서(들)로 하여금 본 개시에 설명된 기능의 특정 양태들을 수행하게 할 수도 있다.
- [86] 소프트웨어로 구현되는 경우, 상기 기법들은 하나 이상의 명령들 또는 코드로서 컴퓨터 판독 가능한 매체 상에 저장되거나 또는 컴퓨터 판독 가능한 매체를 통해 전송될 수도 있다. 컴퓨터 판독가능 매체들은 한 장소에서 다른 장소로 컴퓨터 프로그램의 전송을 용이하게 하는 임의의 매체를 포함하여 컴퓨터 저장 매체들 및 통신 매체들 양자를 포함한다. 저장 매체들은 컴퓨터에 의해 액세스될 수 있는 임의의 이용 가능한 매체들일 수도 있다. 비제한적인

예로서, 이러한 컴퓨터 판독가능 매체는 RAM, ROM, EEPROM, CD-ROM 또는 다른 광학 디스크 스토리지, 자기 디스크 스토리지 또는 다른 자기 스토리지 디바이스들, 또는 소망의 프로그램 코드를 명령들 또는 데이터 구조들의 형태로 이송 또는 저장하기 위해 사용될 수 있으며 컴퓨터에 의해 액세스될 수 있는 임의의 다른 매체를 포함할 수 있다. 또한, 임의의 접속이 컴퓨터 판독가능 매체로 적절히 칭해진다.

- [87] 예를 들어, 소프트웨어가 동축 케이블, 광섬유 케이블, 연선, 디지털 가입자 회선 (DSL), 또는 적외선, 무선, 및 마이크로파와 같은 무선 기술들을 사용하여 웹사이트, 서버, 또는 다른 원격 소스로부터 전송되면, 동축 케이블, 광섬유 케이블, 연선, 디지털 가입자 회선, 또는 적외선, 무선, 및 마이크로파와 같은 무선 기술들은 매체의 정의 내에 포함된다. 본원에서 사용된 디스크(disk)와 디스크(disc)는, CD, 레이저 디스크, 광 디스크, DVD(digital versatile disc), 플로피디스크, 및 블루레이 디스크를 포함하며, 여기서 디스크들(disks)은 보통 자기적으로 데이터를 재생하고, 반면 디스크들(disks)은 레이저를 이용하여 광학적으로 데이터를 재생한다. 위의 조합들도 컴퓨터 판독가능 매체들의 범위 내에 포함되어야 한다.
- [88] 소프트웨어 모듈은, RAM 메모리, 플래시 메모리, ROM 메모리, EPROM 메모리, EEPROM 메모리, 레지스터들, 하드 디스크, 이동식 디스크, CD-ROM, 또는 공지된 임의의 다른 형태의 저장 매체 내에 상주할 수도 있다. 예시적인 저장 매체는, 프로세서가 저장 매체로부터 정보를 판독하거나 저장 매체에 정보를 기록할 수 있도록, 프로세서에 연결될 수 있다. 대안으로, 저장 매체는 프로세서에 통합될 수도 있다. 프로세서와 저장 매체는 ASIC 내에 존재할 수도 있다. ASIC은 유저 단말 내에 존재할 수도 있다. 대안으로, 프로세서와 저장 매체는 유저 단말에서 개별 구성요소들로서 존재할 수도 있다.
- [89] 이상 설명된 실시예들이 하나 이상의 독립형 컴퓨터 시스템에서 현재 개시된 주제의 양태들을 활용하는 것으로 기술되었으나, 본 개시는 이에 한정되지 않고, 네트워크나 분산 컴퓨팅 환경과 같은 임의의 컴퓨팅 환경과 연계하여 구현될 수도 있다. 또 나아가, 본 개시에서 주제의 양상들은 복수의 프로세싱 칩들이나 장치들에서 구현될 수도 있고, 스토리지는 복수의 장치들에 걸쳐 유사하게 영향을 받게 될 수도 있다. 이러한 장치들은 PC들, 네트워크 서버들, 및 휴대용 장치들을 포함할 수도 있다.
- [90] 본 명세서에서는 본 개시가 일부 실시예들과 관련하여 설명되었지만, 본 개시의 발명이 속하는 기술분야의 통상의 기술자가 이해할 수 있는 본 개시의 범위를 벗어나지 않는 범위에서 다양한 변형 및 변경이 이루어질 수 있다. 또한, 그러한 변형 및 변경은 본 명세서에 첨부된 특허청구의 범위 내에 속하는 것으로 생각되어야 한다.

[91]

청구범위

- [청구항 1] DNN 프레임워크를 이용하는 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법에 있어서,
 딥러닝 연산 함수에 대한 호출을 수신하는 단계;
 단일의 가속기에서 상기 딥러닝 연산 함수를 실행하기 위한 가속기 라이브러리 함수에 대한 호출을 수신하는 단계;
 상기 가속기 라이브러리 함수의 호출에 응답하여, 접근 가능한 복수의 가속기의 각각에 상기 가속기 라이브러리 함수를 할당하는 단계;
 상기 복수의 가속기의 각각으로부터 상기 가속기 라이브러리 함수를 처리한 중간 결과 데이터를 수신하는 단계; 및
 상기 수신된 중간 결과 데이터를 기초로, 상기 호출된 가속기 라이브러리 함수에 대한 결과 데이터를 생성하는 단계를 포함하는,
 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.
- [청구항 2] 제1항에 있어서,
 상기 접근 가능한 복수의 가속기의 각각에 상기 가속기 라이브러리 함수를 할당하는 단계는,
 상기 가속기 라이브러리 함수의 입력 데이터를 복수의 부분 입력 데이터 세트로 분할하는 단계; 및
 상기 가속기 라이브러리 함수 및 상기 분할된 복수의 부분 입력 데이터 세트의 각각을 상기 복수의 가속기의 각각에 할당하는 단계를 포함하고,
 상기 복수의 가속기의 각각으로부터 상기 가속기 라이브러리 함수를 처리한 중간 결과 데이터를 수신하는 단계는, 상기 복수의 가속기의 각각에서 상기 가속기 라이브러리 함수를 이용하여 상기 복수의 부분 입력 데이터 세트의 각각을 처리한 중간 결과 데이터를 수신하는 단계를 포함하는,
 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.
- [청구항 3] 제2항에 있어서,
 상기 가속기 라이브러리 함수 및 상기 분할된 복수의 부분 입력 데이터 세트의 각각을 상기 복수의 가속기의 각각에 할당하는 단계는,
 상기 가속기 라이브러리 함수의 메모리 접근 패턴을 분석하는 단계; 및
 상기 분석된 접근 패턴에 기초하여 상기 가속기 라이브러리 함수의 실행 전에 상기 가속기 라이브러리 함수의 입력 데이터를 상기 접근 가능한 복수의 가속기의 각각에 할당하는 단계를 포함하는,
 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.
- [청구항 4] 제2항에 있어서,
 상기 가속기 라이브러리 함수 및 상기 분할된 복수의 부분 입력 데이터

세트의 각각을 상기 복수의 가속기의 각각에 할당하는 단계는,
 상기 복수의 가속기의 각각에 할당되는 부분 입력 데이터 세트가 n 개의
 부분 입력 데이터 세트를 포함하는 경우(여기서, n 은 2이상의 자연수임),
 상기 n 개의 부분 입력 데이터 세트 중에서, m 번째 부분 입력 데이터
 세트를 상기 복수의 가속기의 각각에서 처리하는 동시에 $m+1$ 번째 부분
 입력 데이터 세트가 상기 복수의 가속기의 각각에 할당하는 단계(여기서,
 m 은 n 보다 작은 자연수임)를 포함하는,
 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.

[청구항 5]

제1항에 있어서,
 상기 접근 가능한 복수의 가속기의 각각에 상기 가속기 라이브러리
 함수를 할당하는 단계는,
 상기 가속기 라이브러리 함수의 파라미터 데이터를 복수의 부분
 파라미터 데이터 세트로 분할하는 단계; 및
 상기 가속기 라이브러리 함수 및 상기 분할된 복수의 부분 파라미터
 데이터 세트의 각각을 상기 복수의 가속기의 각각에 할당하는 단계를
 포함하는,
 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.

[청구항 6]

제5항에 있어서,
 상기 복수의 가속기의 각각으로부터 상기 가속기 라이브러리 함수를
 처리한 중간 결과 데이터를 수신하는 단계는 상기 복수의 가속기의
 각각으로부터 상기 가속기 라이브러리 함수의 파라미터 데이터를 처리한
 중간 결과 데이터를 수신하는 단계를 포함하고,
 상기 호출된 가속기 라이브러리 함수에 대한 결과 데이터를 생성하는
 단계는, 상기 가속기 라이브러리 함수의 파라미터 데이터를 처리한 중간
 결과 데이터에 대한 결과 데이터를 생성하는 단계를 포함하는,
 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.

[청구항 7]

제1항에 있어서,
 상기 수신된 중간 결과 데이터를 기초로, 상기 호출된 가속기 라이브러리
 함수에 대한 결과 데이터를 생성하는 단계는 상기 수신된 중간 결과
 데이터를 연결하여(concatenate) 상기 결과 데이터를 생성하는 단계를
 포함하는,
 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.

[청구항 8]

제1항에 있어서,
 상기 수신된 중간 결과 데이터를 기초로, 상기 호출된 가속기 라이브러리
 함수에 대한 결과 데이터를 생성하는 단계는 상기 수신된 중간 결과
 데이터를 연산하여 상기 결과 데이터를 생성하는 단계를 포함하는,
 단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.

[청구항 9]

제1항에 있어서,

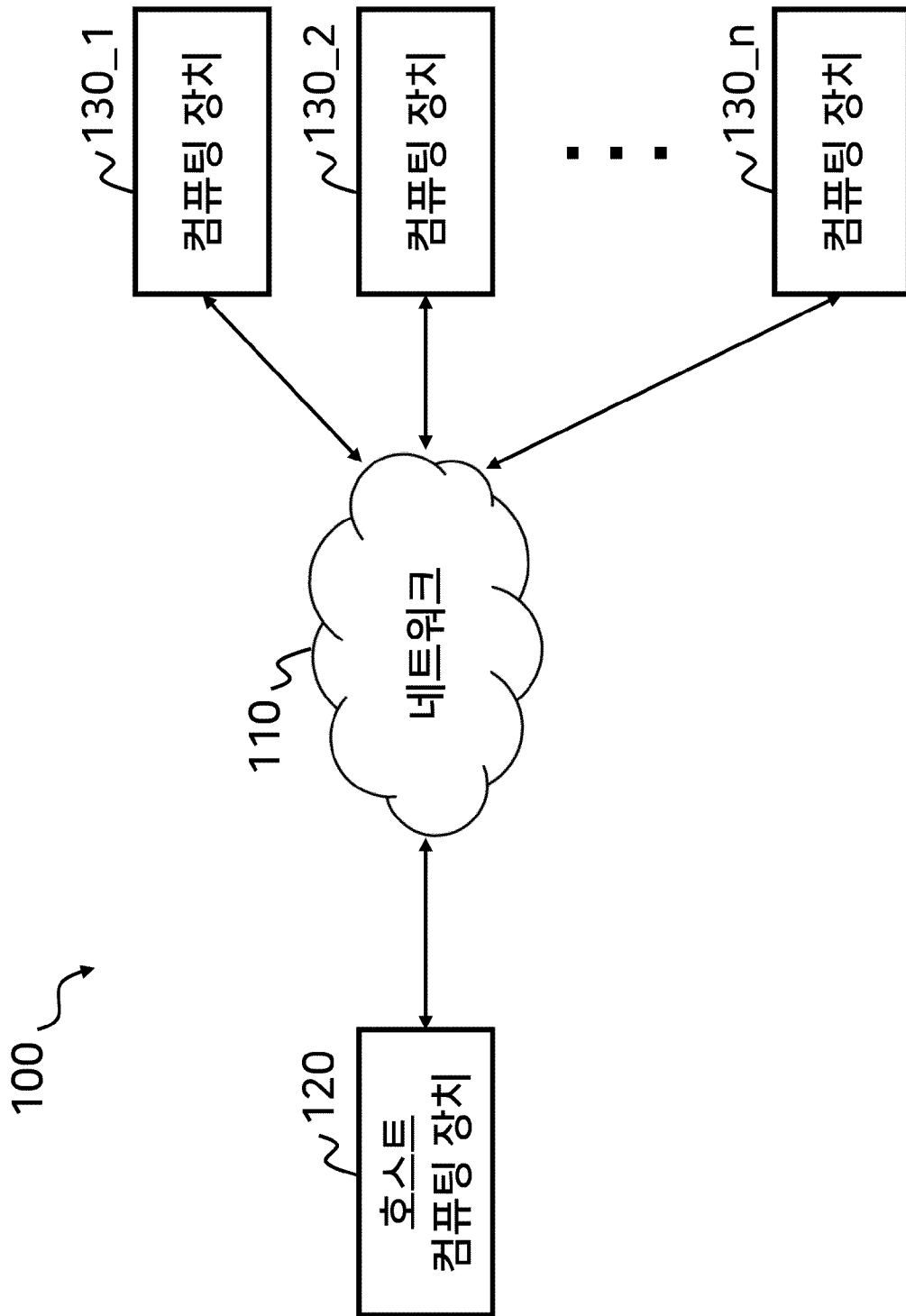
상기 복수의 가속기는 하나의 컴퓨팅 장치에 포함되는,
단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.

[청구항 10] 제1항에 있어서,
상기 복수의 가속기는 복수의 컴퓨팅 장치를 포함하는 클러스터
시스템에 포함되는,

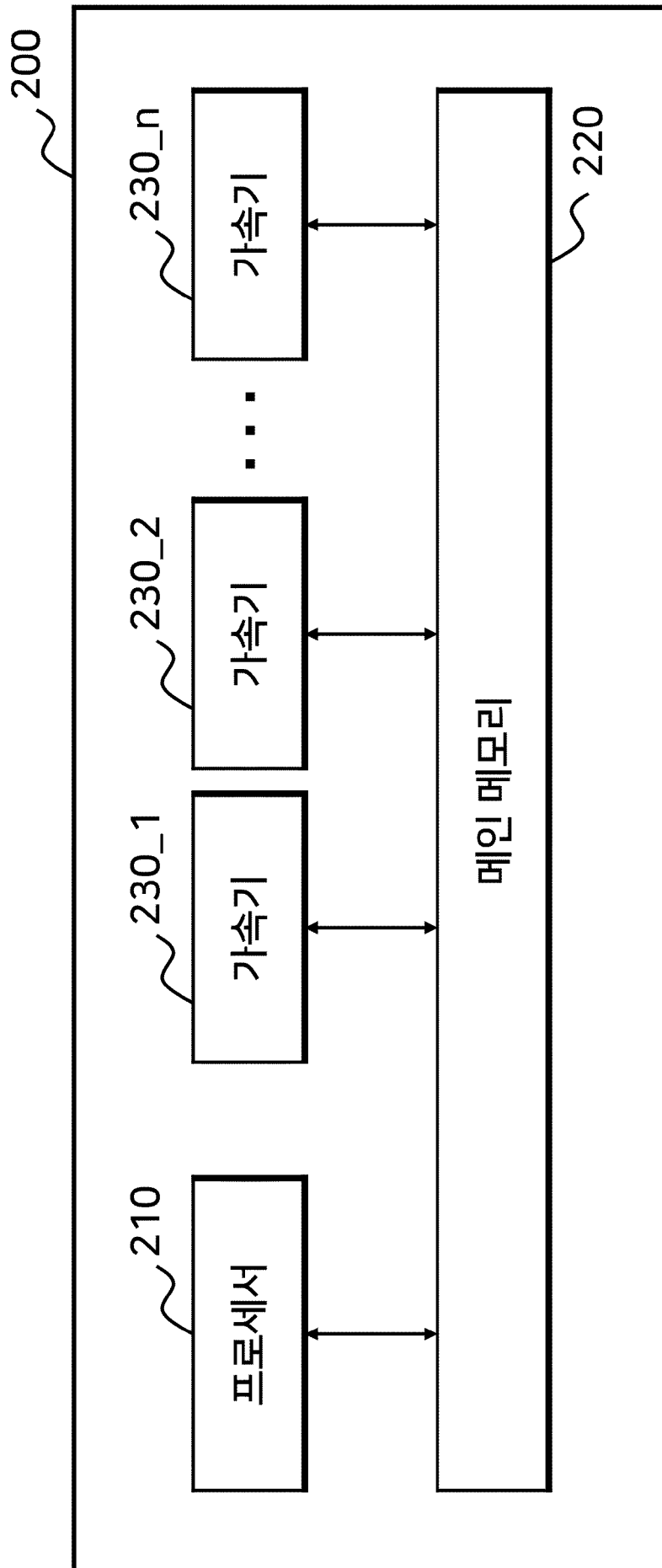
단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법.

[청구항 11] 제1항 내지 제10항 중 어느 한 항에 따른 DNN 프레임워크를 이용하는
단일 가속기용 프로그램을 복수의 가속기에서 처리하는 방법을
컴퓨터에서 실행하기 위해 컴퓨터 판독 가능한 기록 매체에 저장된
컴퓨터 프로그램.

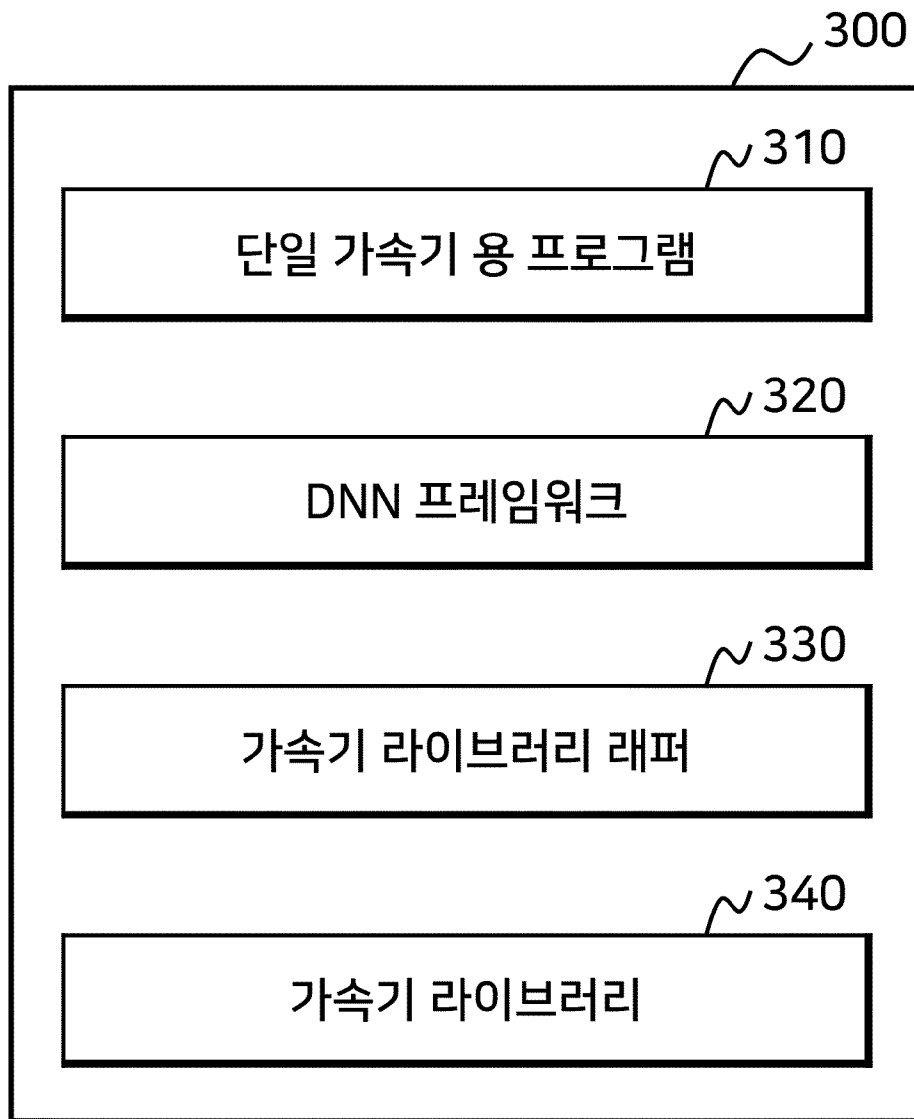
[도 1]



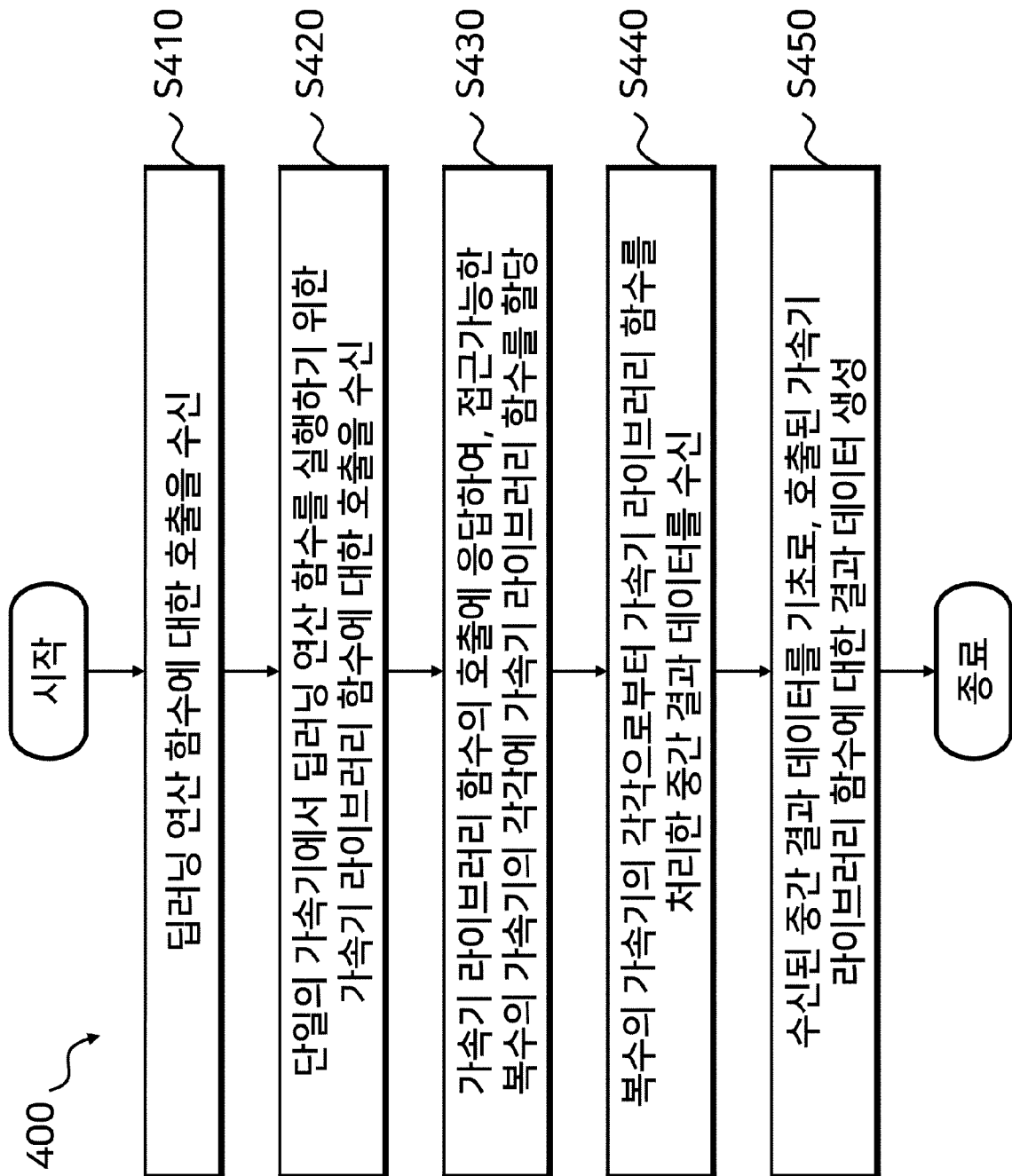
[도2]



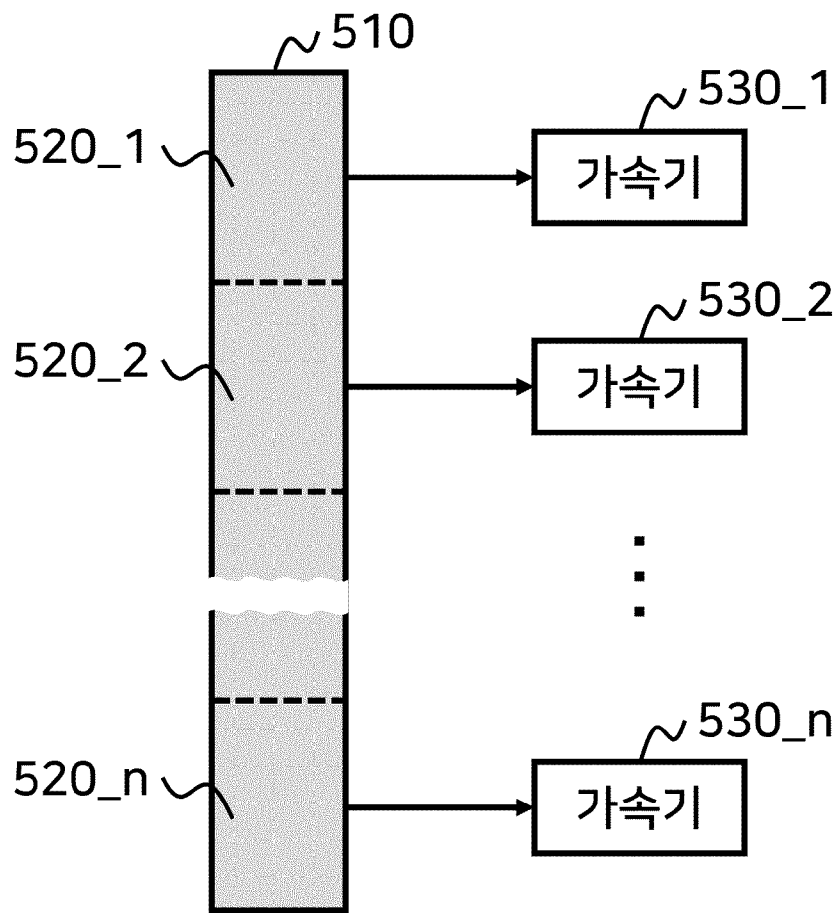
[도3]



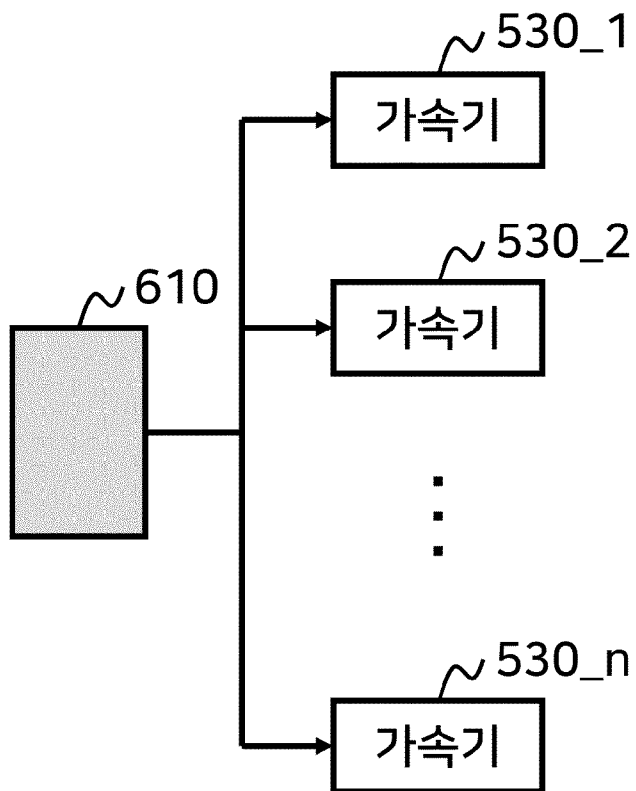
[도4]



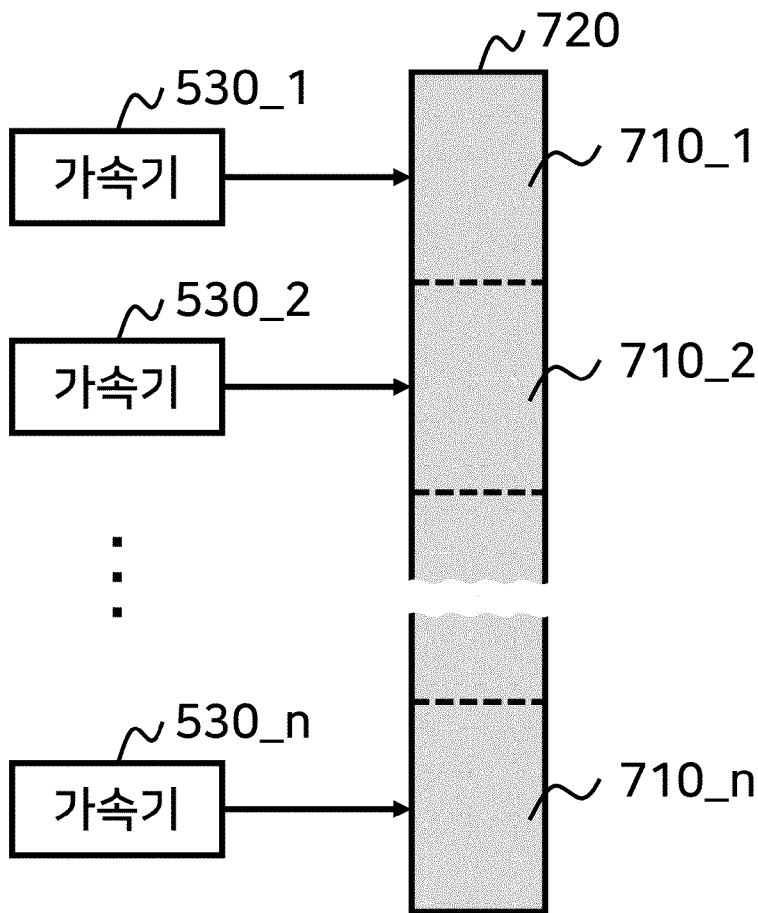
[도5]



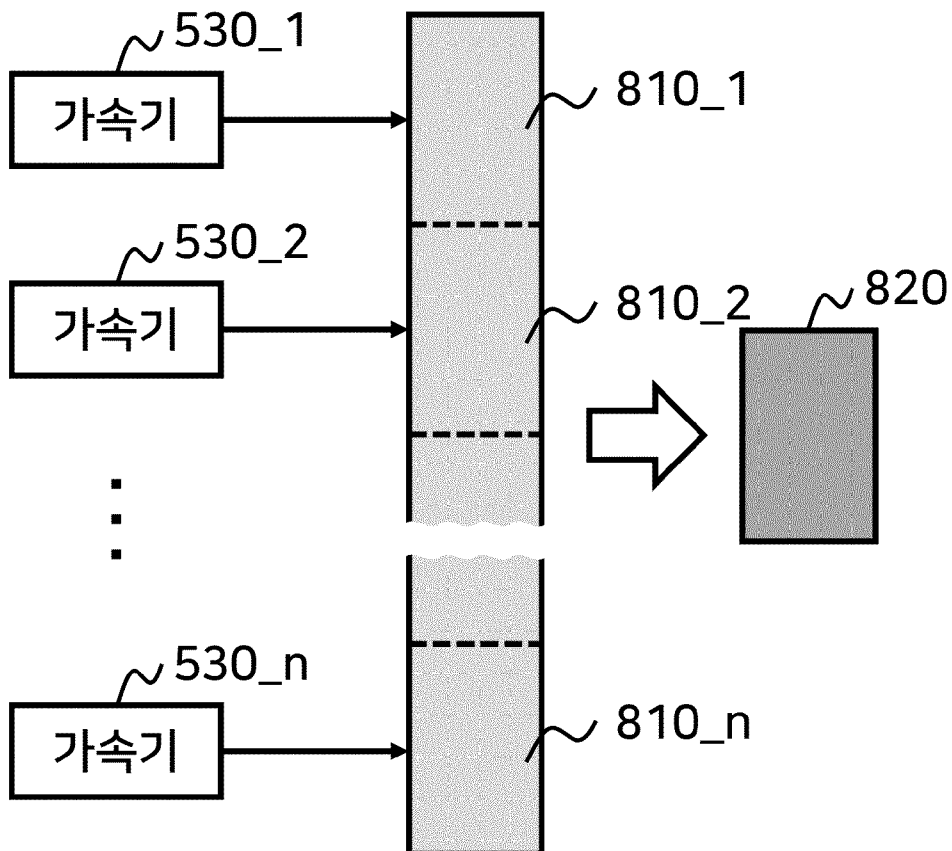
[도6]



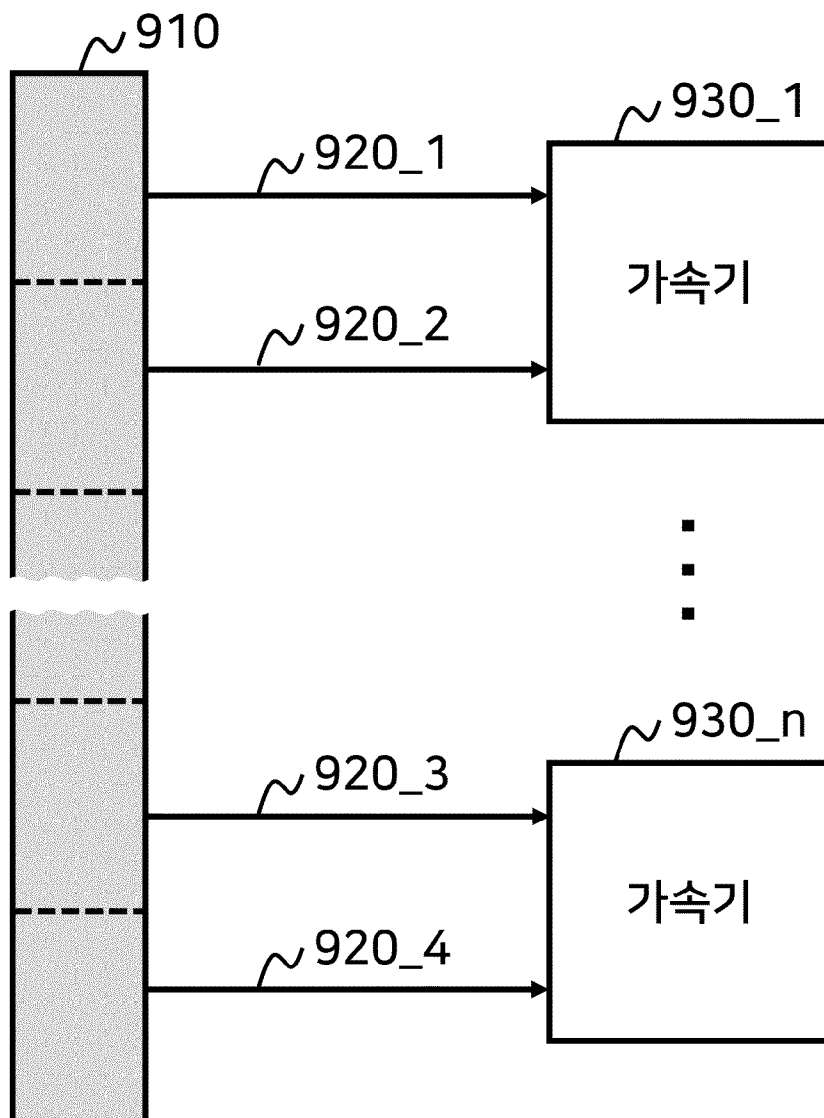
[도7]



[도8]



[도9]



INTERNATIONAL SEARCH REPORT

International application No.

PCT/KR2020/003399

A. CLASSIFICATION OF SUBJECT MATTER

G06T 1/20(2006.01)i, G06T 1/60(2006.01)i, G06N 20/00(2019.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06T 1/20; G06F 17/00; G06F 9/38; G06F 9/46; G06N 3/08; G06T 1/60; G06N 20/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models: IPC as above

Japanese utility models and applications for utility models: IPC as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS (KIPO internal) & Keywords: DNN(deep neural network), framework, deep learning, library function, accelerator

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2018-0293493 A1 (INTEL CORPORATION) 11 October 2018 See paragraph [0138]; and claim 14.	1-11
A	US 2011-0173155 A1 (BECCHI, Michela et al.) 14 July 2011 See claim 1.	1-11
A	GUAN, Yijin et al. FP-DNN: An Automated Framework for Mapping Deep Neural Networks onto FPGAs with RTL-HLS Hybrid Templates. 02 May 2017, pp.152-159, [Retrieved on 04 June 2020], Retrieved from < https://vast.cs.ucla.edu/sites/default/files/publications/fccm2017.pdf > See pages 152-159.	1-11
A	CHETLUR, Sharan et al. cuDNN: Efficient Primitives for Deep Learning. arXiv:1410.0759. 18 December 2014, pp.1-9, [Retrieved on 03 June 2020], Retrieved from < https://arxiv.org/abs/1410.0759 > See pages 1-8.	1-11
A	KR 10-2015-0084098 A (ELECTRONICS AND TELECOMMUNICATIONS RESEARCH INSTITUTE) 22 July 2015 See claims 1-11.	1-11



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

18 JUNE 2020 (18.06.2020)

Date of mailing of the international search report

18 JUNE 2020 (18.06.2020)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex Daejeon Building 4, 189, Cheongsu-ro, Seo-gu,
Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

Telephone No.

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/KR2020/003399

Patent document cited in search report	Publication date	Patent family member	Publication date
US 2018-0293493 A1	11/10/2018	None	
US 2011-0173155 A1	14/07/2011	US 8375392 B2	12/02/2013
KR 10-2015-0084098 A	22/07/2015	US 2015-0199214 A1	16/07/2015

A. 발명이 속하는 기술분류(국제특허분류(IPC))

G06T 1/20(2006.01)i, G06T 1/60(2006.01)i, G06N 20/00(2019.01)i

B. 조사된 분야

조사된 최소문헌(국제특허분류를 기재)

G06T 1/20; G06F 17/00; G06F 9/38; G06F 9/46; G06N 3/08; G06T 1/60; G06N 20/00

조사된 기술분야에 속하는 최소문헌 이외의 문헌

한국등록실용신안공보 및 한국공개실용신안공보: 조사된 최소문헌란에 기재된 IPC

일본등록실용신안공보 및 일본공개실용신안공보: 조사된 최소문헌란에 기재된 IPC

국제조사에 이용된 전산 데이터베이스(데이터베이스의 명칭 및 검색어(해당하는 경우))

eKOMPASS(특허청 내부 검색시스템) & 키워드: DNN(Deep Neural Network), 프레임워크(software framework), 딥러닝(deep learning), 라이브러리 함수(library function), 가속기(accelerator)

C. 관련 문헌

카테고리*	인용문헌명 및 관련 구절(해당하는 경우)의 기재	관련 청구항
A	US 2018-0293493 A1 (INTEL CORPORATION) 2018.10.11 단락 [0138]; 및 청구항 14	1-11
A	US 2011-0173155 A1 (MICHELA BECCHI 등) 2011.07.14 청구항 1	1-11
A	YIJIN GUAN 등, 'FP-DNN: An Automated Framework for Mapping Deep Neural Networks onto FPGAs with RTL-HLS Hybrid Templates', 2017.05.02, pp.152-159, [검색일 2020.06.04.]. 출처< https://vast.cs.ucla.edu/sites/default/files/publications/fccm2017.pdf > 페이지 152-159	1-11
A	SHARAN CHETLUR 등, 'cuDNN: Efficient Primitives for Deep Learning', arXiv:1410.0759, 2014.12.18, pp.1-9 [검색일 2020.06.03]. 출처< https://arxiv.org/abs/1410.0759 > 페이지 1-8	1-11
A	KR 10-2015-0084098 A (한국전자통신연구원) 2015.07.22 청구항 1-11	1-11

☐ 추가 문헌이 C(계속)에 기재되어 있습니다.☒ 대응특허에 관한 별지를 참조하십시오.

* 인용된 문헌의 특별 카테고리:

“A” 특별히 관련이 없는 것으로 보이는 일반적인 기술수준을 정의한 문헌

“D” 본 국제출원에서 출원인이 인용한 문헌

“E” 국제출원일보다 빠른 출원일 또는 우선일을 가지나 국제출원일 이후 “X”에 공개된 선출원 또는 특허 문헌

“L” 우선권 주장에 의문을 제기하는 문헌 또는 다른 인용문헌의 공개일 또는 다른 특별한 이유(이유를 명시)를 밝히기 위하여 인용된 문헌

“O” 구두 개시, 사용, 전시 또는 기타 수단을 언급하고 있는 문헌

“P” 우선일 이후에 공개되었으나 국제출원일 이전에 공개된 문헌

“T” 국제출원일 또는 우선일 후에 공개된 문헌으로, 출원과 상충하지 않으며 발명의 기초가 되는 원리나 이론을 이해하기 위해 인용된 문헌

“X” 특별한 관련이 있는 문헌. 해당 문헌 하나만으로 청구된 발명의 신규성 또는 진보성이 없는 것으로 본다.

“Y” 특별한 관련이 있는 문헌. 해당 문헌이 하나 이상의 다른 문헌과 조합하는 경우로 그 조합이 당업자에게 자명한 경우 청구된 발명은 진보성이 없는 것으로 본다.

“&” 동일한 대응특허문헌에 속하는 문헌

국제조사의 실제 완료일

2020년 06월 18일 (18.06.2020)

국제조사보고서 발송일

2020년 06월 18일 (18.06.2020)

ISA/KR의 명칭 및 우편주소

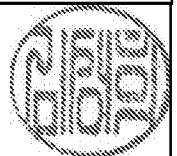
대한민국 특허청
(35208) 대전광역시 서구 청사로 189,
4동 (둔산동, 정부대전청사)

팩스 번호 +82-42-481-8578

심사관

김성희

전화번호 +82-42-481-3516



국제출원번호
PCT/KR2020/003399

국제조사보고서에서 인용된 특허문헌	공개일	대응특허문헌	공개일
US 2018-0293493 A1	2018/10/11	없음	
US 2011-0173155 A1	2011/07/14	US 8375392 B2	2013/02/12
KR 10-2015-0084098 A	2015/07/22	US 2015-0199214 A1	2015/07/16