(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2024-0066718
(43) 공개일자 2024년05월16일

(51) 국제특허분류(Int. Cl.)
G06N 3/04 (2023.01) *G06N 3/08* (2023.01)
H04W 84/18 (2009.01)
(52) CPC특허분류
G06N 3/045 (2023.01)
G06N 3/08 (2023.01)
(21) 출원번호 10-2022-0147865
(22) 출원일자 2022년11월08일
심사청구일자 2022년11월08일

(71) 출원인
고려대학교 산학협력단
서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(72) 발명자
양경식
서울특별시 동대문구 무학로45길 40, 401호(용두동)
신창용
서울특별시 동대문구 무학로49길 25, 대명빌딩 303호
(뒷면에 계속)
(74) 대리인
김등용

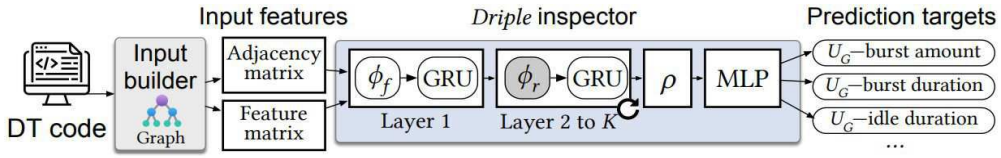
전체 청구항 수 : 총 10 항

(54) 발명의 명칭 **분산 딥러닝 학습 시간 및 자원 소모량 예측 방법 및 장치**

(57) 요 약

분산 딥러닝 학습에 요구되는 학습 시간 및 리소스 소모량을 예측하는 예측 모델을 생성하는 방법 및 예측 모델을 이용한 예측 방법이 개시된다. 상기 예측 모델 생성 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되고, 학습 데이터셋을 구축하는 단계, 및 GNN(Graph Neural Network, 그래프 신경망) 모델을 학습하여 예측 모델을 생성하는 단계를 포함하고, 상기 학습 데이터셋은 입력 데이터와 결과 데이터를 포함하고, 상기 학습 데이터셋을 구축하는 단계는 분산 딥러닝 학습 코드(DT 코드, Distributed Training code)를 그래프로 변환하는 단계, 및 상기 그래프로부터 인접 행렬(adjacency matrix)과 특징 행렬(feature matrix)을 추출하는 단계를 포함한다.

대 표 도 - 도3



(52) CPC특허분류
H04W 84/18 (2013.01)
(72) 발명자
유연호
서울특별시 성북구 고려대로24가길 14-6, 102호
이정환
서울특별시 성북구 안암로9가길 56. 303호
유혁
서울특별시 용산구 한강대로 205, A-803호

이 발명을 지원한 국가연구개발사업
과제고유번호 1711152940
과제번호 2015-0-00280-008
부처명 과학기술정보통신부
과제관리(전문)기관명 정보통신기획평가원
연구사업명 SW컴퓨팅산업원천기술개발
연구과제명 (SW 스타랩) 성능 및 보안 SLA 보장이 가능한 차세대 클라우드 인프라SW 개발
기 여 율 1/2
과제수행기관명 고려대학교산학협력단
연구기간 2022.01.01 ~ 2022.12.31
이 발명을 지원한 국가연구개발사업
과제고유번호 1345347378
과제번호 2021R1A6A1A13044830
부처명 교육부
과제관리(전문)기관명 한국연구재단
연구사업명 이공학학술연구기반구축
연구과제명 융합소프트웨어연구소
기 여 율 1/2
과제수행기관명 고려대학교 산학협력단
연구기간 2022.03.01 ~ 2023.02.28

명 세 서

청구범위

청구항 1

적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는 예측 모델 생성 방법에 있어서,

학습 데이터셋을 구축하는 단계; 및

GNN(Graph Neural Network, 그래프 신경망) 모델을 학습하여 예측 모델을 생성하는 단계를 포함하고,

상기 학습 데이터셋은 입력 데이터와 결과 데이터를 포함하고,

상기 학습 데이터셋을 구축하는 단계는,

분산 딥러닝 학습 코드(DT 코드, Distributed Training code)를 그래프로 변환하는 단계; 및

상기 그래프로부터 인접 행렬(adjacency matrix)과 특징 행렬(feature matrix)을 추출하는 단계를 포함하는,

예측 모델 생성 방법.

청구항 2

제1항에 있어서,

상기 결과 데이터는 GPU 활용률, GPU 메모리 활용률, 네트워크 TX 처리량, 네트워크 RX 처리량, GPU의 점유 시간, GPU 메모리의 점유 시간, 네트워크 TX의 점유 시간, 네트워크 RX의 점유 시간, GPU의 유휴 시간, GPU 메모리의 유휴 시간, 네트워크 TX의 유휴 시간, 및 네트워크 RX의 유휴 시간 중 적어도 하나를 포함하는,

예측 모델 생성 방법.

청구항 3

제1항에 있어서,

상기 GNN은 GCN(Graph Convolutional Network), GIN(Graph Isomorphism Network), 또는 GAN(Graph Attention Network)인,

예측 모델 생성 방법.

청구항 4

제1항에 있어서,

상기 GNN은 복수의 그래프 레이어들, 그래프 리드아웃 레이어, 및 MLP(Multi-Layer Perceptron) 레이어를 포함하는,

예측 모델 생성 방법.

청구항 5

제4항에 있어서,

상기 그래프 레이어들 각각은 GRU(Gated Recurrent Unit)를 포함하는,

예측 모델 생성 방법.

청구항 6

제1항에 있어서,

상기 예측 모델을 생성하는 단계 이후에,

상기 예측 모델에 대한 전이 학습(Transfer Learning, TL)을 수행하는 단계를 더 포함하는,

예측 모델 생성 방법.

청구항 7

제6항에 있어서,

상기 전이 학습을 수행하는 단계는 제2 학습 데이터셋을 이용하여 수행되고,

상기 학습 데이터셋은 제1 DT 세팅(Distributed Training setting)에 대응하는 데이터셋이고,

상기 제2 학습 데이터셋은 제2 DT 세팅에 대응하는 데이터셋이고,

상기 제1 DT 세팅과 상기 제2 DT 세팅은 분산 딥러닝을 수행하는 GPU의 종류, 파라미터 서버(Parameter Server, PS)의 개수, 및 워커 노드(worker node)의 개수 중 적어도 하나가 상이한,

예측 모델 생성 방법.

청구항 8

제6항에 있어서,

상기 전이 학습은 상기 예측 모델에 포함되는 복수의 그래프 레이어들 중 적어도 일부의 그래프 레이어의 파라미터들과 상기 예측 모델에 포함되는 MLP 레이어의 파라미터들 중 적어도 하나만을 업데이트하는,

예측 모델 생성 방법.

청구항 9

제6항에 있어서,

상기 전이 학습은 상기 예측 모델에 포함되는 복수의 그래프 레이어들 중 후반부의 절반의 레이어들의 파라미터들과 상기 MLP 레이어의 파라미터들만을 업데이트하는,

예측 모델 생성 방법.

청구항 10

제1항 내지 제9항에 기재된 예측 모델 생성 방법에 의하여 생성된 예측 모델을 이용한 예측 방법에 있어서,

예측 대상 입력 데이터를 생성하는 단계; 및

상기 예측 대상 입력 데이터를 상기 예측 모델에 입력하여 예측을 수행하는 단계를 포함하는,

예측 방법.

발명의 설명

기 술 분 야

[0001] 본 발명은 분산 딥러닝 모델을 학습하는 시스템에 관한 것으로, 더욱 상세하게는, 분산 딥러닝 모델의 학습에 소모되는 GPU, 네트워크, GPU 메모리 등의 자원 소모량과 그 자원들을 실 점유하는 점유 시간과 유휴하는 유휴 시간을 예측하는 기법에 관한 것이다.

배 경 기 술

[0002] 최근 딥러닝(deep learning)은 컴퓨터 비전(computer vision), 자연어 처리(natural language processing), 금융 거래(financial transactions), 헬스케어(healthcare), 및 자율주행(autonomous driving)에 이르기까지 다양한 응용 분야에서 활용되고 있다. 더욱 다양한 응용 분야를 지원하기 위해, 딥러닝 모델은 더 많은 레이어들과 매개변수들(parameters)를 추가하여, 그 모델의 크기가 점진적으로 그리고 지속적으로 커지고 있다. 가령, 지난 10년 동안 딥 러닝 모델이 지니는 매개변수(파라미터)의 크기는 2,917배 증가하였으며, 이로 인해 하나의 모델을 학습시키는 데에 필요한 학습 시간이 훨씬 길어졌다.

[0003] 급증하는 학습 시간을 감당하기 위해, GPU와 같은 가속기가 학습에 필수적으로 사용되며, 더 나아가 GPU가 탑재된 노드(worker)를 복수 사용하여 하나의 모델을 학습시키는 분산 딥러닝(Distributed Deep Learning, DDL)이 필수적으로 사용되고 있다.

[0004] 분산 딥러닝 중 가장 흔히 사용되는 방식은 여러개의 GPU 노드(워커)에서 학습 데이터를 나누어 학습하고 학습한 모델 파라미터(모델의 가중치 등)를 갱신하는 데이터 병렬 방식이다. 데이터 병렬 방식에서의 각 워커는 나눠진 학습 데이터에 대해 순방향 전파(forward propagation) 및 역방향 전파(backward propagation)를 수행하면서, 모델의 파라미터에 대한 그라디언트(gradient) 값을 생성한다. 그라디언트 값들로부터 모델의 파라미터들이 갱신되는데, 갱신을 위해 워커의 그라디언트 값을 집계하는 네트워크 통신이 필요하다.

[0005] 네트워크 통신 방식으로 크게 두 가지 기법이 사용되는데, PS(Parameter Server) 구조와 올-리듀스(all-reduce) 구조이다. PS 구조는 그라디언트를 수집하고 모델 파라미터를 갱신하여 워커에게 전달하는 별도의 노드(PS)가 존재하는 구조로서, 각각의 워커들이 학습한 모델 파라미터를 PS와 주고 받으면서 파라미터를 갱신한다. 올-리듀스 방식은 PS 없이 각각의 워커가 매 학습마다 모델 파라미터를 모든 워커에게 전송하며 갱신하는 방식이다.

[0006] 갱신 방식에는 크게 동기적 갱신 방식과 비동기적 갱신 방식이 있다. 모든 워커가 학습이 종료될 때까지 기다린 후 모델 파라미터를 취합해 갱신하고 다음 학습을 진행하는 것이 동기적 갱신 방식이다. 반대로, 각각의 워커의 학습 종료마다 모델 파라미터를 주고 받은 후 개별적으로 다음 학습을 진행하는 것이 비동기적 갱신 방식이다.

[0007] 그래프 신경망(Graph Neural Network, GNN)은, 그래프를 예측 모델의 입력값(input feature)으로 하는 특수한 종류의 기계학습 알고리즘이다. 일반적인 기계학습 알고리즘(심층 신경망 및 CNN(Convolutional Neural Network, 컨볼루션 신경망) 등)은 고정된 크기의 입력(예: 이미지, 텍스트 및 숫자 값)을 기반으로 동작한다. 반면, 그래프는 각 그래프가 동일한 종류의 정보를 담고 있더라도, 노드 및 간선(edge) 수가 다르고, 이로 인해 크기가 서로 상이하다(비정형).

[0008] 이러한 비정형 데이터인 그래프를 기반으로 예측을 수행하기 위해, GNN은 그래프를 대표하는 축약값인 임베딩(embedding)을 생성한다. GNN은 여러 개의 그래프 레이어(graph layer)로 구성되는데, 각 그래프 레이어는 각 노드별 임베딩을 생성한다. 구체적으로, 각 레이어는 노드별로 한 홉(hop)씩 떨어진 이웃 노드의 특징들(feature vector)을 집계(aggregate)하여 임베딩을 생성한다. 가령, 그래프 레이어가 n개 존재한다면, 최종적으로는 각 노드 임베딩은 n홉까지 떨어진 이웃 노드들의 특징 벡터를 축약한 값이 된다. 이때, 각 그래프 레이어 내부에서 노드 임베딩을 계산하는 방식은 GCN(Graph Convolutional Network), GIN(Graph Isomorphism Network), GAN(Graph Attention Network) 등 알고리즘별로 상이하다.

[0009] 또한, GNN은 생성된 노드 임베딩을 기반으로 그래프 전체를 축약하는 고정크기 벡터(fixed-sized vector)인 "그래프 임베딩"을 생성한다. 그래프 임베딩을 생성하는 과정을 소위 그래프 리드아웃(graph readout)이라고 부르며, 노드 임베딩에 대한 합계, 평균, 정렬 또는 또 다른 신경망을 이용하는 등 다양한 방법이 존재한다. 생성된 그래프 임베딩이 가변적인 사이즈의 그래프를 축약하는 고정 크기의 벡터이기에, 이 값을 또 다른 머신러닝 알고리즘, 가령 다층 퍼셉트론(Multilayer Perceptron, MLP) 등을 거치게 하여 의도한 예측을 수행한다.

[0010] 분산 딥러닝은 다수의 노드들을 요구하기 때문에, Google Cloud, Amazon AWS, Microsoft Azure와 같은 퍼블릭 클라우드(Public cloud) 또는 각 기업이나 연구소의 on-premise GPU cluster에서 수행된다. 이때, 가장 큰 문제는 분산 딥러닝의 학습 시 필요한 최적의 컴퓨팅 자원의 양과 학습 시간을 알 수 없다는 것이다. 이로 인해, 딥러닝 모델 개발자들은 임시방편적인 시행착오(trial and error) 또는 애드-혹 방법(ad-hoc methods)을 사용하여 자원을 요청(resource provisioning)한다. 1) 컴퓨팅 자원 중 GPU는 매우 고가임을 고려할 때, 그리고 2) 수 시간에서 수 일까지 걸리는 긴 학습 시간을 고려할 때, 실제 연산에 필요한 최적의 자원양을 예측하지 못하고, 수차례 반복해야만 하는 점은 매우 비효율적이다.

[0011] 자원 소모량 예측의 어려움은 우선, 분산 딥러닝의 구동 환경(예컨대, DT(Distributed Training) 세팅)의 복잡성에 있다. 예를 들어, 동일한 모델의 학습이 수 많은 GPU(예컨대, 2080 Ti, Titan RTX, V100, A100 등)를 통해 가능하다. 또한, PS 및 워커의 수를 무궁무진한 조합으로 구성할 수 있다. 다음으로, 워커들은 동일 서버에 배치되어 PCI를 통해 네트워크 통신을 수행할 수도 있고, 서로 다른 물리 서버에 배치되어 40GbE 또는 고속 RDMA를 통해 통신할 수도 있다. 이러한 환경 구성의 다양성은 자원 소모량 예측의 복잡도를 증가시킨다.

[0012] 뿐만 아니라, 분산 딥러닝 학습의 대상이 되는 워크로드 자체(DT 워크로드)도 예측을 복잡하게 만든다. 자원 소모량은 1) 어떤 모델을 학습시키는지, 2) 어떤 데이터셋을 사용하는지, 3) 어떤 하이퍼파라미터를 활용하는지에

따라 그 양상이 매우 달라진다.

[0013] 분산 딥러닝 학습에서의 자원을 전혀 알 수 없기에 발생하는 심각한 문제점들을 해결하고자, 자원 사용량을 미리 예상하려는 여러 시도가 존재했다. Justus et al.은 컨볼루션 레이어와 같이 딥러닝 모델에서 일반적으로 사용되는 레이어의 실행 시간을 예측했다(Daniel Justus, John Brennan, Stephen Bonner, and Andrew Stephen McGough. 2018. Predicting the computational cost of deep learning models. In 2018 IEEE international conference on big data (Big Data). IEEE, 3873-3882.). Habitat은 GPU에서 단일 이터레이션(iteration)을 실행하는 시간을 측정하고 사전 훈련된 다층 퍼셉트론을 통해 다른 GPU의 시간을 예측했다(X Yu Geoffrey, Yubo Gao, Pavel Golikov, and Gennady Pekhimenko. 2021. Habitat: A Runtime-Based Computational Performance Predictor for Deep Neural Network Training. In 2021 USENIX Annual Technical Conference (USENIX ATC21). 503-521.). Lin et al.은 GPU 및 CPU 사용률을 프로파일링하고 하나의 미니 배치에 대한 이터레이션 시간을 계산했다(Zheyu Lin, Xukun Chen, Hanyu Zhao, Yunteng Luan, Zhi Yang, and Yafei Dai. 2020. A Topology-Aware Performance Prediction Model for Distributed Deep Learning on GPU Clusters. In 2020 IEEE International Conference on Big Data (Big Data). IEEE, 2795-2801.).

[0014] 이러한 시도들은, 분산 딥러닝 학습이 이터레이션(1회의 순방향 전파 및 역방향 전파)을 반복하고, 이터레이션 동안은 자원 소모의 방식이 유사하다는 것을 기반으로 한다. 그러나, 핵심은 DT 세팅과 DT 워크로드의 다양성을 고려하지 못하여, 새로운 모델, 새로운 데이터셋, 새로운 하이퍼파라미터가 사용될 경우, 활용될 수 없다. 뿐만 아니라, 상이한 GPU, 상이한 PS와 워커의 수, 상이한 네트워크 연결 종류에 적용될 수 없다. 즉, 다양한 설정과 작업 부하를 함께 수용하는 데 한계가 있다.

[0015] 이에, 본 발명에서는 다양한 DT 워크로드에 대한 자원 소모의 양과 시간에 대한 예측을 가능하게 하면서도, 다양한 DT 세팅에 적용할 수 있는 예측 기법을 제안하고자 한다. 보다 구체적으로, 본 발명은 그래프 신경망을 기반으로 주어진 DT 세팅에 대해 다양한 DT 워크로드에 대한 자원 소모량을 정확하게 예측하는 기법을 제안한다. 더 나아가, 전이 학습(Transfer Learning, TL)을 활용하여 서로 다른 DT 세팅을 위한 예측 모델을 적은 부하로 훈련하는 기법을 제안한다.

[0016] 본 발명의 차별성은 다음과 같다.

[0017] 1) 분산 딥러닝에서 자원 소비 예측을 위해, GNN을 사용하여 무한히 다양한 DT 워크로드 자체를 예측의 입력으로 사용하는 첫 번째 접근이다.

[0018] 2) 800종 이상의 대표적인 딥러닝 모델을 상세하게 분석(프로파일링)하여, 예측에 적합한 자원 소모 측정 지표인 ① 자원 사용량, ② 점유 시간, 및 ③ 유휴 시간을 제안한다.

[0019] 3) 분산 딥러닝 시스템에서 핵심적으로 고려되는 4가지 핵심 리소스(예: GPU 활용률, GPU 메모리 활용률, 네트워크 TX 처리량, 및 네트워크 RX 처리량)에 대해, 자원 사용량, 점유 시간 및 유휴 시간을 예측하여 총 12가지 값을 예측해 낸다.

[0020] 4) 전이 학습을 이용하여, 학습 시간 및 학습에 필요한 데이터셋의 크기를 2.5배, 7.3배 개선하면서도 높은 예측 정확도를 유지하였다.

선행기술문헌

특허문헌

[0021] (특허문헌 0001) 대한민국 공개특허 제2020-0097623호 (2020년 8월 19일 공개)

(특허문헌 0002) 대한민국 공개특허 제2020-0063943호 (2020년 6월 5일 공개)

(특허문헌 0003) 대한민국 등록특허 제1852610호 (2018년 4월 27일 공고)

발명의 내용

해결하려는 과제

[0022] 본 발명이 이루고자 하는 기술적인 과제는 분산 딥러닝의 학습에 필요한 자원 소모량과 그 자원들을 실 점유하

는 점유 시간과 유휴하는 유휴 시간을 예측하는 방법 및 장치를 제공하는 것이다.

과제의 해결 수단

[0023] 본 발명의 일 실시예에 따른 예측 모델 생성 방법은 적어도 프로세서를 포함하는 컴퓨팅 장치에 의해 수행되는 예측 모델 생성 방법으로서, 학습 데이터셋을 구축하는 단계, 및 GNN(Graph Neural Network, 그래프 신경망) 모델을 학습하여 예측 모델을 생성하는 단계를 포함하고, 상기 학습 데이터셋은 입력 데이터와 결과 데이터를 포함하고, 상기 학습 데이터셋을 구축하는 단계는 분산 딥러닝 학습 코드(DT 코드, Distributed Training code)를 그래프로 변환하는 단계, 및 상기 그래프로부터 인접 행렬(adjacency matrix)과 특징 행렬(feature matrix)을 추출하는 단계를 포함한다.

[0024] 또한, 상기 결과 데이터는 GPU 활용률, GPU 메모리 활용률, 네트워크 TX 처리량, 네트워크 RX 처리량, GPU의 점유 시간, GPU 메모리의 점유 시간, 네트워크 TX의 점유 시간, 네트워크 RX의 점유 시간, GPU의 유휴 시간, GPU 메모리의 유휴 시간, 네트워크 TX의 유휴 시간, 및 네트워크 RX의 유휴 시간 중 적어도 하나를 포함할 수 있다.

[0025] 또한, 상기 GNN은 GCN(Graph Convolutional Network), GIN(Graph Isomorphism Network), 또는 GAN(Graph Attention Network)으로 구현될 수 있다.

[0026] 또한, 상기 GNN은 복수의 그래프 레이어들, 그래프 리드아웃 레이어, 및 MLP(Multi-Layer Perceptron) 레이어를 포함할 수 있다.

[0027] 또한, 상기 그래프 레이어들 각각은 GRU(Gated Recurrent Unit)를 포함할 수 있다.

[0028] 또한, 상기 예측 모델을 생성하는 단계 이후에, 상기 예측 모델에 대한 전이 학습(Transfer Learning, TL)을 수행하는 단계가 더 포함될 수도 있다.

[0029] 또한, 상기 전이 학습을 수행하는 단계는 제2 학습 데이터셋을 이용하여 수행되고, 상기 학습 데이터셋은 제1 DT 세팅(Distributed Training setting)에 대응하는 데이터셋이고, 상기 제2 학습 데이터셋은 제2 DT 세팅에 대응하는 데이터셋이고, 상기 제1 DT 세팅과 상기 제2 DT 세팅은 분산 딥러닝을 수행하는 GPU의 종류, 파라미터 서버(Parameter Server, PS)의 개수, 및 워커 노드(worker node)의 개수 중 적어도 하나가 상이할 수 있다.

[0030] 또한, 상기 전이 학습은 상기 예측 모델에 포함되는 복수의 그래프 레이어들 중 적어도 일부의 그래프 레이어의 파라미터들과 상기 예측 모델에 포함되는 MLP 레이어의 파라미터들 중 적어도 하나만을 업데이트할 수 있다.

[0031] 또한, 상기 전이 학습은 상기 예측 모델에 포함되는 복수의 그래프 레이어들 중 후반부의 절반의 레이어들의 파라미터들과 상기 MLP 레이어의 파라미터들만을 업데이트할 수 있다.

[0032] 본 발명의 일 실시예에 따른 예측 방법은 상기의 예측 모델 생성 방법에 의하여 생성된 예측 모델을 이용한 예측 방법으로서, 예측 대상 입력 데이터를 생성하는 단계, 및 상기 예측 대상 입력 데이터를 상기 예측 모델에 입력하여 예측을 수행하는 단계를 포함한다.

발명의 효과

[0033] 본 발명의 실시예에 따른 예측 모델 생성 방법 및 예측 방법에 의할 경우, 분산 딥러닝의 학습에 필요한 학습 시간 및 자원 소모량에 대한 정보를 용이하게 예측할 수 있다.

도면의 간단한 설명

[0034] 본 발명의 상세한 설명에서 인용되는 도면을 보다 충분히 이해하기 위하여 각 도면의 상세한 설명이 제공된다.

도 1a와 도 1b는 상이한 입력 타입에 대한 머신 러닝을 설명하기 위한 도면으로써, 도 1a는 이미지에 적용되는 CNN의 일 예를, 도 1b는 그래프에 적용되는 GNN의 일 예를 도시한다.

도 2a, 도 2b, 및 도 2c는 각각 워크로드 1에 대한 네트워크 RX 처리량(T_{NR}), 워크로드 2에 대한 GPU 활용률(U_G), 및 워크로드 3에 대한 GPU 메모리 활용률(U_{GM})을 도시한다.

도 3은 본 발명이 제안하는 예측 기법을 활용한 예측 과정을 설명하기 위한 개념도이다.

도 4는 VGG16 모델에 대한 입력 구축기의 동작을 설명하기 위한 도면이다.

도 5는 학습 데이터셋의 생성 과정을 설명하기 위한 도면이다.

도 6a는 시나리오별 예측 오차를 나타내는 그래프이다.

도 6b는 시나리오별 학습 시간을 나타내는 그래프이다.

도 7은 본 발명의 일 실시예에 따른 학습 데이터셋 생성 방법을 설명하기 위한 흐름도이다.

도 8은 본 발명의 일 실시예에 따른 예측 모델 생성 방법을 설명하기 위한 흐름도이다.

도 9는 본 발명의 일 실시예에 따른 예측 방법을 설명하기 위한 흐름도이다.

발명을 실시하기 위한 구체적인 내용

[0035] 본 명세서에 개시되어 있는 본 발명의 개념에 따른 실시예들에 대해서 특정한 구조적 또는 기능적 설명들은 단지 본 발명의 개념에 따른 실시예들을 설명하기 위한 목적으로 예시된 것으로서, 본 발명의 개념에 따른 실시예들은 다양한 형태들로 실시될 수 있으며 본 명세서에 설명된 실시예들에 한정되지 않는다.

[0036] 본 발명의 개념에 따른 실시예들은 다양한 변경들을 가할 수 있고 여러 가지 형태들을 가질 수 있으므로 실시예들을 도면에 예시하고 본 명세서에서 상세하게 설명하고자 한다. 그러나, 이는 본 발명의 개념에 따른 실시예들을 특정한 개시 형태들에 대해 한정하려는 것이 아니며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물, 또는 대체물을 포함한다.

[0037] 제1 또는 제2 등의 용어는 다양한 구성 요소들을 설명하는데 사용될 수 있지만, 상기 구성 요소들은 상기 용어들에 의해 한정되어서는 안 된다. 상기 용어들은 하나의 구성 요소를 다른 구성 요소로부터 구별하는 목적으로만, 예컨대 본 발명의 개념에 따른 권리 범위로부터 벗어나지 않은 채, 제1 구성 요소는 제2 구성 요소로 명명될 수 있고 유사하게 제2 구성 요소는 제1 구성 요소로도 명명될 수 있다.

[0038] 어떤 구성 요소가 다른 구성 요소에 "연결되어" 있다거나 "접속되어" 있다고 언급된 때에는, 그 다른 구성 요소에 직접적으로 연결되어 있거나 또는 접속되어 있을 수도 있지만, 중간에 다른 구성 요소가 존재할 수도 있다고 이해되어야 할 것이다. 반면에, 어떤 구성 요소가 다른 구성 요소에 "직접 연결되어" 있다거나 "직접 접속되어" 있다고 언급된 때에는 중간에 다른 구성 요소가 존재하지 않는 것으로 이해되어야 할 것이다. 구성 요소들 간의 관계를 설명하는 다른 표현들, 즉 "~사이에"와 "바로 ~사이에" 또는 "~에 이웃하는"과 "~에 직접 이웃하는" 등도 마찬가지로 해석되어야 한다.

[0039] 본 명세서에서 사용한 용어는 단지 특정한 실시예를 설명하기 위해 사용된 것으로서, 본 발명을 한정하려는 의도가 아니다. 단수의 표현은 문맥상 명백하게 다르게 뜻하지 않는 한, 복수의 표현을 포함한다. 본 명세서에서, "포함하다" 또는 "가지다" 등의 용어는 본 명세서에 기재된 특징, 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것이 존재함을 지정하려는 것이지, 하나 또는 그 이상의 다른 특징들이나 숫자, 단계, 동작, 구성 요소, 부분품 또는 이들을 조합한 것들의 존재 또는 부가 가능성을 미리 배제하지 않는 것으로 이해되어야 한다.

[0040] 다르게 정의되지 않는 한, 기술적이거나 과학적인 용어를 포함해서 여기서 사용되는 모든 용어들은 본 발명이 속하는 기술 분야에서 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.

[0041] 이하, 본 명세서에 첨부된 도면들을 참조하여 본 발명의 실시예들을 상세히 설명한다. 그러나, 특허출원의 범위가 이러한 실시예들에 의해 제한되거나 한정되는 것은 아니다. 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.

[0042] 도 1A는 이미지에 적용되는 CNN의 일 예를 도시한다.

[0043] 고정된 크기의 이미지, 즉 고정된 개수의 픽셀들을 갖는 이미지가 CNN(Convolutional Neural Network, 콘볼루션 신경망)에 입력된다. CNN은 고정된 개수의 픽셀들로부터 정보를 집계(aggreagte, filter로 명명될 수도 있음)하고, 이미지의 섹션을 나타내는 벡터 값(convolved feature)을 출력한다. 예시적으로, 도 1A에서, 이미지의 오른쪽 아래 코너(lower-right corner)로부터 9개의 픽셀들의 필터는 하나의 convolved feature를 도출한다. 입력

과 필터 사이즈는 고정되어 있기 때문에, 각 입력 이미지로부터 동일한 개수의 convolved features가 도출된다.

[0044] 반면에, GNN에서, 그래프의 정보는 기본적으로 노드별로 집계된다(aggregated). 이와 같은 집계의 결과는 노드 임베딩(node embedding)을 의미할 수 있다. 구체적으로, 노드 임베딩은 특정 홉 거리 내의 이웃 노드들의 특징의 집계로 계산된다. 도 1B에서, 1 홉 거리 내에, 노드 1은 4개의 이웃 노드들을, 노드 2는 2개의 이웃 노드들을, 노드 3은 3개의 이웃 노드들을 갖는다. GNN의 노드 임베딩은 상이한 개수의 이웃 노드들을 집계할 수 있다. 따라서, 각 GNN 알고리즘은 노드 임베딩을 위한 상이한 기법들을 수행한다. 또한, CNN의 입력 이미지와는 달리, GNN은 상이한 개수의 노드들을 포함하는 그래프를 입력받는다. 다양한 크기를 수용하기 위하여, GNN은 그래프를 나타내는 고정된 크기의 벡터인 그래프 임베딩(graph embedding)을 생성한다.

[0045] 또한, GNN은 레이어의 개수와 같은 새로운 파라미터들을 포함한다. GNN의 각 레이어는 단일 홉 내의 인접 노드들(nearby nodes)로부터의 정보를 집계함으로써 노드 임베딩을 생성한다. 또한, 노드 임베딩은 n 개의 레이어들을 쌓음으로서(stacking) n 홉 떨어진 이웃 노드들을 집속할 수 있다. 사용되는 레이어의 개수는 GNN의 사용예에 따라 달라진다.

[0046] MLP(Multi-Layer Perceptron)와 같은 전통적인 머신 러닝 알고리즘과의 조합으로 예측을 수행하기 위해 그래프 임베딩이 사용될 수 있다. 본 발명에서는, 다양한 DT 세팅들에 대한 DT 워크로드들의 자원 소모량을 예측하기 위하여, GNN에 기반한 예측 모델을 디자인하였다. 기존의 딥러닝 알고리즘들과 유사하게, GNN은 반복적인 이터레이션들(순방향과 역방향 전파)에 의해 학습된다.

[0047] 이하에서는, 본 명세서에서 이용되는 용어에 대한 설명을 기재한다.

[0048] · 딥러닝 모델 학습: 여러 개의 레이어 및 각 레이어에 구성된 모델을 구성하는 파라미터가, 주어진 데이터셋의 입력값에 대해 예측값을 가장 잘 예측해 내는 값으로 수렴될 때까지 모델을 갱신해 나가는 과정

[0049] · 이터레이션(iteration): 데이터셋의 일부(batch)를 기반으로, 1회의 순방향 전파와 역방향 전파를 수행하는 과정

[0050] · 순방향 전파(순전파): 딥러닝 모델의 입력 레이어로부터 최종 출력 레이어까지 차례대로 레이어를 거치며 변수들을 계산하고 저장해 나가는 과정

[0051] · 역방향 전파(역전파): 순전파 결과로 출력된 예측값과 실측값(ground truth)과의 차이를 기반으로, 최종 출력 레이어에서부터 입력 레이어까지 각 레이어 파라미터들의 그라디어언트를 역순으로 계산해 나가는 과정

[0052] 다음으로, 예측지표의 정의를 설명한다.

[0053] 예측 지표를 정의하기 위해, 대표적인 이미지 분류 및 자연어 처리 모델, 데이터셋, 하이퍼파라미터를 조합하여 약 832종의 DT 워크로드를 구성한다. 구체적으로, 워크로드들은 다음의 표 1로부터 도출된다.

[0054] [표 1]

Model type	Model	Dataset	Hyperparameters
Image classification	AlexNet, GoogLeNet (Inception v1), Inception v3, Inception v4, ResNet101, ResNet101_v2, ResNet152, ResNet20, ResNet20_v2, ResNet32, ResNet32_v2, ResNet44, ResNet44_v2, ResNet50, ResNet50_v2, ResNet56, ResNet56_v2, ResNet110, ResNet110_v2, ResNet152, ResNet152_v2, VGG11, VGG16, VGG19, Overfeat, DenseNet100_k12, DenseNet100_k24, DenseNet40_k12	CIFAR-10, ImageNet	Batch size, parameter precision (floating-point), optimizer, data format, synchronization method
Natural language processing	NMT_Big, NMT_Medium, NMT_Small, Transformer, Transformer_AAN, Transformer_Big	Europarl	

[0055]

[0056] 우선, 각 DT 워크로드를 PS 1개와 워커 2개(각각 V100 GPU 사용)로 학습시킨다. 학습이 진행되는 동안, 1) GPU 활용률(GPU utilization, U_G), 2) GPU 메모리 활용률(GPU memory utilization, U_{GM}), 3) 네트워크 TX 처리량(network TX throughput, T_{NT}), 및 4) 네트워크 RX 처리량(network RX throughput, T_{NR})을 측정한다. 상술한 네 가지 자원들은 관련 연구들에서 DT 시스템을 최적화하는 데 필수적이라 간주된 지표들이다. 100회의 이터레이션 동안 측정이 수행되었다. 측정 결과는 832 워크로드들에 걸쳐 유사한 패턴을 보이기에, 세 가지 예시 워크로드를 중심으로 설명한다.

[0057] · 워크로드 1: NMT_Medium 모델, Europar 데이터셋 (배치 크기 32, 비동기)

[0058] · 워크로드 2: DenseNet40_k12 모델, CIFAR-10 데이터셋 (배치 크기 512, 동기)

[0059] · 워크로드 3: Inception v3 모델, ImageNet 데이터셋 (배치 크기 128, 비동기)

[0060] 도 2A, 도 2B, 및 도 2C에는 각각 워크로드 1에 대한 네트워크 RX 처리량(T_{NR}), 워크로드 2에 대한 GPU 활용률(U_G), 및 워크로드 3에 대한 GPU 메모리 활용률(U_{GM})이 도시되어 있다. 각 그래프의 x축은 시간을, y축은 리소스 사용량을 나타낸다. 세 개의 그래프 모두 리소스 소비량이 많은 데이터 지점과 리소스 소비량이 낮은 데이터 지점이 반복되는 패턴을 보인다.

[0061] 따라서, 본 발명에서는 이러한 패턴을 정량화하여, 자원 소모량의 측정 지표를 정의한다. 리소스 소비가 높은 데이터 지점을 점유 지점(burst points)이라 하고, 리소스 소비가 낮은 데이터 지점을 유휴 지점(idle points)이라고 한다. 이를 x축과 y축을 기준으로 정량화할 수 있다. 이때, 미리 정해진 임계치가 활용될 수 있다. 즉, 임계치를 초과(또는 이상)하는 소비량을 보이는 지점은 점유 지점으로, 임계치 미만의 소비량을 보이는 지점을 유휴 지점이라 할 수 있다. 임계치는 사용자나 관리자에 의해 미리 정해진 값을 가질 수 있고, 실시예에 따라 가변적일 수 있다. 또한, 점유 사용량은 자원 점유 지점들에서 사용되는 평균 리소스 양을 의미할 수 있고, 유휴 시간은 유휴 지점들이 연속적으로 나타나는 평균 시간을 의미할 수 있다.

[0062] 표 2는, 도 2A, 도 2B, 및 도 2C에 도시된, 정의된 지표들에 대한 정량화된 값들(amounts and durations of burst and idle data points)을 보여준다. 점유 지점과 유휴 지점 사이에서, 리소스 사용량은 큰 차이를 보인다. T_{NR}의 경우, 처리량은 83배 차이가 난다. 또한, T_{NR}이 이용되면, 평균적으로 0.33초 동안 소비된다. 역으로, T_{NR}의 유휴 시간은 약 0.7초로, 점유 시간보다 약 2.12배 길다. 또한, U_G와 U_{GM}에서 중요한 차이점이 관찰되었다.

[0063] [표 2]

[0064]

	T_{NR} (workload 1)	U_G (workload 2)	U_{GM} (workload 3)
Burst	374.3 MB/s, 0.33 s	98.8%, 0.19 s	54.4%, 0.64 s
Idle	4.5 MB/s, 0.7 s	84.6%, 0.21 s	16.8%, 0.28 s

[0065] 도 3은 본 발명이 제안하는 예측 기법(Driple이라 명명함)을 활용한 예측 과정을 설명하기 위한 개념도이다. 예측 과정은 크게 입력 구축기(도 3의 Input builder)와 예측기(도 3의 Driple inspector)에 의해 수행된다.

[0066] 입력 구축기는 DT 워크로드의 코드로부터 계산 그래프(computational graph)를 추출하고 인접 행렬(Adjacency matrix)과 특징 행렬(Feature matrix)을 구축할 수 있다. 예측기는 생성된 인접 행렬과 특징 행렬을 입력으로 수신하고, 지표들(예컨대, 12 가지의 지표들)에 대한 예측을 수행한다. 다시 말하면, 예측기는 학습 데이터셋을 이용하여 GNN 모델을 학습함으로써 예측 모델을 생성하고, 생성된 예측 모델을 이용하여 입력 데이터에 대한 예측값을 생성할 수 있다. 실시예에 따라, 입력 구축기는 입력부, 입력 유닛, 또는 입력 생성기 등으로 명명될 수 있고, 예측기는 예측부 또는 예측 유닛, 예측 모델 생성부, 예측 모델 생성 유닛, 학습부, 학습 유닛 등으로 명명될 수도 있다.

[0067] 또한, 입력 구축기, 예측기, 및 측정기는 각각 기능 및 논리적으로 분리될 수 있음을 나타내는 것이며, 반드시 각각의 구성이 별도의 물리적 장치로 구분되거나 별도의 코드로 작성됨을 의미하는 것은 아니다. 또한, 입력 구축기, 예측기, 및 측정기는 본 발명의 기술적 사상을 수행하기 위한 하드웨어 및 상기 하드웨어를 구동하기 위한 소프트웨어의 기능적, 구조적 결합을 의미할 수 있다. 예컨대, 상기 구성들은 소정의 코드와 상기 소정의 코드가 수행되기 위한 하드웨어 리소스의 논리적인 단위를 의미할 수 있으며, 반드시 물리적으로 연결된 코드를 의미하거나, 한 종류의 하드웨어를 의미하는 것은 아니다.

[0068] 본 발명의 핵심은, 분산 딥러닝 학습에 사용되는 딥러닝 라이브러리들이 DT 워크로드 자체를 "그래프"로 변환한다는 점이다. TensorFlow, PyTorch, Caffe, MXNet 등 현존하는 대부분의 라이브러리는 모두 학습과정의 연산, 변수, 및 상수를 그래프의 노드로, 연산과정을 간선으로 표현하는 그래프를 생성한다.

[0069] 본 명세서에서, 계산 그래프는 G = (N,E)로 나타낸다. 여기서, N은 개별 노드들의 집합(set of nodes)을, E는 간선들의 집합(set of edges)을 나타낸다. 개별 노드 n은 노드의 특징들(characteristics)과 정보(information)를 표현하기 위한 노드 특징(node feature) X_n을 갖는다. 그래프의 i번째 노드는 n_i($n_i \in N$)로 표현된다. 모델의 각 레이어는 "op"로 불리는 다중 저-수준 연산들(multiple low-level operations, 예컨대 Add and MatMul)로 변환된다. 이러한 op는 n이 된다. 또한, 데이터셋은 이터레이션을 시작하기 위한 데이터를 로딩하는 op(예컨대, VariableV2 in TensorFlow)로 표현되기 때문에, 또 다른 n이 된다. 각 op에 필요

한 변수들(variables)과 상수들(constants), 예컨대 Const 또한 n이 된다. 각 n은 상이한 특징들 X_n(예컨대, 노드 타입)을 갖는다. 노드들은 연산 종류(computation, 예컨대 MatMul), 데이터셋, 및 변수들과 같이 상이한 타입을 갖기 때문이다.

[0070] 또한, DT 워크로드의 하이퍼파라미터들은 노드들 또는 노드들의 특징(node feature)으로 변환된다. 예컨대, 옵티마이저가 변경될 때(예컨대, RMSProp and Momentum), G 내의 옵티마이저 노드들은 새로운 옵티마이저 노드들로 교체된다. 또한, 배치 사이즈(batch size)는 피딩되는(fed) 데이터셋의 양을 결정한다. 배치 사이즈는 n의 X_n로 반영된다. 특히, ML 라이브러리는 배치 사이즈를 VariableV2의 "tensor size" 특징으로 설정한다. 다른 예로, 표 1의 parameter precision은 부동 소수점 연산 타입(type of floating-point operations)을 나타낸다. 또한, 16-비트 부동 소수점 연산이 이용될 때, TensorFlow 라이브러리는 연관된 n의 X_n을 변경한다(예컨대, DT_HALF). 다른 파라미터들도 유사하게 G에 반영된다.

[0071] 본 명세서에서 n_i와 n_j를 연결하는 간선은 e_{ij}로 표현한다. 간선들은 op들의 실행 순서를 결정한다(예컨대, 이터레이션을 위한 데이터를 로딩하는 VariableV2 op 부터 마지막 레이어의 op들까지). DT 코드로부터 G가 생성되면, 딥러닝 라이브러리는 G를 장치(예컨대, GPU)에 위치시키고, 첫번째 n으로부터 시작하여, 간선들에 따른 학습을 수행한다. G의 각 노드 n에 대하여, 단일의 간선(single edge)을 통해 n에 연결되는 노드들은 이웃으로 불린다. 여기서, $\mathcal{N}(n)$은 이웃들의 집합이다.

[0072] 입력 구축기는 인접 행렬과 특징 행렬을 생성한다. 인접 행렬은 각 그래프의 노드들과 간선들을 나타내기 위한 것이다. 인접 행렬로 표현되기 위해서, 그래프 내의 노드들은 넘버링된다. G = (N,E)가 주어졌을 때, 인접 행렬의 크기는 m×m(m은 |N|)이다. 인접 행렬의 i번째 행과 j번째 열의 요소(element)는, n_i로부터 n_j로의 간선이 존재하는 경우 1의 값을 갖고, 간선이 존재하지 않으면 0의 값을 갖는다. 따라서, 인접 행렬은 1) 노드들의 존재와 2) 노드들간의 연결성(즉, 간선)에 대한 정보만을 포함한다.

[0073] 특징 행렬은 텐서 사이즈(tensor size)와 노드 타입(node type)과 같은 X_n을 포함한다. G 내의 노드들은 노드에 따라 상이한 타입들(ops)과 상이한 개수의 특징들을 갖는다. 예컨대, VariableV2의 X_n은 텐서 사이즈를 가지나, Conv2D는 그렇지 않다. 또한, 모든 노드들은 X_n으로써 노드 타입을 갖는다. 그러나, GNN이 그래프들을 학습하기 위해서는, 각 노드는 동일한 특징들을 가져야 한다. 또한, 특징들의 개수는 너무 크지 않아야 한다. 특징들의 개수가 학습 시간과 예측 정확도에 영향을 미치기 때문이다. X_n 내의 특징의 개수가 f라면, 특징 행렬은 m×f의 크기를 갖는다.

[0074] X_n에 포함될 특징들을 결정하기 위해, 종래 GNN에 관한 연구에서 사용된 특징들을 체크하였다. 종래 연구들은, GPU들 사이에 그래프 파티션들을 배치하는 것과 그래프 최적화(컴파일러)와 같은 딥러닝 모델의 최적화를 위해 GNN을 이용하였다. 그러나, 이러한 연구들은 DT 워크로드들의 자원 소모량을 예측하지 않는다.

[0075] 특징들은 두 가지의 카테고리, 즉 1) 개별 노드들과 연관된 특징들과 2) 그래프의 내부 상태와 연관된 특징들(노드들 사이의 데이터 천이를 포함)로 분류된다. 본 발명에서는 다음과 같이 두 개의 특징들을 선택하였다.

[0076] · 노드 타입(Node type): 노드 타입의 예는 Conv2D와 L2Loss를 포함한다. 따라서, G 내의 노드 타입은 텍스트 값(text value)으로 주어진다. 노드 타입을 입력 특징으로 취하기 위하여, 타입 값은 숫자로 변환될 수 있다. 보다 상세하게, 데이터셋에 나타나는 노드 타입의 프리퀀시를 설정하는 프리퀀시 인코딩(frequency encoding)이 이용될 수 있다.

[0077] · 텐서 사이즈(Tensor size): 텐서 사이즈는 DT에서 계산과 통신을 위해 중요하다. X_n으로써 텐서 사이즈를 갖지 않는 n에 대하여, 텐서 사이즈로 0을 할당할 수 있다.

[0078] 상술한 두 가지의 특징들에 더해, "그룹핑" 메커니즘을 반영하는 다른 특징을 제안한다. 다만, 본 발명에서 그룹핑 동작은 선택적인 구성으로, 반드시 채용되어야 하는 구성은 아니다. 그룹핑 메커니즘은 그래프 노드들을 더 적은 개수의 노드들을 갖는 그래프로 그룹핑하는 것을 의미하고, 이는 학습 효율을 향상시킨다. 이러한 특징은 "그룹핑된 노드 사이즈(grouped node size)"로 불리며, 이는 얼마나 많은 노드들이 해당 노드로 그룹핑되는지를 나타낸다.

[0079] 그룹핑의 목적은 두 가지이다. 첫째, 그룹핑은 G의 사이즈(|N| 및 |E|)를 줄이는 것을 목적으로 한다. 표 1의

DT 워크로드들에 대한 그래프들은 수천개의 노드와 간선(최대 노드와 간선의 개수는 각각 18,758개와 28,276개(트랜스포머 모델)이며, 평균적으로 2,923개와 3,982개임)을 포함한다. 많은 노드와 간선은 GNN의 예측 정확도를 감소시킬 수 있다. 또한, 간선들의 개수가 많기 때문에, 노드 임베딩을 계산하는 데 많은 시간이 필요하다. 이에 따라, 전체 학습 시간은 지수적으로 증가하는 경향을 갖는다.

[0080] 둘째, 그룹핑은 그래프를 학습하는 과정에서 학습 속도에 중요한 배칭(batching)을 가능케 한다. 배칭은 GNN이 개별 입력 데이터(단일의 그래프)에 대해 업데이트되는 것이 아닌 배치로 불리는 데이터의 집합에 의해 업데이트됨을 의미한다. 예컨대, 배치 사이즈 10은 모델 파라미터들이 10개의 그래프들이 GNN 모델을 투과한 후 업데이트됨을 의미한다. 여기서, 문제는 그래프의 노드의 개수 m이 변함에 따라, GNN 모델 내의 레이어의 개수가 달라진다는 것이다. 따라서, 배칭을 위해, m은 배치 내의 각 그래프마다 동일하여야 한다. 명백히, 입력 그래프들은 상이한 m을 갖고, 이로 인해 배칭 작업을 수행하는 것은 어려운 문제이다.

[0081] 본 발명에서는 두 가지의 그룹핑 정책(uniform and proportional)을 디자인하였다. 각 정책은 입력 그래프들을 배치들로 분할하고 "노드 그룹핑"의 스케일(scale)을 설정한다. M을 그래프 내에서 그룹핑된 노드들의 개수라고 하자(예컨대, 500개의 그룹핑된 노드들은 19,758개의 오리지널 노드들로부터 도출될 수 있다). 균일 그룹핑(uniform grouping)은 모든 배치들에서 M을 동일하게 설정한다.

[0082] 비례 그룹핑(proportional grouping)은 우선 입력 그래프들을 m의 오름 차순(ascending order)으로 정렬한다. 그런 다음, 그래프들은 m의 순으로 배치 사이즈로 분할된다. 따라서, 제1 배치는 적은 수의 노드들을 포함하는 그래프들을 포함하고, 이후의 배치들은 보다 큰 그래프들을 포함한다. 각 배치에 대해, 노드들의 평균 개수(v)가 계산되고, M을 $log_{10}v$ 노드로 설정한다(배치의 로딩에 필요한 GPU 메모리 사이즈를 고려하여 경험적으로 선택됨).

[0083] 입력 구축기는 M에 따라 노드 그룹핑을 수행한다. 노드 그룹핑은 fluid communities algorithm(Ferran Pares, Dario Garcia Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguade, Jesus Labarta, Ulises Cortes, and Toyotaro Suzumura. 2018. Fluid Communities: A Competitive, Scalable and Diverse Community Detection Algorithm. In Complex Networks & Their Applications VI, Chantal Cherifi, Hocine Cherifi, Marton Karsai, and Mirco Musolesi (Eds.). Springer International Publishing, Cham, 229-240.)에 기초할 수 있다. 상기 알고리즘은 그래프 내에서 M 시드 노드들(M seed nodes)을 선택하고, 시드 노드들 주위의 나머지 노드들을 그룹핑한다.

[0084] 이하에서는, 입력 구축기의 동작 예를 설명한다. 도 4는 VGG16 모델에 대한 입력 구축기의 동작을 보여준다. VGG16 모델의 전체 레이어들 중 7번째 콘볼루션 레이어(conv7)를 살펴본다. DT 워크로드 코드 내의 7번째 콘볼루션 레이어를 정의하는 라인(도 4의 (a) 참조)은 G 내의 멀티플 노드들을 생성한다. 순방향 전파에서 수학적 연산들을 위한 노드들은 7번째 콘볼루션 레이어의 파선 박스(dashed box)로 도시되어 있다(도 4의 (b) 참조). 다른 노드들은 변수들, 상수들, 및 이전 연산들(preceding operations)과 연관된다. 전체적으로, 7번째 콘볼루션 레이어에 대한 16개의 노드들이 생성된다. 그러면, 7번째 콘볼루션 레이어에서 수학적 연산들은 하나의 그룹(G1)으로 그룹핑된다(도 4의 (c) 참조). 유사하게, 다른 노드들도, G2 내지 G5와 같이 그룹들을 형성한다. 최종적으로, 입력 구축기는 예측기의 입력이 되는 인접 행렬과 특징 행렬을 생성한다(도 4의 (d) 참조).

[0085] 요컨대, 본 발명의 입력 구축기는 주어진 코드로부터 그래프를 추출하고, GNN이 입력받는 행렬의 형태로 변환한다. 변환 과정에서, GNN에서 사용할 노드의 특징값들의 선정과 그래프 규모의 조정(grouping) 등이 수행된다.

[0086] 본 발명의 예측기는 인접 행렬과 특징 행렬을 입력으로 받는 GNN을 기반으로 설계된다. 즉, 예측기는 학습 데이터셋을 이용하여 GNN 모델을 학습시켜 예측 모델을 생성하거나 생성된 예측 모델을 이용하여 예측 동작을 수행하는 구성으로 이해될 수 있다.

[0087] 예측기(또는 예측 모델)의 제1 파트는 다층 그래프 레이어들(multiple graph layers)이다. 그래프 레이어는 업데이트 함수 ϕ를 실행한다. 여기서, 제1 레이어를 ϕ_f로, 나머지 레이어들을 ϕ_r로 표기한다. 업데이트 함수들이 상이하기 때문이다.

[0088] 제1 레이어 ϕ_f는, 모든 n_i에 대하여 $\mathcal{N}(n_i)$에 속하는 n_j의 X_{n_j}를 집계하는(aggregating) 업데이트 함수를 실행한다. ϕ_f에 의해 집계된 값은 n_i의 노드 임베딩이 되고, 이는 $h^1_{n_i}$로 나타낸다. 따라서, 제1 레이어의 노드 임베딩은 수학식 1에 의해 생성될 수 있다.

[0089] [수학식 1]

[0090] $$h_{n_i}^1 = \phi_f = \text{AGGREGATE}^1(X_{n_j} | n_j \in \mathcal{N}(n_i)), i = 1, \ldots, m$$

[0091] 다음 레이어들(예컨대, k번째 레이어)은 n_i별로 두 가지의 업데이트 단계들(ϕ_r)을 수행한다. 즉, 1) ϕ_r은 이전의 (k-1)번째 레이어의, $\mathcal{N}(n_i)$에 속하는 n_j에 대한 노드 임베딩 $h_{n_j}^{k-1}$을 집계하고, 2) 집계된 노드 임베딩들은 n_i의 $h_{n_i}^{k-1}$와 결합된다. 결합의 방법은 GNN 알고리즘별로 상이하다. ϕ_r을 수행하는 복수의 레이어들이 존재하기 때문에, n_i에 대한 k번째 업데이트된 임베딩 값을 $h_{n_i}^k$로 나타낸다. 이는 수학식 2와 같다.

[0092] [수학식 2]

[0093] $$h_{n_i}^k = \phi_r = \text{COMBINE}^k(h_{n_i}^{k-1}, \text{AGGREGATE}^k(h_{n_j}^{k-1} | n_j \in \mathcal{N}(n_i))), i = 1, \ldots, m$$

[0094] 학습 데이터셋 내의 l번째 G가 G_l(G_l =(N_l, E_l) and |N_l|=m)이고, 값 K는 GNN 내의 레이어들의 개수라고 하자. K가 m(G_l 내의 노드들의 개수)으로 설정되면, 노드 임베딩들은 m번 집계된다. 따라서, m 홉 내의 이웃들의 X_n이 집계되고 결합된다. 보통, GNN은 GNN 레이어의 개수(K)를 m과 유사하거나 m의 절반으로 설정한다. 실시예에 의하면, 실험 결과 중 가장 높은 예측 정확도를 보인 m의 1/2로 K를 설정할 수 있다. 그러나, 본 발명의 권리범위가 K의 구체적인 값에 제한되는 것은 아니다. 가중치(weight) 및 바이어스(bias)와 같은 ϕ_r의 파라미터들은 (K-1) 개의 ϕ_r 레이어들에서 동일할 수 있다. 이는, 학습과 추론에서의 계산 비용을 감소시키기 위한 일반적인 디자인 선택이다.

[0095] 수학식 1과 수학식 2의 계산은 GNN 알고리즘에 의존적이다. 예컨대, GCN(graph convolution network)은 업데이트 함수의 집계를 위해 정규화된 평균(normalized mean)을 이용한다. 또한, GAT(graph attention network)는, X_n의 중요도(즉, 가중치)를 계산하는 어텐션 메커니즘(attention mechanism)에 의해 계산된 가중치들로, $h_{n_i}^k$에 대한 가중 합(weighted sum)을 채용한다. 본 발명에서는, 4개의 GNN 알고리즘들, 즉 GCN, GAT, GIN(graph isomorphism network), 및 MPNN(message passing neural network)를 테스트하였고, GCN의 예측 정화도가 가장 높음을 확인하였다. 따라서, 본 발명은 GCN을 이용할 수 있으나, 본 발명이 이제 제한되는 것은 아니다.

[0096] 추가적으로, 도 3에 도시된 바와 같이, 본 발명에서는 ϕ_f와 ϕ_r, 레이어들 각각의 뒷단에 GRUs(gated recurrent units)를 도입하였다. 따라서, 예측기의 제1 레이어(later 1)는 ϕ_f와 GRU의 쌍으로 구성된다. 이후의 레이어들은 ϕ_r와 GRU의 쌍으로 구성된다. 이는 레이어의 개수가 증가함에 따라 이전 레이어에서의 정보 손실에 의해 야기되는 over-smoothing 문제를 방지하기 위함이다.

[0097] $h_{n_i}^K$ 값들을 GRU에 통과시킨 후, 예측기는 G_l에 대한 그래프 임베딩 h_{G_l}을 생성한다. h_{G_l}은 그래프 리드아웃 레이어(ρ)에 의해 생성될 수 있다. ρ 레이어는 G_l 내의 N_l에 속하는 n_i의 $h_{n_i}^K$ 값들에 대한 풀링 함수(pooling function)를 실행한다. ρ 레이어는 수학식 3과 같이 표현된다.

[0098] [수학식 3]

[0099] $$h_{G_l} = \rho = \text{POOL}(h_{n_i}^K | n_i \in N_l)$$

[0100] ϕ_f 및 ϕ_r과 유사하게, 다양한 ρ 레이어들이 선택될 수 있다. 수학적 연산들(예컨대, 평균, 최대값, 또는 합)을 통해 주어진 $h_{n_i}^K$ 값을 h_{G_l}으로 변환하기 위한 간단한 풀링 함수들이 존재한다. 또한, 분류(sort)는 내림차순(또는 오름차순)에서 특정 개수의 $h_{n_i}^K$ 값들을 선택하고 선택된 값들을 최종 h_{G_l}로 반환하는 또다른 간단한 풀링 함수이다. 구현의 용이함과 빠른 학습때문에, 이러한 간단한 풀링 함수들이 널리 사용된다. 또한, ρ 레이어는 별도의 신경망으로 구현될 수도 있다. 예컨대, set2set은 LSTM 신경망을 사용한다. h_{G_l}을 생성하기 위해, set2set은 $h_{n_i}^K$ 값들을 시계열 데이터(time-series data)로 보고, $h_{n_i}^K$ 값들 사이의 중요성에 가중치를 부여한다. 이러한

선택들 중에서, 본 발명에서는 set2set을 이용한다. 그러나, 본 발명의 권리범위가 이에 제한되는 것은 아니다. 이러한 디자인 또는 선택은 예측 정확도에 대한 테스트 결과에 기인할 수 있다.

[0101] 마지막으로, 예측기는 ρ 레이어에 의해 생성된 h_{G_I}를 MLP로 전송한다. MLP와 관련하여, 본 발명에서는 측정된 예측 정확도에 따라, 3개의 완전 연결 레이어들(fully connected layers)을 선택한다. 그러나, 본 발명의 권리범위가 이에 제한되는 것은 아니다. MLP는 주어진 G_I에 대하여 12 가지의 타겟들에 대한 예측 결과를 생성한다.

[0102] 요컨대, 본 발명의 예측기는 GNN을 기반으로 설계된다. 첫 레이어(layer 1)부터 K번째 레이어는 GNN의 그래프 레이어이다. 그래프 레이어로는 GCN, GIN, GAT, MPNN 등이 이용될 수 있다. 테스트 결과로 가장 높은 정확도를 보이는 GCN을 본 발명에서는 선택하였지만, 본 발명의 권리범위가 이에 제한되는 것은 아니다. 또한, 각 그래프 레이어 뒤에 GRU 레이어가 추가될 수 있는데, 이는 레이어의 개수가 증가함에 따라 이전 레이어에서 학습한 정보의 손실 및/또는 over-smooting 문제를 방지하기 위함이다.

[0103] 또한, 그래프 임베딩을 생성하는 그래프 리드아웃 레이어로는, 합계, 평균, 정렬, set2set(LSTM 기반 신경망 사용) 등이 사용될 수 있다. 평가 결과로 가장 높은 정확도를 보이는 set2set을 본 발명에서는 사용하지만, 본 발명의 권리범위가 이에 제한되는 것은 아니다. 생성된 그래프 임베딩 값은 완전 연결 레이어(fully connected layer)에 전달되어 지표(예컨대, 12가지의 지표)가 예측될 수 있다.

[0104] 예측기를 학습시키기 위해서는 데이터셋이 필요하다. 그러나, 분산 딥러닝 과정에서의 다양한 모델 및 자원 소모량에 대한 데이터셋은 부재하다. 이에, 본 발명에서는 데이터셋을 직접 생성하는 기법을 제안한다.

[0105] 데이터셋은 입력 구축기와 측정기(도 5의 DriplePerf)를 통해 생성되며, 각 DT 세팅별로 구축된다. 입력 구축기는 앞서 설명된 것처럼, DT 코드로부터 모델의 입력값인 행렬을 생성한다. 측정기(측정부, 측정 유닛 등으로 명명될 수도 있음)는 주어진 DT 코드에 대한 학습이 진행될 때, GPU 활용률, GPU 메모리 활용률, 네트워크 TX 사용량, 네트워크 RX 사용량을 측정한다. 도 5에 도시된 바와 같이, 입력 특징들과 예측 타겟들은 DT 워크로드의 학습 데이터셋이 된다.

[0106] 입력 구축기로부터의 입력 특징들과 함께, 측정기는 DT 워크로드에 대한 출력 특징들을 생성한다. 출력 특징들은 DT 학습의 결과이고, 따라서 측정기는 첫째로 DT 코드를 실행하는 동안에 자원 소모량(예컨대, U_G, U_{GM}, T_{NT}, 및 T_{NR})을 측정한다. 측정 결과는 2D 좌표의 데이터 포인트들(측정 시간, 소모된 자원량)로 생성된다. 측정기는 측정 결과로부터 출력 특징들을 추출한다.

[0107] 측정기는 측정 데이터의 데이터 포인트들을 분할하기 위해 k-평균 클러스터링 알고리즘(k-means clustering algorithm)을 실행한다. 클러스터링 알고리즘은 각 데이터 포인트를 y축 값에 기초하여 분류하고, 이에 따라 데이터 포인트는 두 개의 파트(점유 및 유휴)로 분류된다. 그런 다음, 측정기는 다음과 같이 주어진 그래프의 출력 특징들을 계산한다. 첫째, 각 자원의 점유량은 점유 데이터 포인트의 소모된 자원량(y축)의 평균 값에 의해 계산된다. 점유 시간과 유휴 시간을 계산하기 위하여, 각 측정에서 나타나는 연속적인 점유 포인트와 유휴 포인트가 카운트될 수 있다. 연속적인 점유 포인트들이 나타나는 구간은 단일의 점유 구간이 된다. 또한, 유휴 포인트들이 나타나는 시간은 유휴 구간이 된다. 점유량, 점유 구간, 및 유휴 구간에 대한 정규 분포를 생성하였다. 이러한 분포로부터, 예측 타겟들에 대한 평균 값을 획득할 수 있다.

[0108] 입력 구축기와 측정기를 통해, 학습 데이터셋이 표 1의 각 DT 워크로드별로 생성된다. 그런 다음, 예측기는 학습될 수 있다.

[0109] 본 발명에서는, 학습 시간을 감소시키기 위해 전이 학습의 도입을 제안한다.

[0110] 우선, 여러 이종의 DT 세팅들에 대해 하나의 예측기(또는 예측 모델)로 정확한 예측, 즉 대응이 가능한지 확인할 필요가 있다. 이론적으로 여러 DT 세팅의 데이터셋을 하나의 예측기에 대해 훈련할 수 있다. 이론을 테스트하기 위해 DT 워크로드들을 1) V100 GPU와 2) 2080Ti GPU를 사용하여 두 가지 다른 DT 세팅에서 프로파일링한다. DT 세팅들은 GPU의 종류만 다르고, 나머지 DT 세팅 구성은 동일하다(PS 1개, 워커 2개, PCI 사용).

[0111] 위의 상황을 두 가지 시나리오를 통해 비교한다. 시나리오 1은 V100과 2080Ti GPU를 통합한 데이터셋으로 훈련된 하나의 예측기(V100+2080Ti)이고, 시나리오 2는 V100의 데이터셋과 2080Ti의 데이터셋에 의해 훈련된 각각의 예측기들이다.

[0112] 도 6A에서, 시나리오 2의 정확도가 시나리오 1에 비해 높음(V100 및 2080Ti에서 각각 170%, 110%)을 알 수 있다. 따라서, 하나의 예측기는 두 개(또는 여러 개)의 DT 설정을 정확하게 예측할 수 없음을 알 수 있으며, 이에 본 발명에서는 각 DT 세팅별로 별도의 예측기를 생성한다.

[0113] 그러나, DT 세팅에 따라 예측기를 훈련하려면 상당한 노력이 필요하다. 사용자가 GPU 유형 또는 GPU 수, PS 수, 워커 수 및 네트워크 연결에 따라 DT 세팅을 변경하고자 할 때 각 DT 세팅에 대한 데이터셋을 생성하는 것은 부담이 될 수 있다. 예를 들어, 도 6B의 시나리오 2에서는 합리적인 예측 정확도를 달성하려면 V100 모델에 대해서만 800개 이상의 DT 워크로드가 필요하다는 것을 발견하였다.

[0114] 더 나아가, 각 모델의 학습 시간은 더 심각한 문제이다. 도 6B에서, 시나리오 1의 학습시간은 149.8분임에 비해, 시나리오 2는 총 228.5분으로 시나리오 1보다 52% 길다. 따라서, 본 발명에서는 전이 학습을 활용하여 이러한 문제점을 극복하고자 한다.

[0115] 전이 학습은 사전에 잘 학습된 모델의 지식을 새로운 타겟 모델에 활용하는 학습 방법이다. 전이 학습 시 사전 학습 모델(pre-trained model)의 레이어(파라미터)를 일부 재사용하고, 일부 레이어는 새롭게 학습될 수 있다.

[0116] 본 발명의 예측기에 전이 학습을 적용하기 위해서는, 전이 학습 시 사용할 데이터셋의 크기 및 레이어를 재사용 또는 갱신할지 결정해야 한다. 본 발명에서는 다양한 데이터셋의 크기 및 레이어의 조합을 검토한 결과, 데이터셋의 크기가 기존 800개에서 320개로 줄어들고, 그래프 레이어의 일부(예컨대, 각 그래프 레이어의 후반부의 일부로써, 절반을 의미할 수도 있음) 및/또는 MLP 레이어(Multi-Layer Perceptron layer)를 갱신하고 나머지를 재사용할 때 정확도가 가장 좋음을 확인하였다. 구체적인 예로 4 개의 그래프 레이어들이 존재하는 경우, 제3 그래프 레이어의 파라미터들과 제4 그래프 레이어의 파라미터들이 갱신될 수 있다. 또다른 예로, 3 개의 그래프 레이어들이 존재하는 경우, 제3 그래프 레이어의 파라미터들과 제2 그래프 레이어의 파라미터들의 절반이 갱신될 수 있다.

[0117] 이하에서는, 본 발명에서 제안한 기법의 실험 결과를 설명한다.

[0118] 본 발명의 효과를 입증하기 위해, 14가지의 DT 세팅과 800종 이상의 DT 워크로드를 활용하였다. 구체적으로, DT 세팅은 아래 표 3과 같다.

[0119] [표 3]

Name	GPU	DP topology	Network	# of GPU machines	Name	GPU	DP topology	Network	# of GPU machines
V100-P1w2/ho-PCIe	V100	PS1/w2/ homo	Co-located	1	2080Ti-P4w4/he-40G	2080Ti	PS4/w4/ hetero	40 GbE	2
V100-P2w2/ho-PCIe	V100	PS2/w2/ homo	Co-located	1	TitanRTX-P4w4/he-40G	Titan RTX			
2080Ti-P1w2/ho-PCIe	2080Ti	PS1/w2/ homo	Co-located	1	V100-P5w5/he-1G	V100	PS5/w5/ hetero	1 GbE	5
2080Ti-P1w3/ho-PCIe	2080Ti	PS1/w3/ homo	Co-located	1	2080Ti-P5w5/he-1G	2080Ti			
2080Ti-P2w2/he-PCIe	2080Ti	PS2/w2/ hetero	Co-located	1	V100-P5w10/he-1G	V100	PS5/w10/ hetero	1 GbE	5
TitanRTX-P2w2/he-PCIe	Titan RTX				2080Ti-P5w10/he-1G	2080Ti			
2080Ti-P2w2/he-40G	2080Ti	PS2/w2/ hetero	40 GbE	2					
TitanRTX-P2w2/he-40G	Titan RTX								

[0120]

[0121] 상기 표 3에서, 가장 첫번째 세팅이 전이 학습의 사전학습 모델로 사용되었다. 구체적으로, 실험의 항목은 다음과 같다.

[0122] · 본 발명에서 제안하는 예측 모델의 예측 구조의 디자인 결정을 위한 실험: GNN 알고리즘, 레이어 개수, 그래프 리드아웃 레이어, 그룹핑 방식에 따른 정확도의 비교 분석

[0123] · 전이 학습의 효과 분석: 전이 학습을 통해 학습시간 7.3배 개선, 정확도는 전이 학습이 없을 때와 유사한 수준 유지

[0124] · 예측 정확도: GPU 활용률, GPU 메모리 활용률, 네트워크 TX 사용량, 네트워크 RX 사용량에 대해 평균 11%, 9%, 17%, 15%의 오차

[0125] · 사용 예시: 모델 학습 시 GPU 선택, 배치 사이즈 선택, 워커 수 선택 시 예측 기법 활용이 예시(동일한 환경에서 학습시간을 최대 2.4배 개선할 수 있는 조합 제시)

[0126] 도 7은 본 발명의 일 실시예에 따른 학습 데이터셋 생성 방법을 설명하기 위한 흐름도이다.

[0127] 학습 데이터셋 생성 방법은 적어도 프로세서 및/또는 메모리를 포함하는 컴퓨팅 장치에 의해 수행될 수 있다. 컴퓨팅 장치는 PC(Personal Computer), 서버(server), 태블릿 PC, 노트북 등을 포함할 수 있다. 실시예에 따라, 학습 데이터셋 생성 방법에 포함되는 단계들 중 적어도 일부는 컴퓨팅 장치의 프로세서에 의한 동작으로 이해될 수도 있다. 또한, 컴퓨팅 장치는 물리적으로 분리된 하나의 장치를 의미하거나, 물리적으로 분리된 복수의 컴퓨팅 장치들을 의미할 수도 있다. 이하, 학습 데이터셋 생성 방법을 설명함에 있어 앞선 기재와 중복되는 내용에 관하여는 그 구체적인 기재를 생략하기로 한다.

[0128] 우선, DT 코드에 대응하는 입력 데이터가 생성된다(S110). 입력 데이터는 GNN 모델의 입력 데이터로써, DT 코드에 대한 인접 행렬과 특징 행렬을 의미한다. 이를 위해, DT 코드는 그래프로 변환되고, 변환된 그래프로부터 인접 행렬과 특징 행렬이 추출될 수 있다. DT 코드는 컴퓨팅 장치에 포함되는 저장 장치에 미리 저장되어 있을 수 있다.

[0129] 다음으로, 결과 데이터가 측정된다(S120). 결과 데이터는 GNN 모델의 예측 결과에 해당하는 데이터로써, DT 코드에 대한 학습을 진행하는 도중에 측정될 수 있다. 측정되는 데이터는 학습 진행 과정에서의 GPU 활용률, GPU 메모리 활용률, 네트워크 TX 처리량, 및 네트워크 RX 처리량일 수 있다. 이를 위해, 학습을 수행하기 위한 분산 딥러닝 시스템이 이용될 수 있다. 즉, 특정한 DT 세팅에 대응하는 분산 딥러닝 시스템을 이용하여 학습을 진행하면서, 상술한 결과 데이터가 측정될 수 있다.

[0130] 마지막으로, 학습 데이터셋이 생성될 수 있다(S130). 학습 데이터는 GNN의 입력 데이터와 출력 데이터를 포함한다. 출력 데이터는 측정된 결과 데이터와 결과 데이터로부터 도출 가능한 복수의 예측 지표들(예컨대, 12 가지의 예측 지표들)에 해당하는 값들을 의미할 수 있다.

[0131] 학습 데이터셋은 DT 코드별로 생성되어야 한다. 따라서, DT 코드의 개수에 해당하는 횟수만큼 순차적으로 S110, S120, 및 S130 단계가 수행될 수 있다.

[0132] 또한, 학습 데이터셋은 DT 세팅별로 생성될 수 있다. 따라서, DT 세팅별로 상기 과정을 반복 수행함으로써 각 DT 세팅별로 DT 코드들 각각에 대응하는 학습 데이터셋이 생성될 수 있다.

[0133] 도 8은 본 발명의 일 실시예에 따른 예측 모델 생성 방법을 설명하기 위한 흐름도이다.

[0134] 예측 모델 생성 방법은 적어도 프로세서 및/또는 메모리를 포함하는 컴퓨팅 장치에 의해 수행될 수 있다. 컴퓨팅 장치는 PC(Personal Computer), 서버(server), 태블릿 PC, 노트북 등을 포함할 수 있다. 실시예에 따라, 예측 모델 생성 방법에 포함되는 단계들 중 적어도 일부는 컴퓨팅 장치의 프로세서에 의한 동작으로 이해될 수도 있다. 또한, 컴퓨팅 장치는 물리적으로 분리된 하나의 장치를 의미하거나, 물리적으로 분리된 복수의 컴퓨팅 장치들을 의미할 수도 있다. 이하, 예측 모델 생성 방법을 설명함에 있어 앞선 기재와 중복되는 내용에 관하여는 그 구체적인 기재를 생략하기로 한다.

[0135] 우선, 학습 데이터가 구축된다(S210). 학습 데이터는 GNN 모델의 입력 데이터와 입력 데이터에 대응하는 결과 데이터를 포함한다. 입력 데이터를 생성하기 위해, DT 코드는 그래프로 변환될 수 있고, 변환된 그래프로부터 인접 행렬과 특징 행렬이 추출될 수 있다. DT 코드 및 결과 데이터는 컴퓨팅 장치에 포함되는 저장 장치에 미리 저장되어 있을 수 있다.

[0136] 학습 데이터는 미리 정해진 DT 세팅의 DT 워크로드별로 생성될 수 있다. 따라서, DT 워크로드의 개수에 해당하는 횟수만큼의 반복적인 동작을 통해 학습 데이터셋이 구축될 수 있다.

[0137] 구축된 학습 데이터셋을 이용하여 GNN 모델을 학습함으로써 예측 모델이 생성될 수 있다(S220). 이때, 생성되는 예측 모델은 미리 정해진 DT 세팅에 대응하는 예측 모델일 수 있다. 하나의 예측 모델로 여러개의 DT 세팅에 대한 예측을 수행하는 것은 예측 정확도를 낮출 수 있기 때문이다.

[0138] 예측 모델 생성 방법은 생성된 예측 모델을 이용하여 예측을 수행하는 단계를 더 포함할 수도 있다. 예측을 수행함에 있어, 학습된 DT 세팅과는 상이한 DT 세팅에 대한 예측을 위해서는 전이 학습이 필요하다. 따라서, S220 단계 이후에 전이 학습을 수행하는 단계가 추가될 수 있다. 전이 학습은 예측을 수행하기 위한 DT 세팅에 대한 학습 데이터셋을 이용하여 수행될 수 있다. 전이 학습이 완료된 예측 모델을 이용하여 예측을 수행하면, 사전학습된 예측 모델이 학습한 DT 세팅과는 상이한 DT 세팅에 대하여도 높은 정확도록 예측을 수행할 수 있다.

[0139] 도 9는 본 발명의 일 실시예에 따른 예측 방법을 설명하기 위한 흐름도이다.

[0140] 예측 방법은 적어도 프로세서 및/또는 메모리를 포함하는 컴퓨팅 장치에 의해 수행될 수 있다. 컴퓨팅 장치는

PC(Personal Computer), 서버(server), 태블릿 PC, 노트북 등을 포함할 수 있다. 실시예에 따라, 예측 방법에 포함되는 단계들 중 적어도 일부는 컴퓨팅 장치의 프로세서에 의한 동작으로 이해될 수도 있다. 또한, 컴퓨팅 장치는 물리적으로 분리된 하나의 장치를 의미하거나, 물리적으로 분리된 복수의 컴퓨팅 장치들을 의미할 수도 있다. 이하, 예측 방법을 설명함에 있어 앞선 기재와 중복되는 내용에 관하여는 그 구체적인 기재를 생략하기로 한다.

[0141] 우선, DT 코드에 대응하는 입력 데이터가 생성된다(S310). 입력 데이터는 DT 코드에 대응하는 인접 행렬과 특징 행렬을 의미할 수 있다. 구체적으로 DT 코드에 대응하는 그래프가 생성되고, 생성된 그래프로부터 인접 행렬과 특징 행렬이 추출될 수 있다.

[0142] 다음으로, 예측 모델을 이용하여 타겟 지표에 대한 예측 동작이 수행될 수 있다(S320). 여기서, 예측 모델은 미리 정해진 DT 세팅에 대한 예측 모델로써, 미리 정해진 DT 세팅 하에서의 타겟 지표에 대한 예측을 수행할 수 있다. 예측 모델은 도 8을 통해 설명한 예측 모델 생성 방법을 통해 생성된 예측 모델로써, 컴퓨팅 장치에 포함되는 저장 장치에 미리 저장되어 있을 수 있다.

[0143] 또한, 전체 학습 시간은 예측된 유휴 시간과 점유 시간의 합에 전체 이터레이션의 횟수를 곱함으로써 도출될 수 있다.

[0144] 이상에서 설명된 장치는 하드웨어 구성 요소, 소프트웨어 구성 요소, 및/또는 하드웨어 구성 요소 및 소프트웨어 구성 요소의 집합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성 요소는, 예를 들어, 프로세서, 콘트롤러, ALU(Arithmetic Logic Unit), 디지털 신호 프로세서(Digital Signal Processor), 마이크로컴퓨터, FPA(Field Programmable 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(Operation System, 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 기술 분야에서 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.

[0145] 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, 구성 요소(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.

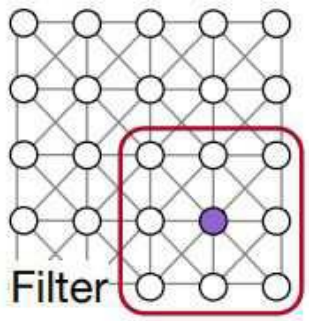
[0146] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(Magnetic Media), CD-ROM, DVD와 같은 광기록 매체(Optical Media), 플롭티컬 디스크(Floptical Disk)와 같은 자기-광 매체(Magneto-optical Media), 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.

[0147] 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. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성

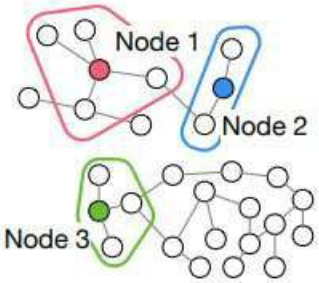
요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성 요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 따라서, 본 발명의 진정한 기술적 보호 범위는 첨부된 등록청구범위의 기술적 사상에 의해 정해져야 할 것이다.

도면

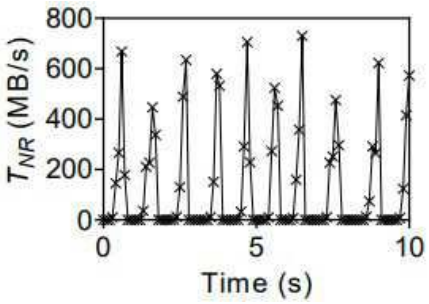
도면1a



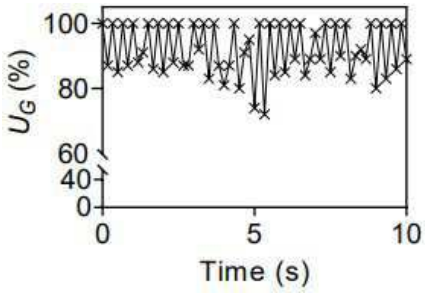
도면1b



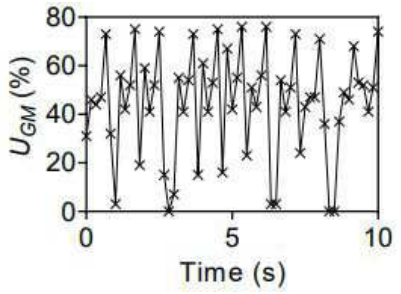
도면2a



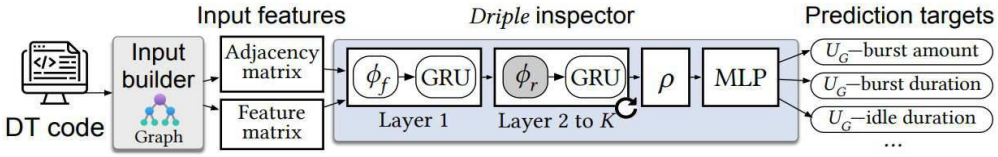
도면2b



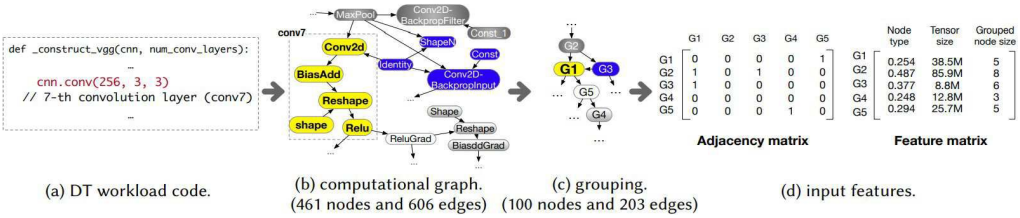
도면2c



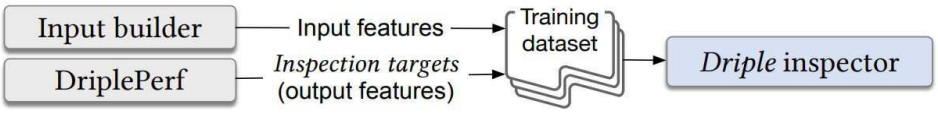
도면3



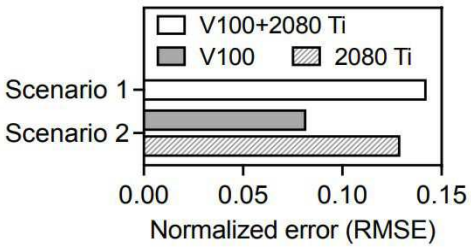
도면4



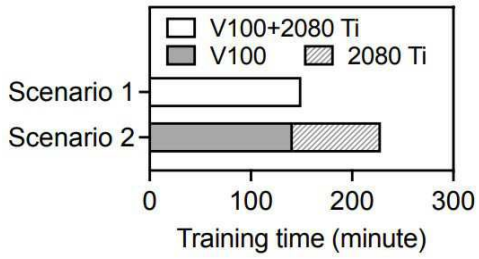
도면5



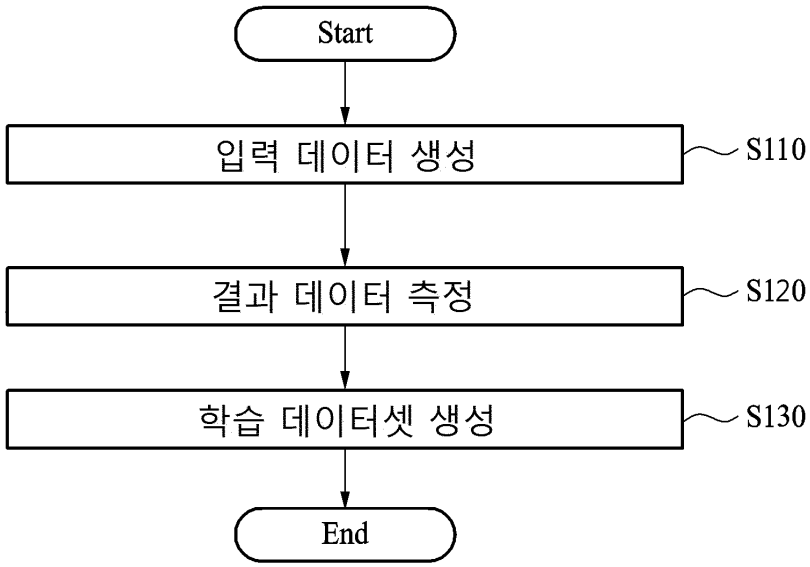
도면6a



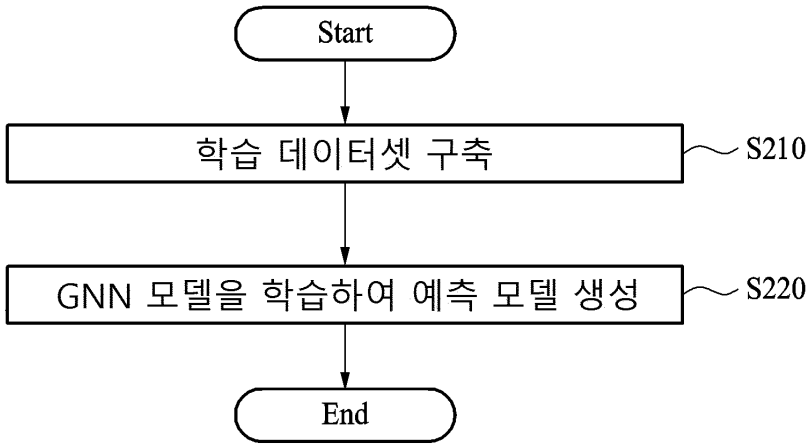
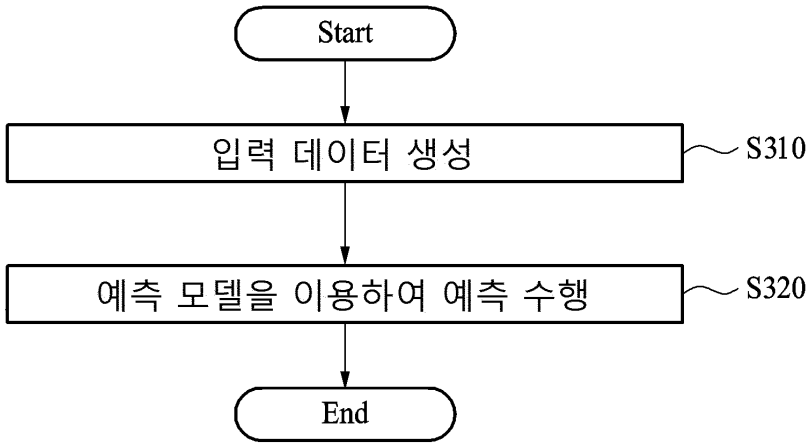
도면6b


V100+2080 Ti
V100
2080 Ti
Scenario 1
Scenario 2
0
100
200
300
Training time (minute)

도면7


Start
입력 데이터 생성
S110
결과 데이터 측정
S120
학습 데이터셋 생성
S130
End

도면8


Start
학습 데이터셋 구축
S210
GNN 모델을 학습하여 예측 모델 생성
S220
End

도면9